COLLEGEDICTAAT

GRAFENTHEORIE

DEEL 2

DOOR

L. C. M. KALLENBERG

Rijksuniversiteit Leiden

Mag. Uitgifte No. 69

INHOUDSOPGAVE Pagina

DEEL 1

INHOUDSOPGAVE

Pagina

DEEL 2

I. GRAFEN EN MATRICES

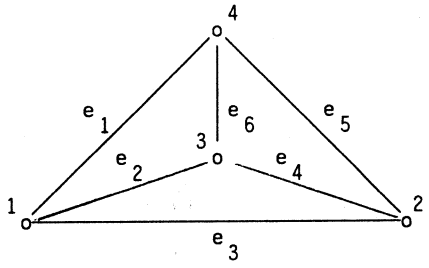
1. Grafen en vectorruimten

Literatuur:

* Hoofdstuk 6 in:
  N.Deo: "Graph theory with applications to engineering and computer science", Prentice-Hall, Englewood Cliffs (1974).
* Hoofdstuk 4 in:
  M.N.S.Swamey and K.Thulasiraman: "Graphs, networks and algorithms", Wiley, New York (1981).

Indien men een bepaalde graaf wil aanduiden, dan gebeurt dat veelal door een plaatje (zie figuur (1.1)) of door een bepaalde notatie (K_4).

(1.1)



Indien de computer wordt gebruikt, bv. als een algoritme wordt toegepast om de kortste paden in een graaf op te sporen, dan wordt de graaf gewoonlijk gerepresenteerd door een lijst of een matrix. Zo kunnen we een graaf representeren door een matrix die voor ieder knooppunt een rij en voor iedere tak een kolom heeft. De kolom behorende bij een zekere tak heeft 1'en in de twee rijen behorende bij de eindpunten van de tak en verder 0'en. Voor de graaf uit figuur (1.1) is deze matrix:

(1.2)

	e_1	e_2	e_3	e_4	e_5	e_6	← tak
1	1	1	1	0	0	0	
2	0	0	1	1	1	0	
3	0	1	0	1	0	1	
4	1	0	0	0	1	1	

↑ knoopppunt

Laat ons wederom de graaf K_4 beschouwen. Iedere deelverz. van de 6 takken kan worden gerepresenteerd door een 6-tal $(x_1,x_2,x_3,x_4,x_5,x_6)$ met:

$$x_i = \begin{cases} 1 \text{ als } e_i \text{ tot de deelverz. behoort} \\ 0 \text{ anders} \end{cases}$$

Zo wordt $\{e_1,e_3,e_4\}$ gerepresenteerd door (1,0,1,1,0,0). Voor de K_4 zijn er 2^6 van dergelijke 6-tallen.

In het algemeen hebben we bij een graaf met m takken 2^m m-tallen bestaande uit 0'en en 1'en. Zij W(G) de verz. van deze m-tallen.
Laat $\mathbb{Z}_2 = \{0,1\}$, met optelling (notatie $\oplus$) en vermenigvuldiging (notatie $\otimes$) modulo 2. Definiëer een optelling + op W(G) door:

$$x + y := (x_1 \oplus y_1, x_2 \oplus y_2, \ldots, x_m \oplus y_m),$$

en een vermenigvuldiging $\cdot$ tussen $\mathbb{Z}_2$ en W(G) door:

$$\alpha \cdot y := (\alpha \otimes x_1, \alpha \otimes x_2, \ldots, \alpha \otimes x_m).$$

Stelling 1.1
W(G) is een m-dimensionale vectorruimte over het lichaam $\mathbb{Z}_2$.
Bewijs
De gewenste eigenschappen zijn eenvoudig te verifiëren (ga dit zelf na). □

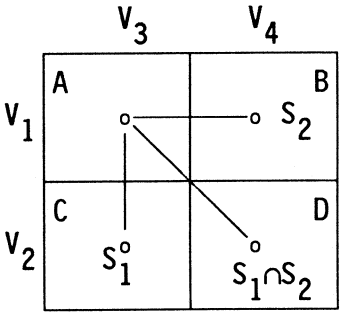
We introduceren de volgende deelverz. van W(G):
$W_C(G)$: de m-tallen van W(G) die behoren bij de nulgraaf, de enkelvoudige kringen en de vereniging van tak-disjuncte kringen.
$W_S(G)$: de m-tallen van W(G) die behoren bij de nulgraaf, de minimale sneden en de vereniging van tak-disjuncte sneden.

Stelling 1.2
$W_C(G)$ en $W_S(G)$ zijn lineaire deelruimten van W(G).
Bewijs
Uit de axioma's volgt dat het voldoende is om aan te tonen dat de som van twee elementen van $W_C(G)$ of van $W_S(G)$ weer tot $W_C(G)$ resp. $W_S(G)$ behoort.
Merk op dat een element tot $W_C(G)$ behoort d.e.s.d. als ieder knooppunt in de bijbehorende graaf een even graad heeft. Laat $x,y \in W_C(G)$ en zij $z := x + y$. We moeten nu aantonen dat in de door z geïnduceerde deelgraaf ieder knooppunt een even graad heeft. Er geldt voor een willekeurig knooppunt:

graad t.o.v. z = graad t.o.v. x + graad t.o.v. y - 2·(aantal takken van x dat ook tot y behoort en met dit knooppunt incident is)
≡ 0 (modulo 2).

Beschouw vervolgens twee sneden: $S_1 = (V_1,V_2)$ en $S_2 = (V_3,V_4)$. Definiëer:

$A = V_1 \cap V_3$, $B = V_1 \cap V_4$, $C = V_2 \cap V_3$, $D = V_2 \cap V_4$,
$V_5 = A \cup D$, $V_6 = B \cup C$.
Zij x en y de vectoren behorende bij S_1 resp. S_2, en laat z := x + y. De bij z behorende takkenverz. bestaat uit de takken die lopen tussen V_5 en V_6.
z behoort dus tot $W_S(G)$. □



Opgave 1.1

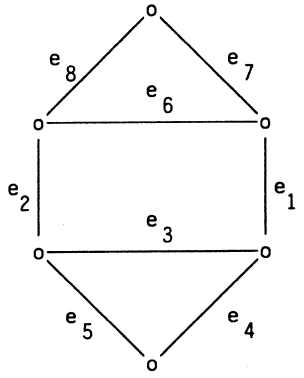
Als x behoort tot $W_C(G) \cap W_S(G)$, dan bevat x een even aantal 1'en. Toon dit aan.

Opgave 1.2

Laat $W_C(G) + W_S(G) = \{x + y \mid x \in W_C(G), y \in W_S(G)\}$.
Toon aan: $W_C(G) + W_S(G) = W(G)$ d.e.s.d. als $W_C(G) \cap W_S(G) = \{0\}$.
(Hint: Gebruik de eigenschap dat $\dim(X \cup Y) + \dim(X \cap Y) = \dim(X) + \dim(Y)$ voor X en Y lineaire vectorruimten).

Opgave 1.3

Beschouw nevenstaande graaf.
a. Ga na of $W_C(G) + W_S(G) = W(G)$.
b. Bepaal een $c \in W_C(G)$ en een $s \in W_S(G)$ zdd. $c + s = (1,1,\ldots,1)$.



2. De incidentiematrix

Literatuur:

* De paragrafen 7.1 en 7.2 van:
  N.Deo: "Graph theory with applications to engineering and computer science", Prentice-Hall, Englewood Cliffs (1974).
* De paragrafen 6.1 en 6.6 van:
  M.N.S.Swamey and K.Thulasiraman: "Graphs, networks and algorithms", Wiley, New York (1981).

Zij G een graaf met n knooppunten, m takken en zonder lussen. De <u>incidentiematrix A(G)</u> is een nxm matrix (a_{ij}) die het verband aangeeft tussen de knooppunten en de takken, en als volgt is gedefiniëerd:

niet-gerichte graaf: $a_{ij} = \begin{cases} 1 & \text{als tak j knooppunt i als eindpunt heeft} \\ 0 & \text{anders} \end{cases}$

gerichte graaf : $a_{ij} = \begin{cases} +1 & \text{als pijl j knooppunt i als beginpunt heeft} \\ -1 & \text{als pijl j knooppunt i als eindpunt heeft} \\ 0 & \text{anders} \end{cases}$

Voor een niet-gerichte graaf bevat iedere kolom van A(G) twee 1'en en verder nullen; als G een gerichte graaf is, dan heeft iedere kolom één +1, één -1 en verder nullen. De matrix (1.2) is de incidentiematrix van de graaf uit figuur (1.1).

Een matrix is <u>totaal unimodulair</u> als de determinant van iedere vierkante deelmatrix de waarde 0, +1 of -1 heeft (dus ook alle elementen van de matrix zijn 0, +1 of -1).

<u>Stelling 1.3</u>
De incidentiematrix A(G) van een niet-gerichte graaf G is totaal unimodulair d.e.s.d. als G bipartiet is.

<u>Bewijs</u>
⇒ Veronderstel dat A(G) totaal unimodulair is en dat G geen bipartiete graaf is. Dan bevat G een kring van oneven lengte, zeg $[v_1,v_2,\ldots,v_{2p+1},v_1]$.

Beschouw de vierkante deelmatrix van A(G) voortgebracht door $v_1, v_2, \ldots, v_{2p+1}$:

	v_1	v_2			.....				v_{2p+1}
v_1	1	0	0	0	.....	0	0	0	1
v_2	1	1	0	0	.....	0	0	0	0
.	0	1	1	0	.....	0	0	0	0
.	.....								
.	.....								
.	0	0	0	0	.....	0	1	1	0
v_{2p+1}	0	0	0	0	.....	0	0	1	1

Ontwikkel de determinant naar de eerste rij:
waarde = $1 + (-1)^{2p} \cdot 1 = 2$:
De matrix A(G) is niet totaal unimodulair: tegenspraak.

⇐ Veronderstel dat G bipartiet is, en dat de takken tussen V_1 en V_2 lopen. Neem een willekeurige kxk deelmatrix A_k van A(G). We bewijzen de bewering met inductie naar k (klopt voor k = 1). Kies een willekeurige kolom van A_k. Deze kolom bevat geen, één of twee 1'en. Als de kolom geen 1'en bevat, dan is $\det(A_k) = 0$; als de kolom één 1 bevat, dan ontwikkelen we de determinant naar deze kolom en is de waarde gelijk aan $\pm \det(A_{k-1})$ voor een zekere (k-1)x(k-1) matrix A_{k-1}: volgens de inductie-veronderstelling klopt de bewering ook in dit geval.
We moeten nu alleen nog het geval beschouwen dat iedere kolom precies twee 1'en bevat, één in een rij van V_1 en één in een rij van V_2. Nu is echter de som van de rijen van V_1 gelijk aan de som van de rijen van V_2 (nl. overal staat een 1). Er is dus afhankelijkheid: $\det(A_k) = 0$. □

Stelling 1.4

De incidentiematrix A(G) van een gerichte graaf G is totaal unimodulair.

Bewijs

Neem een vierkante kxk deelmatrix van A(G). We passen inductie naar k toe (klopt voor k = 1). Indien er een kolom is met hoogstens één element ongelijk aan 0, dan klopt de bewering (ontwikkel naar die kolom als in Stelling 1.3). Veronderstel dat iedere kolom twee elementen ongelijk aan 0 bevat, dus één +1 en één -1. Optellen van alle rijen geeft de 0-vector: afhankelijkheid en de determinant = 0. □

Veronderstel dat G een niet-gerichte graaf is met p componenten. Door de knooppunten en de takken geschikt te nummeren kan A(G) in blokvorm worden geschreven, met A_i de incidentiematrix van de i-de component:

$$(1.3) \quad A(G) = \begin{bmatrix} A_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & A_2 & 0 & \cdots & 0 & 0 \\ \cdot & & & & & \cdot \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdot & & & & & \cdot \\ \cdot & & & & & 0 \\ 0 & 0 & \cdots & \cdots & 0 & A_p \end{bmatrix}.$$

Iedere rij kan beschouwd worden als een element van de vectorruimte W(G). Omdat er in iedere kolom precies twee 1'en staan is de som van alle rijen 0 (in W(G) tellen we modulo 2): de rang van A_i is dus hoogstens n_i-1, met n_i = aantal knooppunten in i-de component.
Beschouw vervolgens de som van k van deze vectoren ($1 \leq k \leq n_i$-1), en veronderstel dat deze 0 is. Dit houdt voor iedere tak in: òf deze tak is met geen enkel van deze k knoopppunten incident òf heeft beide uiteinden in deze verz. van k knooppunten. Deze k knooppunten vormen dus één of meer componenten: tegenspraak. De rang van A_i is dus n_i-1. M.b.v. (1.3) is het volgende bewezen:

Stelling 1.5
Als G een niet-gerichte graaf is met n knooppunten en p componenten, dan is in de vectorruimte W(G): rang A(G) = n-p.

Stelling 1.6
Zij G een niet-gerichte graaf met n knooppunten. Dan geldt:
Een (n-1)x(n-1) deelmatrix A_1 van A(G) heeft onafhankelijke rijen d.e.s.d als de n-1 kolommen van A(G) corresponderen met de takken van een voortbrengende boom van G.
Bewijs
⇒ Zij G_1 de deelgraaf van G voortgebracht door de n-1 takken die corresponderen met A_1. Omdat rang $A(G_1)$ = rang (A_1) = n-1, is G_1 volgens Stelling 1.5 samenhangend, en omdat G_1 n-1 takken bevat is G_1 een voortbrengende boom.
⇐ Beschouw de deelgraaf corresponderend met de voortbrengende boom. Uit de redenering die tot Stelling 1.5 heeft geleid volgt dat iedere n-1 vectoren linear onafhankelijk zijn. □

Veronderstel vervolgens dat G een gerichte graaf is. De rijen van A(G) zullen we nu beschouwen als vectoren in $\mathbb{R}^m$ met de gebruikelijke optelling en vermenigvuldiging.

Stelling 1.7

Als G een gerichte graaf is met n knoopunten, m pijlen en p componenten, dan geldt in de vectorruimte $\mathbb{R}^m$: rang A(G) = n-p.

Bewijs

Ook voor een gerichte graaf met p componenten kan A(G), na hernummering, geschreven worden in de gedaante (1.3). Het is dus voldoende om het resultaat aan te tonen voor een samenhangende graaf.

De som van alle rijen geeft de 0-vector: rang A(G) $\leq$ n-1.

Beschouw een lineaire combinatie van n-1 rijen die ook de 0-vector oplevert. Daar in iedere kolom precies één +1 en één -1 staat, moeten de coëfficiënten van de rijen die "verbonden" zijn met de niet in de combinatie voorkomende rij 0 zijn. Van rijen verbonden met deze rij moet de coëfficiënt eveneens 0 zijn, etc. Vanwege de samenhang geldt dus dat alle coëfficiënten 0 zijn: de rang van A(G) = n-1. □

Opgave 1.4

Zij G een boom en laat A_1 uit A(G) verkregen worden door een willekeurige rij weg te laten. Toon aan dat $\det(A_1) = \pm 1$.

Opgave 1.5

Zij G een getekende vlakke, samenhangende graaf. Laat C de matrix zijn met als rijen de vectoren van de m-n+2 kringen de behoren bij de m-n+2 gebieden van G

a. Geef een voorbeeld van een normale graaf G waarvoor de bijbehorende matrix C niet totaal unimodulair is.

b. Toon aan:

C is totaal unimodulair d.e.s.d. als de duale graaf G^* bipartiet is.

3. De kringenmatrix

Literatuur:

* De paragrafen 7.3, 7.4 en 7.5 van:
  N.Deo: "Graph theory with applications to engineering and computer science", Prentice-Hall, Englewood Cliffs (1974).
* De paragrafen 6.3, 6.4, 6.5 en 6.6 van:
  M.N.S.Swamey and K.Thulasiraman: "Graphs, networks and algorithms", Wiley, New York (1981).

Zij G een samenhangende, niet-gerichte graaf met n knooppunten en m takken. Laat T een willekeurige voortbrengende boom zijn. T bestaat dus uit n-1 takken, zeg $e_{m-n+2}, e_{m-n+3}, \ldots, e_m$. Door aan T de tak e_k ($k = 1,2,\ldots,m-n+1$) toe te voegen ontstaat precies één kring. De m-n+1 aldus te construeren kringen hebben lineair onafhankelijke vectoren in $W_C(G)$, nl. e_k komt alleen voor in de k-de kring ($k = 1,2,\ldots,m-n+1$).

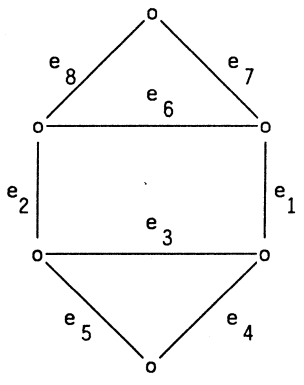
De bij T behorende <u>kringenmatrix</u> $C_T(G)$ is de matrix met als rijen deze m-n+1 vectoren. $C_T(G)$ heeft dus de gedaante:

$$C_T(G) = \left[\begin{array}{c|c} I_{m-n+1} & D_T(G) \end{array}\right] \begin{array}{l} \leftarrow C_1 \\ \leftarrow C_2 \\ \vdots \\ \leftarrow C_{m-n+1} \end{array} \qquad (1.4)$$

$$\begin{array}{c} e_1 \; e_2 \; \cdots \; e_{m-n+1} \quad e_{m-n+2} \; \cdots \; e_m \end{array}$$

<u>Voorbeeld 1.1</u>

Beschouw nevenstaande graaf en neem als voortbrengende boom T de takken $\{e_2, e_4, e_5, e_7, e_8\}$.
Dit geeft de volgende kringenmatrix:



$$C_T(G) = \left[\begin{array}{ccc|ccccc} e_1 & e_3 & e_6 & e_2 & e_4 & e_5 & e_7 & e_8 \\ \hline 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

De vector van de kring $[e_1, e_3, e_2, e_8, e_7]$ wordt verkregen door de kringen van e_1 en e_3 op te tellen. De volgende stelling toont aan dat dit algemeen geldt.

Stelling 1.8

Zij C een (som van tak-disjuncte) kring(en), bestaande uit de takken $\{e_j \mid j \in J\}$, en laat c de bijbehorende vector van $W_C(G)$ zijn.

Zij $J^* = \{ j \in J \mid e_j \notin T\} = \{j_1, j_2, \ldots, j_k\}$ met j_i de index van de rij van $C_T(G)$ behorende bij het i-de element van J^*. Dan geldt: $c = \Sigma_{i=1}^{k} [C_T(G)]_{j_i \cdot}$, waarbij $[C_T(G)]_{j_i \cdot}$ de j_i-de rij van $C_T(G)$ is.

Bewijs

Beschouw de vector $c + \Sigma_{i=1}^{k} [C_T(G)]_{j_i \cdot}$. Dit is weer een element van $W_C(G)$, dus de corresponderende takkenverz. is òf leeg òf bevat een kring.

Laat p zdd. $e_p \notin T$:

als $e_p \in C$: $p \in J^*$, zeg $p = j_s$, zodat:

$$\{c + \Sigma_{i=1}^{k} [C_T(G)]_{j_i \cdot}\}_p = c_p \oplus [C_T(G)]_{j_s p} = 1 \oplus 1 = 0.$$

als $e_p \notin C$: $p \notin J^*$, zodat:

$$\{c + \Sigma_{i=1}^{k} [C_T(G)]_{j_i \cdot}\}_p = c_p = 0.$$

Hier volgt dat $\{c + \Sigma_{i=1}^{k} [C_T(G)]_{j_i \cdot}\}_p = 0$ voor alle p met $e_p \notin T$.

De takken behorende bij positieve elementen van $c + \Sigma_{i=1}^{k} [C_T(G)]_{j_i \cdot}$ zijn dus een deelverz. van de boom T en kunnen daarom geen kring bevatten. Deze verz. is dus leeg, d.w.z. $c + \Sigma_{i=1}^{k} [C_T(G)]_{j_i \cdot} = 0$.

Omdat het een 0-1 vector is geldt: $c = \Sigma_{i=1}^{k} [C_T(G)]_{j_i \cdot}$. □

Gevolg

De rijen van $C_T(G)$ vormen een basis voor $W_C(G)$ en de vectorruimte $W_C(G)$ heeft dimensie m-n+1.

Stelling 1.9

Als $c \in W_C(G)$, dan is $A(G) \cdot c = 0$ (rekenen modulo 2).

Bewijs

Laat C de bij c behorende takkenverz. zijn. Beschouw een knooppunt v_i.

Als v_i geen eindpunt is van een tak van C:

de takken van G incident met v_i behoren niet tot C:

$c_j = 0$ voor alle j met $a_{ij} \neq 0$: $\Sigma_j a_{ij} c_j = 0$.

Als v_i wel een eindpunt is van een tak van C:
  v_i is eindpunt van een even aantal takken van C:
    $c_j = 1$ voor een even aantal j met $a_{ij} = 1$: $\Sigma_j\, a_{ij}c_j \equiv 0$ (modulo 2). □

Gevolg
$A(G)\cdot[C_T(G)]^t = 0$, waarbij $[C_T(G)]^t$ de getransponeerde van $C_T(G)$ is.

Tenslotte zullen we laten zien hoe de kringenmatrix uit de incidentiematrix bepaald kan worden.
Neem daartoe een voortbrengende boom T. In Stelling 1.6 is aangetoond dat de bij T behorende takken een (n-1)x(n-1) deelgraaf met onafhankelijke rijen oplevert. Na hernummering (zet de takken van de boom achteraan) kunnen we dus schrijven:

$$A(G) = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad \text{en} \quad C_T(G) = \left[I_{m-n+1} \quad D_T(G) \right],$$

met A_{12} de niet-singuliere (n-1)x(n-1) deelmatrix behorend bij de takken van T. Omdat $A(G)\cdot[C_T(G)]^t = 0$ (zie bovenstaand gevolg) kunnen we schrijven:

$$A_{11}I_{m-n+1} + A_{12}[D_T(G)]^t = 0 \;\rightarrow\; [D_T(G)]^t = -A_{12}^{-1}A_{11} = A_{12}^{-1}A_{11}.$$

De inverse A_{12}^{-1} kan via een standaard inverteermethode worden bepaald, waarbij de berekeningen modulo 2 moeten worden uitgevoerd.

Voorbeeld 1.1 (vervolg)
Met als voortbrengende boom $\{e_2,e_4,e_5,e_7,e_8\}$ krijgen we:

$$A(G) = \begin{array}{c} \begin{matrix} e_1 & e_3 & e_6 & e_2 & e_4 & e_5 & e_7 & e_8 \end{matrix} \\ \left[\begin{array}{ccc|ccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right] \end{array} \begin{matrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{matrix}$$

$$A_{12}^{-1} = \begin{array}{c} \begin{matrix} e_2 & e_4 & e_5 & e_7 & e_8 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$D_T(G) = [A_{12}^{-1}A_{11}]^t = \begin{array}{c} \begin{matrix} e_2 & e_4 & e_5 & e_7 & e_8 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}$$

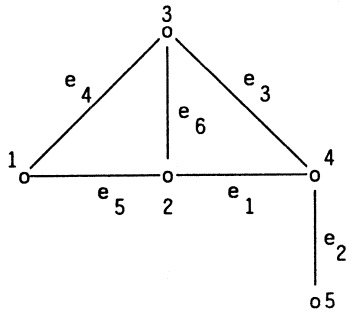
Opgave 1.6

Beschouw nevenstaande graaf, en neem

$T = \{e_2, e_3, e_5, e_6\}$.

a. Bepaal $C_T(G)$ door de onafhankelijke kringen op te sporen.

b. Bepaal $C_T(G)$ via $A(G)$.



Opgave 1.7

Zij G de graaf uit voorbeeld 1.1 en $T = \{e_2, e_4, e_5, e_7, e_8\}$.
Bepaal G_1 zdd. $C_T(G) = C_T(G_1)$, terwijl G en G_1 niet isomorf zijn.

4. De snedenmatrix

Literatuur:

* De paragrafen 7.6 en 7.7 van:
  N.Deo: "Graph theory with applications to engineering and computer science", Prentice-Hall, Englewood Cliffs (1974).
* De paragrafen 6.2 en 6.5 van:
  M.N.S.Swamey and K.Thulasiraman: "Graphs, networks and algorithms", Wiley, New York (1981).

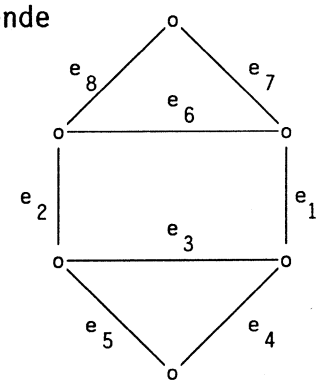
Zij G een samenhangende, niet-gerichte graaf met n knooppunten en m takken. Laat T een willekeurige voortbrengende boom zijn,bestaande uit de takken $e_{m-n+2}, e_{m-n+3}, \ldots, e_m$. Door een van de takken van T weg te laten is de resterende graaf niet meer samenhangend. Dus iedere tak e_k (k = m-n+2,m-n+3, ...,m) genereert een minimale snede. De n-1 aldus geconstrueerde sneden geven n-1 lineair onafhankelijke vectoren in $W_S(G)$, nl. e_k komt alleen voor in de k-de snede (k = m-n+2,m-n+3,...,m).

De bij T behorende <u>snedenmatrix</u> $S_T(G)$ is de matrix met als rijen deze n-1 vectoren. $S_T(G)$ heeft dus de gedaante:

$$S_T(G) = \begin{array}{c} \begin{matrix} e_1 & e_2 & \cdots & e_{m-n+1} & e_{m-n+2} & \cdots & e_m \end{matrix} \\ \left[\begin{array}{ccc|ccc} & & & & & & \\ & F_T(G) & & & I_{n-1} & & \\ & & & & & & \end{array}\right] \end{array} \begin{matrix} S_1 \\ S_2 \\ \cdot \\ \cdot \\ \cdot \\ S_{n-1} \end{matrix} \qquad (1.5)$$

<u>Voorbeeld 1.1 (vervolg)</u>

Beschouw weer nevenstaande graaf en neem als voortbrengende boom T de takken $\{e_2, e_4, e_5, e_7, e_8\}$.
Dit geeft de volgende snedenmatrix:



$$S_T(G) = \begin{array}{c} \begin{matrix} e_1 & e_3 & e_6 & e_2 & e_4 & e_5 & e_7 & e_8 \end{matrix} \\ \left[\begin{array}{ccc|ccccc} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \end{array}$$

De vector van de snede $[e_7,e_8]$ wordt verkregen door de sneden van e_7 en e_8 op te tellen. De volgende stelling toont aan dat dit algemeen geldt.

Stelling 1.10
Zij S een (som van tak-disjuncte) snede(n), bestaande uit de takken $\{e_j \mid j \in J\}$, en laat s de bijbehorende vector van $W_S(G)$ zijn.
Zij $J^* = \{ j \in J \mid e_j \in T\} = \{j_1,j_2,\ldots,j_k\}$ met j_i de index van de rij van $S_T(G)$ behorende bij het i-de element van J^*. Dan geldt: $s = \Sigma_{i=1}^{k} [S_T(G)]_{j_i\cdot}$.

Bewijs
Beschouw de vector $s + \Sigma_{i=1}^{k} [S_T(G)]_{j_i\cdot}$. Dit is weer een element van $W_S(G)$, dus de corresponderende takkenverz. is òf leeg òf bevat een kring.

Laat p zdd. $e_p \in T$:

als $e_p \in S$: $p \in J^*$, zeg $p = j_t$, zodat:

$$\{s + \Sigma_{i=1}^{k} [S_T(G)]_{j_i\cdot}\}_p = s_p \oplus [S_T(G)]_{j_tp} = 1 \oplus 1 = 0.$$

als $e_p \notin S$: $p \notin J^*$, zodat:

$$\{s + \Sigma_{i=1}^{k} [S_T(G)]_{j_i\cdot}\}_p = s_p = 0.$$

Hier volgt dat $\{s + \Sigma_{i=1}^{k}[S_T(G)]_{j_i\cdot}\}_p = 0$ voor alle p met $e_p \in T$.
De takken behorende bij positieve elementen van $s + \Sigma_{i=1}^{k}[S_T(G)]_{j_i\cdot}$ zijn dus disjunct met de boom T en kunnen daarom geen snede bevatten. Deze verz. is dus leeg, d.w.z. $s + \Sigma_{i=1}^{k}[S_T(G)]_{j_i\cdot} = 0$. Hieruit volgt: $s = \Sigma_{i=1}^{k} [S_T(G)]_{j_i\cdot}$. □

Gevolg
De rijen van $S_T(G)$ vormen een basis voor $W_S(G)$ en de vectorruimte $W_S(G)$ heeft dimensie n-1.

Stelling 1.11
$F_T(G) = [D_T(G)]^t$.
Bewijs
Omdat iedere snede een even aantal takken gemeen heeft met iedere kring, geldt:

$\Sigma_j [S_T(G)]_{ij}\cdot[C_T(G)]_{kj}$ = aantal takken in de doorsnede van de i-de snede en de k-de kring
$\equiv 0$ (modulo 2) voor iedere i en k.

Dus:

$$0 = S_T(G)\cdot[C_T(G)]^t = \left(F_T(G), I_{n-1}\right)\begin{pmatrix} I_{m-n+1} \\ [D_T(G)]^t \end{pmatrix} = F_T(G) + [D_T(G)]^t,$$

d.w.z. $F_T(G) = [D_T(G)]^t$. □

Gevolgen

1. De matrix $S_T(G)$ volgt direct uit $C_T(G)$ en omgekeerd.
2. De rijen van $C_T(G)$ vormen een basis voor de nulruimte van $S_T(G)$.
   immers: Uit het bewijs van Stelling 1.11 volgt dat iedere rij van $C_T(G)$ in de nulruimte van $S_T(G)$ zit. Omdat de m-n+1 rijen van $C_T(G)$ lineair onafhankelijk zijn, en omdat:
   dimensie nulruimte $S_T(G)$ = m - dimensie $S_T(G)$ = m-n+1,
   is $C_T(G)$ een basis voor de nulruimte van $S_T(G)$.
3. De rijen van $S_T(G)$ vormen een basis voor de nulruimte van $C_T(G)$.
   (analoog aan 2).

Stelling 1.12

S is een minimale snede d.e.s.d. als S een minimale verz. takken is die tenminste één tak gemeen heeft met iedere voortbrengende boom.

Bewijs

⇒ Triviaal.

⇐ De takken van G die niet tot S behoren bevatten geen enkele voortbrengende boom. Deze takken vormen dus geen samenhangende deelgraaf meer. S is dus een splitsende verz. en omdat S minimaal is is S een minimale snede. □

Opgave 1.8

De verz. van alle voortbrengende bomen van een graaf G bestaat uit: (e_2,e_3,e_5), (e_1,e_4,e_5), (e_1,e_3,e_4), (e_2,e_3,e_4), (e_1,e_2,e_3), (e_1,e_2,e_4), (e_2,e_4,e_5) en (e_1,e_3,e_5).
Bepaal voor $T = \{e_2,e_3,e_5\}$ de snedenmatrix $S_T(G)$.

Opgave 1.9

Zij G een samenhangende graaf. Laat S_i de snede zijn bestaande uit de takken incident met v_i, $i = 1,2,\ldots,n-1$.
Toon aan dat de vectoren $S_1,S_2,\ldots,S_{n-1}$ een basis vormen voor $W_S(G)$.

Opgave 1.10*

Toon aan dat er voor iedere graaf G vectoren $c \in W_C(G)$ en $s \in W_S(G)$ zijn zdd. $c + s = (1,1,\ldots,1)$.
(zie ook opgave 1.3).

Opgave 1.11*

a. Zij C_1 een $(m-n+1)x(m-n+1)$ deelmatrix van $C_T(G)$.
   Toon aan:
   C_1 is niet-singulier d.e.s.d. als de takken van $E\backslash C_1$ een voortbrengende boom vormen.

b. Zij S_1 een $(n-1)x(n-1)$ deelmatrix van $S_T(G)$.
   Toon aan:
   S_1 is niet-singulier d.e.s.d. als de takken van S_1 een voortbrengende boom vormen.

Opgave 1.12*

Toon via de volgende stappen aan dat de snedenmatrix en kringenmatrix van een bipartiete graaf totaal unimodulair is.

a. Beschouw een $(n-1)x(n-1)$ deelmatrix S_1 van $S_T(G)$.
   Toon aan dat $\det(S_1) = \pm 1$ of 0.

b. Beschouw een kxm deelmatrix S_2 van $S_T(G)$. Laat zien dat er een graaf G_1 is met eveneens m takken en waarvoor geldt: $S_{T_1}(G_1) = S_2$ voor een zekere voortbrengende boom T_1 van G_1.

c. Toon aan dat $S_T(G)$ totaal unimodulair is.

d. Toon aan dat $C_T(G)$ totaal unimodulair is.

* = opgave met een moeilijkheidsgraad die het tentamen niveau te boven gaat.
(als ze behandeld zijn op college behoren ze wel tot de tentamenstof)

5. De padenmatrix

Literatuur:

* Paragraaf 7.8 van:
  N.Deo: "Graph theory with applications to engineering and computer science", Prentice-Hall, Englewood Cliffs (1974).
* Paragraaf 5.6 van:
  R.G.Busacker and T.L.Saaty: "Finite graphs and networks: an introduction with applications", McGraw-Hill, New York, 1965.

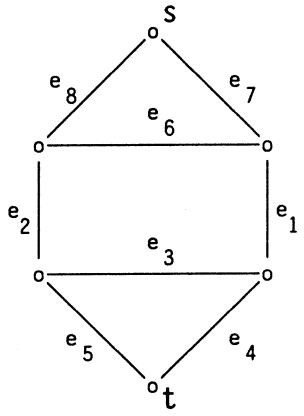
De <u>padenmatrix</u> is een matrix die meestal bij gerichte grafen wordt beschouwd, maar ook voor niet-gerichte grafen bestaat (deze zou dan eigenlijk ketensmatrix moeten heten).
De matrix is gedefiniëerd voor een gespecificeerd tweetal knooppunten, zeg s en t, van de graaf, en wordt genoteerd met $P(s,t) = (p_{ij})$. De rijen van de matrix corresponderen met de paden (ketens) van s naar t.
Gerichte graaf:

$$p_{ij} = \begin{cases} 1 \text{ als pijl } a_j \text{ ligt op het i-de pad van s naar t} \\ 0 \text{ anders} \end{cases}$$

Niet-gerichte graaf:

$$p_{ij} = \begin{cases} 1 \text{ als tak } e_j \text{ ligt op de i-de keten tussen s en t} \\ 0 \text{ anders} \end{cases}$$

<u>Voorbeeld 1.1 (vervolg)</u>
Beschouw nevenstaande graaf.
De ketens tussen s en t zijn:
$[e_7,e_1,e_4]$, $[e_7,e_1,e_3,e_5]$, $[e_7,e_6,e_2,e_5]$,
$[e_7,e_6,e_2,e_3,e_4]$, $[e_8,e_2,e_5]$, $[e_8,e_2,e_3,e_4]$,
$[e_8,e_6,e_1,e_4]$, $[e_8,e_6,e_1,e_3,e_5]$.



$$P(s,t) = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array}$$

Stelling 1.13

Zij G een niet-gerichte graaf.

De rijen van de matrix $A(G)\cdot[P(s,t)]^t$ niet behorend bij s en t leveren de nulvector op; de rijen van s en t bestaan uit allemaal 1'en (rekenen modulo 2).

Bewijs

Het element (i,j) van $A(G)\cdot[P(s,t)]^t$ is gelijk aan:

$$\Sigma_k\, a_{ik}p_{jk} = \text{aantal takken } e_k \text{ incident met } v_i \text{ en j-de keten}$$

$$= \begin{cases} 2 \text{ als } v_i \text{ op de j-de keten ligt en } i \neq s,t \\ 0 \text{ als } v_i \text{ niet op de j-de keten ligt en } i \neq s,t \\ 1 \text{ als } i = s \text{ of } i = t \end{cases}$$

□

Opgave 1.13

Zij G een gerichte graaf. Toon het volgende aan:

De rijen van de matrix $A(G)\cdot[P(s,t)]^t$ niet behorend bij s en t leveren de nulvector op; de rijen van s bestaan uit allemaal 1'en, en die van t uit -1'en.

6. De structuurmatrix

Literatuur:

* Paragraaf 7.9 van:
  N.Deo: "Graph theory with applications to engineering and computer science", Prentice-Hall, Englewood Cliffs (1974).
* Paragraaf 6.10 van:
  M.N.S.Swamey and K.Thulasiraman: "Graphs, networks and algorithms", Wiley, New York (1981).

De structuurmatrix $Q(G) = (q_{ij})$ is een nxn matrix die het verband aangeeft tussen de knooppunten:

$$q_{ij} = \begin{cases} \text{het aantal takken tussen } v_i \text{ en } v_j \text{ als de graaf niet-gericht is} \\ \text{het aantal pijlen van } v_i \text{ naar } v_j \text{ als de graaf gericht is} \end{cases}$$

Stelling 1.14

$[Q(G)]^k$ geeft het aantal takkenreeksen (pijlenreeksen) van de lengte k aan.

Bewijs

We passen volledige inductie naar k toe (voor k = 1 klopt de bewering).

$$\{[Q(G)]^k\}_{ij} = \Sigma_{p=1}^{n} \{[Q(G)]^{k-1}\}_{ip} \cdot q_{pj}$$

$= \Sigma_{p=1}^{n}$ aantal takken (pijlen) reeksen van de lengte k-1 van v_i naar v_p, vermenigvuldigd met het aantal takken (pijlen) van v_p naar v_j

$= \Sigma_{p=1}^{n}$ aantal takken (pijlen) reeksen van de lengte k tussen v_i en v_j met v_p vóór v_j

= aantal takken (pijlen) reeksen van de lengte k van v_i naar v_j. □

Gevolg

Voor een gerichte graaf G geldt:
G heeft geen ronden d.e.s.d. als $[Q(G)]^k = 0$ voor zekere $k \leq n$.

Behalve de structuurmatrix wordt voor de representatie van een normale graaf ook vaak gebruik gemaakt van de structuurlijst. Bij een structuurlijst wordt voor ieder knooppunt v bijgehouden met welke andere knooppunten v incident is:

$$L[v] = \begin{cases} \{w \mid v \text{ en } w \text{ verbonden door een tak}\} & \text{als } G \text{ niet-gericht is} \\ \{w \mid \text{er is een pijl van } v \text{ naar } w\} & \text{als } G \text{ gericht is} \end{cases}$$

Vervolgens zullen we laten zien dat m.b.v. de structuurmatrix in een streng samenhangende graaf (of component) een natuurlijke klassenindeling kan worden gemaakt. Tevens kan de zogenaamde periode met een elegant algoritme berekend worden.

Zij $G = (V,A)$ een gerichte graaf met structuurmatrix $Q(G) = (q_{ij})$, en laat $q_{ij}^{(k)}$ het (i,j)-de element van $[Q(G)]^k$ zijn.
Definiëer voor iedere $v_i \in V$ de indexverz. $I(v_i)$ en de periode $d(v_i)$ door:

$$I(v_i) = \{k \geq 1 \mid q_{ii}^{(k)} > 0\} \quad \text{en} \quad d(v_i) = \text{g.g.d. } \{k \mid k \in I(v_i)\}.$$

Stelling 1.15
Als $I(v_i) \neq \emptyset$, dan is er een k_i zódanig dat $kd(v_i) \in I(v_i)$ voor alle $k \geq k_i$.
Bewijs
Omdat $k \in I(v_i)$ d.e.s.d. als er een pijlenreeks is ter lengte k van v_i naar v_i geldt: als $k_1,k_2 \in I(v_i)$, dan ook $k_1 + k_2 \in I(v_i)$.
Definiëer d_i, $k_1(i)$ en $k_2(i)$ door:

$$d_i := \min \{k \geq 1 \mid k = k_1 - k_2 \text{ met } k_1,k_2 \in I(v_i)\} = k_1(i) - k_2(i).$$

We zullen aantonen dat $d_i = d(v_i)$.
Omdat $d(v_i)$ een deler is van $k_1(i)$ en $k_2(i)$, is $d(v_i)$ ook een deler van d_i.
Neem $k \in I(v_i)$ willekeurig en laat $k = pd_i + r$ met $p \geq 0$ en $0 \leq r \leq d_i - 1$.
Omdat $k + pk_2(i) \in I(v_i)$, $pk_1(i) \in I(v_i)$ en $r = k + pk_2(i) - pk_1(i)$, geldt:
òf $r = 0$ òf $r \geq d_i$, waaruit volgt $r = 0$: d_i is deler van elke $k \in I(v_i)$: d_i is ook een deler van $d(v_i)$. Hiermee is aangetoond dat $d_i = d(v_i)$.
Laat $k_i := [k_2(i)\{k_2(i) - 1\}]/d_i^2$, en neem $k \geq k_i$.
Schrijf $kd_i = pk_2(i) + r$ met $p \geq 0$ en $0 \leq r \leq k_2(i) - 1$.
d_i is een deler van $k_2(i)$, dus ook van $kd_i - pk_2(i) = r$: $r = sd_i$. Nu geldt:

$$kd_i = pk_2(i) + s\{k_1(i) - k_2(i)\} = (p - s)k_2(i) + sk_1(i).$$

Omdat $k \geq k_i$, kunnen we schrijven:

$$pk_2(i) + r = kd_i \geq k_i d_i = k_2(i)\{k_2(i) - 1\}/d_i \geq k_2(i) \cdot r/d_i = sk_2(i),$$

waaruit volgt dat $p - s \geq 0$. Er geldt dus:

$$kd_i = (p - s)k_2(i) + sk_1(i), \text{ met } (p - s) \geq 0 \text{ en } s \geq 0: kd_i \in I(v_i). \qquad \square$$

Stelling 1.16

In een streng samenhangende graaf hebben alle knooppunten dezelfde periode.

Bewijs

Het is voldoende om het volgende aan te tonen: als er een pad is van v naar w en een pad van w naar v, dan is d(v) = d(w). Zeg er is een pad ter lengte k_1 van v naar w en een pad ter lengte k_2 van w naar v.
Laat $d(v) = k_1(v) - k_2(v)$ met $k_1(v), k_2(v) \in I(v)$.
Dan is er dus een pijlenreeks ter lengte $k_2 + k_1(v) + k_1$ van w naar w en ook ter lengte $k_2 + k_2(v) + k_1$. Hieruit volgt voor het verschil:

$$k_1(v) - k_2(v) \geq d(w), \text{ d.w.z. } d(v) \geq d(w).$$

Geheel analoog is aan te tonen dat $d(w) \geq d(v)$. Dus geldt: $d(v) = d(w)$. □

Gevolg

We kunnen over de periode van een streng samenhangende graaf spreken.

Stelling 1.17

Zij G = (V,A) een streng samenhangende graaf met periode d.
Laat T = (V,B) een voortbrengende gerichte boom zijn met wortel v_1 en laat k(i) het aantal pijlen in T zijn van v_1 naar v_i (i = 2,3,...,n). Dan geldt:
De periode van G = g.g.d. $\{k(i) + 1 - k(j) \mid (v_i,v_j) \in A - B\}$.

Bewijs

Laat C een gesloten pijlenreeks in G zijn, bestaande uit k pijlen.

$$\Sigma_{(v_i,v_j)\in C} \{k(i) + 1 - k(j)\} = k + \Sigma_{(v_i,v_j)\in C} \{k(i) - k(j)\} = k.$$

Anderzijds geldt omdat $k(j) = k(i) + 1$ als $(v_i,v_j) \in B$:

$$\Sigma_{(v_i,v_j)\in C-B} \{k(i) + 1 - k(j)\} =$$

$$\Sigma_{(v_i,v_j)\in C} \{k(i) + 1 - k(j)\} - \Sigma_{(v_i,v_j)\in C\cap B} \{k(i) + 1 - k(j)\} = k - 0 = k.$$

Laat nu g := g.g.d. $\{k(i) + 1 - k(j) \mid (v_i,v_j) \in A - B\}$.
Uit het bovenstaande volgt dat g een deler is van k en, omdat k willekeurig is, dat g een deler is van de lengte van alle gesloten pijlenreeksen, d.w.z. g is een deler van de periode d.
Zij $(v_i,v_j) \in A - B$ willekeurig en k_j de lengte van een pad van v_j naar v_1 in G. Dan geldt:

d deler van $k(i) + 1 + k_j$
d deler van $k(j) + k_j$
} d deler van $k(i) + 1 - k(j)$: d deler van g.

Hiermee is bewezen dat d = g. □

Stelling 1.18

Zij G = (V,A) een streng samenhangende graaf met periode d.
Dan is er een partitie $V_1,V_2,\ldots,V_d$ van V zódanig dat als $(v_i,v_j) \in A$ en $v_j \in V_{k+1}$, dan $v_i \in V_k$, waarbij $V_{d+1} = V_1$.

Bewijs

Beschouw de graaf $G^{(d)}$ behorende bij de structuurmatrix $[Q(G)]^d$. Neem als partitie de streng samenhangende componenten van $G^{(d)}$.

We zullen allereerst bewijzen dat deze partitie minstens d verz. heeft. Zij V_1 de verz. waarin knooppunt v_1 zit.

Als voor i en j geldt dat $q_{ij}^{(k)} > 0$ en $q_{ij}^{(s)} > 0$, dan is d een deler van k - s (immers: laat t zódanig dat $q_{ji}^{(t)} > 0$, dan is d een deler van k + t en van s + t, dus ook van het verschil k - s).

Als er in G een pijlenreeks is van v_i naar v_j ter lengte $k < d$, dan zitten v_i en v_j in verschillende deelverz. (immers: stel v_i en v_j in dezelfde verz., dan is er in $G^{(d)}$ een pijlenreeks in beide richtingen tussen v_i en v_j; dus in G is er een pijlenreeks ter lengte pd van v_i naar v_j voor zekere p, wat inhoudt dat d een deler is van pd - k, dus ook van k: tegenspraak met $k < d$). Hieruit volgt dat de partitie minstens d deelverz. bevat.

Beschouw nu een pad $v_1 = v_{i_1},v_{i_2},\ldots,v_{i_d}$ en nummer de deelverz. van de partities zódanig dat $v_{i_k} \in V_k$ $(k = 1,2,\ldots,d)$. Kies een willekeurig knooppunt v_j, laat $q_{1j}^{(k)} > 0$ voor zekere k, en schrijf $k = pd + r$, met $0 \leq r \leq d-1$.

We zullen aantonen dat $v_j \in V_{r+1}$.

Laat s zódanig dat $q_{i_{r+1}j}^{(s)} > 0$. We kunnen nu schrijven:

$$\left.\begin{array}{l} q_{1j}^{(r+pd)} > 0 \\ q_{1j}^{(r+s)} > 0 \end{array}\right\} \text{ d is deler van pd - s: d is deler van s.}$$

In $G^{(d)}$ is er dus een pad van $v_{i_{r+1}}$ naar v_j. Daar de lengte van iedere ronde een veelvoud is van d, en het pad van $v_{i_{r+1}}$ naar v_j in G een veelvoud is van d, heeft ieder pad van v_j naar $v_{i_{r+1}}$ ook als lengte een veelvoud van d, d.w.z. in $G^{(d)}$ is er een pad van v_j naar $v_{i_{r+1}}$: $v_j \in V_{r+1}$.

Uit bovenstaande volgt dat er precies d deelverz. zijn, en dat als $(v_i,v_j) \in A$ en $v_j \in V_{k+1}$, dan $v_i \in V_k$. □

Bovenstaande stellingen tonen de correctheid aan van het volgende algoritme.

Algoritme 1.1: *Bepaling perioden en partities in gerichte graaf.*

Input: G = (V,A) een gerichte graaf.
Output: Voor ieder van de streng samenhangende componenten wordt de periode en de partitie (volgens Stelling 1.18) bepaald.

1. Bepaal de streng samenhangende componenten van G.
2. Doe voor ieder van de streng samenhangende componenten het volgende:
   a. Kies een knooppunt v_1 en bepaal een voortbrengende gerichte boom T = (V,B) met wortel v_1.
   b. Bereken voor iedere v_i het aantal pijlen k(i) om van v_1 in T naar v_i te komen.
   c. Bepaal d := g.g.d. $\{k(i) + 1 - k(j) \mid (v_i,v_j) \in A - B\}$.
   d. Bepaal $V_i := \{v_j \mid k(j) \equiv i-1 \pmod d\}$, $i = 1,2,\ldots,d$.

Opmerkingen

1. In het volgende hoofdstuk zullen we afleiden hoe de streng samenhangende componenten van een gerichte graaf kunnen worden bepaald. De daar gebruikte methode levert tevens een voortbrengende gerichte boom vanuit knooppunt v_1 op.
2. De g.g.d. van een aantal getallen kan met het algoritme van Euclides worden bepaald. Dit algoritme berekent de g.g.d. van a en b (met $a \geq b$) en is gebaseerd op het herhaald toepassen van de eigenschap dat g.g.d. (a,b) = g.g.d. (a-b,b). We geven dit algoritme expliciet hieronder.

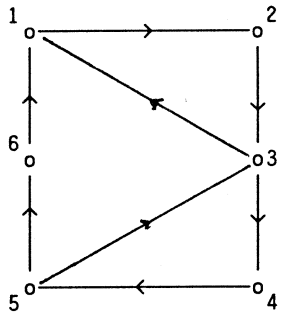
Algoritme 1.2: *Bepaling g.g.d. van a en b.*

Input: Twee natuurlijke getallen a en b.
Output: De grootste gemene deler van a en b.

```
function ggd(a,b:integer): integer;
  begin
    if b = 0 then  ggd := a
    else ggd := ggd(b,a mod b)
  end
```

Voorbeeld 1.2

Beschouw nevenstaande streng samenhangende graaf.



Neem als voortbrengende boom:
{(1,2),(2,3),(3,4),(4,5),(5,6)}.
Dan is A - B = {((6,1),(3,1),(5,3)}
Boomafstand k(i) = i-1, 1 = 1,2,...,6.
d = g.g.d. {6,3,3} = 3.
V_1 = {1,4}; V_2 = {2,5}; V_3 = {3,6}.

Opgave 1.14

Zij G een normale, niet-gerichte graaf. Toon het volgende aan:

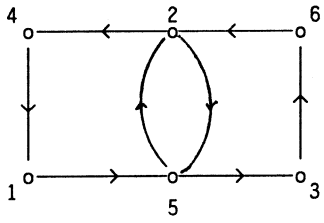
(i) Het aantal kringen in G bestaande uit 3 takken = $\frac{1}{6} \Sigma_{i=1}^{n} q_{ii}^{(3)}$;

(ii)* Het aantal kringen in G bestaande uit 4 takken =

$$\frac{1}{8} \cdot [\Sigma_{i=1}^{n} q_{ii}^{(4)} - 2m - 2 \cdot \Sigma_{i=1}^{n} \Sigma_{j \neq i} q_{ij}^{(2)}].$$

Opgave 1.15

Bepaal met algoritme 1.1 de periode en de partitie van nevenstaande streng samenhangende graaf



II. COMPLEXITEIT EN STANDAARDALGORITMEN

1. Complexiteitstheorie

Literatuur:

* A.V.Aho, J.E.Hopcroft and J.D.Ullman: "The design and analysis of computer algorithms", Addison-Wesley, Reading (1974).
* M.R.Garey and D.S.Johnson: "Computers and intractability: a guide to the theory of NP-completeness", Freeman, San Francisco (1979).
* C.H.Papadimitriou and K.Steiglitz: "Combinatorial optimization: algorithms and complexity", Prentice-Hall, Englewood Cliffs (1982).

Onder het woord probleem zullen we een algemene vraagstelling met een aantal parameters verstaan. Bijvoorbeeld: bepaal een voortbrengende gerichte boom T in een graaf G; vind een optimale oplossing van een lineair programmerings-probleem. Als de parameters zijn gespecificeerd (graaf G is gegeven; de doelfunctie en de beperkingen van het LP-probleem zijn bekend) dan spreken we van een instantie van een probleem.

Om problemen op te lossen gebruiken we algoritmen. Het is vaak gewenst om algoritmen, zowel voor dezelfde als voor verschillende problemen, met elkaar te vergelijken. Als maat voor de performance van een algoritme is het gebruikelijk om te kijken naar de tijdsduur voordat het eindantwoord wordt gegeven. We spreken dan over de tijdscomplexiteit. Soms wordt ook gekeken naar de hoeveelheid (geheugen)ruimte die door het algoritme in beslag wordt genomen. Dit correspondeert met de ruimtecomplexiteit. We zullen ons hier beperken tot de tijdscomplexiteit.

De tijdsduur is sterk afhankelijk van de gebruikte computer. Vandaar dat we niet de tijd zelf nemen, maar dat we het aantal elementaire stappen tellen. Een elementaire stap is een optelling, vermenigvuldiging, toekenning, vergelijking e.d. op een hypothetische computer, de Turing machine. Deze machine zelf zullen we hier niet bespreken. Wat geteld wordt zal later uit de context duidelijk worden. We maken dus ook geen onderscheid tussen de verschillende operaties; ze hebben alle tijdsduur één. We zullen nu de complexiteitstheorie op een wat informele manier bespreken.

We zullen de tijdsduur uitdrukken in de inputparameters van het probleem. Bij een probleem zullen we steeds afspreken wat de inputparameters zijn. Bij een nxn matrix zou dit de parameter n kunnen zijn. Verder vergelijken we alle mogelijke performances tegelijkertijd als functie van de inputparameters en wel zó dat we letten op de slechts mogelijke performance. We letten dus op het worst-case gedrag van het algoritme.

Dit worst-case gedrag vergelijken we globaal (d.w.z. bij parameter n maken we geen onderscheid tussen n^2 en $2n^2$ stappen) en alleen voor grotere waarden van de inputparameters. We zijn vooral geïnteresserd in de snelheid waarmee de tijdsduur groeit als functie van de parameter(s). Om deze groeisnelheid te beschrijven gebruiken we de volgende begrippen. De gebruikte functies zijn functies van de natuurlijke getallen $\mathbb{N}$ naar de positieve reële getallen $\mathbb{R}^+$.

a. We schrijven $f(n) = \mathcal{O}(g(n))$ als er constanten c en n_0 bestaan zódanig dat $f(n) \leq cg(n)$ voor alle $n \geq n_0$; in dit geval zeggen we dat f niet sneller groeit dan g.
b. We schrijven $f(n) = \Omega(g(n))$ als er constanten c en n_0 bestaan zódanig dat $f(n) \geq cg(n)$ voor alle $n \geq n_0$; in dit geval zeggen we dat f niet langszamer groeit dan g.
a. We schrijven $f(n) = \Theta(g(n))$ als er constanten c,d en n_0 bestaan zódanig dat $cg(n) \leq f(n) \leq dg(n)$ voor alle $n \geq n_0$; in dit geval zeggen we dat f en g even snel groeien.

Opmerking

Als $\lim_{n\to\infty} \frac{f(n)}{g(n)} \leq c$, dan is $f(n) = \mathcal{O}(g(n))$.

Voorbeeld 2.1

Laat $f(n) = 10n^3 + 100n^2 + 200$. Dan geldt dat $f(n) = \Theta(n^3)$, want:

$n^3 \leq f(n) \leq 11n^3$ voor alle $n \geq 101$.

We meten de complexiteit van een algoritme als functie van de grootte van de input. Maar wat is de grootte van de input? Dit hangt af van de representatie van het object. Een graaf kan bijvoorbeeld gegeven zijn door de nxm incidentiematrix, of door de nxn structuurmatrix, of door de structuurlijst. Deze laatste representatie bevat 2m elementen. In de volgende paragrafen

zullen we bij een probleem steeds aangeven wat de representatie en/of de grootte van de input is.

Wanneer vinden we dat een algoritme voldoende snel is? In de praktijk blijkt dat algoritmen met een polynomiale groeisnelheid, d.w.z. dat de tijdscomplexiteit $\mathcal{O}(n^k)$ is voor zekere k, bevredigend werken. Dit dan in tegenstelling tot algoritmen met een exponentiele groeisnelheid, waarvoor de tijdscomplexiteit $\Omega(c^n)$ is voor zekere constante $c > 1$. De praktische toepasbaarheid van deze laatste categorie is beperkt, zoals ook onderstaande tabel toont.

functie	n = 10	n = 100	n = 1000
nlogn	33	664	9966
n^3	1000	10^6	10^9
10^6n^8	10^{14}	10^{22}	10^{30}
$n^{\log n}$	2.099	$1{,}93\text{x}10^{13}$	$7{,}9\text{x}10^{29}$
2^n	1.024	$1{,}27\text{x}10^{30}$	$1{,}1\text{x}10^{300}$
n!	3.628.800	10^{158}	$4\text{x}10^{2567}$

Een groeisnelheid van n^{80} is polynomiaal, maar zal toch niet bevredigend werken. Dit lijkt in tegenspraak met de bewering dat polynomiale algoritmen goed werken. Toch is dit het geval: in de praktijk blijkt polynomiaal bijna steeds $\mathcal{O}(n^3)$ of lager te zijn. Op grond van deze theoretische en praktische overwegingen noemen we polynomiale algoritmen goed en exponentiële slecht. Een functie als $n^{\log n}$ zit er tussen en wordt wel subexponentieel genoemd (zie ook Opgave 2.2).

We zullen de klassen van problemen verder gaan indelen. Voor deze indeling geldt dat we de problemen moeten formuleren als beslissingsproblemen (problemen die met ja of nee worden beantwoord): gegeven een instantie I, is een bepaalde voorwaarde waar voor I?

Hieronder is een aantal problemen geformuleerd als beslissingsprobleem. Onder een graaf wordt een niet-gerichte graaf verstaan; onder een netwerk een gerichte graaf met een lengtefunctie op de pijlen.

SAMENHANG PROBLEEM
Gegeven een graaf G, is deze samenhangend?

ISOMORFIE PROBLEEM
Gegeven twee grafen G_1 en G_2, zijn deze isomorf?

KORTSTE PAD PROBLEEM
Gegeven een network N, twee knooppunten s en t en een natuurlijk getal k, is er een pad van s naar t met lengte $\leq$ k?

MAXIMALE KOPPELING PROBLEEM
Gegeven een graaf G en een natuurlijk getal k, is er een koppeling met k of meer takken?

PLANARITEITSPROBLEEM
Gegeven een graaf G, is deze vlak?

MINIMALE VOORTBRENGENDE BOOM PROBLEEM
Gegeven een graaf G met een lengtefunctie en een natuurlijk getal k, is er een voortbrengende boom met lengte $\leq$ k?

HAMILTON KRING PROBLEEM
Gegeven een graaf G, is er een kring die alle knooppunten bevat?

HANDELSREIZIGERSPROBLEEM
Gegeven een graaf G met een lengtefunctie en een natuurlijk getal k, is er een kring die alle knooppunten bevat met lengte $\leq$ k?

KLIEKENPROBLEEM
Gegeven een graaf G en een natuurlijk getal k, is er een deelgraaf van G met k knooppunten en waarin ieder tweetal knooppunten door een tak verbonden is?

KNOOPPUNTEN BEDEKKINGSPROBLEEM
Gegeven een graaf G en een natuurlijk getal k, is er een verz. W van k knooppunten zódanig dat iedere tak minstens één eindpunt heeft in W?

LINEAIRE PROGRAMMERINGSPROBLEEM
Gegeven een (maximaliserings) LP-probleem en een natuurlijk getal k, is er een toelaatbare oplossing met waarde van de doelfunctie $\geq$ k?

LINEAIRE GEHEELTALLIGE PROGRAMMERINGSPROBLEEM
Gegeven een (maximaliserings) LP-probleem met de extra voorwaarde dat de variabelen geheel moeten zijn en een natuurlijk getal k, is er een toelaatbare oplossing met waarde van de doelfunctie $\geq$ k?

MULTIPROCESSOR SCHEDULING
Gegeven een verz. taken $\{J_1,J_2,\ldots,J_n\}$ met tijdsduren $\{T_1,T_2,\ldots,T_n\}$, een aantal van m machines en een natuurlijk getal (dealine) D, bestaat er een schedule van de taken over de machines zódanig dat de laatste machine niet later dan op tijdstip D klaar is?

Laat $\mathcal{P}$ de klasse van problemen zijn waarvoor een polynomiaal algoritme mogelijkerwijs bestaat. Als een polynomiaal algoritme bekend is, dan behoort het probleem zeker tot de klasse $\mathcal{P}$; echter, indien nog geen polynomiaal algoritme bekend is zou dit misschien wel kunnen bestaan, hoewel het nog niet gevonden is.

Zonder bewijs (we komen op een aantal problemen later nog terug) vermelden we nu dat de volgende problemen tot de klasse $\mathcal{P}$ behoren:
* Samenhang probleem;
* Kortste pad probleem;
* Maximale koppeling probleem;
* Planariteitsprobleem;
* Minimale voortbrengende boom;
* Lineaire programmeringsprobleem.

Vervolgens introduceren we klasse $\mathcal{NP}$. Deze klasse bevat die problemen waarvoor het volgende geldt: Indien I een instantie van het probleem is met een ja-antwoord, dan kan in polynomiale tijd worden gecontroleerd of dit antwoord juist is.

Het handelsreizigersprobleem zit in $\mathcal{NP}$. Immers, als we een graaf hebben met een lengtefunctie en een natuurlijk getal k, dan kan in $\mathcal{O}(n)$ worden gecontroleerd of een gegeven kring door alle knooppunten lengte $\leq$ k heeft.

Het is duidelijk dat $\mathcal{P} \subseteq \mathcal{NP}$ (pas het polynomiale algoritme toe). Een nog onopgelost probleem is de vraag of $\mathcal{P} = \mathcal{NP}$. Omdat het uiterst onwaarschijnlijk is dat voor het handelsreizigersprobleem een polynomiaal algoritme gevonden vermoedt men het volgende:

Vermoeden 1: $\mathcal{P} \neq \mathcal{NP}$.

Problemen uit $\mathcal{NP}$ kunnen worden vergeleken met het begrip polynomiale reduceerbaarheid. We zeggen dat een probleem P polynomiaal gereduceerd kan worden tot probleem Q als iedere instantie van P in polynomiale tijd kan worden omgezet naar een instantie van Q zódanig dat I een ja-instantie is van P dan en slechts dan als de corresponderende instantie een ja-instantie van Q is.

Informeel gezegd betekent dit dat (op polynomialiteit na) P "niet moeilijker" is dan Q. Dit impliceert dat als er een polynomiaal algoritme is voor Q er ook een polynomiaal algoritme is voor P. We kunnen nu op zoek gaan naar de moeilijkste problemen in $\mathcal{NP}$.

Deze klasse van problemen noemen we de $\mathcal{NP}$-complete problemen, afgekort tot $\mathcal{NPC}$. Dit zijn de problemen P die voldoen aan:
a. Het probleem P behoort tot $\mathcal{NP}$.
b. Ieder probleem Q uit $\mathcal{NP}$ is polynomiaal reduceerbaar tot P.

Stelling 2.1
Als $\mathcal{P} \neq \mathcal{NP}$, dan is $\mathcal{P} \cap \mathcal{NPC} = \emptyset$.
Bewijs
Stel $P \in \mathcal{P} \cap \mathcal{NPC}$. Omdat $P \in \mathcal{NPC}$ is iedere $Q \in \mathcal{NP}$ polynomiaal reduceerbaar tot P. Omdat $P \in \mathcal{P}$, is er voor P en dus ook voor iedere $Q \in \mathcal{NP}$ een polynomiaal algoritme: $\mathcal{NP} = \mathcal{P}$: tegenspraak. □

Men veronderstelt (o.a. omdat door vele onderzoekers hier tevergeefs naar is gezocht) dat er voor $\mathcal{NP}$-complete problemen geen polynomiale algoritmen bestaan. Dit betekent dat een algoritme dat (in de worst case) een exponentiëel aantal elementaire stappen vereist voor een dergelijk probleem acceptabel is. In dat geval zijn grote instanties nauwelijks meer exact oplosbaar en worden benaderingesalgoritmen interessant.

Indien men voor een probleem geen polynomiaal algoritme kan vinden, dan kan men zich afvragen of het misschien een $\mathcal{NP}$-compleet probleem is. Dit is voor vele problemen gedaan. Voorbeelden van problemen uit $\mathcal{NPC}$ zijn:

* Hamilton kring probleem;
* Handelsreizigerprobleem;
* Klieken probleem;
* Knooppunten bedekkingsprobleem;
* Lineaire geheeltallige programmeringsprobleem.

Het bewijs dat P tot $\mathcal{NPC}$ behoort gaat dan als volgt:

a. Bewijs dat P tot $\mathcal{NP}$ behoort.
b. Reduceer Q polynomiaal tot P, waarbij voor Q reeds is bewezen dat het tot $\mathcal{NPC}$ hoort.

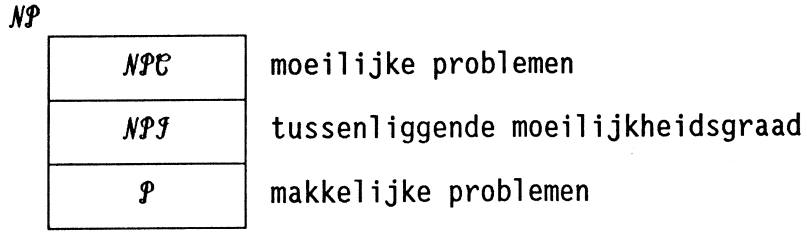
De meest bekende problemen uit de grafentheorie of uit de combinatorische optimalisatie liggen in $\mathcal{P}$ of $\mathcal{NPC}$. De volgende stelling (het bewijs laten we achterwege) laat zien dat het zeer waarschijnlijk is dat er nog andere problemen zijn.

Stelling 2.2

Als $\mathcal{P} \neq \mathcal{NP}$, dan geldt $\mathcal{P} \cup \mathcal{NPC} \neq \mathcal{NP}$.

Een probleem dat wel in $\mathcal{NP}$ ligt en waarvan niet bekend is of het tot $\mathcal{NPC} \cup \mathcal{P}$ behoort is het isomorfie probleem. Laat $\mathcal{NPI} = \mathcal{NP} - (\mathcal{P} \cup \mathcal{NPC})$, dan is (onder de veronderstelling dat $\mathcal{P} \neq \mathcal{NP}$) $\mathcal{NPI} \neq \emptyset$.

Schematisch kunnen de problemen van $\mathcal{NP}$ als volgt worden ingedeeld (onder de aannname dat $\mathcal{P} \neq \mathcal{NP}$):



We kunnen ook spreken over het complement P^C van een probleem P: gegeven een instantie I van P, is een bepaalde voorwaarde niet waar voor I?

Het complement van het handelsreizigersprobleem is:
Gegeven een graaf met een lengtefunctie en een natuurlijk getal k, heeft iedere kring door alle knooppunten een lengte > k?

We definiëren de volgende verzamelingen:

$$\text{co-}\mathcal{P} = \{P^C \mid P \in \mathcal{P}\};$$

$$\text{co-}\mathcal{NP} = \{P^C \mid P \in \mathcal{NP}\};$$

$$\text{co-}\mathcal{NPC} = \{P^C \mid P \in \mathcal{NPC}\}.$$

Stelling 2.3

(i) co-$\mathcal{P}$ = $\mathcal{P}$;

(ii) als $P \in \mathcal{NPC} \cap$ co-$\mathcal{NP}$, dan geldt: $\mathcal{NP}$ = co-$\mathcal{NP}$;

(iii) als co-$\mathcal{NP} \neq \mathcal{NP}$, dan geldt: $\mathcal{P} \neq \mathcal{NP}$;

(iv) als $\mathcal{P} \neq \mathcal{NP}$, dan $\mathcal{P} \neq$ co-$\mathcal{NP}$.

Bewijs

(i) Als een probleem tot $\mathcal{P}$ behoort dan behoort het complement ook tot $\mathcal{P}$. Immers: voor iedere instantie is op polynomiale wijze na te gaan of het een ja-instantie of nee-instantie is, waarmee ook de complementaire vraag is beantwoord. Hieruit volgt: co-$\mathcal{P} \subseteq \mathcal{P}$. Het omgekeerde geldt ook omdat $\mathcal{P}$ = co-(co-$\mathcal{P}$) $\subseteq$ co-$\mathcal{P}$.

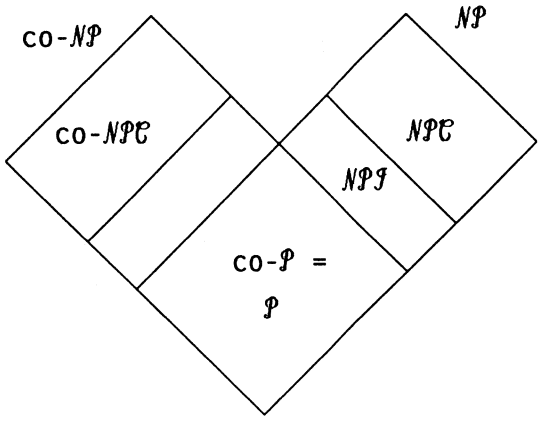
(ii) We zullen eerst bewijzen dat $\mathcal{NP} \subseteq$ co-$\mathcal{NP}$. Stel $Q \in \mathcal{NP}$, dan is te bewijzen dat ook het complement $Q^C \in \mathcal{NP}$, d.w.z. dat een ja-instantie van Q^C polynomiaal controleerbaar is. Omdat een ja-instantie van Q^C een nee-instantie van Q is, moet een nee-instantie van Q polynomiaal controleerbaar zijn.
Laat I een nee-instantie van Q zijn. Omdat $Q \in \mathcal{NP}$ en $P \in \mathcal{NP}$C, is I polynomiaal te reduceren tot een nee-instantie J van P. $P \in$ co-NP, dus een nee-instantie J is polynomiaal controleerbaar : I ook polynomiaal controleerbaar. Omdat $\mathcal{NP} \subseteq$ o-$\mathcal{NP}$ geldt: co-$\mathcal{NP} \subseteq$ co-(co-$\mathcal{NP}$) = $\mathcal{NP}$, dus: $\mathcal{NP}$ = co-$\mathcal{NP}$.

(iii) Stel $\mathcal{P} = \mathcal{NP}$, dan: $\mathcal{P}$ = co-$\mathcal{P}$ = co-$\mathcal{NP} \neq \mathcal{NP}$: tegenspraak.

(iv) Stel $\mathcal{P}$ = co-$\mathcal{NP}$. Dan geldt: $\mathcal{P}$ = co-$\mathcal{P}$ = co-(co-$\mathcal{NP}$) = $\mathcal{NP}$: tegenspraak. □

Gevolgen

1. Uit onderdeel (ii) volgt :als co-$\mathcal{NP} \neq \mathcal{NP}$, dan geldt: $\mathcal{NPC} \cap$ co-$\mathcal{NP} = \emptyset$, d.w.z. $\mathcal{NPC} \subseteq (\mathcal{NP} \backslash$co-$\mathcal{NP})$. Analoog kan worden aangetoond dat uit co-$\mathcal{NP} \neq \mathcal{NP}$ volgt dat co-$\mathcal{NPC} \subseteq$ (co-$\mathcal{NP} \backslash \mathcal{NP}$).
2. Onder de aanname dat co-$\mathcal{NP} \neq \mathcal{NP}$ hebben de verschillende verzamelingen de hieronder schematisch weergegeven relaties:



Uit bovenstaande tekening zou kunnen worden geconcludeerd dat $\mathcal{P} = \mathcal{NP} \cap$ co-$\mathcal{NP}$. Of dit geldt is echter een open vraag.

Opgave 2.1

Laten de functies f en g gedefiniëerd zijn door:

$$f(n) = \begin{cases} n^2 & \text{als n oneven is} \\ n^3 & \text{anders} \end{cases} \qquad g(n) = \begin{cases} n^2 & \text{als n een priemgetal is} \\ n^3 & \text{anders} \end{cases}$$

Ga na welke van de volgende uitspraken waar zijn:

a. $f(n) = \Omega(n^2)$.

b. $f(n) = \mathcal{O}(g(n))$.

c. $f(n) = \Theta(g(n))$.

Opgave 2.2

Beschouw de volgende recurrente methode om elementen te sorteren (we nemen aan dat het aantal elementen een macht van twee is).

Veronderstel dat de linker en rechter helft gesorteerd zijn: zeg $[x_1,x_2,\ldots,x_N]$ en $[x_{N+1},x_{N+2},\ldots,x_{2N}]$. Vergelijk de eerste elementen van beide helften, en plaats het kleinste, zeg x_1,voorop in de nieuwe rij; daarna worden x_2 en x_{N+1} vergeleken en wordt de kleinste op plaats 2 gezet, etc. Zo wordt een gesorteerde rij gemaakt in 2N stappen.

a. Toon aan dat als er n elementen aldus gesorteerd worden de complexiteit van deze methode $\mathcal{O}(n\log n)$ is.

b.* Toon aan ieder algoritme voor het sorteerprobleem een complexiteit $\Omega(n\log n)$ heeft.

Opgave 2.3

Toon aan dat $n^{\log n} = \mathcal{O}(2^n)$ en $n^{\log n} \neq \Theta(2^n)$.

Opgave 2.4

Toon aan dat het Hamiltonkring probleem polynomiaal reduceerbaar is tot het handelsreizigersprobleem.

Opgave 2.5

Zij G = (V,E) een graaf.

$W \subseteq V$ heet een onafhankelijke verz. als geen enkel tweetal knooppunten van W door een tak van E verbonden is.

a. Toon aan dat de volgende beweringen equivalent zijn:

(i) W is een knooppuntenbedekking;

(ii) V - W is een onafhankelijke verz.;

(iii) V - W is een kliek in de complementaire graaf $\bar{G}$.

b. Het *ONAFHANKELIJKSPROBLEEM* luidt:

Gegeven een graaf G = (V,E) en een natuurlijk getal k, is er een $W \subseteq V$ met #W = k zódanig dat W een onafhankelijke verzameling is?

Toon aan dat het knooppunten bedekkingsprobleem, het klieken probleem en het onafhankelijkheidsprobleem via polynomiale reducties in elkaar over te voeren zijn.

2. Voorwaarts zoeken

Literatuur:

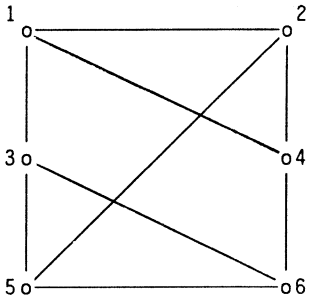
* A.V.Aho, J.E.Hopcroft and J.D.Ullman: "Data structures and algorithms", Addison-Wesley, Reading (1983).
* S.Baase: "Computer algorithms: introduction to design and analysis", Addison-Wesley, Reading (1988).
* R.Sedgewick: "Algorithms", Addison-Wesley, Reading (1983).

In deze paragraaf beschijven we een systematische methode om "door een graaf te lopen". We houden bij welke knooppunten (en in welke volgorde) we tegenkomen en hoe de verbindingen lopen. Hiermee kunnen we o.a. vragen over de samenhang beantwoorden. De techniek die we gebruiken is die van het voorwaarts zoeken. We zullen dit allereerst voor een niet-gerichte graaf G = (V,E) bespreken.

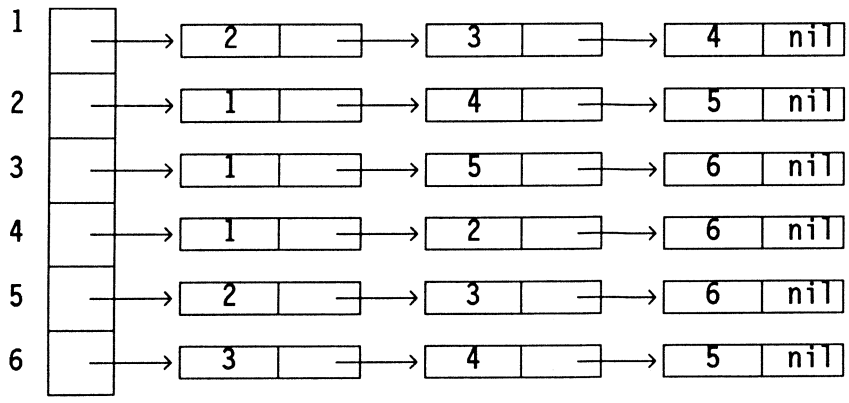
We veronderstellen dat G wordt gerepresenteerd door structuurlijsten. De structuurlijst $L[v_i]$ van een knooppunt v_i is een lijst waarin de aan v_i grenzende knooppunten worden opgeborgen. We kunnen dan G representeren door een pointer array *Header*[1:n] met *Header*[i] een pointer naar het begin van de structuurlijst van v_i. De elementen in de structuurlijst hebben twee velden: in het eerste staat het nummer van het volgende aangrenzende knooppunt, het tweede is een pointer, die verwijst naar het volgende element. Bij deze representatie wordt iedere tak bij beide eindpunten. De inputgrootte die hierbij hoort is $\Theta(p)$, waarbij p = max(n,m).

Voorbeeld 2.1

Beschouw de hiernaast getekende graaf.
De representitie m.b.v. structuurlijsten is hieronder gegeven.



Header



De procedure laat zich als volgt beschrijven:
Start in een willekeurig knooppunt, zeg v, en begin het zoeken vanuit v. Dit knooppunt v noemen we de wortel, en we zeggen dat v bezocht is. Het bezocht zijn houden we bij door de knooppunten in de volgorde waarin ze bezocht worden te nummeren (niet-bezochte knooppunten hebben nummer -1). Vervolgens kiezen we het eerste knooppunt van L[v], zeg w, d.w.z. we bekijken de tak (v&w). We noteren dat v de vader is van w (v = PRED[w]). We gaan nu verder in w.
Indien in een knooppunt alle takken onderzocht zijn gaan we terug naar de vader totdat we helemaal terug zijn bij de wortel en daar alle takken hebben bekeken. De component waar v toe behoort is dan geheel onderzocht. Indien er nog niet bezochte knooppunten zijn, dan gaan we in een volgende component verder.

In onderstaande procedure *DFS* (depth-first-search) gebruiken we de volgende parameters:

i : het nummer van het eerstvolgende bezoek.
k : het nummer van de huidige component.
PRED[1:n] : integer array met PRED[i] = vader van knooppunt i (een wortel heeft PRED-waarde 0; als een knooppunt nog niet is bezocht ,is PRED = -1).
NUMBER[1:n]: integer array met NUMBER[i] = het bezoeknummer van knooppunt i (nog niet bezochte knooppunten hebben NUMBER = -1).

Zij de procedure *VISIT*(v) één zoekstap vanuit v, dan werkt deze als volgt:
als een knooppunt w van L[v] nog niet eerder bezocht is:
    ga verder met *VISIT*(w);
als alle knooppunten van L[v] bezocht zijn:
    ga naar *PRED*[v].

Onderstaand programma past bovenstaande procedure recursief toe. We schrijven dit programma in een pseudo-Pascal taal. Dit kan vrij eenvoudig in een echt Pascal programma worden omgezet. We kiezen voor de pseudo versie omdat deze het meeste inzicht geeft in wat er gebeurt.

```
procedure DFS;
  var i,j,k: integer;
      PRED,NUMBER: array [1:n] of integer;
  procedure VISIT(v: integer);
    var w: integer;
    begin NUMBER[v] := i; i := i+1;
          for iedere w van L[v] do
              if NUMBER[w] < 0 then
                begin PRED[w] := v; VISIT(w) end
    end VISIT;

begin i := 1; k := 0;
      for j := 1 to n do
          begin PRED[j] := -1; NUMBER[j] := -1 end;
      for j := 1 to n do
          if NUMBER[j] < 0 then
             begin k := k+1; PRED[j] := 0; VISIT(j) end
end DFS.
```

Op de volgende pagina staat een voorbeeld.

Stelling 2.4

Het algoritme van het voorwaarts zoeken heeft een complexiteit $\mathcal{O}(p)$, waarbij $p = \max(n,m)$.

Bewijs

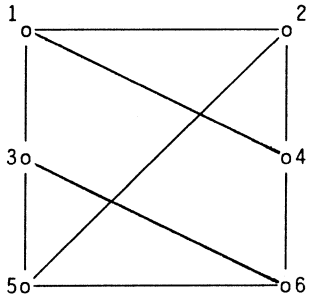
De initialisatie vereist $\mathcal{O}(n)$ stappen.

Merk op dat de aanroep van *VISIT*(v) $\mathcal{O}(\delta(v))$ stappen kost (de binnen de recurrente procedure aangeroepen *VISIT*(w) tellen we apart bij knooppunt w; zie ook het voorbeeld op de volgende pagina).

Omdat $\Sigma_{v \in V}\ \delta(v) = 2m$ leveren alle *VISIT* aanroepen tezamen $\mathcal{O}(m)$ elementaire stappen. De totale complexiteit is dus: $\mathcal{O}(n) + \mathcal{O}(m) = \mathcal{O}(p)$. □

Voorbeeld 2.1 (vervolg; figuur staat nogmaals rechtsonder)

```
Initialisatie: i = 1; k = 0; PRED[j] = NUMBER[j] = -1 voor j = 1,2,...,6.
j = 1:k = 1; PRED[1] = 0;
      VISIT(1):
        v = 1:NUMBER[1] = 1; i = 2;
              w = 2: PRED[2] = 1;
              VISIT(2):
                v = 2:NUMBER[2] = 2; i = 3;
                      w = 1:
                      w = 4: PRED[4] = 2;
                      VISIT(4):
                        v = 4:NUMBER[4] = 3; i = 4;
                              w = 1:
                              w = 2:
                              w = 6: PRED[6] = 4;
                              VISIT(6):
                                v = 6:NUMBER[6] = 4; i = 5;
                                      w = 3: PRED[3] = 6;
                                      VISIT(3):
                                         v = 3:NUMBER[3] = 5; i = 6;
                                               w = 1:
                                               w = 5: PRED[5] = 3;
                                               VISIT(5):
                                                 v = 5:NUMBER[5] = 6; i = 7;
                                                       w = 2:
                                                       w = 3:
                                                       w = 6:
                                               w = 6:
                                      w = 4:
                                      w = 5:
                      w = 5:
              w = 3:
              w = 4:
j = 2:
j = 3:
j = 4:
j = 5:
j = 6:
```



Maak van G een gerichte graaf door tijdens de procedure *DFS* (als $w \in L[v]$ wordt bekeken) de tak (v&w) van v naar w te richten; stop (v,w) in een verz. F ("forward") als NUMBER[w] < 0 en in een verz. B ("backward") als NUMBER[w] > 0

Stelling 2.5

(i) De pijlen van F vormen een voortbrengend gericht bos met als wortels de knoopppunten v met PRED[v] = 0.

(ii) De met F geassocieerde kringen zijn ronden, d.w.z. de enige pijl van B loopt "de goede kant" op.

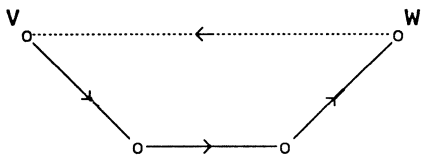
(iii) Iedere enkelvoudige ronde in de gerichte graaf is een met F geassocieerde kring.

Bewijs

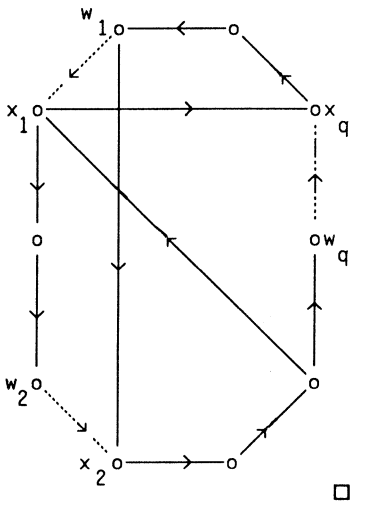
Voor alle onderdelen van het bewijs mogen we wel aannemen dat de graaf samenhangend is (anders passen we het op iedere component toe).

(i) Behalve de wortel wordt ieder knooppunt één keer bekeken terwijl zijn NUMBER-waarde negatief is : ieder knooppunt heeft precies één binnenkomende pijl van F. Door vanuit een knooppunt terug te lopen kunnen we de wortel bereiken: de pijlen van F vormen een voortbrengende gerichte boom

(ii) Veronderstel dat de tak tussen v en w niet tot F behoort en dat NUMBER[w] > NUMBER[v]. Dan was toen tijdens *VISIT*(w) v werd bekeken NUMBER[v] > 0 en dus $(w,v) \in B$, en w eerder afgehandeld dan v. Uit het algoritme volgt dat vanuit een nog niet afgehandeld knooppunt alle knooppunten met een hoger NUMBER via pijlen van F bereikbaar zijn: pad van v naar w met pijlen van F. Dit geeft tezamen met (w,v) de ronde.



(iii) Stel C is een ronde met $q \geq 2$ pijlen van B. Stel deze pijlen (w_i,x_i), $1 \leq i \leq q$. Voor iedere pijl (w_i,x_i) is er volgens (ii) een pad van x_i naar w_i met louter pijlen van F. Omdat ieder knooppunt hoogstens één binnenkomende pijl van F heeft, komt het pad van x_i naar w_i bij w_i aan met de pijlen van C vanaf x_{i-1}: er is een pad van x_i naar x_{i-1} met louter pijlen van F, $1 \leq i \leq q$ ($x_0 = x_q$).
Dit geeft echter de volgende ronde binnen F:
$x_1 \to x_q \to x_{q-1} \to \cdots \to x_2 \to x_1$: tegenspraak. □



Vervolgens bespreken we het voorwaarts zoeken in een gerichte graaf. De lijst L[v] bevat in dit geval alleen de knooppunten w zódat er een pijl is van v naar w. De procedure is verder precies hetzelfde.

Het indelen van de pijlen (als $w \in L[v]$ wordt bekeken) gaat als volgt:
als NUMBER[w] < 0: (v,w) in F;
als NUMBER[w] > 0: als NUMBER[w] > NUMBER[v]: $(v,w) \in I$;
als NUMBER[w] $\leq$ NUMBER[v]:
als we nog in *VISIT*(w) zitten : $(v,w) \in D$;
als we niet meer in *VISIT*(w) zitten: $(v,w) \in C$.
De letter F staat voor "forward", I voor "increasing", D voor "decreasing" en C voor "crossing".

Stelling 2.6

(i) De complexiteit van het voorwaarts zoeken in een gerichte graaf is $\mathcal{O}(p)$.
(ii) De pijlen van F vormen een voortbrengend gericht bos T met als wortels de knooppunten v met PRED[v] = 0.

Bewijs

Geheel analoog aan het geval van een niet-gerichte graaf. □

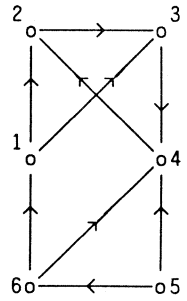
Lemma 2.1

(i) De pijl $(v,w) \in I$ d.e.s.d. als er in T pad van v naar w is.
(ii) De pijl $(v,w) \in D$ d.e.s.d. als er in T pad van w naar v is.
(iii) De pijl $(v,w) \in C$ d.e.s.d. als er in T geen pad van v naar w is en ook geen pad van w naar v.

Bewijs

(i) Als $(v,w) \in I$, dan is v vóór w bezocht, *VISIT*(w) is afgehandeld en we zitten op het moment van de toekenning in *VISIT*(v): in T kunnen we van v naar w gaan. Het omgekeerde gaat analoog.
(ii) Als $(v,w) \in D$, dan is w vóór v bezocht en we zitten nog in *VISIT*(w): in T kunnen van w naar v. Het omgekeerde gaat analoog.
(iii) Als $(v,w) \in C$, dan is w vóór v bezocht en *VISIT*(w) is al afgehandeld: v wordt in T bereikt vanuit een andere wortel dan w, d.w.z. in beide richtingen is er geen pad. Het omgekeerde gaat analoog. □

Voorbeeld 2.2



Beschouw nevenstaande graaf. De structuurlijsten zijn:
L[1] = {2,3}; L[2] = {3}; L[3] = {4}; L[4] = {2}; L[5] = {4,6};
L[6] = {1,4}.
Initialisatie: i = 1; k = 0; PRED[j] = NUMBER[j] = -1, $1 \leq j \leq 6$.

```
j = 1: k = 1; PRED[1] = 0;
       VISIT(1):
         v = 1: NUMBER[1] = 1; i = 2;
                w = 2: PRED[2] = 1; (1,2) ∈ F;
                VISIT(2):
                v = 2: NUMBER[2] = 2; i = 3;
                       w = 3: PRED[3] = 2; (2,3) ∈ F;
                       VISIT(3):
                         v = 3: NUMBER[3] = 3; i = 4;
                                w = 4: PRED[4] = 3; (3,4) ∈ F;
                                VISIT(4):
                                  v = 4: NUMBER[4] = 4; i = 5;
                                         w = 2: (4,2) ∈ D;
                w = 3: (1,3) ∈ I;
j = 2:
j = 3:
j = 4:
j = 5: k = 2; PRED[5] = 0;
       VISIT(5):
         v = 5: NUMBER[5] = 5; i = 6;
                w = 4: (5,4) ∈ C;
                w = 6: PRED[6] = 5; (5,6) ∈ F;
                VISIT(6):
                  v = 6: NUMBER[6] = 6; i = 7;
                         w = 1: (6,1) ∈ C;
                         w = 4: (6,4) ∈ C;
j = 6:
```

Stelling 2.7

Laten de pijlen van F het voortbrengende gerichte bos T in G = (V,A) vormen. Zij $G_1 = (V_1,A_1)$ een streng samenhangende component van G. Dan is de deelgraaf van T bestaande uit de knooppunten V_1 en de daartussen lopende pijlen van F samenhangend.

Bewijs

Kies twee knooppunten $v_1,v_2 \in V_1$. Zonder beperking der algemeenheid mogen we aannemen dat NUMBER[v_1] < NUMBER[v_2]. Omdat G_1 streng samenhangend is, is er in G_1 een pad P van v_1 naar v_2. Laat z_1 het knooppunt op dit pad zijn met het laagste NUMBER, en laat T_1 de deelboom zijn van T met wortel z_1.

Beschouw $(v,w) \in A$ met $v \in T_1$ en $w \notin T_1$. Omdat $(v,w) \notin T_1$ is knooppunt w eerder bezocht dan alle knooppunten van T_1: NUMBER[w] < NUMBER[z_1].

Omdat voor alle knooppunten v op het deelpad $[z_1,\ldots,v_2]$ van P geldt: NUMBER[v] ≥ NUMBER[z_1] en $z_1 \in T_1$ behoren alle opvolgende knooppunten tot en met v_2 dus ook tot T_1.

Omdat NUMBER[z_1] ≤ NUMBER[v_1] < NUMBER[v_2], z_1 en v_2 tot T_1 behoren geldt dit ook voor de tussenliggende waarden van NUMBER, dus ook $v_1 \in T_1$. In T_1 zijn er dus paden van de wortel z_1 naar v_1 en naar v_2.

Kies vervolgens een $v_3 \in V_1$ en $v_3 \neq v_1,v_2,z_1$. Herhaal het bovenstaande met voor v_1 nu z_1 en voor v_2 de zojuist gekozen v_3. Dit geeft een z_2 zódanig dat er in T paden zijn van z_2 naar z_1 en v_3, dus ook van z_2 naar v_1,v_2 en v_3.

Op deze wijze voortgaande is in te zien dat er in V_1 een z is zódat er binnen T paden zijn van z naar alle $v \in V_1$. Laat T^* deze deelboom van T zijn, met dus als eindpunten knooppunten van V_1.

Het is nu voldoende om te laten zien dat deze deelgraaf geen andere knooppunten dan die van V_1 bevat.

Neem een willekeurige v uit deze deelgraaf. Er is een pad van z naar v. Er is echter ook een pad terug van v naar z: Loop in de deelboom naar een eindpunt w; $w \in V_1$ en omdat V_1 streng samenhangend is, is er een pad van w naar z.

v zit dus in dezelfde streng samenhangende component als z, d.w.z. in V_1. □

Gevolg

Als G zelf streng samenhangend is, dan vormen de pijlen van F een gerichte voortbrengende boom.

Opgave 2.6

Pas het voorwaarts zoeken toe (inclusief de toekenning van de pijlen van F en B) op de niet-gerichte graaf met 5 knooppunten, gerepresenteerd door de volgende lijsten:

L[1] = {2,3,5}; L[2] = {1,3,4}; L[3] = {1,2,3,4,5}; L[4] = {2,3}; L[5] = {1,3}

Opgave 2.7

Zij G een niet-gerichte samenhangende graaf en T de voortbrengende gerichte boom bestaande uit de pijlen van F.

Laat G_1 een deelgraaf van G zijn die volledig is.

a. Toon aan dat er een pad in T is waar alle knooppunten van G_1 op liggen.

b. Ga na of er op dit pad ook andere knooppunten kunnen liggen.

Opgave 2.8

Zij G een niet-gerichte graaf en T het voortbrengend gerichte bos bestaande uit de pijlen van F. Zij T[v] het aantal knooppunten dat in T vanuit v kan worden bereikt, inclusief v zelf.

a. Toon aan dat w in T vanuit v te bereiken is d.e.s.d als
   NUMBER[v] $\leq$ NUMBER[w] < NUMBER[v] + T[v].

b. Pas de procedure *DFS* aan zódat daarmee ook de getallen T[v] worden berekend.

Opgave 2.9

Stel een $\mathcal{O}(p)$ algoritme op, waarbij p = max(n,m), om na te gaan of een niet-gerichte graaf bipartiet is.

Opgave 2.10

Stel een algoritme op, gebaseerd op *DFS*, om een kringbasis te bepalen.
Bepaal de complexiteit van het voorgestelde algoritme.

Opgave 2.11

Pas het voorwaarts zoeken, inclusief de bepaling van de pijlenverz. F, I, D en C, toe op de graaf met de volgende structuurlijsten:

L[1] = {2}; L[2] = {3,4}; L[3] = {1}; L[4] = {3,6}; L[5] = {2,6,9}; L[6] = $\varnothing$;
L[7] = {6,8}; L[8] = {4}; L[9] = {7,8}.

Opgave 2.12

Zij G een gerichte graaf, C een ronde in G en T het voortbrengende gerichte bos met pijlen van F. Laat v het knooppunt op C zijn met het kleinste NUMBER. Toon aan dat T_v, de deelboom van T met wortel v, alle knooppunten van C bevat.

Opgave 2.13

Beschouw een gerichte graaf G = (V,A).
Stel een algoritme met complexiteit $\mathcal{O}(p)$, waarbij p = max(n,m), op om de knooppunten te nummeren zódat als $(v,w) \in A$, dan is NUMBER[v] < NUMBER[w].
Als een dergelijke nummering niet bestaat, toon aan dat G dan een ronde bevat en geef deze situatie aan door de Boolean "cycle" de waarde true toe te kennen.

3. Zijwaarts zoeken

Literatuur:

* A.V.Aho, J.E.Hopcroft and J.D.Ullman: "Data structures and algorithms", Addison-Wesley, Reading (1983).
* S.Baase: "Computer algorithms: introduction to design and analysis", Addison-Wesley, Reading (1988).
* R.Sedgewick: "Algorithms", Addison-Wesley, Reading (1983).

Deze methode verschilt van het voorwaarts zoeken doordat - als een knooppunt wordt bezocht - eerst alle met dat knooppunt incidente takken worden bekeken voordat we verder gaan. De nog niet eerder bezochte eindpunten van deze takken worden in een queue geplaatst. Als nieuw te onderzoeken knooppunt nemen we het eerste uit de queue. Als v de wortel is, dan worden aldus achtereenvolgens bekeken: de knooppunten die via één tak (pijl) vanuit v bereikbaar zijn, daarna de knooppunten die via twee takken (pijlen) vanuit v bereikbaar zijn , etc.

Een queue is een lijst waar aan één kant (de staart) elementen worden toegevoegd en aan de andere kant (de kop) elementen worden verwijderd. We gebruiken de volgende operaties voor een queue Q:

MAKENULL(Q) : maakt van Q een lege lijst;
EMPTY(Q) : Boolean die waarde true heeft d.e.s.d. als Q leeg is.
FRONT(Q) : deze geeft het eerste element van Q;
ENQUEUE(v,Q): voegt v aan de staart van Q toe;
DEQUEUE(Q) : verwijdert het eerste element van Q;

In onderstaande procedure *BFS* (breadth-first-search) gebruiken we dezelfde parameters als bij *DFS*. Tevens zullen we weer een aantal pijlen in F stoppen, waarbij de pijlen van F aan het einde van de procedure een voortbrengend gericht bos vormen.

```
procedure BFS;
  var i,j,k: integer;
      PRED,NUMBER: array[1:n] of integer;
      Q: queue of integer;
      F: queue of pair of integers;
  procedure VISIT(v: integer);
    var x,y: integer;
    begin ENQUEUE(v,Q);
          while not EMPTY(Q) do
                begin x := FRONT(Q); DEQUEUE(Q);
                      for iedere y van L[x] do
                          if NUMBER[y] < 0 then
                             begin NUMBER[y] := i; i := i+1; ENQUEUE((x,y),F);
                                   PRED[y] := x; ENQUEUE(y,Q)
                             end
                end
    end VISIT;
begin i := 1; k := 0; MAKENULL(F);
      for j := 1 to n do
          begin PRED[j] := -1; NUMBER[j] := -1 end;
      for j := 1 to n do
          if NUMBER[j] < 0 then
             begin k := k+1; PRED[j] := 0; NUMBER[j] := i; i := i+1;
                   MAKENULL(Q); VISIT(j)
             end
end BFS.
```

Stelling 2.8

De procedure van het zijwaarts zoeken heeft de volgende eigenschappen:

(i) De complexiteit is $\mathcal{O}(p)$, waarbij $p = \max(n,m)$;

(ii) De pijlen van F vormen een voortbrengend gericht bos T met als wortels de knooppunten v met PRED[v] = 0.

(iii) De lengte in T van een wortel v naar w = het minimum aantal takken (pijlen) van v naar w.

Bewijs

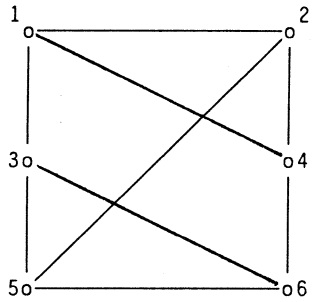
(i) De initialisatie is van $\mathcal{O}(n)$. Iedere tak wordt twee keer bekeken en per keer is het werk $\mathcal{O}(1)$: in totaal $\mathcal{O}(m)$. Bij elkaar dus $\mathcal{O}(p)$.

(ii) Analoog aan onderdeel (i) van Stelling 2.5.

(iii) Zie de opmerking aan het begin van deze paragraaf. □

Voorbeeld 2.1 (vervolg)

Initialisatie: i = 1; k = 0; F = ∅; PRED[j] = NUMBER[j] = -1, 1≤ j ≤ 6.
j = 1:k = 1; PRED[1] = 0; NUMBER[1] = 1; i = 2; Q = ∅;
VISIT(1):
Q = {1};
x = 1: Q = ∅;
y = 2: NUMBER[2] = 2; i = 3; F = {(1,2)};
PRED[2] = 1; Q = {2};
y = 3: NUMBER[3] = 3; i = 4; F = {(1,2),(1,3)};
PRED[3] = 1; Q = {2,3};
y = 4: NUMBER[4] = 4; i = 5; F = {(1,2),(1,3),(1,4)};
PRED[4] = 1; Q = {2,3,4};
x = 2: Q = {3,4};
y = 1:
y = 4:
y = 5: NUMBER[5] = 5; i = 6; F = {(1,2),(1,3),(1,4),(2,5)};
PRED[5] = 2; Q = {3,4,5};
x = 3: Q = {4,5};
y = 1:
y = 5:
y = 6: NUMBER[6] = 6; i = 7; F = {(1,2),(1,3),(1,4),(2,5),(3,6)};
PRED[6] = 3; Q = {4,5,6};
x = 4: Q = {5,6};
y = 1:
y = 2:
y = 6:
x = 5: Q = {6};
y = 2:
y = 3:
y = 6:
x = 6: Q = ∅;
y = 3:
y = 4:
y = 5:
j = 2:
j = 3:
j = 4:
j = 5:
j = 6:



Opgave 2.14

Pas het zijwaarts zoeken toe (inclusief de toekenning van de pijlen van F) op de niet-gerichte graaf met 5 knooppunten, gerepresenteerd door de volgende lijsten:

L[1] = {2,3,5}; L[2] = {1,3,4}; L[3] = {1,2,3,4,5}; L[4] = {2,3}; L[5] = {1,3}

Opgave 2.15

a. Geef een voorbeeld, waarmee met voorwaarts zoeken eerder een kring kan worden gevonden dan met zijwaarts zoeken.
b. Geef een voorbeeld, waarmee met zijwaarts zoeken eerder een kring kan worden gevonden dan met voorwaarts zoeken.

Opgave 2.16

Zij G = (V,E) een niet-gerichte samenhangende graaf. Kies een knooppunt v.

a. Stel een $O(m)$ algoritme op om voor ieder knooppunt w het minimum aantal takken tussen v en w te bepalen.
b. Pas dit algoritme toe (met v = knooppunt 1) op de graaf met structuurlijsten:

   L[1] = {2,3,4}; L[2] = {1,4,6}; L[3] = {1,4,5}; L[4] = {1,2,3,5}
   L[5] = {3,4} ; L[6] = {2,7} ; L[7] = {6}.

4. Streng samenhangende componenten

Literatuur:

* S.Baase: "Computer algorithms: introduction to design and analysis", Addison-Wesley, Reading (1988).
* K.Mehlhorn: "Data structures and algorithms 2: Graph algorithms and NP-completeness", Springer, Berlin (1984).
* R.Sedgewick: "Algorithms", Addison-Wesley, Reading (1983).

In deze paragraaf gaan we uit van een gerichte graaf G = (V,A) en willen we de streng samenhangende componenten (afgekort: s.s.c.) van G bepalen. We gebruiken daartoe een variant van *DFS*. *DFS* verdeelt de graaf in componenten, die i.h.a. niet streng samenhangend zijn. De verdeling van de knooppunten over de componenten is afhankelijk van het gekozen startpunt (de wortel) van de zoekmethode.

Wel is het zo (Stelling 2.7) dat de knooppunten van een s.s.c. in dezelfde deelgraaf terecht komen. De met *DFS* geconstrueerde componenten bevatten elk een aantal s.s.c.'s, we weten echter nog niet hoe deze indeling er uitziet.
Zo bevat in voorbeeld 2.2 de eerste component de knooppunten 1 t/m 4, terwijl {1} en {2,3,4} de s.s.c. zijn; de tweede component bestaat uit de knooppunten 5 en 6 die ieder een s.s.c. zijn.

Zij $G_1 = (V_1,A_1)$ een s.s.c., dan is in het met *DFS* geconstrueerde voortbrengende bos T = (V,F), de bij G_1 behorende deelgraaf een voortbrengende gerichte deelboom $T_1 = (V_1,F_1)$ van T.
Bovendien geldt voor de wortel z van T_1: NUMBER[z] = min{NUMBER[w] | w ∈ V_1}.

We zullen nu eerst een iets gewijzigde versie van *DFS* geven. Veronderstel dat we voor de knooppunten bijhouden in welke volgorde ze afgehandeld worden: READY[v] = k d.e.s.d als v als k-de knooppunt wordt afgehandeld. Veronderstel bovendien dat we voor iedere pijl bijhouden in welk van de verz. F, I, D of C deze wordt geplaatst.

Het algoritme ziet er dan als volgt uit.

```
procedure DFS;
  var i,j,k,l: integer;
      PRED,NUMBER,READY: array [1:n] of integer;
      F,I,D,C: queue of pair of integers;
  procedure VISIT(v: integer);
    var w: integer;
    begin NUMBER[v] := i; i := i+1;
          for iedere w van L[v] do
              if NUMBER[w] < 0 then
                 begin PRED[w] := v; ENQUEUE((v,w),F); VISIT(w);
                       l := l+1; READY[w] := l
                 end
              else
                 begin if   NUMBER[w] > NUMBER[v] then ENQUEUE((v,w),I)
                       else begin if   READY[w] < 0 then ENQUEUE((v,w),D)
                                  else ENQUEUE((v,w),C)
                            end
                 end
    end VISIT;

begin i := 1; k := 0; l := 0;
      MAKENULL(F); MAKENULL(I); MAKENULL(D); MAKENULL(C);
      for j := 1 to n do
          begin PRED[j] := -1; NUMBER[j] := -1; READY[j] := -1 end;
      for j := 1 to n do
          if NUMBER[j] < 0 then
             begin k := k+1; PRED[j] := 0; VISIT(j); l := l+1; READY[j] := l
             end
end DFS.
```

Om de s.s.c. te kunnen ontdekken introduceren we de volgende twee begrippen:

CURRENT(v) = {w ∈ V | READY[z] ≥ READY[v] met z de wortel van de s.s.c. van w}

LOW[v] = min {NUMBER[v], min{NUMBER[w] | (v,w) ∈ D ∪ C èn w ∈ CURRENT(v)},
min{LOW[w] | (v,w) ∈ F}}.

Bij iedere s.s.c. is er een deelboom van T die de knooppunten van die s.s.c. bevat. CURRENT(v) bevat die deelbomen van s.s.c.'en waarvan de wortel niet eerder is afgehandeld dan v zelf.

In voorbeeld 2.2 zijn {1}, {2,3,4}, {5} en {6} de s.s.c.
De getallen READY worden: READY[1] = 4; READY[2] = 3; READY[3] = 2;
READY[4] = 1; READY[5] = 6; READY[6] = 5.
Voor de verz. CURRENT geldt:
CURRENT(1) = {1,5,6}; CURRENT(2) = {1,2,3,4,5,6}; CURRENT(3) = {1,2,3,4,5,6};
CURRENT(4) = {1,2,3,4,5,6}; CURRENT(5) = {5}; CURRENT(6} = {5,6}.
De getallen LOW kunnen worden berekend in de volgorde waarin de knooppunten worden afgehandeld, d.w.z. in de volgorde van de getallen READY:
LOW[4] = min{4,NUMBER[2]} = 2;LOW[3] = min{3,LOW[4]} = 2;
LOW[2] = min{2,LOW[3]} = 2; LOW[1] = min{1,LOW[2]} = 1;
LOW[6] = min{6} = 6; LOW[5] = min{5,LOW[6]} = 5.

De waarde die LOW krijgt levert een karakterisering op van de wortels van de voortbrengende gerichte bomen die behoren bij de s.s.c.'s. De karakterisering wordt gegeven door de volgende stellingen.

Stelling 2.9
Voor ieder knooppunt v geldt: LOW[v] ≥ NUMBER[r(v)], waarbij r(v) de wortel is van de deelboom van T die behoort bij de s.s.c. van v.
Bewijs
We beschouwen de volgende drie mogelijkheden voor LOW[v]:
a. LOW[v] = NUMBER[v]: NUMBER[r(v)] ≤ NUMBER[v] = LOW[v].
b. LOW[v] = NUMBER[w] < NUMBER[v], (v,w) ∈ D ∪ C en w ∈ CURRENT(v):
Met r(w) de wortel van de boom van de s.s.c. van w kunnen we schrijven:
NUMBER[r(w)] ≤ NUMBER[w] < NUMBER[v] en READY[r(w)] ≥ READY[v]:
v is eerder afgehandeld dan r(w), maar heeft een hoger NUMBER: v is vanuit r(w) bereikbaar in T. Omgekeerd is (in G) r(w) bereikbaar vanuit v, nl. (eerst (v,w) en dan kunnen we van w naar r(w), want w en r(w) liggen in dezelfde s.s.c.): r(w), v en w in dezelfde s.s.c.: r(v) = r(w).
Nu geldt: NUMBER[r(v)] = NUMBER[r(w)] ≤ NUMBER[w] = LOW[v].

c. LOW[v] = LOW[w] < NUMBER[v] met (v,w) ∈ F:

We passen inductie toe naar READY[v].

Als READY[v] = 1, dan zijn er in v geen uitgaande pijlen van F. Dan is LOW[v] = min {NUMBER[v], min{NUMBER[w] | (v,w) ∈ D ∪ C èn w ∈ CURRENT(v)}

Dus geldt de Stelling volgens de reeds bewezen onderdelen a en b.

Omdat w maar één binnenkomende pijl van F heeft geldt: òf r(w) = w òf r(w) = r(v). In beide gevallen geeft dit NUMBER[r(v)] ≤ NUMBER[r(w)].

READY[w] < READY[v] en volgens de inductieveronderstelling geldt: NUMBER[r(w)] ≤ LOW[w], waaruit volgt: NUMBER[r(v)] ≤ LOW[w] = LOW[v]. □

Stelling 2.10

Als LOW[z] = NUMBER[z] dan is z is de wortel van een voortbrengende gerichte boom van een s.s.c.

Bewijs

Zij $G_1 = (V_1,A_1)$ de s.s.c. waar z toe behoort, en laat $T_1 = (V_1,F_1)$ de bijbehorende gerichte boom met pijlen van F zijn.

Stel z is niet de wortel van T_1, maar r. Dan is er in G_1 een pad P van z naar r, zeg $P = [z = v_0,v_1,\ldots,v_k = r]$, terwijl we via T_1 van r naar z kunnen lopen.

Er is dus een $1 \leq i \leq k$ zódat $[z = v_0,\ldots,v_{i-1}] \in T_1$ en $(v_{i-1},v_i) \in D \cup C$ (zou $(v_{i-1},v_i) \in I$, dan is er in T_1 een pad van v_{i-1} naar v_i en nemen we dit pad).

Dit houdt in dat $LOW[z] = LOW[v_0] \leq LOW[v_1] \leq \cdots \leq LOW[v_{i-1}]$, en dat $NUMBER[v_i] < NUMBER[v_0] = NUMBER[z]$, dit laatste omdat v_i eerder bezocht moet zijn dan $\{v_0,v_1,\ldots,v_{i-1}\}$.

We moeten nu nog aantonen dat $v_i \in CURRENT(v_{i-1})$, want dan hebben als volgt een tegenspraak met LOW[z] = NUMBER[z]:

$$LOW[z] \leq LOW[v_{i-1}] \leq NUMBER[v_i] < NUMBER[z].$$

Omdat v_i en v_{i-1} beide tot dezelfde s.s.c. G_1 behoren, zitten ze ook in dezelfde component T_1 van T, die wortel r heeft : $READY[r] \geq READY[v_{i-1}]$, d.w.z. $v_i \in CURRENT(v_{i-1})$. □

Gevolg

z is de wortel van een voortbrengende gerichte boom van een s.s.c. dan en slechts dan als LOW[z] = NUMBER[z].

Bewijs

Als z de wortel is van een voortbrengende gerichte boom van een s.s.c. dan geldt volgens Stelling 2.9: LOW[z] ≥ NUMBER[z]. De andere ongelijkheid volgt direct uit de definitie van LOW.
De omgekeerde bewering staat in Stelling 2.10. □

De getallen LOW kunnen als volgt worden berekend.

* Als een knooppunt v voor het eerst wordt bezocht (via aanroep VISIT(v)) dan maken we LOW[v] = NUMBER[v];
* Als (v,w) ∈ F, dan wordt tijdens VISIT(v) VISIT(w) aangeroepen. Zodra VISIT(w) klaar is, wordt LOW[w] vergeleken met de huidige waarde van LOW[v] wordt LOW[v] vervangen door LOW[w] als deze laatste waarde kleiner is.
* Als een (v,w) ∈ D ∪ C èn w ∈ CURRENT(v) wordt bekeken, dan wordt LOW[v] vergeleken met NUMBER[w] en wordt LOW[v] vervangen door NUMBER[w] als deze laatste waarde kleiner is. We weten dat (v,w) ∈ D ∪ C als NUMBER[w] ≤ NUMBER[v].
  Om na te kunnen gaan of w ∈ CURRENT(v) houden we een verz. CURRENT, gerepresenteerd door een stapel, bij die als volgt is gedefiniëerd:
  CURRENT = {w | w is reeds bezocht en r(w) is nog niet afgehandeld},
  met r(w) de wortel van de s.s.c. van w. Als een pijl (v,w) ∈ D ∪ C wordt bekeken, dan is w reeds bezocht. r(w) is nog niet afgehandeld als READY[r(w)] ≥ READY[v], d.w.z. de controle of w ∈ CURRENT(v) kan worden uitgevoerd door bij de berekening van LOW[v] na te gaan of w ∈ CURRENT. Dit geschiedt door de bezochte knooppunten op CURRENT te plaatsen totdat de wortel van een s.s.c. is ontdekt, waarna deze wortel en de overige knooppunten van deze s.s.c. uit CURRENT worden verwijderd.

Een stapel is een lijst waar de elementen aan één kant worden ingevoerd, en aan dezelfde kant weer worden verwijderd: het laatste element gaat er als eerste af. We gebruiken de volgende operaties op een stapel S:

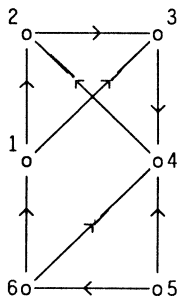
TOP(S) : bovenste element van S;
POP(S) : verwijdert het bovenste element van S;
PUSH(x,S) : voegt x op de top van S toe.

Tenslotte, als een wortel r van een s.s.c. is gevonden (NUMBER[r] = LOW[r]), dan moeten we nagaan welke opvolgers van r tot dezelfde s.s.c. behoren. Dit zijn de knooppunten v met NUMBER[v] ≥ NUMBER[r] en die niet eerder aan een s.s.c zijn toegekend, d.w.z. alle v ∈ CURRENT met NUMBER[v] ≥ NUMBER[r]. Dit zijn de elementen die vanaf v op de stapel CURRENT zijn geplaatst.

Het bovenstaande leidt tot de volgende procedure *SCC* (strongly connected components). Hierin is COMPONENT[j] = k als knooppunt j in de k-de s.s.c. zit. De pijlen van F, I, D en C worden niet bijgehouden.

```
procedure SCC;
  var i,j,k: integer;
      NUMBER,COMPONENT: array [1:n] of integer;
      CURRENT: stack of integers;
  procedure VISIT(v: integer);
    var w,x: integer;
    begin NUMBER[v] := i; LOW[v] := i; i := i+1;
          for iedere w van L[v] do
              if NUMBER[w] < 0 then
                 begin PUSH(w,CURRENT); VISIT(w);
                       if LOW[w] < LOW[v] then LOW[v] := LOW[w]
                 end
              else
                 begin if NUMBER[w] ≤ NUMBER[v] ∧ w ∈ CURRENT then
                       begin if NUMBER[w] < LOW[v] then LOW[v] := NUMBER[w]
                       end
                 end
          if LOW[v] = NUMBER[v] then
             begin x := TOP(CURRENT);
                   while  NUMBER[x] ≥ NUMBER[v] ∧ not EMPTY(CURRENT) do
                          begin POP(CURRENT); COMPONENT[x] := k;
                                if not EMPTY(CURRENT) then x := TOP(CURRENT)
                          end;
                   k := k+1
             end
    end VISIT;

begin i := 1; k := 1; MAKENULL(CURRENT);
      for j := 1 to n do NUMBER[j] := -1;
      for j := 1 to n do
          if NUMBER[j] < 0 then
             begin PUSH(j,CURRENT); VISIT(j) end
end SCC.
```

Voorbeeld 2.2 (vervolg)



Initialisatie: i = 1; k = 1; CURRENT = ∅; NUMBER[j] = -1, 1 ≤ j ≤ 6
j = 1: PRED[1] = 0; CURRENT = {1};
VISIT(1):
v = 1: NUMBER[1] = 1; LOW[1] = 1; i = 2;
w = 2: CURRENT = {1,2}
VISIT(2):
v = 2: NUMBER[2] = 2; LOW[2] = 2; i = 3;
w = 3: CURRENT = {1,2,3};
VISIT(3):
v = 3: NUMBER[3] = 3; LOW[3] = 3; i = 4;
w = 4: CURRENT = {1,2,3,4};
VISIT(4):
v = 4: NUMBER[4] = 4; LOW[4] = 4; i = 5;
w = 2: LOW[4] = 2;
LOW[3] = 2;
x = 4: CURRENT = {1,2,3}; COMPONENT[4] = 1;
x = 3: CURRENT = {1,2}; COMPONENT[3] = 1;
x = 2: CURRENT = {1}; COMPONENT[2] = 1;
k = 2;
w = 3:
x = 1: CURRENT = ∅; COMPONENT[1] = 2; k = 3;
j = 2:
j = 3:
j = 4:
j = 5: CURRENT= {5};
VISIT(5):
v = 5: NUMBER[5] = 5; LOW[5] = 5; i = 6;
w = 4:
w = 6: CURRENT= {5,6};
VISIT(6):
v = 6: NUMBER[6] = 6; LOW[6] = 6; i = 7;
w = 1:
w = 4:
x = 6: CURRENT = {5}; COMPONENT[6] = 3; k = 4;
x = 5: CURRENT = ∅; COMPONENT[5] = 4; k = 5;
j = 6:

Stelling 2.11

Bij een geschikte implementatie van CURRENT is de complexiteit van *SCC* $\bar{O}(p)$, waarbij $p = \max(n,m)$.

Bewijs

Naar analogie van Stelling 2.4 is het algoritme $\bar{O}(p)$ als we de opdrachten die te maken hebben met CURRENT niet meerekenen.

Beschouw nu de opdrachten met CURRENT. Ieder knooppunt wordt één keer in CURRENT geplaatst (op het moment dat het knooppunt voor de eerste keer bereikt wordt) en één keer uit CURRENT verwijderd: bij elkaar geeft dit $\bar{O}(n)$.

Blijft over de controle $w \in$ CURRENT. Dit is i.h.a. niet $\bar{O}(1)$. Dit probleem kan worden opgelost door CURRENT twee keer te representeren: als stapel èn als boolean array. De waarde van de boolean B[v] = true d.e.s.d als $v \in$ CURRENT. Voor ieder knooppunt is het bijhouden van de boolean en de controle of $v \in$ CURRENT dan $\bar{O}(1)$. In totaal dus eveneens $\bar{O}(n)$. □

Opgave 2.17

Pas het *SCC* algoritme toe op de gerichte graaf, gerepresenteerd door de lijsten:

L[1] = {2}; L[2] = {4,8}; L[3] = {9}; L[4] = {6,7}; L[5] = {3,4}; L[6] = {4};
L[7] = {6}; L[8] = {1,5}; L[9] = {10}; L[10] = {3,11}; L[11] = {9}.

Opgave 2.18

Geef de expliciete procedure van *SCC* zoals beschreven in het bewijs van Stelling 2.11.

Opgave 2.19*

In deze opgave bespreken we een ander algoritme om de streng samenhangende componenten van een gerichte graaf G = (V,A) te vinden. Dit gaat als volgt:

1. Voer *DFS* uit en nummer de knoopunten in de volgorde waarin ze worden afgehandeld (het eerst afgehandelde knooppunt krijgt nummer 1, etc.).
2. Construeer een andere graaf $G_1 = (V,A_1)$ met $(v,w) \in A_1 \Leftrightarrow (w,v) \in A$.
3. Voer *DFS* uit in G_1, beginnend met het hoogst genummerde knooppunt (volgens de in onderdeel 1 aangebrachte nummering). Indien met een nieuwe boom moet worden gestart, dan beginnen we weer met het overgebleven knooppunt dat het hoogste nummer heeft.

Beantwoord de volgende vragen over dit algoritme:

a. Pas het toe op voorbeeld 2.2.

b. Toon aan dat de componenten van *DFS* uit onderdeel 3 de s.s.c. van G zijn.

c. Ga de complexiteit van deze methode na.

III. KORTSTE PADEN IN NETWERKEN

1. Inleiding

Literatuur

* Paragraaf 3 van hoofdstuk 3 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds",
  Holt, Rinehart and Winston, New York (1976).

Zij $G = (V,A)$ een gerichte graaf. Aan iedere pijl $(v_i,v_j) \in A$ is een reëel getal l_{ij} toegevoegd dat we de lengte van de pijl zullen noemen. Een graaf met een dergelijke lengtefunctie heet een netwerk. Als lengte van een pad nemen we de som van de lengten van de pijlen van dit pad.

We veronderstellen dat de lengte van iedere ronde positief is. We kunnen dan vragen naar (de lengte van) het kortste pad vanuit een knooppunt v_1 naar een willekeurig ander knooppunt v_i. We zullen laten zien hoe dit kortste pad probleem kan worden opgelost en dat dit tegelijk de kortste paden bepaalt van v_1 naar alle overige knooppunten.

Door voor $l_{ij} = +\infty$ te nemen als $(v_i,v_j) \notin A$ mogen we wel aannemen dat de graaf volledig is. Definiëer:

$$\begin{cases} u_1^* = 0 \\ u_i^* = \text{lengte van het kortste pad van } v_1 \text{ naar } v_i, \ i = 2,3,\ldots,n. \end{cases}$$

Stelling 3.1

(i) De getallen u_i^*, $i = 1,2,\ldots,n$ voldoen aan de zgn. Bellman vergelijkingen

$$(3.1) \quad \begin{cases} u_1 = 0 \\ u_i = \min\limits_{k \neq i} (u_k + l_{ki}) \ , \ i = 2,3, \ \ldots \ ,n \end{cases}$$

(ii) Met iedere oplosing u van de Bellman-vergelijkingen correspondeert een voortbrengende boom met wortel v_1 zódanig dat u_i de afstand in de boom is van v_1 naar v_i.

(iii) Er is een boom van kortste paden.

(iv) u_i^* , $i = 1,2,\ldots,n$ is de enige oplossing van de Bellmanvergelijkingen.

Bewijs

(i) Het is duidelijk dat $u_i^* \leq u_k^* + l_{ki}$. Dus $u_i^* \leq \min_{k \neq i}(u_k^* + l_{ki})$.
Voor ieder knooppunt v_i $(i \neq 1)$ heeft een kortste pad van v_1 naar v_i een laatste pijl, zeg $(v_{k(i)}, v_i)$. Het gedeelte van het pad tot aan $v_{k(i)}$ moet een kortste pad zijn van v_1 naar $v_{k(i)}$.
Dus: $u_i^* = u_{k(i)}^* + l_{k(i)i} \geq \min_{k \neq i}(u_k^* + l_{ki})$.

(ii) Stel u_i , $i = 1,2,\ldots,n$ is een oplossing van de Bellmanvergelijkingen. Voor iedere $i \neq 1$ is er dus een $k(i)$ zdd. $u_i = u_{k(i)} + l_{k(i)i}$. Beschouw nu de deelgraaf bestaande uit de pijlen $(v_{k(i)}, v_i)$, $i = 2,3,\ldots,n$, en de bijbehorende knooppunten. Dit geeft een deelgraaf met n-1 pijlen. Stel er is een ronde in deze deelgraaf, zeg $[v_{i_1}, v_{i_2}, \ldots, v_{i_j} = v_{i_1}]$. Dan geldt: $l_{i_k i_{k+1}} = u_{i_{k+1}} - u_{i_k}$, $k = 1,2,\ldots,j-1$.
De lengte van de ronde is dus: $\Sigma_{k=1}^{j-1} l_{i_k i_{k+1}} = \Sigma_{k=1}^{j-1} u_{i_{k+1}} - \Sigma_{k=1}^{j-1} u_{i_k} = 0$, omdat $i_j = i_1$. Dit geeft een tegenspraak en de deelgraaf heeft dus geen ronden. Omdat er naar iedere knooppunt $v_i \neq v_1$ precies één pijl gericht is kunnen we, als we in een willekeurig knooppunt beginnen en tegen de richting van de pijlen inlopen, slechts eindigen in v_1. Dit impliceert dat de deelgraaf een voortbrengende boom met wortel v_1 is. Uit de constructie volgt tevens dat u_i de afstand in de boom is van v_1 naar v_i.

(iii) Volgt uit (i) en (ii).

(iv) Stel u_i , $i = 1,2,\ldots,n$ is ook een oplossing. Volgens (ii) is dan $u_i > u_i^*$ als $u_i \neq u_i^*$ en $u_{k(i)} = u_{k(i)}^*$ met $(v_{k(i)}, v_i)$ een pijl in de boom van kortste paden. Omdat $u_1 = u_1^* = 0$ moet er zo'n i en k(i) bestaan. We krijgen nu de volgende tegenspraak:

$$u_i > u_i^* = u_{k(i)}^* + l_{k(i)i} = u_{k(i)} + l_{k(i)i} \geq \min_{k \neq i}(u_k + l_{ki}) = u_i. \quad \square$$

Helaas zijn de Bellman vergelijkingen een niet-lineair stelsel. Dit maakt het oplossen lastig. We zullen in de volgende paragrafen verschillende methoden beschrijven om deze vergelijkingen op te lossen.

Opgave 3.1

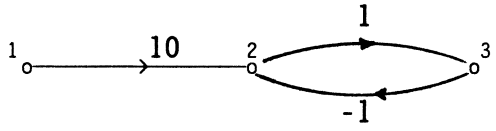
Beschouw een volledige gerichte graaf met 102 knooppunten.

a. Toon aan dat er $\Sigma_{k=0}^{100} \binom{100}{k} k!$ verschillende enkelvoudige paden zijn van v_1 naar v_{102}.

b. Toon aan dat dit aantal het grootste gehele getal is dat niet groter is dan $100! \cdot e$.

Opgave 3.2

Beschouw het hieronder getekende netwerk. Toon aan dat voor dit netwerk de oplossing van de Bellman vergelijkingen niet uniek is. Karakteriseer bovendien de verz. van alle oplossingen.



Opgave 3.3

Laat de lengte van de pijlen van een graaf met 5 knooppunten gegeven zijn door nevenstaande matrix.
Neem aan dat de lengten van kortste paden voldoen aan:
$u_1^* = 0$, $u_2^* = -1$, $u_3^* = 3$, $u_4^* = 5$ en $u_5^* = 6$.
Construeer een boom van kortste paden. Is deze uniek?

$$\begin{pmatrix} 0 & -1 & 3 & \infty & 6 \\ \infty & 0 & 5 & 6 & 8 \\ \infty & \infty & 0 & \infty & 10 \\ \infty & 0 & -1 & 0 & \infty \\ \infty & 0 & 0 & -1 & 0 \end{pmatrix}$$

Opgave 3.4

Veronderstel dat $G = (V,A)$ een gerichte graaf is waarvoor geldt dat $(v_i,v_j) \in A$ impliceert dat $i < j$.
Construeer een $\mathcal{O}(n^2)$ algoritme om de (lengten van de) kortste paden in G te bepalen.

2. Methode van Dijkstra

Literatuur

* Paragraaf 5 van hoofdstuk 3 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds",
  Holt, Rinehart and Winston, New York (1976).
* Hoofdstuk 31 in:
  R.Sedgewick: "Algorithms", Addison-Wesley, Reading (1983).

We veronderstellen in deze paragraaf dat iedere lengte niet-negatief is. Aan de knooppunten van het netwerk worden labels u toegekend. Bij iedere iteratie zullen sommige labels definitief zijn en de andere tijdelijk.
Een definitief label bij knooppunt i is gelijk aan u_i^*, de lengte van het kortste pad van v_1 naar v_i; een tijdelijk label geeft de afstand van het kortste pad van v_1 naar v_i met als tussenpunten alleen punten met een definitief label.

Bij de start heeft alleen v_1 een definitief label, nl. $u_1 = u_1^* = 0$; de andere labels zijn $u_j = l_{1j}$, $j = 2,3,\ldots,n$.

De algemene stap van deze methode gaat als volgt:

a. Zoek een knooppunt waarvoor het tijdelijke label minimaal is, zeg v_p.
b. Maak v_p tot een knooppunt met een definitief label.
c. Bereken voor alle knooppunten v_j met een tijdelijk label het nieuwe label:
   $u_j := \min(u_j, u_p + l_{pj})$.

Het algoritme eindigt als alle knooppunten een definitief label hebben gekregen.

Hieronder volgt het algoritme. In het algoritme worden de volgende parameters gebruikt:

l[i,j]: lengte van v_i naar v_j (een pijl die niet bestaat krijgt lengte max).
u[j] : label voor het knooppunt v_j.
T[j] : boolean met waarde true als u[j] definitief is.
k[j] : voorganger van j op het pad van v_1 naar v_j.
p : index van het knooppunt met kleinste tijdelijk label.
i : iteratieteller.

```
procedure Dijkstra;
  var i,j,p: integer;
      u: array [1:n] of real;
      k: array [1:n] of integer;
      T: array [1:n] of boolean;
begin T[1] := false; u[1] := 0;
      for j := 2 to n do
          begin u[j] := l[1,j]; k[j] := 1; T[j] := true end;
      for i := 2 to n do
          begin min := n × max;
                for j:= 2 to n do if T[j] ∧ u[j] < min then
                    begin p := j; min := u[j] end;
                T[p] := false;
                for j:= 2 to n do if T[j] ∧ u[j] > u[p] + l[p,j] then
                    begin k[j] := p; u[j] := u[p] + l[p,j] end
          end
end Dijkstra.
```

Voorbeeld 3.1

Beschouw het netwerk met nevenstaande lengte-matrix (∞ geeft aan dat de pijl niet bestaat). Voor max nemen we 100.

$$\begin{pmatrix} 0 & 4 & 5 & 2 & 12 & \infty & \infty & \infty \\ \infty & 0 & 3 & \infty & 1 & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty & \infty & 13 \\ \infty & \infty & 1 & 0 & \infty & 11 & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & 6 & 9 \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty & 8 \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 & 7 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

Iteratie 1

T[1] = false; u[1] = 0;
u[2] = 4; k[2] = 1; T[2] = true;
u[3] = 5; k[3] = 1; T[3] = true;
u[4] = 2; k[4] = 1; T[4] = true;
u[5] = 12; k[5] = 1; T[5] = true; u[6] = 100; k[6] = 1; T[6] = true;
u[7] = 100; k[7] = 1; T[7] = true; u[8] = 100; k[8] = 1; T[8] = true.

Iteratie 2

min = 800; p = 2; min = 4; p = 4; min = 2;
T[4] = false; k[3] = 4; u[3] = 3; k[6] = 4; u[6] = 13.

Iteratie 3

min = 800; p = 2; min = 4; p = 3; min = 3; T[3] = false; k[8] = 3; u[8] = 16.

Iteratie 4

min = 800; p = 2; min = 4; T[2] = false; k[5] = 2; u[5] = 5.

Iteratie 5

min = 800; p = 5; min = 5;

T[5] = false; k[7] = 5; u[7] = 11; k[8] = 5; u[8]= 14.

Iteratie 6

min = 800; p = 6; min = 13; p = 7; min = 11; T[7] = false.

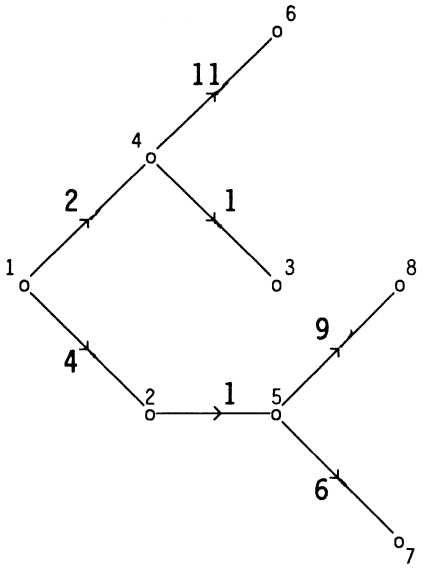
Iteratie 7

min = 800; p = 6; min = 13; T[6] = false.

Iteratie 8

min = 800; p = 8; min = 14; T[8] = false.

De boom van kortste paden is:



Stelling 3.2

(i) Het algoritme van Dijkstra eindigt met de lengten van de kortste paden en de boom $\{(k[j],j),\ j = 2,3,\ldots,n\}$ is een boom van kortste paden.

(ii) De complexiteit van het algoritme is $\mathcal{O}(n^2)$.

Bewijs

(i) We tonen met inductie naar het iteratienummer aan dat tijdens het algoritme de definitieve knooppunten als label de lengte van het kortste pad hebben en dat voor de overige knooppunten v_j geldt:

$$u[j] = \min_i\{u[i] + l[i,j] \mid v_i \text{ definitief}\}.$$

Bij de eerste iteratie, als alleen v_1 definitief is, is dit correct.
Veronderstel dat de berekening juist is geschied tot en met iteratie i-1.

Stel v_p is zó gekozen dat $u[p] = \min_i\{u[i] \mid v_i \text{ tijdelijk}\}$, en dat $u[p] > u_p^*$. Dan is er dus een pad P, korter dan u[p], van v_1 naar v_p met minstens één tijdelijk knooppunt op P. Laat v_j het eerste tijdelijke knooppunt op P zijn. De lengte van het kortste pad naar v_j is dus wel de kortste lengte met als tussenpunten alleen definitieve punten:
$u_j^* = u[j] \geq u[p] > u_p^* \geq u_j^*$, de laatste ongelijkheid omdat alle lengten niet-negatief zijn en v_p op P verder dan v_j van v_1 af ligt. Dit geeft de gewenste tegenspraak.
Uit het algoritme is het duidelijk dat de u[j] goed wordt bijgehouden en dat k[j] de voorganger van j is op het pad van v_1 naar v_j met lengte u[j] We hebben zojuist aangetoond dat de u[j]-waarden aan het einde de u^*-waarden zijn. Omdat we voor de u[j]'s de voorganger van v_j bijhouden met k[j], is dit ook de voorganger van v_j op het kortste pad met lengte u^*: {(k[j],j), j = 2,3,...,n} is een boom van kortste paden.

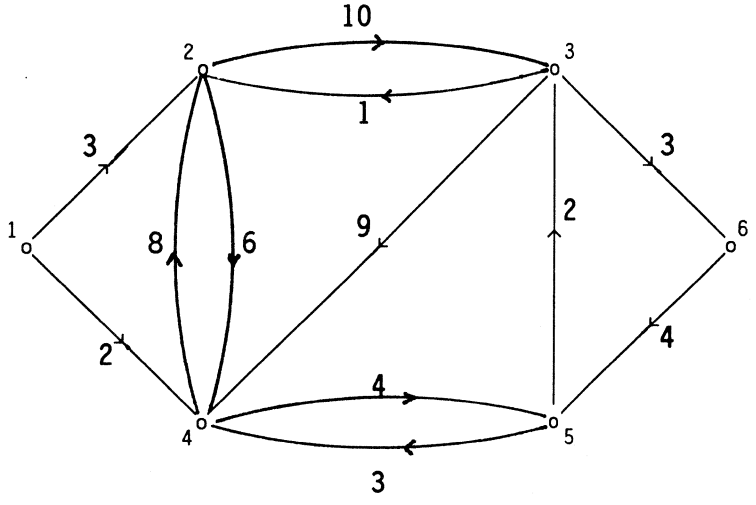
(ii) De eerste iteratie is van $\mathcal{O}(n)$. Het is een voudig in te zien dat in de volgende iteraties beide for-statements van $\mathcal{O}(n)$ zijn. Aangezien er n-1 volgende iteraties zijn is de totale complexiteit $\mathcal{O}(n^2)$. □

Opgave 3.5

Toon met een voorbeeld aan dat Dijkstra's algoritme niet mag worden toegepast op netwerken waarin ook negatieve lengten voorkomen.

Opgave 3.6

Pas het algoritme van Dijkstra toe op onderstaand netwerk.



Opgave 3.7

Veronderstel dat G = (V,A) een netwerk is met $l_{ij} = 1$ voor alle $(v_i,v_j) \in A$. Laat de vanuit v met één pijl bereikbare knooppunten gegeven zijn door de structuurlijst L[v], $v \in V$.

Modificeer het algoritme van Dijkstra zódat de (lengten van de) kortste paden in $O(n+m)$ worden bepaald.

3. Methode van Bellman en Ford

Literatuur

* Paragraaf 6 van hoofdstuk 3 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds", Holt, Rinehart and Winston, New York (1976).
* Paragraaf 7.3 in:
  K.Mehlhorn: "Data structures and algorithms 2: Graph algorithms and NP-completeness", Springer, Berlin, (1984).

In deze paragraaf mogen de lengten ook negatief zijn (wel nemen we aan dat iedere ronde een niet-negatieve lengte heeft).
De Bellman vergelijkingen kunnen worden opgelost door de volgende iteratieve methode, ook wel successieve approximatie genoemd.
We beginnen met een eerste benadering u^1 van u^*, nl. $u_j^1 = l_{1j}$, $2 \leq j \leq n$.
In het algemeen is u_j^k de lengte van het kortste pad van v_1 naar v_j met hoogstens k pijlen. u^{k+1} kan dan als volgt uit u^k worden berekend:

$$u_j^{k+1} = \min \{u_j^k, \min_i [u_i^k + l_{ij}]\}, \; j = 2,3,\ldots,n.$$

Omdat de lengte van het kortste pad hoogstens n-1 pijlen heeft is $u^{n-1} = u^*$.

Deze methode kan als volgt worden geïmplementeerd.

```
procedure Bellman_Ford_1;
  var i,j,k: integer;
      u,d: array [1:n] of real;
      k: array [1:n] of integer;
begin u[1] := 0;
      for j := 2 to n do
          begin u[j] := l[1,j]; k[j] := 1 end;
      for k := 2 to n-1 do
          begin for j:= 2 to n do
                begin d[j] := u[j];
                      for i:= 2 to n do if u[i] + l[i,j] < d[j] then
                    begin k[j] := i; d[j] := u[i] + l[i,j] end
                end;
                for j := 2 to n do u[j] := d[j]
          end
end Bellman_Ford_1.
```

Bovenstaand algoritme kan op de volgende wijze worden versneld:

1. Zodra $u^{k+1} = u^k$, zal er nooit meer iets veranderen en is $u^* = u^k$.
2. Zij $w_j^{k+1} = \min \{w_j^k, \min_{i<j}[w_i^{k+1} + l_{ij}], \min_{i>j}[w_i^k + l_{ij}]\}$, $j = 2,3,\ldots,n$ (start met $w^1 = u^1$). Dan geldt:
   Voor alle k en j is $w_j^k \leq u_j^k$ en is er een pad van v_1 naar v_j met lengte w_j^k. Omdat $u_j^* \leq w_j^{n-1} \leq u_j^{n-1} = u_j^*$, $2 \leq j \leq n$, leidt de berekening van de getallen w^{n-1} ook tot de lengten van de kortste paden. Meestal convergeert w^k veel sneller dan u^k.

```
procedure Bellman_Ford_2;
  var i,j,k: integer;
      ready: boolean;
      u: array [1:n] of real;
      k: array [1:n] of integer;
begin u[1] := 0; k := 1;
      for j := 2 to n do
            begin u[j] := l[1,j]; k[j] := 1 end;
      repeat k := k+1; ready := true;
            for j:= 2 to n do
               begin for i:= 2 to n do if u[i] + l[i,j] < u[j] then
                     begin ready := false; k[j] := i; u[j] := u[i] + l[i,j]
                     end
               end
      until k = n-1 or ready
end Bellman_Ford_2.
```

Voorbeeld 3.2

Beschouw het netwerk met nevenstaande lengte-matrix (∞ geeft aan dat de pijl niet bestaat; we nemen hiervoor weer de waarde 100).

$$\begin{pmatrix} 0 & -3 & \infty & \infty & 2 & 3 & \infty \\ \infty & 0 & -5 & 15 & 12 & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty & \infty \\ 6 & \infty & 8 & 0 & \infty & 4 & 11 \\ \infty & \infty & \infty & -7 & 0 & \infty & \infty \\ \infty & \infty & -10 & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{pmatrix}$$

Iteratie 1

k = 1; u[1] = 0; u[2] = -3; k[2] = 1; u[3] = 100; k[3] = 1; u[4] = ∞; k[4] = 1; u[5] = 2; k[5] = 1; u[6] = 3; k[6] = 1; u[7] = 100; k[7] = 1.

Iteratie 2

k = 2: ready = false; k[3] = 2; u[3] = -8; ready = false; k[4] = 5; u[4] = -5.
k[6] = 4; u[6] = -1; ready = false; k[7] = 4; u[7] = 6.

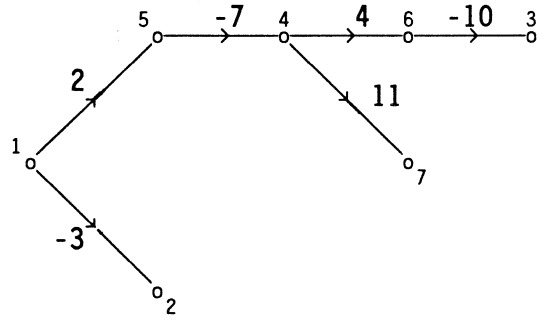
Iteratie 3

k = 3: ready = false; k[3] = 6; u[3] = -11.

Iteratie 4

k = 4: geen veranderingen meer.

De boom van kortste paden is:



Stelling 3.3

(i) Het algoritme 2 van Bellman en Ford eindigt met de lengten van de kortste paden en de boom {(k[j],j), j = 2,3,...,n} is een boom van kortste paden.

(ii) De complexiteit van het algoritme is $O(n^3)$.

Bewijs

(i) We hebben reeds gezien bij de afleiding van de methode dat u_j^k (in de versie van algoritme 2) een pad is van v_1 naar v_j, en dat $u^{n-1} = u^*$. Het algoritme eindigt dus met de lengten van de kortste paden. De voorgangers worden weer bijgehouden met k[j], dus {(k[j],j), j = 2,3,...,n} is een boom van kortste paden.

(ii) De eerste iteratie vereist $O(n)$ stappen. Daarna komen er nog hoogstens n-2 iteraties, die ieder twee geneste for-statements hebben en binnen deze for-statements is de hoeveelheid werk $O(1)$. Per iteratie dus $O(n^2)$ en in totaal $O(n^3)$. □

Opgave 3.8

Veronderstel dat:

u_j^k = de lengte van de kortste pijlenreeks van v_1 naar v_j met precies k pijlen, $j = 2,3,\ldots,n$.

Geef een betrekking waar u^{k+1} aan voldoet.

Opgave 3.9

Veronderstel dat $G = (V,A)$ een netwerk is, waarbij $(v_i,v_j) \in A$ impliceert dat $i < j$.

Na hoeveel iteraties convergeert algoritme 2 van Bellman en Ford?

Opgave 3.10

Beschouw het netwerk met nevenstaande lengtematrix (∞ geeft aan dat de pijl niet bestaat)
Pas de methode van Bellman en Ford toe om de kortste paden in dit netwerk te bepalen.

$$\begin{pmatrix} 0 & \infty & 4 & 10 & 3 & \infty & \infty \\ \infty & 0 & -1 & -3 & 2 & 11 & 0 \\ \infty & 9 & 0 & 8 & 3 & 2 & 1 \\ \infty & 4 & 0 & 0 & 8 & 6 & 3 \\ \infty & 0 & 1 & 2 & 0 & 3 & -1 \\ \infty & 1 & -1 & 3 & 2 & 0 & 0 \\ \infty & 4 & 3 & \infty & \infty & 2 & 0 \end{pmatrix}$$

Opgave 3.11

Beschouw een netwerk $G = (V,A)$.

Veronderstel dat v_1 alleen uitgaande pijlen heeft en dat er voor ieder knooppunt $v_j \neq v_1$ een pad is van v_1 naar v_j.

a. Toon aan dat G een ronde met een negatieve lengte bevat d.e.s.d. als $u_j^n < u_j^{n-1}$ voor zekere j.

b. Gebruik onderdeel a om na te gaan of het netwerk uit opgave 3.10, met daarin $l_{62} = -1$ en de andere lengten ongewijzigd, een ronde van negatieve lengte bevat.

Opgave 3.12*

Veronderstel dat G = (V,A) een netwerk is zonder een ronde van negatieve lengte. Beschouw het volgende algoritme.

```
procedure SHORTEST_PATH;
  var i,j: integer;
      u: array [1:n] of real;
      k: array [1:n] of integer;
      T: array [1:n] of boolean;
      Q: queue of integer;
begin T[1] := true; u[1] := 0; MAKENULL(Q); ENQUEUE(1,Q);
      for j := 2 to n do
          begin u[j] := ∞; T[j] := false end;
      while not EMPTY(Q) do
            begin i := FRONT(Q); DEQUEUE(Q); T[i] := false;
                  for all (i,j) ∈ A do
                      begin if u[i] + l[i,j] < u[j] then
                                  begin u[j] := u[i] + l[i,j]; k[j] := i;
                                        if not T[j] then
                                           begin ENQUEUE(j,Q); T[j] := true end
                                  end
                      end
            end
end SHORTEST_PATH.
```

Beantwoord de volgende vragen over dit algoritme.

a. Pas het toe op voorbeeld 3.2.

b. Toon aan dat voor een $i \notin Q$ geldt: $u[i] + l[i,j] \geq u[j]$ voor alle j.

c. Toon aan dat Q tijdens iedere iteratie een j bevat waarvoor geldt: $u[j] = u_j^*$.

d. Toon aan dat ieder knooppunt hoogstens n-1 keer in Q geplaatst kan worden.

e. Toon aan dat het algoritme eindigt met de (lengten van de) kortste paden.

f. Indien de datastructuur gegeven is door structuurlijsten $L[v_j]$, $1 \leq j \leq n$, dan is de complexiteit $O(n \cdot m)$. Toon dit aan.

4. Methode van Floyd en Warshall

Literatuur

* Paragraaf 9 van hoofdstuk 3 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds",
  Holt, Rinehart and Winston, New York (1976).
* Hoofdstuk 37 in:
  R.Sedgewick: "Algorithms", Addison-Wesley, Reading (1983).

In plaats van de kortste paden vanuit één knooppunt beschouwen we in deze paragraaf de kortste paden tussen ieder tweetal knooppunten.
We nemen weer aan dat het netwerk geen ronden met een negatieve lengte bevat.

Met achtereenvolgens ieder knooppunt als wortel zouden we bijvoorbeeld n keer het algoritme van Bellman en Ford kunnen toepassen. Dit geeft een methode met complexiteit $O(n^4)$. Het kan echter ook met $O(n^3)$, zoals door Floyd en Warshall is aangetoond.

Definiëer: u_{ij}^k = lengte van het kortste pad van v_i naar v_j met als tussenpunten slechts punten van $\{v_1, v_2, \ldots, v_{k-1}\}$.

Voor u^1 geldt: $u_{ij}^1 = l_{ij}$ voor alle (i,j). Omdat voor het kortste pad van v_i naar v_j met als tussen punten slechts elementen van $\{v_1, v_2, \ldots, v_k\}$ geldt: dit pad bevat v_k òf wel òf niet:

$$u_{ij}^{k+1} = \min [u_{ij}^k, u_{ik}^k + u_{kj}^k] \text{ voor alle } (i,j).$$

Als u_{ij}^* de lengte is van het kortste pad van v_i naar v_j, dan is $u_{ij}^* = u_{ij}^{n+1}$.
Dit laat zich als volgt eenvoudig implementeren. Met k[i,j] houden we de voorganger van j bij op het pad van i naar j.
Bij deze methode moeten we - in tegenstelling tot die van Bellman en Ford - altijd u^{n+1} uitrekenen, ook al is $u^k = u^{k+1}$ voor zekere k.

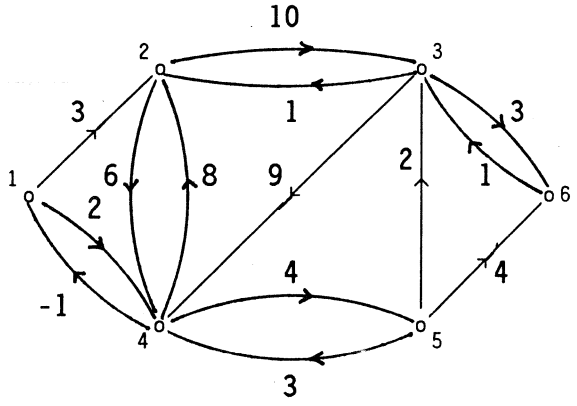
```
procedure Floyd_Warshall;
  var i,j,p: integer;
      u: array [1:n,1:n] of real;
      k: array [1:n,1:n] of integer;
begin for i := 1 to n do
          for j := 1 to n do
              begin u[i,j] := l[i,j]; k[i,j] := i end;
      for p := 1 to n do
          for i := 1 to n do
              for j := 1 to n do
                  if u[i,p] + u[p,j] < u[i,j] then
                      begin u[i,j] := u[i,p] + u[p,j]; k[i,j] := k[p,j] end
end Floyd_Warshall.
```

Voorbeeld 3.3

Beschouw het netwerk met nevenstaande lengtematrix (∞ geeft aan dat de pijl niet bestaat)

$$\begin{pmatrix} 0 & 3 & \infty & 2 & \infty & \infty \\ \infty & 0 & 10 & 6 & \infty & \infty \\ \infty & 1 & 0 & 9 & \infty & 3 \\ -1 & 8 & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 3 & 0 & \infty \\ \infty & \infty & 1 & \infty & 4 & 0 \end{pmatrix}$$

Het netwerk is hieronder getekend.



Per iteratie geven we de matrices u en k; de elementen die veranderd zijn onderstrepen we.

Iteratie 1

$$u^1 = \begin{pmatrix} 0 & 3 & \infty & 2 & \infty & \infty \\ \infty & 0 & 10 & 6 & \infty & \infty \\ \infty & 1 & 0 & 9 & \infty & 3 \\ -1 & 8 & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 3 & 0 & \infty \\ \infty & \infty & 1 & \infty & 4 & 0 \end{pmatrix} \qquad k^1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 \end{pmatrix}$$

Iteratie 2

$$u^2 = \begin{bmatrix} 0 & 3 & \infty & 2 & \infty & \infty \\ \infty & 0 & 10 & 6 & \infty & \infty \\ \infty & 1 & 0 & 9 & \infty & 3 \\ -1 & \underline{2} & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 3 & 0 & \infty \\ \infty & \infty & 1 & \infty & 4 & 0 \end{bmatrix} \qquad k^2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & \underline{1} & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 \end{bmatrix}$$

Iteratie 3

$$u^3 = \begin{bmatrix} 0 & 3 & \underline{13} & 2 & \infty & \infty \\ \infty & 0 & 10 & 6 & \infty & \infty \\ \infty & 1 & 0 & \underline{7} & \infty & 3 \\ -1 & 2 & \underline{12} & 0 & 4 & \infty \\ \infty & \infty & 2 & 3 & 0 & \infty \\ \infty & \infty & 1 & \infty & 4 & 0 \end{bmatrix} \qquad k^3 = \begin{bmatrix} 1 & 1 & \underline{2} & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & \underline{2} & 3 & 3 \\ 4 & 1 & \underline{2} & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 6 & 6 & 6 & 6 & 6 & 6 \end{bmatrix}$$

Iteratie 4

$$u^4 = \begin{bmatrix} 0 & 3 & 13 & 2 & \infty & \underline{16} \\ \infty & 0 & 10 & 6 & \infty & \underline{13} \\ \infty & 1 & 0 & 7 & \infty & 3 \\ -1 & 2 & 12 & 0 & 4 & \underline{15} \\ \infty & \underline{3} & 2 & 3 & 0 & \underline{5} \\ \infty & \underline{2} & 1 & \underline{8} & 4 & 0 \end{bmatrix} \qquad k^4 = \begin{bmatrix} 1 & 1 & 2 & 1 & 1 & \underline{3} \\ 2 & 2 & 2 & 2 & 2 & \underline{3} \\ 3 & 3 & 3 & 2 & 3 & 3 \\ 4 & 1 & 2 & 4 & 4 & \underline{3} \\ 5 & \underline{3} & 5 & 5 & 5 & \underline{3} \\ 6 & \underline{3} & 6 & \underline{2} & 6 & 6 \end{bmatrix}$$

Iteratie 5

$$u^5 = \begin{bmatrix} 0 & 3 & 13 & 2 & \underline{6} & 16 \\ \underline{5} & 0 & 10 & 6 & \underline{10} & 13 \\ \underline{6} & 1 & 0 & 7 & \underline{11} & 3 \\ -1 & 2 & 12 & 0 & 4 & 15 \\ \underline{2} & 3 & 2 & 3 & 0 & 5 \\ \underline{7} & 2 & 1 & 8 & 4 & 0 \end{bmatrix} \qquad k^5 = \begin{bmatrix} 1 & 1 & 2 & 1 & \underline{4} & 3 \\ \underline{4} & 2 & 2 & 2 & \underline{4} & 3 \\ \underline{4} & 3 & 3 & 2 & \underline{4} & 3 \\ 4 & 1 & 2 & 4 & 4 & 3 \\ \underline{4} & 3 & 5 & 5 & 5 & 3 \\ \underline{4} & 3 & 6 & 2 & 6 & 6 \end{bmatrix}$$

Iteratie 6

$$u^6 = \begin{bmatrix} 0 & 3 & \underline{8} & 2 & 6 & \underline{11} \\ 5 & 0 & 10 & 6 & 10 & 13 \\ 6 & 1 & 0 & 7 & 11 & 3 \\ -1 & 2 & \underline{6} & 0 & 4 & \underline{9} \\ 2 & 3 & 2 & 3 & 0 & 5 \\ \underline{6} & 2 & 1 & \underline{7} & 4 & 0 \end{bmatrix} \qquad k^6 = \begin{bmatrix} 1 & 1 & \underline{5} & 1 & 4 & \underline{3} \\ 4 & 2 & 2 & 2 & 4 & 3 \\ 4 & 3 & 3 & 2 & 4 & 3 \\ 4 & 1 & \underline{5} & 4 & 4 & \underline{3} \\ 4 & 3 & 5 & 5 & 5 & 3 \\ \underline{4} & 3 & 6 & \underline{5} & 6 & 6 \end{bmatrix}$$

Iteratie 7

$$u^7 = \begin{bmatrix} 0 & 3 & 8 & 2 & 6 & 11 \\ 5 & 0 & 10 & 6 & 10 & 13 \\ 6 & 1 & 0 & 7 & \underline{7} & 3 \\ -1 & 2 & 6 & 0 & 4 & 9 \\ 2 & 3 & 2 & 3 & 0 & 5 \\ 6 & 2 & 1 & 7 & 4 & 0 \end{bmatrix} \qquad k^7 = \begin{bmatrix} 1 & 1 & 5 & 1 & 4 & 3 \\ 4 & 2 & 2 & 2 & 4 & 3 \\ 4 & 3 & 3 & 2 & \underline{6} & 3 \\ 4 & 1 & 5 & 4 & 4 & 3 \\ 4 & 3 & 5 & 5 & 5 & 3 \\ 4 & 3 & 6 & 5 & 6 & 6 \end{bmatrix}$$

Hiermee zijn alle kortste paden bepaald. Willen we bijvoorbeeld het kortste pad hebben van v_6 naar v_1, dan lezen we in $u^7[6,1]$ af dat de lengte 6 is; de route krijgen we door in de matrix k^7 op plaats (6,1) te kijken en via dat getal "terug te lopen": eerst komen we bij 4, $k^7[6,4] = 5$, dus vervolgens krijgen we knooppunt 5, daarna omdat $k^7[6,5] = 6$ eindigen we in knooppunt 6: in omgekeerde richting is het pad: $1 \leftarrow 4 \leftarrow 5 \leftarrow 6$.

Stelling 3.4

(i) Het algoritme van Floyd en Warshall eindigt met de lengten van de kortste paden, opgeslagen in de matrix u, en de routes zijn af te leiden uit de matrix k.

(ii) De complexiteit van het algoritme is $\mathcal{O}(n^3)$.

Bewijs

(i) Dit volgt direct uit de voorgaande beschouwingen.

(ii) De initialisatie is $\mathcal{O}(n^2)$. Daarna volgt een drievoudige for-statement met daarachter opdrachten van $\mathcal{O}(1)$. De totale complexiteit is dus $\mathcal{O}(n^3)$. □

Opgave 3.13

Pas het algoritme van Floyd en Warshall toe op het netwerk uit opgave 3.10.

Opgave 3.14

Veronderstel dat het kortste pad van v_i naar v_j niet uniek is. Bijvoorbeeld, veronderstel dat er twee kortste paden zijn: $[v_i, v_2, v_4, v_j]$ en $[v_i, v_{10}, v_1, v_j]$. Welk pad wordt gekozen met het algoritme van Floyd en Warshal?
Stel een regel op die deze vraag in het algemeen beantwoordt.

Opgave 3.15

Beschrijf een methode met complexiteit $O(n^3)$ om de kortste ronde te bepalen in een graaf die geen ronde van negatieve lengte heeft.

Opgave 3.16

Zij A een n×n matrix. Toon de volgende bewering aan.
A is een matrix van de lengten van de kortste paden in een zeker netwerk d.e.s.d. als voor iedere i geldt: $a_{ii} = 0$ en $a_{ij} \leq a_{ik} + a_{kj}$ voor alle k en j.

Opgave 3.17*

Beschouw het volgende, van Dantzig afkomstige, algoritme.

```
procedure Dantzig;
  var i,j,k,l: integer;
      u: array [1:n,1:n] of real;
begin for i := 1 to n do u[i,i] := 0;
      for k := 2 to n do
          begin for l := 1 to k-1 do
                    begin u[k,l] := min_{1≤j≤k-1} {l[k,j] + u[j,l]};
                          u[l,k] := min_{1≤j≤k-1} {l[j,k] + u[l,j]}
                    end;
                for i := 1 to k-1 do
                    for j := 1 to k-1 do u[i,j] := min{u[i,j],u[i,k] + u[k,j]}
          end
end Dantzig.
```

Beantwoord de volgende vragen.

a. Pas dit algoritme toe op voorbeeld 3.3.

b. Toon aan dat dit algoritme de lengten van de kortste paden tussen ieder tweetal knooppunten bepaalt.

c. Toon aan dat de complexiteit $O(n^3)$ is.

5. De simplexmethode

Literatuur:

* D.Goldfarb, J.Hao and S.R.Kai: "Efficient shortest path simplex algorithms", Operations Research, 38 (1990) pp.624-628.
* Paragraaf 8 van hoofdstuk 3 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds", Holt, Rinehart and Winston, New York (1976).

Het verband tussen de lineaire programmering en de lengten van de kortste paden wordt gegeven door de volgende stelling.

Stelling 3.5

De getallen u_i^*, $i = 1,2,\ldots,n$ zijn de unieke oplossing van het LP-probleem:

$$(3.2) \qquad \max \left\{ \Sigma_{i=1}^{n} u_i \;\middle|\; \begin{array}{l} u_1 = 0 \\ u_i - u_k \leq l_{ki} \text{ voor alle } (k,i) \text{ met } (v_k,v_i) \in A \end{array} \right\}$$

Bewijs

Uit Stelling 3.1 volgt dat u^* een toegelaten oplossing van (3.2) is. Laat ook u een toegelaten oplossing zijn en neem een willekeurige v_i. Met inductie naar het aantal pijlen op het pad van v_1 naar v_i in de boom bij u^* bewijzen we dat $u_i^* \geq u_i$.

Als deze lengte 1 is: $u_i^* = l_{1i} = u_1 + l_{1i} \geq u_i$.

Als deze lengte j is: Laat $(v_{k(i)},v_i)$ de pijl in de boom zijn die naar v_i loopt.

Dan is volgens de inductieveronderstelling $u_{k(i)}^* \geq u_{k(i)}$, waaruit volgt:

$$u_i^* = u_{k(i)}^* + l_{k(i)i} \geq u_{k(i)} + l_{k(i)i} \geq u_i. \qquad \square$$

Met probleem (3.2) kunnen dus de lengten van de kortste paden worden bepaald. De routes worden er echter niet door gegeven. Het zal blijken dat deze te halen zijn uit het duale LP-probleem. Dit duale probleem luidt:

$$(3.3)\quad \min\left\{ \Sigma_k \Sigma_i\, l_{ki}\, x_{ki} \;\middle|\; \begin{array}{l} x_1 - \Sigma_j\, x_{1j} = 1 \\ \Sigma_j\, x_{ji} - \Sigma_j\, x_{ij} = 1 \quad i = 2,3,\ldots,n \\ x_{ki} \geq 0 \quad \text{voor alle } (k,i) \text{ met } (v_k,v_i) \in A \end{array} \right\}$$

Laat x nu een basisoplossing van (3.3) zijn, dan heeft x maximaal n positieve componenten. Uit de vergelijkingen volgt:

$$x_1 = 1 + \Sigma_j\, x_{1j} \geq 1, \text{ dus } x_1 \text{ is positief.}$$

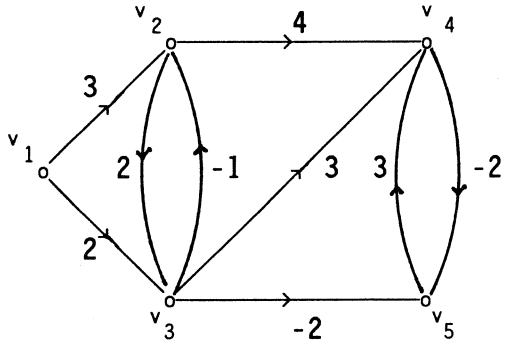
$$\Sigma_j\, x_{ji} = 1 + \Sigma_j\, x_{ij} \geq 1, \text{ dus } \Sigma_j\, x_{ji} \text{ is positief, } i = 2,3,\ldots n.$$

Maar dan is van de variabelen x_{ji}, $1 \leq j \leq n$, er precies één positief, zeg $x_{k(i)i} > 0$, $i = 2,3,\ldots,n$.
Beschouw nu de deelgraaf bestaande uit de pijlen $(v_{k(i)},v_i)$, $i = 2,3,\ldots,n$ en de bijbehorende knooppunten. Net zoals in het bewijs van 3.1 (ii) is in te zien dat dit een voortbrengende boom met wortel v_1 is.
Omgekeerd, zij $(v_{k(i)},v_i)$, $2 \leq i \leq n$, de pijlen van een voortbrengende boom met wortel v_1. In de matrix behorende bij de kolommen van x_1 en die van $x_{k(i)i}$, $1 \leq i \leq n$, staat in rij i alleen het getal +1 (verder slechts nullen) als v_i een eindpunt is van de boom.
Een lineaire combinatie van deze kolommen zódat dit de 0-vector oplevert heeft dus coëfficiënten 0 voor kolommen behorende bij eindpunten van de boom. Door nu deze eindpunten en de pijlen er naar toe weg te laten is inductief in te zien dat de kolommen van de matrix lineair onafhankelijk zijn, en dus behoren bij een basisoplossing, mits de oplossing niet-negatief is. Voor het niet-negativiteitsbewijs en een interpretatie van de waarden van de basisvariabelen verwijzen we naar Opgave 3.19.

Conclusie: Er is een één-éénduidig verband tussen de hoekpunten van het LP-probleem (3.3) en de voortbrengende bomen met wortel v_1.

Voorbeeld 3.4

Beschouw het hieronder getekende netwerk:



Probleem (3.2) luidt: $\max \left\{ \sum_{i=1}^{5} u_i \;\middle|\; \begin{array}{llll} u_1 = 0 & ; & u_4 - u_2 \le 4 \\ u_2 - u_1 \le 3; & & u_4 - u_3 \le 3 \\ u_2 - u_3 \le -1; & & u_4 - u_5 \le 3 \\ u_3 - u_1 \le 2; & & u_5 - u_3 \le -2 \\ u_3 - u_2 \le 2; & & u_5 - u_4 \le -2 \end{array} \right\}$

Het duale probleem (3.3) wordt:

$$\min \left\{ \begin{array}{l} 3x_{12} - x_{32} + \\ 2x_{13} + 2x_{23} + \\ 4x_{24} + 3x_{34} + \\ 3x_{54} - 2x_{35} - \\ 2x_{45} \end{array} \;\middle|\; \begin{array}{l} x_1 - x_{12} - x_{13} = 1; \\ x_{12} + x_{32} - x_{23} - x_{24} = 1; \\ -x_{32} + x_{13} + x_{23} - x_{34} - x_{35} = 1; \\ x_{24} + x_{34} + x_{54} - x_{45} = 1; \\ -x_{54} + x_{35} + x_{45} = 1; \end{array} \right\} \begin{array}{l} x_{ki} \ge 0 \\ \text{voor alle} \\ (k,i) \in A \end{array}$$

De voortbrengende boom $[v_1,v_2,v_3,v_4,v_5]$ correspondeert met de basisoplossing: $x_{45} = 1$, $x_{34} = 2$, $x_{23} = 3$, $x_{12} = 4$, $x_1 = 5$ en de overige variabelen 0.

De simplex methode werkt als volgt.
We starten met een basisoplossing x met basisvariabelen $x_{k(i)i} > 0$, $2 \le i \le n$.
Beschouw de daarmee corresponderende boom T, zeg $(v_{k(i)},v_i)$, $2 \le i \le n$.
Laten v_{ki} de verschilvariabelen zijn van het LP-probleem (3.2), d.w.z. de beperkingen zijn van de vorm:

$$u_i - u_k + v_{ki} = l_{ki} \text{ voor alle } (i,k).$$

Voor de met x corresponderende oplossing (u,v) van (3.3) gelden de orthogonaliteitsrelaties:

$$v_{k(i)i} = l_{k(i)i} + u_{k(i)} - u_i = 0 \Rightarrow u_i = u_{k(i)} + l_{k(i)i}, \; i = 2,3,\ldots,n,$$

d.w.z. u_i is de boomafstand in T van v_1 naar v_i, 2,3,...,n.

Uit deze waarden u_i kunnen de overige duale variabelen worden berekend, nl.:

$$v_{ki} = l_{ki} + u_k - u_i.$$

Als $v_{ij} \ge 0$ voor alle (i,j), dan is de boom T een boom van kortste paden.
Is dit niet het geval, dan moet volgens de simplexmethode een negatieve duale variabele, zeg v_{rp}, worden gekozen. Dit betekent dat in de boom de pijl $(v_{k(p)},v_p)$ wordt vervangen door (v_r,v_p). Voor deze nieuwe boom moeten weer de nieuwe duale u-variabelen, d.w.z. de lengten in het boompad, worden berekend.

Als we de pijl $(v_{k(p)},v_p)$ uit de boom T verwijderen, dan valt de boom uiteen in twee deelbomen: T_1 die v_1 bevat en T_2. De boomafstand in T_1 blijft onveranderd, we moeten alleen de boomafstand in T_2 herberekenen.

Stel de oude boomafstand van v_1 naar v_p is u_p en de nieuwe $\bar{u}_p$, dan is de verbetering van de afstand van v_1 naar v_p (en dus ook van alle afstanden van v_1 naar de knooppunten van T_2):

$$u_p - \bar{u}_p = u_p - (\bar{u}_r + l_{rp}) = u_p - u_r - l_{rp} = -v_{rp} > 0.$$

De methode is dus als volgt.

Stap 1 (start):

a. Kies een voortbrengende boom met wortel v_1 en bereken u_j = afstand v_1 tot v_j in de boom, $j = 1,2,\ldots,n$.
b. Bereken $v_{ij} = l_{ij} + u_i - u_j$ voor alle (i,j).

Stap 2 (optimaliteitscontrole):

Als $v_{ij} \geq 0$ voor alle (i,j): $u_j^* = u_j$, $j = 1,2,\ldots,n$ en de bijbehorende boom is de boom van kortste paden; STOP.

Stap 3 (nieuwe voortbrengende boom):

a. Kies (r,p) zodat $v_{rp} < 0$.
b. Bepaal T_1 en T_2, de deelbomen die ontstaan door de huidige pijl naar p te verwijderen.
c. Bereken de nieuwe boomafstand u_i voor de knooppunten $v_i \in T_2$:
   $u_i := u_i + v_{rp}$ voor alle i met $v_i \in T_2$.
d. Bereken $v_{ij} = l_{ij} + u_i - u_j$ voor alle (i,j).
e. Vervang in de boom de huidige pijl naar p door de pijl (v_r,v_p).
f. Ga naar stap 2.

Omdat $x_{k(i)i} > 0$ voor alle i, is er geen degeneratie en geeft deze methode dus na een eindig aantal iteraties een optimale oplossing. Om meer te kunnen zeggen over de complexiteit zullen we een speciale implementatie van bovenstaand algoritme bekijken.

In stap 1 nemen we als voortbrengende boom $\{(v_1,v_i) \mid i = 2,3,\ldots,n\}$.
De bepaling van een $v_{rp} < 0$ doen we als volgt. We berekenen de vectoren $v_{\cdot j} = (v_{1j},v_{2j},\ldots,v_{nj})$ cyclisch: als de laatste keer een negatieve waarde v_{ij} is gekozen gaan we bij de volgende stap verder bij de pijlen naar $v_{j+1},v_{j+2},\ldots$, d.w.z. met de vectoren $v_{\cdot j+1},v_{\cdot j+2},\ldots$ totdat er negatieve waarde bijzit. Stel dat dit het geval is voor $v_{\cdot p}$, dan kiezen we r zódat:

$$v_{rp} = \min\{v_{sp} \mid v_{sp} < 0\}.$$

Deze implementatie geeft het volgende algoritme.

Stap 1 (start):

a. Bereken $u_j = l_{1j}$ voor $j = 2,3,\ldots,n$.
b. Bereken $v_{ij} = l_{ij} + u_i - u_j$ voor $j = 2,3,\ldots$ totdat een vector $v_{\cdot j} = (v_{1j},v_{2j},\ldots,v_{nj})$ een negatieve component heeft, zeg voor $j = p$, òf totdat alle $v_{\cdot j}$, $2 \leq j \leq n$, niet-negatief blijken te zijn.

Stap 2 (optimaliteitscontrole):

Als $v_{ij} \geq 0$ voor alle (i,j): $u_j^* = u_j$, $j = 1,2,\ldots,n$ en de bijbehorende boom is de boom van kortste paden; STOP.

Stap 3 (nieuwe voortbrengende boom):

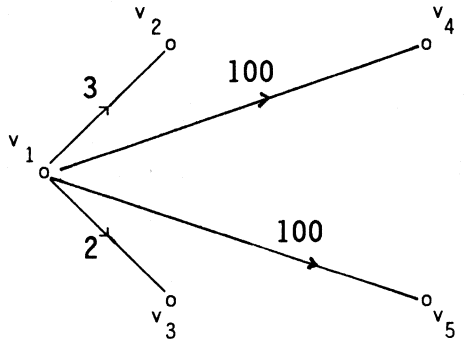
a. Kies r zodat $v_{rp} = \min_i v_{ip}$.
b. Bepaal T_1 en T_2, de deelbomen die ontstaan door de huidige pijl naar p te verwijderen.
c. Bereken de nieuwe boomafstand u_i voor de knooppunten $v_i \in T_2$:
   $u_i := u_i + v_{rp}$ voor alle i met $v_i \in T_2$.
d. Bereken $v_{ij} = l_{ij} + u_i - u_j$ voor $j = p+1,p+2,\ldots$ en voor $i = 1,2,\ldots,n$ totdat een vector $v_{\cdot j} = (v_{1j},v_{2j},\ldots,v_{nj})$ een negatieve component heeft, òf totdat alle $v_{\cdot p+1},v_{\cdot p+2},\ldots,v_{\cdot n},v_{\cdot 2},\ldots,v_{\cdot p}$ niet-negatief blijken te zijn.
e. Vervang in de boom de huidige pijl naar p door de pijl (v_r,v_p).
f. Ga naar stap 2.

Passen we deze implementatie toe op voorbeeld 3.1, dan krijgen we:

Voorbeeld 3.4 (vervolg)

Voor pijlen die er niet zijn nemen we de lengte 100.

Iteratie 1



$u = (0,3,2,100,100);$

$v_{.2} = (0,0,-2,197,197);$

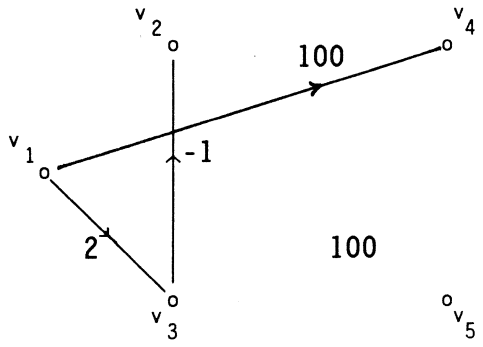
$p = 2;\ r = 3;\ v_{rp} = -2;\ T_2 = \{2\};$

$u = (0,1,2,100,100);$

$v_{.3} = (0,1,0,198,198);$

$v_{.4} = (0,-95,-95,0,3);$

Iteratie 2

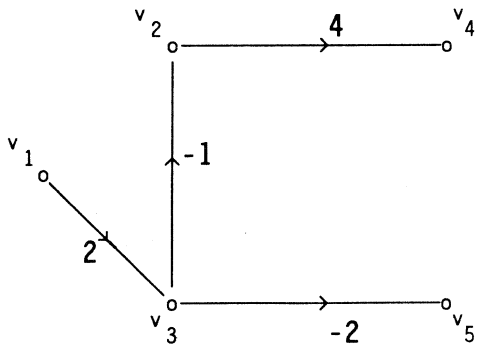


$p = 4;\ r = 2;\ v_{rp} = -95;\ T_2 = \{4\};$

$u = (0,1,2,5,100);$

$v_{.5} = (0,3,-100,-97,0);$

Iteratie 3



$p = 5;\ r = 3;\ v_{rp} = -100;\ T_2 = \{5\};$

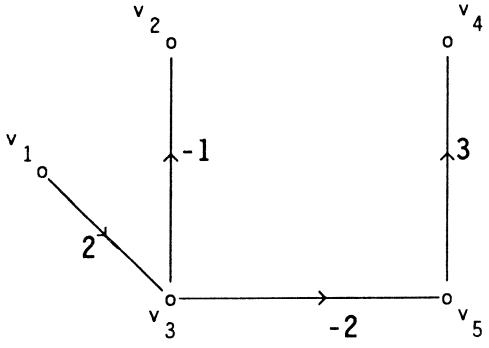
$u = (0,1,2,5,0);$

$v_{.2} = (2,0,0,104,99)$

$v_{.3} = (0,1,0,103,98)$

$v_{.4} = (95,0,0,0,-2)$

Iteratie 4



$p = 4;\ r = 5;\ v_{rp} = -2;\ T_2 = \{4\};$

$u = (0,1,2,3,0);$

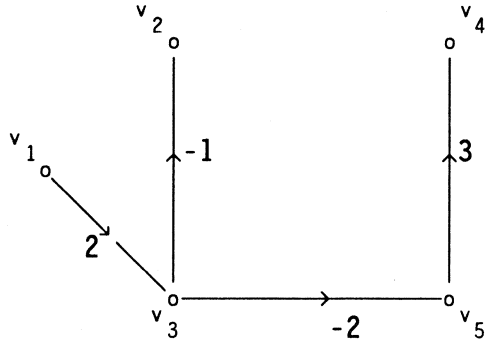
$v_{.5} = (100,101,0,1,0)$

$v_{.2} = (2,0,0,102,99)$

$v_{.3} = (0,1,0,101,98)$

$v_{.4} = (97,2,2,0,0)$

Iteratie 5



alle $v_{ij} \geq 0$: nevenstaande boom is de boom van kortste paden en $u_1 = 0$; $u_2 = 1$; $u_3 = 2$; $u_4 = 3$; $u_5 = 0$.

We zullen nu nader ingaan op de complexiteit van het algoritme.
Laat T^* de boom van kortste paden zijn, waarmee het algoritme eindigt.

Lemma 3.1
Zij T een voortbrengende boom met boomafstanden u_i, $1 \leq i \leq n$.
Als $u_r = u_r^*$, $u_p > u_p^*$ en $(v_r,v_p) \in T^*$, dan is $v_{rp} = \min_i v_{ip} < 0$.
Bewijs
$v_{rp} - v_{ip} = [l_{rp} + u_r^* - u_p] - [l_{ip} + u_i - u_p] = u_p^* - [l_{ip} + u_i] \leq 0.$
$v_{rp} = l_{rp} + u_r - u_p = l_{rp} + u_r^* - u_p = u_p^* - u_p < 0.$ □

Bij aanvang van de eerste iteratie (wanneer T bestaat uit de pijlen (v_1,v_i), $2 \leq i \leq n$) hebben de knooppunten die in T^* ook via één pijl vanuit v_1 bereikbaar zijn reeds de goede boomafstand.
Volgens Lemma 3.1 krijgt in een de volgende iteraties, nl. zodra de desbetreffende $v_{\cdot p}$ wordt bekeken, een knooppunt v_p dat in T^* vanuit v_1 met 2 pijlen bereikt wordt de goede boomafstand u_p^*. Dit gaat zo verder.
De iteraties kunnen dus in stadia worden ingedeeld: stadium k begint met de eerste iteratie waarin alle knooppunten die in T^* met k pijlen bereikbaar zijn de goede boomafstand u^* hebben; de laatste iteratie van stadium k is die iteratie aan het eind waarvan voor de eerste keer alle knooppunten v_i die in T^* met k+1 pijlen bereikbaar zijn de goede boomafstand u_i^* hebben gekregen.

In voorbeeld 3.4 bestaat stadium 1 uit de iteraties 1, 2 en 3, stadium 2 uit iteratie 4 (iteratie 5 bevestigt de optimaliteit).

Stelling 3.6

(i) Het algoritme heeft hoogstens (n-1)(n-2)/2 iteraties.

(ii) De complexiteit van het algoritme is $\mathcal{O}(n^3)$.

Bewijs

(i) Beschouw stadium k. Minstens k+1 knooppunten hebben de goede u^*-waarde, dus na hoogstens n-k-1 iteraties hebben alle knooppunten die in T^* met k+1 pijlen worden bereikt de goede u^*-waarde. Het maximum aantal iteraties is dus: $\Sigma_{k=1}^{n-2}$ (n-k-1) = (n-1)(n-2)/2. □

(ii) Stap 1 wordt éénmaal uitgevoerd en heeft complexiteit $\mathcal{O}(n^2)$.
Er zijn $\mathcal{O}(n^2)$ iteraties en, stap 3d uitgezonderd, de hoeveelheid werk is per iteratie $\mathcal{O}(n)$.
Beschouw tenslotte stap 3d over alle iteraties genomen. In ieder stadium wordt een $v_{\cdot i}$ hoogstens éénmaal berekend, wat per $v_{\cdot i}$ $\mathcal{O}(n)$ vereist, per stadium dus $\mathcal{O}(n^2)$. Omdat er maximaal n-2 stadia zijn is de totale hoeveelheid werk van stap 3d $\mathcal{O}(n^3)$. □

Opgave 3.18

Bepaal met de simplex methode de (lengte van de) kortste paden in het netwerk van opgave 3.10.

Opgave 3.19

Laat x een basisoplossing zijn van het LP-probleem (3.3) met corresponderende voortbrengende boom T.
Als x_{ki} een basisvariabele is, dan is de waarde ervan gelijk aan het aantal in T vanuit v_i bereikbare knooppunten, inclusief v_i zelf.
Toon dit aan.

Opgave 3.20

Veronderstel dat van te voren niet bekend is of het netwerk een ronde van negatieve lengte bevat.

a. Veronderstel dat tijdens het simplex algoritme de pijl (v_i,v_p) wordt vervangen door de pijl (v_r,v_p).
Als blijkt dat $v_r \in T_2$, dan bevat het netwerk een ronde van negatieve lengte. Toon dit aan.

b. Pas het algoritme aan zodat ook ronden met een negatieve lengte ontdekt kunnen worden.
c. Gebruik het aangepaste algoritme om eventueel een ronde van negatieve lengte te vinden in het netwerk van opgave 3.10 met $l_{62} = -1$ en de andere lengten ongewijzigd (zie ook opgave 3.11).

Opgave 3.21

Laat T' een op T volgende boom zijn tijdens het simplex algoritme, waarbij de pijl (v_i,v_p) van T wordt vervangen door de pijl (v_r,v_p) van T'.
Als voor de bij T' behorende verschilvariabelen v geldt dat $\min_j\{v_{jp}\} < 0$, dan bevat het netwerk een negatieve ronde. Toon dit aan.

Opgave 3.22*

Zij G = (V,A) de graaf met n knooppunten en met een pijl van i naar j d.e.s.d. als i > j. De lengte $l_{ij} = -i$ voor alle $(i,j) \in A$.
Beschouw de kortste paden vanuit wortel v_n.
a. Voer de simplex methode uit voor n = 5.
b. Toon voor algemene n het volgende aan:
   (i) Stadium k vereist n-k-1 pivot stappen, waarin de pijlen (n-k,j), voor j = 1,2,...,n-k-1 aan de boom worden toegevoegd.
   (ii) Aan het eind van het k-de stadium bestaat de voortbrengende boom uit de pijlenverz. {(n,n-1), (n-1,n-2),...,(n-k+1,n-k), (n-k,1), (n-k,2), ...,(n-k,n-k-1)}.
   (iii) De simplex methode gebruikt precies (n-1)(n-2)/2 iteraties.

INHOUDSOPGAVE Pagina

DEEL 1

INHOUDSOPGAVE Pagina

DEEL 2

IV. STROMEN IN NETWERKEN

1. Inleiding

Literatuur

* Hoofdstuk 9 in:
  M.S.Bazaraa and J.J.Jarvis: "Linear programming and network flows", Wiley, New York (1977).
* Hoofdstuk 4 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds", Holt, Rinehart and Winston, New York (1976).
* Hoofdstuk 3 in:
  J.F.Shapiro: "Mathematical programming: structures and algorithms", Wiley, New York (1979).

Zij $G = (V, A \cup (n,1))$ een samenhangende graaf. De pijl (v_n,v_1) is een speciale pijl, waarvan de betekenis na de volgende definitie zal worden verklaard. Een stroom is een functie $x: A \cup (n,1) \longrightarrow \mathbb{Z}$ zódanig dat

(4.1) $\quad \Sigma_j x_{ij} - \Sigma_j x_{ji} = 0 \quad$ voor $i = 1,2,\ldots,n,$

d.w.z. een geheeltallige functie op de pijlen van G die voor ieder knooppunt evenveel stroom invoert als uitvoert. Vandaar dat we ook wel spreken over circulatiestroom. De pijl (v_n,v_1) wordt gebruikt om wat in v_n aankomt weer naar v_1 te sturen. We willen namelijk zo veel mogelijk stroom van v_1 naar v_n brengen. x_{n1} heet de waarde van de stroom.
Bovendien zijn aan de pijlen $(i,j) \in A$ positieve, gehele getallen b_{ij}, capaciteiten geheten, toegekend ($b_{ij} = \infty$ is toegestaan).
Een toelaatbare stroom is een stroom x waarvoor geldt:

(4.2) $\quad 0 \leq x_{ij} \leq b_{ij} \quad$ voor alle $(i,j) \in A.$

Het maximale stroom probleem is het probleem om een stroom te vinden met maximale waarde.

Een (1,n)-snede is een partitie (W,V-W) van V zódanig dat $v_1 \in W$ en $v_n \notin W$. De capaciteit c(W) van de (1,n)-snede (W,V-W) is gedefiniëerd door:

(4.3) $\quad c(W) = \Sigma_{i \in W} \Sigma_{j \notin W} b_{ij}.$

Het minimale snede probleem is het probleem van het vinden van een snede met minimale capaciteit. Het maximale stroom probleem en het minimale snede probleem zijn sterk verwant, ze zijn in feite elkaars duale.

Het maximale stroomprobleem kan als volgt als LP-probleem worden geformuleerd:

$$(4.4) \qquad \max \left\{ x_{n1} \;\middle|\; \begin{array}{ll} \Sigma_j x_{ij} - \Sigma_j x_{ji} = 0 & \text{voor } i = 1,2,\ldots,n \\ 0 \le x_{ij} \le b_{ij} & \text{voor alle } (i,j) \in A \end{array} \right\}.$$

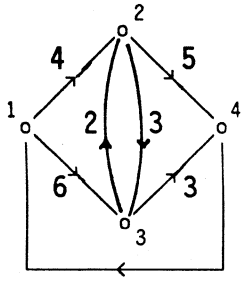
De geheeltalligheid van x wordt in (4.4) niet expliciet geëist. We zullen aantonen dat in ieder hoekpunt (dus ook in het optimale) van (4.4) de oplossing x geheeltallig is.

Het duale probleem van (4.4) luidt:

$$(4.5) \qquad \min \left\{ \Sigma_i \Sigma_j b_{ij} w_{ij} \;\middle|\; \begin{array}{ll} u_i - u_j + w_{ij} \ge 0 & \text{voor alle } (i,j) \in A \\ -u_1 + u_n = 1 & \\ w_{ij} \ge 0 & \text{voor alle } (i,j) \in A \end{array} \right\}.$$

Voorbeeld 4.1

Beschouw nevenstaand netwerk.



Het bijbehorende LP-probleem is:

$$\max \left\{ x_{43} \;\middle|\; \begin{array}{l} x_{12} + x_{13} - x_{41} = 0 \\ -x_{12} - x_{32} + x_{23} + x_{24} = 0 \\ -x_{13} + x_{32} - x_{23} + x_{34} = 0 \\ -x_{24} - x_{34} + x_{41} = 0 \\ 0 \le x_{12} \le 4;\ 0 \le x_{13} \le 6;\ 0 \le x_{32} \le 2 \\ 0 \le x_{23} \le 3;\ 0 \le x_{24} \le 5;\ 0 \le x_{34} \le 3 \end{array} \right\}$$

Het duale hiervan is:

$\min[4w_{12} + 6w_{13} + 2w_{32} + 3w_{23} + 5w_{24} + 3w_{34}]$ onder de voorwaarden

$$\begin{array}{rrrrrrrrrrrrl}
u_1 & - & u_2 & & & + & w_{12} & & & & & & \ge 0 \\
u_1 & & & - & u_3 & & & + & w_{13} & & & & \ge 0 \\
 & - & u_2 & + & u_3 & & & & & + & w_{32} & & \ge 0 \\
 & & u_2 & - & u_3 & & & & & & & + w_{23} & \ge 0 \\
 & & u_2 & & & - & u_4 & & & & & + w_{24} & \ge 0 \\
 & & & & u_3 & - & u_4 & & & & & + w_{34} & \ge 0 \\
-u_1 & & & & & + & u_4 & & & & & & = 1 \\
 & & & & & & & w_{12}, & w_{13}, & w_{32}, & w_{23}, & w_{24}, & w_{34} \ge 0
\end{array}$$

Neem een willekeurige (1,n)-snede, zeg (W,V-W). Definiëer:

$$(4.6)\quad u_i = \begin{cases} 0 \text{ als } i \in W \\ 1 \text{ als } i \notin W \end{cases} \qquad w_{ij} = \begin{cases} 1 \text{ als } i \in W,\ j \notin W \\ 0 \text{ anders} \end{cases}$$

Deze (u,w) is toelaatbaar voor (4.5) en heeft als waarde van de doelfunctie $\Sigma_{i\in W}\ \Sigma_{j\notin W}\ b_{ij} = c(W)$. Iedere snede heeft dus een capaciteit die minstens zo groot is als het minimum van (4.5).

Zij x een toelaatbare stroom en (W,V-W) een (1,n)-snede. Door de gelijkheden van (4.4) op te tellen voor $i \in W$ krijgen we (de variabele x_{n1} die alleen in de eerste rij voorkomt nemen we apart):

$$\begin{aligned} x_{n1} &= \Sigma_{i\in W}[\Sigma_j\ x_{ij} - \Sigma_j\ x_{ji}] \\ &= \text{stroom die W uitgaat minus stroom die W inkomt} \\ (4.7)\qquad &= \Sigma_{i\in W}\ \Sigma_{j\notin W}\ [x_{ij} - x_{ji}]. \end{aligned}$$

<u>Stelling 4.1</u>

Zij x een toelaatbare oplossing van (4.4) en (W,V-W) een (1,n)-snede, dan geldt:

(i) $c(W) \geq x_{n1}$.

(ii) $c(W) = x_{n1} \Leftrightarrow x_{ij} = \begin{cases} b_{ij} \text{ als } i \in W,\ j \notin W \\ 0 \quad \text{als } i \notin W,\ j \in W \end{cases} \Rightarrow \begin{cases} x \text{ maximale stroom} \\ (W,V\text{-}W) \text{ minimale snede} \end{cases}$

<u>Bewijs</u>

Volgens (4.7) geldt: $x_{n1} = \Sigma_{i\in W}\ \Sigma_{j\notin W}\ [x_{ij} - x_{ji}]$

(i) $x_{n1} = \Sigma_{i\in W}\ \Sigma_{j\notin W}\ [x_{ij} - x_{ji}] \leq \Sigma_{i\in W}\ \Sigma_{j\notin W}\ [b_{ij} - 0] = \Sigma_{i\in W}\ \Sigma_{j\notin W}\ b_{ij} = c(W)$.

(ii) $x_{n1} = c(W) \Leftrightarrow x_{ij} = b_{ij}$ en $x_{ji} = 0$ voor alle $i \in W$ en $j \notin W$

Uit (i) volgt: als $x_{n1} = c(W)$, dan is x maximaal en W minimaal. □

Laat x een toelaatbare stroom zijn, en zij P een keten vanuit v_1 die gebruik maakt van de pijlen van A, maar deze pijlen ook in tegengestelde richting mag doorlopen. Een pijl die doorlopen wordt in de "goede" richting heet een <u>voorwaartse pijl</u>, wordt de pijl in tegengestelde richting doorlopen dan heet deze <u>achterwaarts</u>. De keten P heet een <u>groeiketen</u> als $x_{ij} < b_{ij}$ voor alle voorwaartse pijlen (i,j) en als $x_{ij} > 0$ voor alle achterwaartse pijlen.

Indien er een groeiketen P van v_1 naar v_n bestaat, dan kan de waarde van de stroom worden vergroot door de waarde over de pijlen van P als volgt te veranderen.

$\Delta_1 := \min \{b_{ij} - x_{ij} \mid (i,j) \in P \text{ als voorwaartse pijl}\};$

$\Delta_2 := \min \{x_{ij} \mid (j,i) \in P \text{ als achterwaartse pijl}\};$

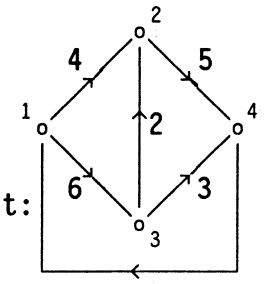
$\Delta := \min (\Delta_1, \Delta_2).$

$$x_{ij} := \begin{cases} x_{ij} + \Delta & \text{voor alle } (i,j) \in P \text{ als voorwaartse pijl} \\ x_{ij} - \Delta & \text{voor alle } (j,i) \in P \text{ als achterwaartse pijl} \\ x_{ij} & \text{anders} \end{cases}$$

Deze nieuwe waarde van met $x_{n1} := x_{n1} + \Delta$ is een toelaatbare stroom met een waarde die Δ hoger is dan de vorige.

Voorbeeld 4.2

Beschouw het netwerk van Voorbeeld 4.1 zonder de pijl (v_2,v_3). Laat $x_{12} = 2$, $x_{13} = 4$, $x_{32} = 2$, $x_{24} = 4$, $x_{34} = 2$ en $x_{41} = 6$. Neem voor P de keten $[v_1,v_2,v_3,v_4]$, met (v_2,v_3) als achterwaartse pijl, d.w.z. dat we de pijl (v_3,v_2) tegen de richting van de pijl inlopen. P is inderdaad een groeiketen, en er geldt: $\Delta_1 = \min \{4-2,3-2\} = 1$; $\Delta_2 = \min \{2\} = 2$; $\Delta = \min\{1,2\} = 1$.



De waarde van de stroom kan met 1 worden verhoogd. Deze stroom wordt: $x_{12} = 3$, $x_{13} = 4$, $x_{32} = 1$, $x_{24} = 4$, $x_{34} = 3$ en $x_{41} = 7$.

Stelling 4.2 *(Groeiketen-stelling)*

Een toelaatbare stroom x is maximaal d.e.s.d. als er geen groeiketen van v_1 naar v_n bestaat.

Bewijs

We hebben reeds gezien dat als er een groeiketen bestaat de waarde van de stroom kan worden vermeerderd. Resteert nog te bewijzen dat als er geen groeiketen is de stroom maximaal is.

Laat $W = \{v \in V \mid v \text{ vanuit } v_1 \text{ bereikbaar via een groeiketen}\}$. Dan is $x_{ij} = b_{ij}$ en $x_{ji} = 0$ als als $i \in W$ en $j \notin W$. Uit Stelling 4.1 volgt dat x maximaal is, en dat $x_{n1} = c(W)$, d.w.z. dat W minimaal is. □

Gevolg *(Maximale stroom = minimale snede resultaat)*
De waarde van het maximale stroom probleem is gelijk aan de waarde van het minimale snede probleem.

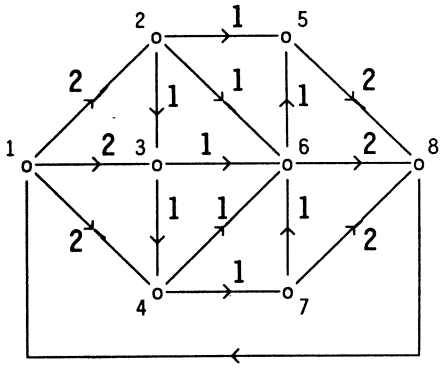
Laat y_{ij} de verschilvariabelen zijn van de beperkingen $x_{ij} \leq b_{ij}$, $(i,j) \in A$, d.w.z. we schrijven het LP-probleem als:

$$(4.8) \qquad \max \left\{ x_{n1} \;\middle|\; \begin{array}{rl} \Sigma_j x_{ij} - \Sigma_j x_{ji} = 0 & \text{voor } i = 1,2,\ldots,n \\ x_{ij} + y_{ij} = b_{ij} & \text{voor alle } (i,j) \in A \\ x_{ij}, y_{ij} \geq 0 & \text{voor alle } (i,j) \in A \end{array} \right\}.$$

Stelling 4.3 *(Geheeltalligheidsstelling)*
Het maximale stroom probleem (4.8) heeft een geheeltallige oplossing.
Bewijs
De beperkingen van (4.6) zijn van de vorm $\begin{pmatrix} D & 0 \\ I_m^* & I_m \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ b \end{pmatrix}$, waarbij D de incidentiematrix van de graaf G is en I_m^* een $m \times (m+1)$ matrix is (m is het aantal pijlen van A), namelijk $I_m^* = (I_m \; 0)$, met I_m de eenheidsmatrix van dimensie m.
Omdat D totaal unimodulair is (Stelling 1.4), is ook $\begin{pmatrix} D & 0 \\ I_m^* & I_m \end{pmatrix}$ totaal unimodulair. Iedere basismatrix heeft dus determinantwaarde ± 1. Uit de regel van Cramer volgt nu dat iedere basisoplossing eveneens geheeltallig is. Omdat er een optimale basisoplossing bestaat, is er een optimale geheeltallige oplossing. □

Opgave 4.1

Beschouw nevenstaand netwerk met de bij de pijlen aangegeven capaciteiten.



Toon aan dat de volgende stroom een toegelaten en maximale stroom is:

$x_{12} = 2$; $x_{13} = 2$; $x_{14} = 1$; $x_{23} = 0$; $x_{25} = 1$;
$x_{26} = 1$; $x_{34} = 1$; $x_{36} = 1$; $x_{46} = 1$; $x_{47} = 1$;
$x_{58} = 2$; $x_{65} = 1$; $x_{68} = 2$; $x_{76} = 0$; $x_{78} = 1$;
$x_{81} = 5$.

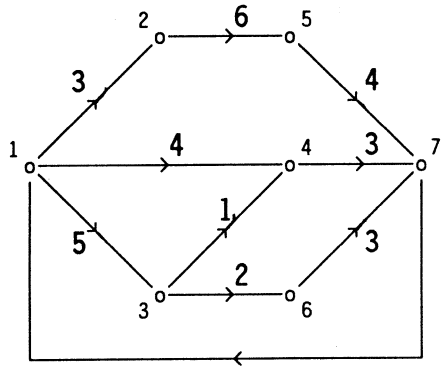
Opgave 4.2

Beschouw een netwerk, waarin behalve aan de pijlen ook aan de knooppunten capaciteitsbeperkingen zijn opgelegd, zeg $\Sigma_j x_{ji} \leq c_i$, $1 \leq i \leq n$, d.w.z. dat de stroom die in knooppunt i binnenkomt (en er dus ook weer uitgaat) maximaal c_i is.

Hoe is dit netwerk te modelleren als een "gewoon" netwerk, met alleen capaciteiten op de pijlen?

Opgave 4.3

Beschouw het hiernaast getekende netwerk (de getallen bij de pijlen zijn de capaciteiten).



a. Formuleer het LP-probleem om de maximale stroom te vinden.
b. Stel het duale LP-probleem op.
c. Toon aan dat de (1,7)-snede (W,V-W) met W = {1,3,4} een snede met minimale capaciteit is.

Opgave 4.4

Ga na of de volgende uitspraken juist zijn:

a. Als alle capaciteiten verschillend zijn, dan is de minimale (1,n)-snede uniek.
b. Als alle capaciteiten verschillend zijn, dan is de maximale stroom uniek.

Opgave 4.5

Noem een pijl in een netwerk "meest vitaal" als door het verwijderen van deze pijl de waarde van de maximale stroom minstens zoveel daalt als bij het verwijderen van een willekeurig andere pijl.
Ga na of de volgende uitspraken juist of onjuist zijn:

a. Een meest vitale pijl zit in een minimale (1,n)-snede.
b. Indien een meest vitale pijl in een minimale (1,n)-snede zit, dan is het daarin een pijl met de grootste capaciteit.

Opgave 4.6

Toon het volgende aan:

Als $(W_1, V-W_1)$ en $(W_2, V-W_2)$ beide minimale (1,n)-sneden zijn, dan zijn $(W_1 \cup W_2,\ V - [W_1 \cup W_2])$ en $(W_1 \cap W_2,\ V - [W_1 \cap W_2])$ ook minimale (1,n)-sneden

Hint: Toon aan dat $c(W_1 \cup W_2) + c(W_1 \cap W_2) \leq c(W_1) + c(W_2)$.

2. Simplex methode

Literatuur

* Hoofdstuk 9 in:
  M.S.Bazaraa and J.J.Jarvis: "Linear programming and netwerk flows", Wiley, New York (1977).
* Hoofdstuk 3 in:
  J.F.Shapiro: "Mathematical programming: structures and algorithms", Wiley, New York (1979).

Bij de simplex methode gaan we van hoekpunt (basisoplossing) naar hoekpunt totdat het optimum bereikt is. We zullen daarom eerst de basisoplossingen van (4.8) gaan karakteriseren.

Beschouw een basisoplossing van (4.8). Het eerste deel van de matrix van de beperkingen is de incidentiematrix D van het netwerk. Omdat rang(D) = n-1 (Stelling 1.7) kunnen we de eerste rij van de beperkingen wel weglaten. Voor het duale probleem betekent dit dat de bijbehorende variabele $u_1 = 0$, wat impliceert dat $u_n = 1$.
Een basisoplossing van (4.8) bevat dan n+m-1 variabelen, daarnaast zijn er m-n+2 niet-basis variabelen (in totaal zijn er 2m+1 variabelen: m x_{ij}'s, m y_{ij}'s en de variabele x_{n1}). Omdat $x_{ij} + y_{ij} = b_{ij} \geq 1$, kunnen x_{ij} en y_{ij} niet beide tegelijk niet-basis variabelen zijn.

Lemma 4.1
Zij (x,y) een basisoplossing van (4.8).
Laat $T = \{\{(i,j) \in A \mid x_{ij} \text{ èn } y_{ij} \text{ in de basis}\} \cup (n,1)\}$.
Dan is T een voortbrengende boom van G.

Bewijs
Omdat er precies m-n+2 niet-basis variabelen zijn en x_{ij} en y_{ij} niet beide niet-basis variabelen kunnen zijn bevat $\{(i,j) \in A \mid x_{ij}$ en y_{ij} in de basis$\}$ m - [m-n+2] = n-2 elementen. De bij deze variabelen behorende kolommen zijn lineair onafhankelijk. Trek van de kolom van de x_{ij} de kolom van y_{ij} af: dit geeft een vector met in de rijen behorende bij $x_{ij} + y_{ij} = b_{ij}$, $(i,j) \in A$, slechts nullen. Deze vectoren tezamen met die van x_{n1} geven n-1 vectoren. Omdat x_{n1} ook in de basis zit (een onbeperkte variabele zit altijd in de basis) zijn dit n-1 onafhankelijke vectoren van de incidentiematrix D. Volgens Stelling 1.7 heeft T dan één component, d.w.z. T is een boom. □

Lemma 4.2

Indien T een voortbrengende boom van G is die (n,1) bevat, en x toelaatbaar is voor (4.4) en aan de voorwaarde voldoet dat $x_{ij} = 0$ òf b_{ij} als $(i,j) \notin T$, dan is (x,y), met $y_{ij} = b_{ij} - x_{ij}$, een basisoplossing van (4.8).

Bewijs

Omdat de toelaatbaarheid gegarandeerd is, is het voldoende om te bewijzen dat de bij positieve variabelen behorende kolommen lineair onafhankelijk zijn. Beschouw eerst alleen kolommen van de x-variabelen. Daar in (4.8) onder de incidentiematrix D een eenheidsmatrix staat zijn deze kolommen onafhankelijk. Bekijk nu de kolom van een y_{ij}-variabele. Deze kan alleen afhankelijk van de andere kolommen zijn als dit "via de kolom van x_{ij}" gaat. Maar in dat geval zou het eerste deel van de x'en afhankelijk zijn. Indien echter zowel x_{ij} als y_{ij} positief is, zit (i,j) in T. Het eerste deel van de kolommen van T is evenwel onafhankelijk, omdat T een boom is en geen kringen bevat. □

We zullen nu het verband tussen het maximale stroom probleem en het minimale snede probleem via de lineaire programmering uitwerken. In Stelling 4.3 hebben we gezien dat de matrix van de beperkingen van (4.4) totaal unimodulair is. De gespiegelde matrix, d.i. de matrix van (4.5), is dan eveneens totaal unimodulair. Hieruit volgt dat een basisoplossing van (4.5) ook geheeltallig is.

Laat (x,y) een basisoplossing zijn van (4.8) en (u,w) de bijbehorende duale oplossing. Dan gelden de volgende orthogonaliteitsrelaties:

(4.9) x_{ij} in basis $\Rightarrow u_i - u_j + w_{ij} = 0$

(4.10) y_{ij} in basis $\Rightarrow w_{ij} = 0$

(4.11) x_{ij} niet in basis $\Rightarrow y_{ij} = b_{ij}$ en $x_{ij} = 0$

(4.12) y_{ij} niet in basis $\Rightarrow x_{ij} = b_{ij}$ en $y_{ij} = 0$

Bij de basisoplossing (x,y) behoort volgens Lemma 4.1 een voortbrengende boom die de tak (n,1) bevat. Indien we uit deze boom de tak (n,1) weglaten ontstaan twee deelbomen W en V-W, waarbij we aannemen dat W de deelboom is die v_1 bevat. De volgende stelling laat zien wat de kandidaten zijn om in de basis te komen, d.w.z. dat de bijbehorende duale variabelen w_{ij} of $u_i - u_j + w_{ij}$ negatief zijn. Uit de theorie van de lineaire programmering is bekend dat de oplossingen van de LP-problemen optimaal zijn zodra de bijbehorende duale variabelen niet-negatief zijn.

Stelling 4.4

Zij (x,y) een basisoplossing van (4.8) met correponderende duale oplossing (u,w) van (4.5), dan geldt:

(i) $w_{ij} < 0 \Leftrightarrow i \notin W,\ j \in W$ en $x_{ij} = b_{ij}$.

(ii) $u_i - u_j + w_{ij} < 0 \Leftrightarrow i \in W,\ j \notin W$ en $x_{ij} = 0$.

(iii) $$\begin{cases} w_{ij} \geq 0 \text{ voor alle } (i,j) \\ u_i - u_j + w_{ij} \geq 0 \text{ voor alle } (i,j) \end{cases} \Leftrightarrow x_{ij} = \begin{cases} b_{ij}, & i \in W \text{ en } j \notin W \\ 0, & i \notin W \text{ en } j \in W \end{cases}$$

Bewijs

De bij de basisoplossing behorende voortbrengende boom bestaat uit (n,1) en de takken (i,j) waarvoor x_{ij} èn y_{ij} in de basis zitten. Voor deze takken (i,j) geldt dus volgens (4.9) en (4.10) dat $u_i = u_j$. Omdat $u_1 = 0$ en $u_n = 1$, hebben de knooppunten van W u-waarde 0, en de andere u-waarde 1.

(i) Stel $w_{ij} < 0$, dan zit y_{ij} niet in de basis: $x_{ij} = b_{ij}$ en x_{ij} in de basis. Hieruit volgt (4.9) $w_{ij} = u_j - u_i = -1$: $j \in W,\ i \notin W$.
Omgekeerd, stel $i \notin W,\ j \in W,\ x_{ij} = b_{ij}$: $w_{ij} = u_j - u_i = -1$.

(ii) Stel $u_i - u_j + w_{ij} < 0$: $u_i = w_{ij} = 0$, $u_j = 1$ en volgens (4.9) zit x_{ij} niet in de basis, d.w.z. $x_{ij} = 0$.
Omgekeerd, stel $i \in W,\ j \notin W$ en $x_{ij} = 0$: $u_i = 0$, $u_j = 1$ en y_{ij} in basis. Volgens (4.10) is $w_{ij} = 0$, d.w.z. $u_i - u_j + w_{ij} = -1$.

(iii) Beide voorwaarden zijn equivalent met optimaliteit van x voor (4.4) en (u,w) voor (4.5). □

We zullen nu de simplex methode toegepast op het maximale stroom probleem gaan bespreken. Bij een hoekpunt behoort een voortbrengende boom T en duale variabelen (u,w). De boom bestaat uit (n,1) en de takken (i,j) waarvoor x_{ij} èn y_{ij} in de basis zitten. Indien (n,1) uit T wordt verwijderd valt de boom uiteen in twee stukken: W met v_1 en V-W met v_n. De negatieve duale variabelen (dit zijn precies de kandidaten om in de basis te komen) zijn direct uit x en W te bepalen volgens Stelling 4.4. Als alle duale variabelen niet-negatief zijn, dan is x een maximale stroom en (W,V-W) een (1,n)-snede met minimale capaciteit.

Een iteratiestap verloopt globaal als volgt (enkele onderdelen worden later verduidelijkt).

1. Kies een $(r,p) \in A$ waarvan een duale variale negatief is, d.w.z. een element van $\{(i,j) \in A \mid x_{ij} = 0,\ i \in W,\ j \notin W;\ x_{ij} = b_{ij},\ i \notin W,\ j \in W\}$.

2.a. Voeg (r,p) aan T toe. Dit geeft een keten P van v_1 naar v_n, die tezamen met de pijl $(n,1)$ een kring C oplevert, die (r,p) en $(n,1)$ bevat.

b. Bepaal de stroom Δ die maximaal over P kan worden vervoerd en een "blokkerende" tak (k,l) op de keten P.

c. $x := x + \Delta$; $T := T + (r,p) - (k,l)$.

In de keuze van (r,p) waarvoor een duale variabele w_{rp} of $u_r - u_p + w_{rp}$ negatief is zijn we vrij. Deze keuze is wel van invloed op de complexiteit. Recentelijk is aangetoond[1] dat de keuze (onder de mogelijke kandidaten) van de tak die op P het dichtst bij v_1 ligt een $O(n^2m)$ algoritme oplevert.

Keten P kan zowel voorwaartse als achterwaartse pijlen bevatten. De hoeveelheid Δ die over P vervoerd kan worden wordt op dezelfde manier berekend als de Δ in paragraaf 1: $\Delta := \min(\Delta_1, \Delta_2)$, waarbij:

$\Delta_1 := \min \{b_{ij} - x_{ij} \mid (i,j) \in P$ als voorwaartse pijl$\}$;

$\Delta_2 := \min \{x_{ij} \mid (j,i) \in P$ als achterwaartse pijl$\}$.

P hoeft geen groeiketen te zijn: het is mogelijk dat $\Delta = 0$. In dat geval is de iteratiestap gedegenereerd, en gaat de waarde van de stroom niet omhoog.

Een blokkerende tak is een tak waarvoor de hoeveelheid die er over vervoerd gaat worden Δ bepaalt: een voorwaartse tak $(k,l) \in P$ als $\Delta = \Delta_1 = b_{kl} - x_{kl}$ of een achterwaartse tak $(k,l) \in P$ met $\Delta = \Delta_2 = x_{lk}$. Er is altijd minstens één blokkerende tak. zijn er meer, dan kiezen we er een willekeurig. Het is mogelijk dat $(k,l) = (r,p)$. De tak (k,l) die de boom verlaat heeft stroom b_{kl} of 0. Volgens Lemma 4.2 is de nieuwe oplossing (x,y) weer een basisoplossing.

Behalve de iteratiestap moeten we ook het starthoekpunt vastleggen. We zullen starten met $x = 0$ (is toegelaten stroom) en een willekeurige voortbrengende boom T die $(n,1)$ bevat. De basisvariabelen zijn dan alle y-variabelen en de x_{ij} variabelen voor $(i,j) \in T$.

1) D.Goldfarb and J.Hao: "A primal simplex algorithm that solves the maximum flow problem in at most nm pivots and $O(n^2m)$ time", Mathematical Programming 47 (1990) pp. 353-366.

Resumerend is de simplex methode voor het maximale stroom probleem als volgt.

Stap 1 (start)

Kies een willekeurige voortbrengende boom die (n,1) bevat; x := 0.

Stap 2 (optimaliteitscontrole)

a. Bepaal de knooppunten W van de deelboom die v_1 bevat en die ontstaat door (n,1) uit T weg te laten.

b. Bepaal: $I_1 := \{(i,j) \in A \mid i \in W,\ j \notin W \text{ en } x_{ij} = 0\}$;
$I_2 := \{(i,j) \in A \mid i \notin W,\ j \in W \text{ en } x_{ij} = b_{ij}\}$.

c. Als $I_1 \cup I_2 = \emptyset$: x is een maximale stroom en (W,V-W) een snede met minimale capaciteit (STOP).

Stap 3 (iteratiestap)

a. Kies een $(r,p) \in I_1 \cup I_2$.

b. Bepaal de keten P (met pijlen van A) van v_1 naar v_n die ontstaat door (r,p) aan T toe te voegen.

c. $P_1 := \{(i,j) \in P \mid (i,j) \text{ is een voorwaartse pijl}\}$;
$P_2 := \{(i,j) \in P \mid (i,j) \text{ is een achterwaartse pijl}\}$.

d. $\Delta_1 := \min \{b_{ij} - x_{ij} \mid (i,j) \in P_1\}$; $\Delta_2 := \min \{x_{ji} \mid (i,j) \in P_2\}$;
$\Delta := \min(\Delta_1, \Delta_2)$.

e. Bepaal (k,l) zódat $\Delta = \begin{cases} b_{ij} - x_{ij} & \text{als } \Delta = \Delta_1 \\ x_{ji} & \text{als } \Delta = \Delta_2. \end{cases}$

f. $x_{ij} := \begin{cases} x_{ij} + \Delta & \text{als } (i,j) \in P_1 \\ x_{ij} - \Delta & \text{als } (j,i) \in P_2 \\ x_{ij} & \text{anders} \end{cases}$

g. T := T + (r,p) - (k,l).

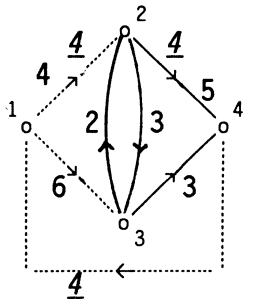
h. Ga naar stap 2.

Voorbeeld 4.1 (vervolg)

De bomen tekenen we gestippeld. De positieve stroomwaarden geven we met onderstreepte cursieve getallen aan. Voor (r,p) nemen we de eerste mogelijkheid die we tegenkomen.

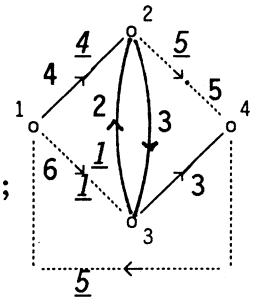
Iteratie 1

$T = \{(1,2), (1,3),(4,1)\}$; $x = 0$;
$W = \{1,2,3\}$; $I_1 = \{(2,4),(3,4)\}$; $I_2 = \varnothing$;
$(r,p) = (2,4)$; $P = [1,2,4]$; $P_1 = \{(1,2),(2,4)\}$; $P_2 = \varnothing$;
$\Delta_1 = 4$; $\Delta_2 = \infty$; $\Delta = 4$; $(k,l) = (1,2)$.



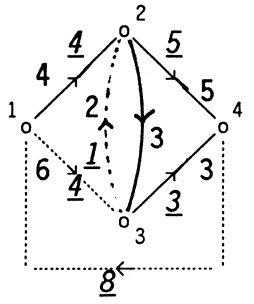
Iteratie 2

$T = \{(1,3), (4,1),(2,4)\}$;
$W = \{1,3\}$; $I_1 = \{(3,2),(3,4)\}$; $I_2 = \varnothing$;
$(r,p) = (3,2)$; $P = [1,3,2,4]$; $P_1 = \{(1,3),(3,2),(2,4)\}$; $P_2 = \varnothing$;
$\Delta_1 = 1$; $\Delta_2 = \infty$; $\Delta = 1$; $(k,l) = (2,4)$.



Iteratie 3

$T = \{(1,3), (4,1),(3,2)\}$;
$W = \{1,2,3\}$; $I_1 = \{(3,4)\}$; $I_2 = \varnothing$;
$(r,p) = (3,4)$; $P = [1,3,4]$; $P_1 = \{(1,3),(3,4)\}$; $P_2 = \varnothing$;
$\Delta_1 = 3$; $\Delta_2 = \infty$; $\Delta = 3$; $(k,l) = (3,4)$.

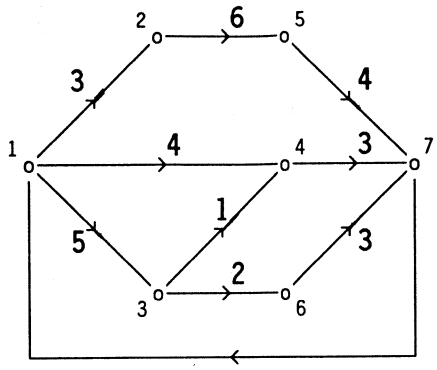


Iteratie 4

$T = \{(1,3), (4,1),(3,2)\}$;
$W = \{1,2,3\}$; $I_1 = \varnothing$; $I_2 = \varnothing$;
Maximale stroom: $x_{12} = 4$; $x_{13} = 4$; $x_{23} = 0$; $x_{32} = 1$; $x_{24} = 5$; $x_{34} = 3$; $x_{41} = 8$.
Minimale snede : $W = \{1,2,3\}$; capaciteit = 8.

Opgave 4.7

Bepaal met de simplex methode een maximale stroom en een snede met minimale capaciteit in onderstaand netwerk. Start met als voortbrengende boom:
$T = \{(1,2),(1,3),(2,5),(1,4),(3,6),(7,1)\}$.



Opgave 4.8*

Beschouw de padenmatrix (paragraaf 5 van hoofdstuk I) P(1,n) en noem deze P.

a. Als A de incidentiematrix van een netwerk is, toon aan dat de matrix AP^t, waarbij P^t de getransponeerde is van P, in de i-de rij de nulvector heeft staan voor alle $i \neq 1,n$.

b. Is P totaal unimodulair? Bewijs dit of geef een tegenvoorbeeld.

c. Toon aan dat het maximale stroom probleem geformuleerd kan worden als:

$$(*) \quad \max \left\{ \sum_{i=1}^{q} x_i \;\middle|\; \begin{array}{r} \sum_{i=1}^{q} p_{ij} x_i \leq b_j,\ j = 1,2,\ldots,m \\ x_i \geq 0\ ,\ i = 1,2,\ldots,q \end{array} \right\},$$

waarbij q het aantal paden van 1 naar n is.

d. Formuleer het duale probleem van (*) en noteer dit met (**).

e. Geef een interpretatie van het duale probleem (**), als we aannemen dat daarin de variabelen de waarden 0 of 1 hebben.

f. Geef de formulering van (*) en (**) voor het netwerk uit opgave 4.3.

g. Los (**) op voor het netwerk uit onderdeel f.

h. Als x een optimale basisoplossing is van (*), is x dan geheeltallig? Bewijs dit of geef een tegenvoorbeeld.

3. Methode van Ford en Fulkerson

Literatuur

* J.Edmonds and R.M.Karp: "Theoretical improvements in algorithmic efficiency for network flow problems", Journal of the ACM 19 (1972) pp.248-264.
* L.R.Ford and D.R.Fulkerson: "Flows in networks", Princeton University Press, Princeton (1962)
* Hoofdstuk 4 in:
E.L.Lawler: "Combinatorial optimization: networks and matroïds", Holt, Rinehart and Winston, New York (1976).

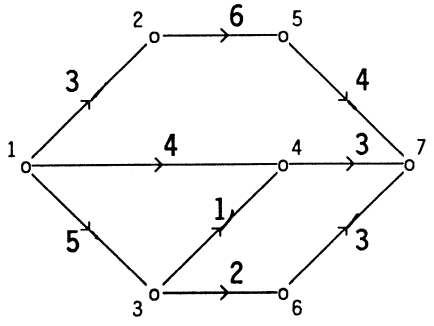
De methode van Ford en Fulkerson is een iteratieve methode waarbij in iedere iteratie een groeiketen van v_1 naar v_n wordt bepaald totdat er geen meer bestaat (dan is de stroom maximaal). Bij deze methode kunnen we de pijl (n,1) achterwege laten. De methode is dus als volgt.

Stap 1 (start)
Kies een toelaatbare beginstroom (bijv. $x_{ij} = 0$ voor iedere (i,j)).

Stap 2 (iteratiestap)
a. Bepaal een groeiketen P van v_1 naar v_n (als deze niet bestaat: ga naar 3).
b. Bepaal de maximale hoeveelheid Δ die over P kan worden vervoerd.
c. Pas de stroom x aan met Δ over P.
d. Ga naar onderdeel 2a.

Stap 3 (einde)
a. Laat $W := \{v_i \mid v_i$ vanuit v_1 bereikbaar met een groeiketen$\}$.
b. x is een maximale stroom en (W,V-W) een snede met minimale capaciteit.

Voorbeeld 4.3
Beschouw nevenstaand netwerk met daarin de aangegeven capaciteiten. Start met $x = 0$.



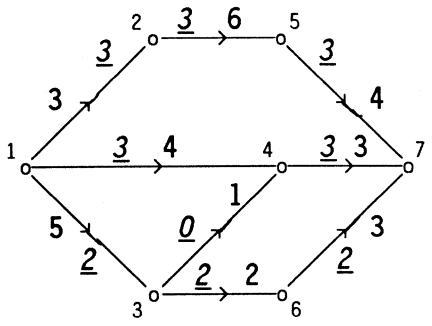
Iteratie 1
$P = [1,2,5,7]$; $\Delta = 3$; $x_{12} = x_{25} = x_{57} = 3$.

Iteratie 2
$P = [1,3,6,7]$; $\Delta = 2$; $x_{13} = x_{36} = x_{67} = 2$.

Iteratie 3
$P = [1,4,7]$; $\Delta = 3$; $x_{14} = x_{47} = 3$.

Iteratie 4

Er kan geen groeiketen meer worden gevonden. W = {1,3,4}. De snede (W,V-W) bestaat ui de pijlenverz. {(1,2),(3,6),(4,7)} en heeft capaciteit 3 + 2 + 3 = 8.
De waarde van de maximale stroom is ook 8.
De stroom zelf is in het netwerk aangegeven met de onderstreepte getallen.



Stelling 4.5

De methode van Ford en Fulkerson is correct en heeft complexiteit $\mathcal{O}(m \cdot w)$, waarbij w de waarde van de maximale stroom is.

Bewijs

Het algoritme eindigt als er geen groeiketen meer bestaat. Er geldt dan:

$$x_{ij} = \begin{cases} b_{ij} & \text{als } i \in W,\ j \notin W \\ 0 & \text{als } i \notin W,\ j \in W \end{cases} \Rightarrow \text{(Stelling 4.1)} \begin{cases} x \text{ maximale stroom} \\ (W,V\text{-}W) \text{ minimale snede} \end{cases}$$

Voor de complexiteit geldt het volgende:

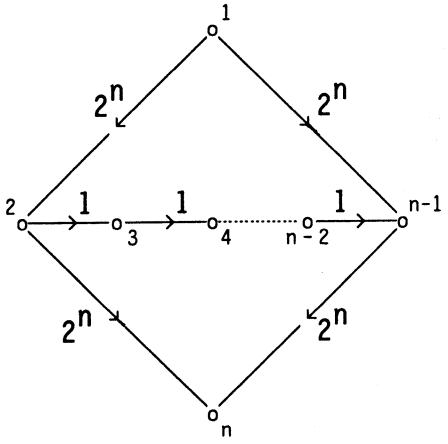
* in iedere iteratie neemt de waarde van de stroom met minstens 1 toe: er zijn $\mathcal{O}(w)$ iteraties;
* per iteratie wordt gedaan:
  - constructie groeiketen P: omdat iedere pijl hoogstens één keer wordt bekeken is de hoeveelheid werk $\mathcal{O}(m)$;
  - bepaling Δ en x aanpassen: P heeft maximaal n-1 pijlen: $\mathcal{O}(n) \leq \mathcal{O}(m)$.

De totale hoeveelheid werk is dus $\mathcal{O}(m \cdot w)$. □

Bovenstaand resultaat betekent niet dat de methode polynomiaal is. De input-parameters van een netwerk probleem zijn n en m. w kan echter exponentieel in deze parameters zijn, zoals onderstaand voorbeeld laat zien.

Voorbeeld 4.4

Als we in nevenstaand netwerk starten met x = 0, dan kunnen afwisselend de ketens [1,2,3,...,n-1,n] en [1,n-1,n-2,...,2,n] als groeiketen worden gekozen. In iedere iteratie is $\Delta = 1$, zodat er 2^{n+1} iteraties zijn.



Edmonds and Karp hebben in 1972 aangetoond dat de techniek van het zijwaarts zoeken een polynomiaal algoritme met complexiteit $O(m^2n)$ oplevert.
Bij het zijwaarts zoeken geldt voor iedere v_i die bezocht wordt dat de boomafstand gelijk is aan het minimum aantal pijlen om via een groeiketen van v_1 naar v_i te komen.

Definiëer: σ_i^k := minimum aantal pijlen om via een groeiketen van v_1 naar v_i te gaan in de k-de iteratie;
τ_i^k := minimum aantal pijlen om via een groeiketen van v_i naar v_n te gaan in de k-de iteratie.

Lemma 4.3
Als in de k-de iteratie een groeiketen met zo min mogelijk pijlen wordt gekozen, dan geldt voor iedere $1 \le i \le n$: $\sigma_i^{k+1} \ge \sigma_i^k$ en $\tau_i^{k+1} \ge \tau_i^k$.
Bewijs
Veronderstel dat $\sigma_i^{k+1} < \sigma_i^k$ voor zekere $2 \le i \le n$ ($\sigma_1^{k+1} = \sigma_1^k = 0$).
Dan is er dus een $2 \le j \le n$ zódat $\sigma_j^{k+1} = \min \{ \sigma_p^{k+1} \mid \sigma_p^{k+1} < \sigma_p^k \} \ge 1$.
Laat (v_q,v_j) de pijl naar v_j zijn in de groeiketen tijdens de (k+1)-ste iteratie: $\sigma_j^{k+1} = \sigma_q^{k+1} + 1$ en $\sigma_q^{k+1} \ge \sigma_q^k$ (volgt uit definitie van j). Omdat

(4.13) $$\sigma_j^k \ge \sigma_j^{k+1} + 1 = \sigma_q^{k+1} + 2 \ge \sigma_q^k + 2,$$

is (v_q,v_j) tijdens de k-de iteratie geen pijl van een mogelijke groeiketen (anders was $\sigma_j^k \le \sigma_q^k + 1$), terwijl de pijl (v_q,v_j) wel in de groeiketen zit tijdens de (k+1)-ste iteratie.

Dit betekent dat (v_j,v_q) een pijl is van de groeiketen tijdens de k-de iteratie: $\sigma_q^k = \sigma_j^k + 1$. Dit geeft met (4.13) de gewenste tegenspraak:
$\sigma_q^k = \sigma_j^k + 1 \ge \sigma_q^k + 3$.
De ongelijkheid $\tau_i^{k+1} \ge \tau_i^k$ is analoog te bewijzen. □

Stelling 4.6
Als in iedere iteratie een groeiketen met zo min mogelijk pijlen wordt gekozen, dan zijn er hoogstens $\frac{1}{2}m \cdot n$ iteraties.

Bewijs

In iedere groeiketen is minstens één blokkerende pijl. Veronderstel dat (v_i,v_j) een blokkerende pijl is tijdens de k-de iteratie. Deze pijl kan in dezelfde richting pas weer in de groeiketen worden opgenomen nadat de pijl (v_j,v_i) in een groeiketen is opgenomen, en laat dit voor het eerst gebeuren in iteratie $p \geq k+1$. We kunnen nu schrijven:

$$\sigma_i^p + \tau_i^p = \sigma_j^p + 1 + \tau_i^p \geq \sigma_j^k + 1 + \tau_i^k = (\sigma_i^k + 1) + 1 + \tau_i^k = \sigma_i^k + \tau_i^k + 2.$$

Het aantal pijlen in een groeiketen die (v_j,v_i) bevat is dus minstens 2 groter dan het aantal pijlen in de groeiketen die (v_i,v_j) voor de laatste keer als blokkerende pijl bevat. Omdat iedere groeiketen hoogstens n-1 pijlen bevat, en de lengten van de groeiketens niet dalen, kan iedere (v_i,v_j) of (v_j,v_i) niet meer dan $\frac{1}{2}n$ keer als blokkerende pijl voorkomen. Er zijn dus hoogstens $\frac{1}{2}m \cdot n$ iteraties. □

Gevolg

De door Edmonds en Karp voorgestelde implementatie van de methode van Ford en Fulkerson heeft complexiteit $O(m^2n)$.

Hieronder volgt een uitwerking van dit algoritme. Deze procedure, *MAXFLOW* geheten, gaat uit van een toelaatbare stroom x = 0, zoekt in iedere iteratie een groeiketen *(CHAINSEARCH)* en verandert de waarde van de stroom *(FLOWCHANGE)*.

We nemen aan dat de datastructuur wordt gegeven door lijsten:

$L^+[v] = \{w \mid (v,w) \in A\}$ en $L^-[v] = \{w \mid (w,v) \in A\}$ voor alle $v \in V$.

Tijdens het zoeken van een groeiketen wordt bijgehouden of een knooppunt v al bereikt is (in dat geval is SCAN[v] = true). Knooppunten v die bereikt zijn krijgen twee labels (*LABELING*):

FLOW[v] = de maximale hoeveelheid die over een groeiketen vanuit v_1 naar v kan worden vervoerd;

$$\text{PRED}[v] = \begin{cases} w & \text{als } (w,v) \text{ een voorwaartse pijl is op de groeiketen naar } v \\ -w & \text{als } (w,v) \text{ een achterwaartse pijl is op de groeiketen naar } v \end{cases}$$

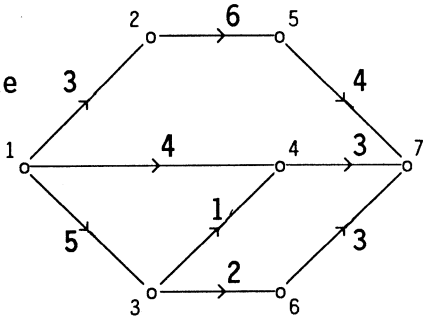
Aan het einde van de procedure geeft x de maximale stroom met x[n,1] de waarde en (W,V-W) is een (1,n)-snede van minimale capaciteit indien voor W wordt genomen: W = {v | SCAN[v] = true}.

```
procedure MAXFLOW;
  var i,j: integer;
      SCAN: array [1:n] of boolean;
      PRED,FLOW: array [1:n] of integer;
      x: array [1:n,1:n] of integer;
      Q: queue of integer;

  procedure CHAINSEARCH;
    var i,j: integer;
  begin SCAN[1] := true; PRED[1] := 0; FLOW[1] := maxint;
        for i := 2 to n do SCAN[i] := false;
        MAKENULL(Q); ENQUEUE(1,Q);
        while not EMPTY(Q) and not SCAN[n] do
              begin j := FRONT(Q); DEQUEUE(Q); LABELING(j) end
  end CHAINSEARCH;

  procedure LABELING(j);
    var k: integer;
  begin for all k ∈ L⁺[j] do
            if x[j,k] < b[j,k] and not SCAN[k] then
               begin SCAN[k] := true; ENQUEUE(k,Q); PRED[k] := j;
                     FLOW[k] := FLOW[j];
                     if b[j,k]-x[j,k] < FLOW[k] then FLOW[k] := b[j,k]-x[j,k]
               end;
        for all k ∈ L⁻[j] do
            if x[k,j] > 0 and not SCAN[k] then
               begin SCAN[k] := true; ENQUEUE(k,Q); PRED[k] := -j;
                     FLOW[k] := FLOW[j];
                     if x[k,j] < FLOW[k] then FLOW[k] := x[k,j]
               end
  end LABELING;

  procedure FLOWCHANGE;
    var Δ,p,q: integer;
  begin Δ := FLOW[n]; x[n,1] := x[n,1] + Δ; q := n; p := PRED[q];
        while p <> 0 do
              begin if p > 0 then x[p,q] := x[p,q] + Δ
                    else begin p := -p; x[q,p] := x[q,p] - Δ end
                    q := p; p := PRED[q]
               end
  end FLOWCHANGE;
```

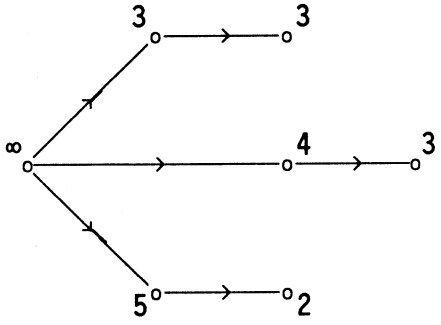
```
begin for i := 1 to n do
          for j := 1 to n do x[i,j] := 0;
      repeat CHAINSEARCH;
             if SCAN[n] = true then FLOWCHANGE
      until  SCAN[n] = false
end MAXFLOW.
```

Voorbeeld 4.3 (vervolg)

Passen we de implementatie van Edmonds en Karp toe op nevenstaand voorbeeld, dan krijgen we de volgende iteraties. Per iteratie is aangegeven welke knooppunten via *CHAINSEARCH* worden bereikt en wat de *FLOWWAARDE* is.



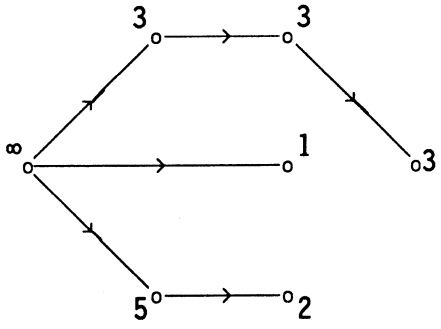
Iteratie 1



Groeiketen: [1,4,7]; Δ = 3;

$x_{14} = x_{47} = 3$.

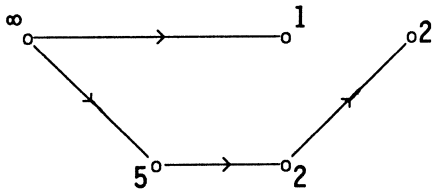
Iteratie 2



Groeiketen: [1,2,5,7]; Δ = 3;

$x_{12} = x_{25} = x_{57} = 3$.

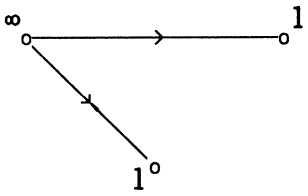
Iteratie 3



Groeiketen: [1,3,6,7]; Δ = 2;

$x_{13} = x_{36} = x_{67} = 2$.

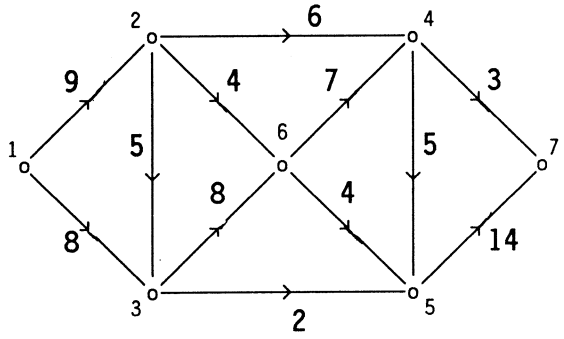
Iteratie 4



Er is geen groeiketen meer.
De huidige stroom is maximaal met waarde 8 en voor de snede (W,V-W) met minimale capaciteit geldt: W = {1,3,4}.

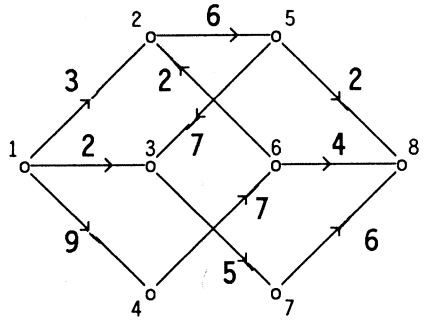
Opgave 4.9

Bepaal een maximale stroom en een (1,7)-snede van minimale capaciteit voor nevenstaand netwerk.
Pas daartoe de implementatie van Edmonds en Karp toe.



Opgave 4.10

Bepaal een maximale stroom en een (1,8)-snede van minimale capaciteit voor nevenstaand netwerk.
Pas daartoe de implementatie van Edmonds en Karp toe.



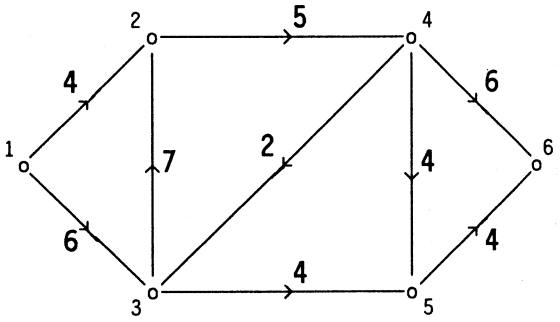
Opgave 4.11

Is de volgende uitspraak juist of onjuist?
De maximale stroom kan niet worden gevonden zonder iedere pijl van het netwerk tenminste één keer te bekijken.

Opgave 4.12

a. Beschrijf een alternatieve procedure, analoog aan die van Ford en Fulkerson, om een maximale stroom te bepalen die groeiketens vanuit v_n in plaats vanuit v_1 tracht op te sporen.

b. Pas de procedure toe op onderstaand netwerk.



c. Beschrijf een procedure om een pijl op te sporen met de eigenschap dat het verhogen van de capaciteit van deze pijl de waarde van de stroom doet toenemen.

d. Pas de in onderdeel c beschreven methode toe op het netwerk uit onderdeel b

e. Bestaat er altijd zo'n pijl als in onderdeel c genoemd? Licht uw antwoord toe.

4. Methode van Dinic, Malhotra, Kumar en Maheshwari

Literatuur

* Sectie 5 van hoofdstuk 3 in:
  M.M.Syslo, N.Deo and J.S.Kowalik: "Discrete optimization algorithms with Pascal programs", Prentice-Hall, Englewood Cliffs (1983).
* Paragraaf 4 van hoofdstuk 9 in:
  C.H.Papadimitriou and K.Steiglitz: Combinatorial optimization: algorithms and complexity", Prentice-Hall, Englewood Cliffs (1982).

Bij de methode van Ford en Fulkerson wordt de stroom in iedere iteratie verbeterd door een stroom over een groeiketen toe te voegen. De methode van Dinic, Malhotra, Kumar en Maheshwari (DMKM) werkt anders. De extra stroom wordt niet noodzakelijkerwijs over een keten gestuurd, maar zo mogelijk over een aantal ketens tegelijk. Het zal blijken dat op deze manier een $O(n^3)$ algoritme verkregen kan worden.

We gebruiken hiervoor een van het netwerk N en stroom x afgeleid gelaagd netwerk N_x. Een gelaagd netwerk is een netwerk waarin de knoopunten van V zijn verdeeld in $V_1 \cup V_2 \cup \cdots \cup V_p$ zódanig dat $V_1 = \{v_1\}$, $V_p = \{v_n\}$, en er zijn alleen pijlen van V_i naar V_{i+1}, $i = 1,2,\ldots,p-1$.

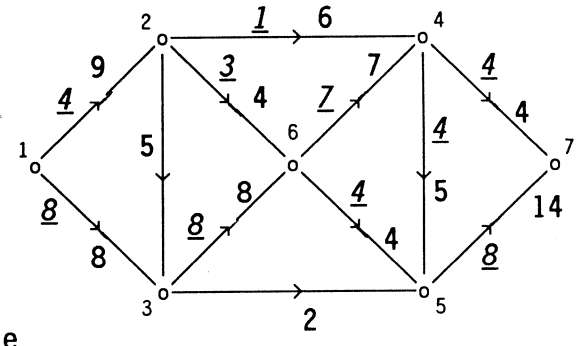
De constructie van N_x geschiedt met de methode van het zijwaarts zoeken en neemt voorwaartse en achterwaartse pijlen van groeiketens in N_x op. De pijlenverz. van N_x wordt bijgehouden in een array cap, waarin de aan de stroom "aangepaste" capaciteiten staan. Voor een voorwaartse pijl (i,j) is dit $b_{ij} - x_{ij}$; voor een achterwaartse pijl x_{ji}. Pijlen met een aangepaste capaciteit 0 komen niet voor in N_x. Indien v_j reeds bezocht is, dan wordt de pijl (v_i,v_j) toch in N_x opgenomen indien $v_j \in V_k$ en $v_i \in V_{k-1}$ voor zekere k.

De knooppunten krijgen labels (LABEL[j] = i betekent $v_j \in V_i$), met verz. PRED[i] en SUCC[i] houden we bij wat de voorgangers resp. de opvolgers van v_i zijn. Indien er geen groeiketen bestaat van v_1 naar v_n, dan krijgt de boolean chain de waarde false. Knooppunten $v \neq v_n$ die geen opvolger hebben gekregen worden aan het einde van de procedure uit N_x verwijderd.

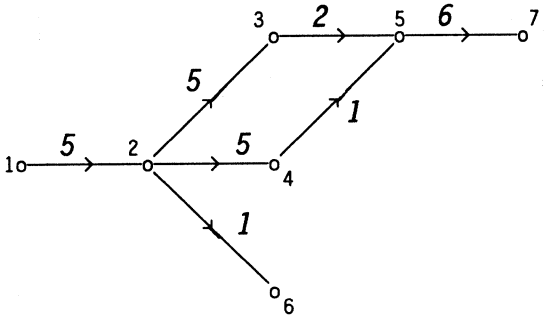
```
procedure LAYER (chain,p,cap);
  var i,j,k,l: integer;
      LABEL: array [1:n] of integer;
      V,PRED,SUCC: array [1:n] of queue of integer;
begin chain := true; MAKENULL(V[1]); LABEL[1] := 1; i := 1; ENQUEUE(1,V[1]);
      for j := 1 to n do
          for k := 1 to n do cap[j,k] := 0;
      for j := 2 to n do
          begin LABEL[j] := -1; MAKENULL(PRED[j]); MAKENULL(SUCC[j]) end;
      while not EMPTY(V[i]) and LABEL[n] = -1 do
      begin MAKENULL(V[i+1]);
            for all j ∈ V[i] do
            begin for all k ∈ L+[j] do
                  if x[j,k] < b[j,k] and (LABEL[k] = -1 or LABEL[k] = i+1) then
                  begin if LABEL[k] = -1 then ENQUEUE(k,V[i+1]);
                        ENQUEUE(k,SUCC[j]); ENQUEUE(j,PRED[k]);
                        cap[j,k] := b[j,k] - x[j,k]; LABEL[k] := i+1;
                  end
                   for all k ∈ L-[j] do
                  if x[k,j] > 0 and (LABEL[k] = -1 or LABEL[k] = i+1) then
                  begin if LABEL[k] = -1 then ENQUEUE(k,V[i+1]);
                        ENQUEUE(k,SUCC[j]); ENQUEUE(j,PRED[k]);
                        cap[j,k] := x[k,j]; LABEL[k] := i+1
                  end
            end;
            i := i +1
      end;
      p := i;
      if   LABEL[n] = -1 then chain := false
      else begin j := i;
             while j <> 1 do
               begin for all k ∈ V[j] do if k <> n and EMPTY(SUCC[k]) then
                        begin for all l ∈ PRED[k] do
                              begin cap[l,k] := 0; DELETE(k,SUCC[l]) end;
                              DELETE(k,V[j]); MAKENULL(PRED[k]);
                        end;
                     j := j-1
               end
end LAYER.
```

Voorbeeld 4.5

Beschouw nevenstaand netwerk met daarin de capaciteiten en een toegelaten stroom (de onderstreepte getallen).
Het netwerk N_x zullen we in twee stappen opbouwen: eerst zonder te letten op knooppunten zonder opvolger, en daarna worden deze knooppunten uit N_x verwijderd.
De getallen bij de pijlen van N_x stellen de aangepaste capaciteiten voor.

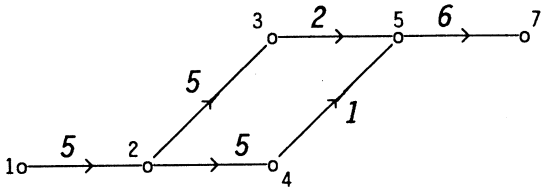


Eerste fase



$V_1 = \{1\}$; $V_2 = \{2\}$; $V_3 = \{3,4,6\}$;
$V_4 = \{5\}$; $V_5 = \{7\}$.

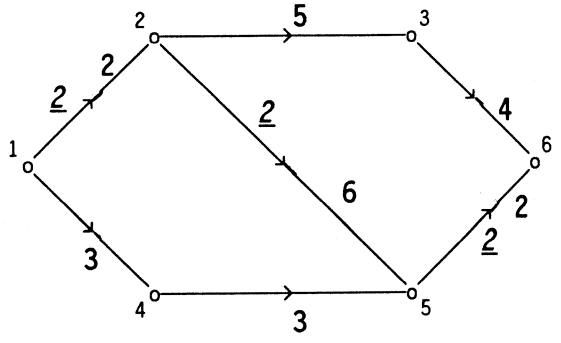
Tweede fase



$V_1 = \{1\}$; $V_2 = \{2\}$; $V_3 = \{3,4\}$;
$V_4 = \{5\}$; $V_5 = \{7\}$.

We zullen nu gaan beschrijven hoe een verzadigde stroom in een gelaagd netwerk kan worden gevonden. Een verzadigde stroom is een stroom waarvoor geldt dat op ieder pad van v_1 naar v_n er tenminste één pijl (i,j) is met cap[i,j] = 0, d.w.z. dat er in N_x geen enkel pad van v_1 naar v_n is.
Het volgende voorbeeld laat zien dat een verzadigde stroom in het algemeen geen maximale stroom is.

Voorbeeld 4.6



De aangegeven stroom is een verzadigde stroom: de drie paden van 1 naar 6, [1,2,3,6], [1,2,5,6] en [1,4,5,6], bevatten alle drie tenminste één pijl waarvan de aangepaste capaciteit 0 is.
De stroom is echter niet maximaal:
$x_{12} = x_{14} = x_{23} = x_{45} = x_{36} = x_{56} = 2$
en $x_{23} = 0$ is de maximale stroom.

Het bepalen van een verzadigde stroom gaat als volgt. We berekenen eerst voor ieder knooppunt v_i de maximale hoeveelheid d_i die langs v_i kan worden vervoerd: $d_i = \min(d_i^+, d_i^-)$, waarbij $d_i^+ = \Sigma_j \text{ cap}[i,j]$ en $d_i^- = \Sigma_j \text{ cap}[j,i]\}$.
Merk op dat als we starten met de N_x, verkregen met de procedure LAYER, $d_i \geq 1$ voor alle in N_x voorkomende knooppunten.

Vervolgens bepalen we een knooppunt v_r met $d_r = \min_i d_i$. Deze hoeveelheid, zeg d, kan langs ieder knooppunt worden vervoerd. We "duwen" d vanuit v_r naar de knooppunten in de volgende laag. Wat daar aankomt wordt weer doorgeduwd totdat de hoeveelheid d in knooppunt v_n is aangekomen.
De stroom in N_x wordt bijgehouden met y[i,j]; de capaciteiten cap[i,j] worden steeds aangepast, evenals de getallen $d^+[i]$ en $d^-[i]$. Met y[j] houden we voor ieder knooppunt bij wat vanuit knooppunt v_i verder moet worden geduwd.
We initialiseren y[j] op 0 voor alle j. Stel $r \in V[i]$. Dan starten we met y[r] = d. In het algemeen duwen we als volgt de hoeveelheden y[j] van V[i] naar V[i+1]:

```
procedure PUSH(i);
begin for all j ∈ V[i] do if y[j] > 0 then
        begin d+[j] := d+[j] - y[j];
              for all k ∈ SUCC[j[] do
              begin if cap[j,k] > y[j] then  q := y[j] else q := cap[j,k];
                    y[j,k] := y[j,k] + q; y[k] := y[k] + q; y[j] := y[j] - q;
                    cap[j,k] := cap[j,k] - q; d-[k] := d-[k] - q
              end
        end
end PUSH.
```

Op analoge wijze is de hoeveelheid d vanuit v_r terug te "trekken" naar v_1 (de hierbij behorende procedure zullen we PULL noemen).

Indien de doorvoer van een knooppunt 0 is geworden, dan wordt dit knooppunt met alle binnenkomende en uitgaande pijlen verwijderd. Dit zal zeker gebeuren met knooppunt v_r, mogelijk ook met andere knooppunten.

Aan het einde van een deel-iteratie, d.z.w. als de hoeveelheid d vanuit v_r naar v_n en naar v_1 is gestuurd, wordt het verwijderen van knooppunten uitgevoerd, ook weer beginnend met $v_r \in V_i$ zowel gaande naar $V_{i+1},V_{i+2},\ldots,V_p$ als de andere kant op naar $V_{i-1},V_{i-2},\ldots,V_1$. Zodra v_1 of v_n wordt verwijderd stoppen we en is de stroom y inderdaad verzadigd. Opeenvolgende deel-iteraties hebben steeds minder knooppunten: na maximaal n-1 deel-iteraties wordt een verzadigde stroom in N_x verkregen.

Dit verwijderen kan gebeuren met de procedure REMOVE(i), waarin de knooppunten met d[j] = 0 uit V[i] worden verwijderd.

```
procedure REMOVE(i);
begin for all j ∈ V[i] do if (d+[j] = 0 or d-[i] = 0) then
      begin for all k ∈ PRED[j] do
                begin d+[k] := d+[k] - cap[k,j]; cap[k,j] := 0;
                      DELETE(j,SUCC[k]);
                end;
            for all k ∈ SUCC[j] do
                begin d-[k] := d-[k] - cap[j,k]; cap[j,k] := 0;
                      DELETE(j,PRED[k])
                end;
                DELETE(j,V[i]); MAKENULL(PRED[j]); MAKENULL(SUCC[j])
      end
end REMOVE.
```

Samenvattend gaat het vinden van een verzadigde stroom in N_x aldus.

Stap 1 (initialisatie)

a. y[i,j] := 0 voor alle (i,j);

b. $d^+[i] := \Sigma_j$ cap[i,j] en $d^-[i] := \Sigma_j$ cap[j,i]} voor i = 1,2,...,n.

Stap 2 (iteratie)

a. d[1] := d^+[1]; d[n] := d^-[n]; d[i] := min(d^+[i],d^-[i]), i = 2,3,...,n-1;

b. Bepaal r en d zódat d = d[r] = $\min_i$ d[i];

c. s := LABEL[r]; y[j] := 0 voor alle knooppunten j; y[r] := d;

d. PUSH(i) voor alle i = s,s+1,...,p; PULL(i) voor alle i = s,s-1,...,2;

e. REMOVE(i) voor i = s,s+1,...,p; REMOVE(i) voor i = s-1,s-2,...,1;

f. Indien v_1 en v_n niet verwijderd zijn: ga naar 2a;
   Anders: y is een verzadigde stroom (STOP).

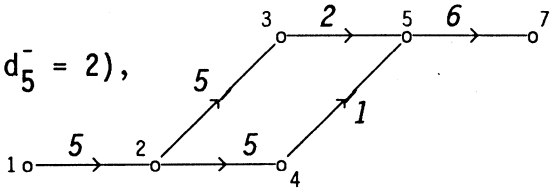
Voorbeeld 4.5 (vervolg)

Initialisatie

$d_1^+ = 5$, $d_1^- = 0$; $d_2^+ = 10$, $d_2^- = 5$; $d_3^+ = 2$, $d_3^- = 5$; $d_4^+ = 1$, $d_4^- = 5$;
$d_5^+ = 6$, $d_5^- = 3$; $d_7^+ = 0$, $d_7^- = 6$.

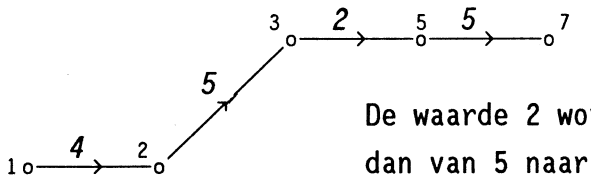
Iteratie 1

$d_1 = 5$; $d_7 = 6$; $d_2 = 5$; $d_3 = 2$; $d_4 = 1$; $d_5 = 3$.
$r = 4$; $d = 1$; $s = 3$.
Deze d = 1 sturen we van 4 naar 5 ($d_4^+ = 0$; $d_5^- = 2$), van 5 naar 7 ($d_5^+ = 5$; $d_7^- = 5$); de andere kant op wordt 1 van 4 naar 2 ($d_4^- = 4$; $d^+ = 9$) en van 2 naar 1 ($d^- = 4$; $d^+ = 4$) gestuurd.



$y_{12} = y_{24} = y_{45} = y_{57} = 1$; $d_1 = 4$.
Knooppunt 4 verdwijnt ($d_2^+ = 5$; $d_5^- = 2$).

Iteratie 2



$d_1 = 4$, $d_2 = 4$, $d_3 = 2$, $d_5 = 2$, $d_7 = 5$;
$r = 3$ en $d = 2$.
De waarde 2 wordt van 3 naar 5 vervoerd ($d_3^+ = 0$; $d_5^- = 0$) dan van 5 naar 7 ($d_5^+ = 3$; $d_7^- = 3$); de andere kant op van 3 naar 2 ($d_3^- = 3$; $d_2^+ = 3$) en van 2 naar 1 ($d_2^- = 2$; $d_1^+ = 2$). $d_1 = 2$; $d_2 = 2$; $d_3 = 0$; $d_5 = 0$; $d_7 = 3$.

De knooppunten 3, 5, 7, 2 en 1 worden verwijderd.
De stroom y met $y_{12} = 3$, $y_{23} = 2$, $y_{24} = 1$, $y_{35} = 2$, $y_{45} = 1$, $y_{57} = 3$ is een verzadigde stroom.

De totale methode is nu:

Stap 1 (initialisatie)

x[i,j] := 0 voor alle (i,j); stel N_x op via LAYER(chain,p,cap).

Stap 2 (optimaliteitstest)

Als chain = true: ga naar stap 3.
Anders: x is een maximale stroom (STOP).

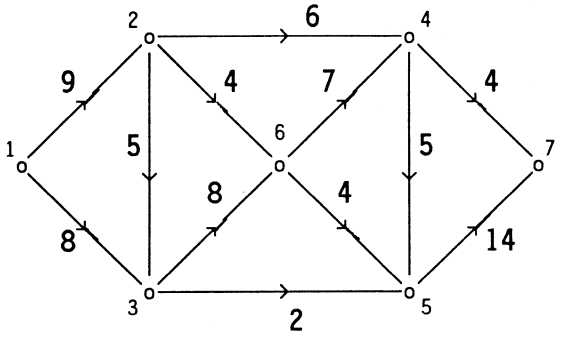
Stap 3 (Iteratiestap)

a. Vind een verzadigde stroom y in N_x (met algoritme vorige pagina).
b. x[i,j] := x[i,j] + y[i,j] voor alle voorwaartse pijlen (i,j) in N_x;
   x[i,j] := x[i,j] - y[j,i] voor alle achterwaartse pijlen (j,i) in N_x.
c. Stel N_x op via LAYER(chain,p,cap) en ga naar stap 2.

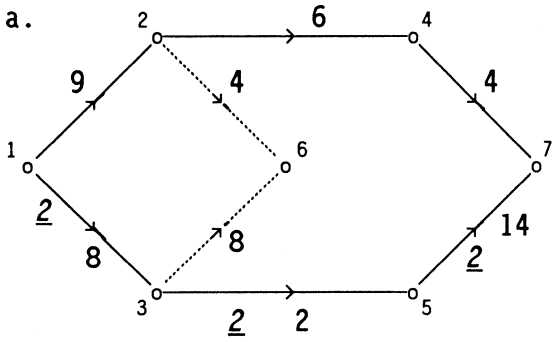
Voorbeeld 4.5 (vervolg)

Iteratie 1

Het gelaagde netwerk N_x staat hieronder. De niet onderstreepte getallen geven de aangepaste capaciteiten; de stroom y wordt met onderstreepte getallen aangeduid. Wat uit N_x is verwijderd in fase 2 is gestippeld aangegeven.

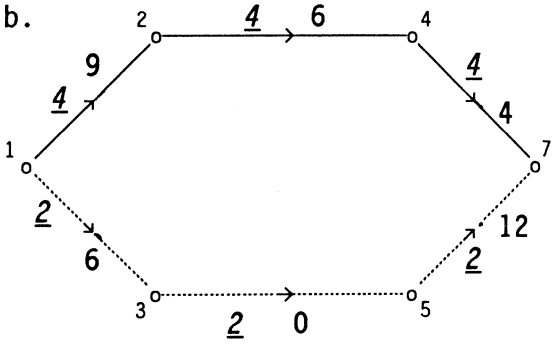


a.



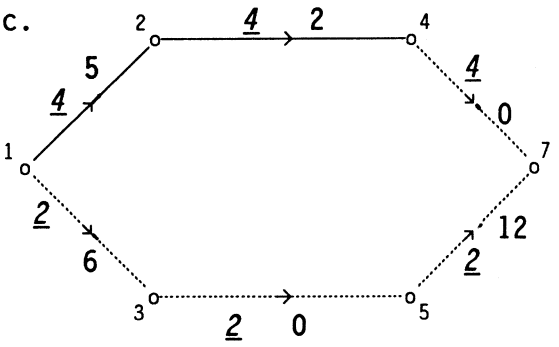
$d_1 = 17$; $d_2 = 6$; $d_3 = 2$; $d_4 = 4$; $d_5 = 2$;
$d_7 = 18$; r = 3; d = 2.
y[3] = 2.
y[3,5] = 2; y[5] = 2; y[3] = 0;
y[5,7] = 2; y[7] = 2; y[5] = 0.
y[3] = 2.
y[1,3] = 2; y[3] = 0; y[1] = 2.

b.



$d_1 = 9$; $d_2 = 6$; $d_4 = 4$; $d_7 = 4$;
r = 4; d = 4
y[4] = 4.
y[4,7] = 4; y[4] = 0; y[7] = 4.
y[4] = 0.
y[2,4] = 4; y[2] = 4; y[4] = 0;
y[1,2] = 4; y[1] = 4; y[2] = 0.

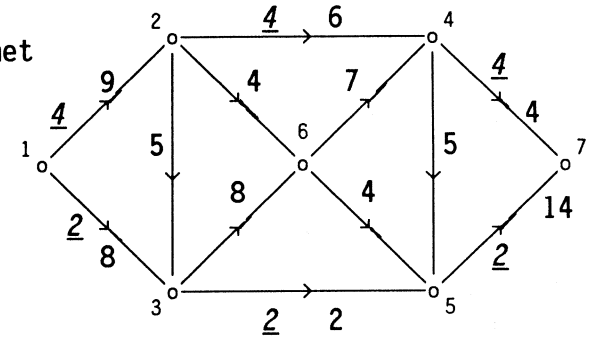
c.



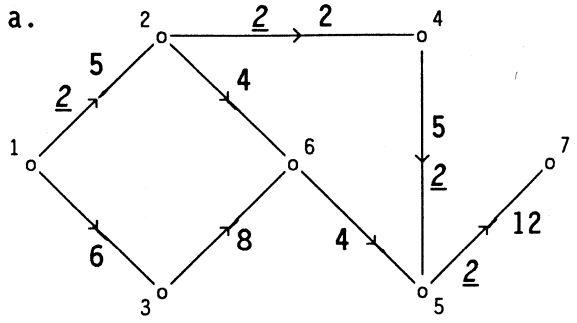
knooppunt 7 is verwijderd: de stroom is een verzadigde stroom.

Iteratie 2

Rechts staat het oorspronkelijke netwerk met de huidige stroom x; hieronder staat N_x.



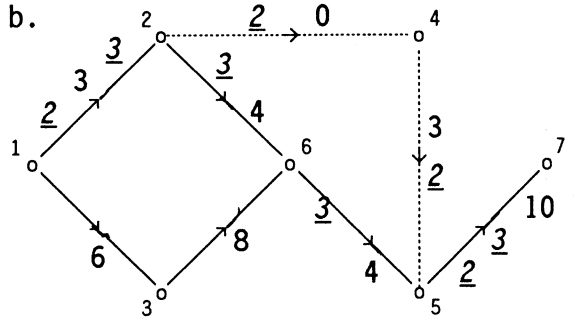
a.



$d_1 = 11$; $d_2 = 5$; $d_3 = 6$; $d_4 = 2$; $d_6 = 4$;
$d_5 = 9$; $d_7 = 9$; $r = 4$; $d = 2$.

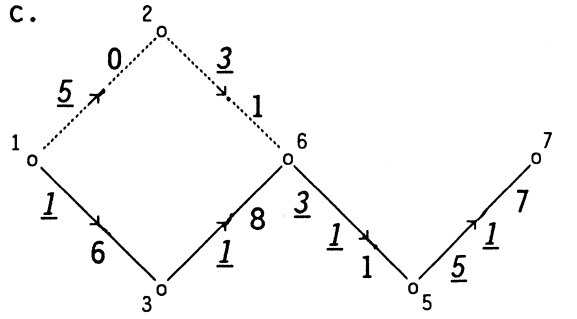
y[4] = 2; y[4,5] = 2; y[5] = 2; y[4] = 0; y[5,7] = 2; y[7] = 2; y[5] = 0.
y[4] = 2; y[2,4] = 2; y[4] = 0; y[2] = 2; y[1,2] = 2; y[1] = 2; y[2] = 0.

b.



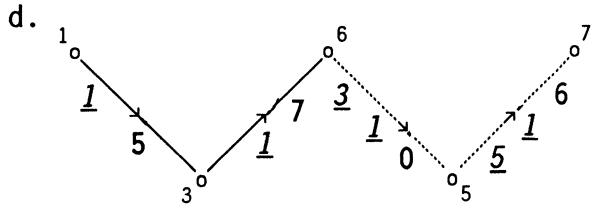
$d_1 = 9$; $d_2 = 3$; $d_3 = 6$; $d_6 = 4$; $d_5 = 4$;
$d_7 = 4$; $r = 2$; $d = 3$.
y[2] = 3.
y[2,6] = 3; y[6] = 3; y[2] = 0.
y[6,5] = 3; y[5] = 3; y[6] = 0.
y[5,7] = 3; y[7] = 3; y[5] = 0.
y[2] = 3.
y[1,2] = 3; y[1] = 3; y[2] = 0.

c.



$d_1 = 6$; $d_3 = 6$; $d_6 = 1$; $d_5 = 1$; $d_7 = 1$;
$r = 6$; $d = 1$.
y[6] = 1.
y[6,5] = 1; y[5] = 1; y[6] = 0.
y[5,7] = 3; y[7] = 1; y[5] = 0.
y[6] = 1.
y[3,6] = 1; y[3] = 1; y[6] = 0.
y[1,3] = 1; y[1] = 1; y[3] = 0.

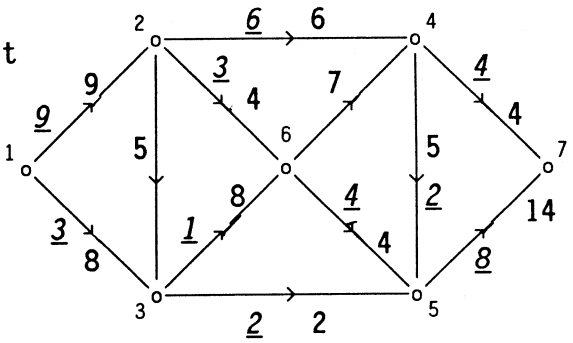
d.



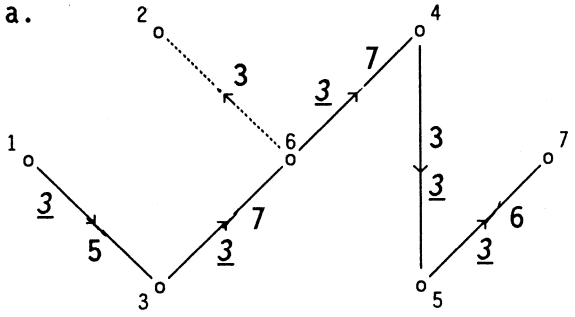
knooppunt 7 is verwijderd: de stroom is een verzadigde stroom.

Iteratie 3

Rechts staat het oorspronkelijke netwerk met de huidige stroom x; hieronder staat N_x.



a.

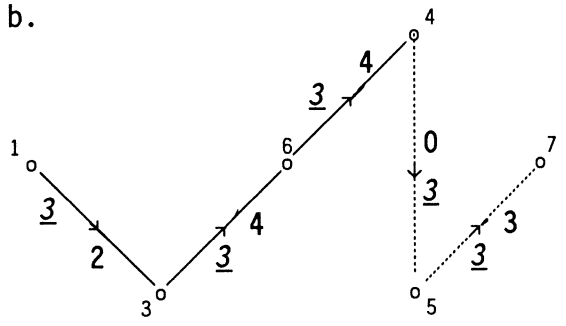


Merk op dat (6,3) een achterwaartse pijl is.

$d_1 = 5$; $d_3 = 5$; $d_6 = 7$; $d_4 = 3$; $d_5 = 3$; $d_7 = 6$; $r = 4$; $d = 3$.

y[4] = 3. y[4,5] = 3; y[5] = 3; y[4] = 0
y[5,7] = 3; y[7] = 3; y[5] = 0.
y[4] = 3. y[6,4] = 3; y[6] = 3; y[4] = 0
y[3,6] = 3; y[3] = 3; y[6] = 0.
y[1,3] = 3; y[1] = 3; y[3] = 0.

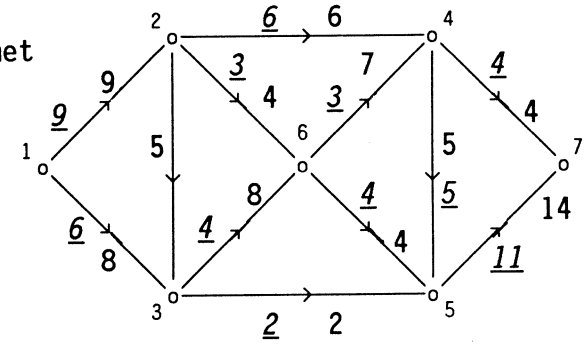
b.



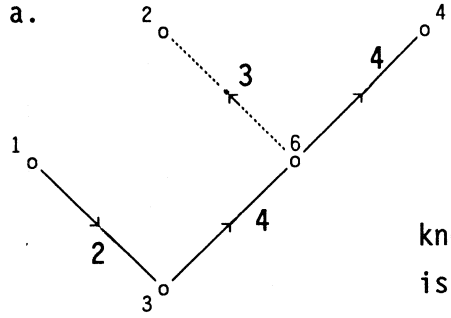
knooppunt 7 is verwijderd: de stroom is een verzadigde stroom.

Iteratie 4

Rechts staat het oorspronkelijke netwerk met de huidige stroom x; hieronder staat N_x.



a.



knooppunt 7 is niet bereikbaar: de huidige stroom is maximaal met waarde 15; de snede (W,V-W) met W = {1,3,6,2,4} heeft minmale capaciteit 15.

Stelling 4.7

In opeenvolgende iteraties is het minimum aantal pijlen om in N_x van v_1 naar v_n te gaan strict stijgend.

Bewijs

In het hulpnetwerk N_x heeft ieder pad van v_1 naar v_n dezelfde lengte, zeg p, welke gelijk is aan het minimum aantal pijlen om van v_1 naar v_n te gaan.
In N_x wordt een verzadigde stroom y gevonden, waarna x wordt aangepast met y tot x'. Vervolgens wordt $N_{x'}$ geconstrueerd.
Veronderstel dat hierin een pad P van v_1 naar v_n is met eveneens p pijlen, zeg $[v_1,v_2,\ldots,v_p,v_n]$.
Dit pad bestaat niet in de aan y aangepaste N_x (anders is y geen verzadigde stroom in N_x). Stel (v_i,v_{i+1}) is een pijl van P die niet in de aan y aangepaste N_x zit, en laat (v_i,v_{i+1}) een voorwaartse pijl zijn (voor een achterwaartse pijl gaat het bewijs analoog).
Omdat $(v_i,v_{i+1}) \in N_{x'}$, geldt: $x'_{i,i+1} < b_{i,i+1}$.
We kunnen de volgende gevallen onderscheiden:

a. (v_i,v_{i+1}) is een pijl van N_x bij aanvang van de iteratie:
   $x_{i,i+1} < b_{i,i+1}$, $y_{i,i+1} = b_{i,i+1} - x_{i,i+1}$, dus $x'_{i,i+1} = b_{i,i+1}$:
   tegenspraak.

b. (v_i,v_{i+1}) is geen pijl van N_x bij aanvang van de iteratie:
   Laat $\hat{N}_x$ het hulpnetwerk zijn van <u>alle</u> voorwaartse en achterwaartse pijlen t.o.v. x, dus N_x is een deelgraaf van $\hat{N}_x$. Geheel analoog aan Lemma 4.3 kan worden aangetoond dat in opeenvolgende iteraties de paden van v_1 naar een willekeurig knooppunt v_j en van een willekeurig knooppunt v_j naar v_n nooit korter.
   Laat $V_j := \{v \mid$ het kortste pad in $\hat{N}_x$ van v_1 naar v_j heeft lengte j-1$\}$,
   dus $v_1 \in V_1$ en $v_n \in V_{p+1}$. Veronderstel dat $v_{i+1} \in V_k$.
   (i) $x_{i,i+1} = b_{i,i+1}$: $x'_{i,i+1} < b_{i,i+1}$, dus $y_{i+1,i} > 0$ en $(v_{i+1},v_i) \in N_x$,
   d.w.z. $v_i \in V_{k+1}$.
   (ii) $x_{i,i+1} < b_{i,i+1}$: $(v_i,v_{i+1}) \notin N_x$, dus $v_i \in V_l$ met $l \geq k$.
   In beide gevallen is $v_i \in V_l$ met $l \geq k$.

We kunnen nu schrijven:

p = het aantal pijlen op het kortste pad van v_1 naar v_n (in $N_{x'}$)

= aantal pijlen van v_1 naar v_i + 1 + aantal pijlen van v_{i+1} naar v_n (in $N_{x'}$)

$\geq$ aantal pijlen van v_1 naar v_i + 1 + aantal pijlen van v_{i+1} naar v_n (in $\hat{N}_x$)

= (l-1) + 1 + (p+1-k) $\geq$ (k-1) + 1 + (p+1-k) = p + 1: tegenspraak. □

Stelling 4.8

De methode van DMKM is correct en heeft een implementatie met complexiteit $O(n^3)$.

Bewijs

De methode eindigt als er geen groeiketen is van v_1 naar v_n en is dus correct. Volgens Stelling 4.7 zijn er maximaal n-1 iteraties. We moeten dus aantonen dat de hoeveelheid werk per iteratie $O(n^2)$ is.

Per iteratie wordt het volgende gedaan:

* Stap 3a: Het vinden van een verzadigde stroom in N_x.
  - De initialisatie is $O(m) \leq O(n^2)$.
  - Per iteratiestap wordt het volgende gedaan met de pijlen van N_x (dit werk domineert al het andere).
    (i) Het bepalen van r en d:
    Aan het begin van een deel-iteratie worden de getallen d[i] berekend; dit kost $O(n)$ en er zijn $O(n)$ deel-iteraties.
    De veranderingen in $d^+[i]$ en $d^-[i]$ bijgehouden via berekeningen met de pijlen in PULL/PUSH en REMOVE.
    (ii) De PUSH en PULL procedures:
    Als een hoeveelheid aan een pijl wordt toegekend, dan wordt òf de capaciteit volledig vol gemaakt (deze pijlen worden verder niet meer bekeken) òf de capaciteit is voldoende om alles wat nog moet worden doorgestuurd op te nemen.
    Bekijk nu de totale hoeveelheid werk per iteratie, uitgesplitst over het volmaken en het gedeeltelijk vullen. Iedere keer als dit gebeurt is de hoeveelheid werk $O(1)$, nl. aanpassen van capaciteiten, van de y en de d^+ of d^-.

Volmaken: een pijl wordt hoogstens één keer volgemaakt tijdens een iteratie; er zijn $O(n^2)$ pijlen, dus de hoeveelheid werk is $O(n^2)$.

Het gedeeltelijk vullen wordt vanuit één knooppunt maar hoogstens één keer gedaan voor de PUSH en PULL operaties tijdens één deel-iteraties.

Er zijn maximaal n deel-iteraties en n knooppunten: de totale hoeveelheid werk per iteratie is $O(n^2)$.

(iii) De REMOVE procedures:

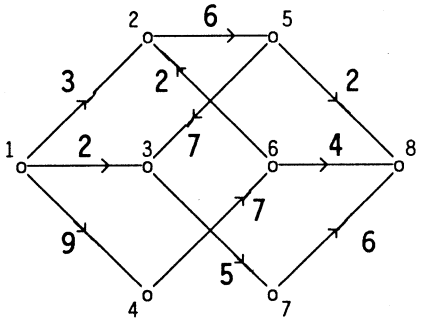
De REMOVE procedure wordt per iteratie hoogstens n keer uitgevoerd (er verdwijnt iedere keer een knooppunt). Als knooppunt v_j wordt verwijderd is de hoeveelheid werk $O(n)$ bij een geschikte (dubbele zoals bij SSC) implementatie van SUCC[k] en PRED[k].

De totale hoeveelheid werk van de REMOVE procedures is per iteratie $O(n^2)$.

* Stap 3b: Het aanpassen van de stroom x kost $O(n^2)$.
* Stap 3c: De procedure LAYER is een aanpassing van het zijwaarts zoeken. Het is direct in te zien dat de complexiteit daarvan $O(n^2)$ is. □

Opgave 4.13

Bepaal een maximale stroom en een (1,8)-snede van minimale capaciteit voor nevenstaand netwerk met de DMKM methode.



Opgave 4.14*

Beschouw een netwerk waarin alle capaciteiten gelijk aan 1 zijn.

a. Toon aan dat één iteratie van de DMKM methode in $\mathcal{O}(m)$ kan worden uitgevoerd.

b. Toon aan dat voor ieder tweetal V_i en V_{i+1} geldt dat het aantal knooppunten in minstens één van de twee groter dan of gelijk is aan $\sqrt{w}$, met w = waarde van de maximale stroom.

c. Toon aan dat ieder pad van v_1 naar v_n maximaal $2n/\sqrt{w}$ pijlen bevat.

d. Toon aan dat de DMKM methode complexiteit $\mathcal{O}(m \cdot n^{2/3})$ heeft.

Opgave 4.15*

Beschouw een enkelvoudig netwerk, d.z.w. een netwerk waarin alle capaciteiten gelijk aan 1 zijn en met bovendien $\min[\delta^+(v),\delta^-(v)] \leq 1$ voor alle knooppunten.

a. Toon aan dat V_i minstens w elementen bevat, met w de waarde van de maximale stroom, voor i = 2,3,...,p-1.

b. Toon aan dat de DMKM methode complexiteit $\mathcal{O}(m \cdot \sqrt{n})$ heeft.

5. Stromen met onder- en bovengrenzen

Literatuur:
Hoofdstuk 7 in:
R.G.Busacker and T.L.Saaty: "Finite graphs and networks: an introduction with applications, McGraw-Hill, New York (1965).

In deze paragraaf veronderstellen we dat een stroom x toelaatbaar is als:

$$0 \leq a_{ij} \leq x_{ij} \leq b_{ij} \text{ voor alle } (i,j) \in A.$$

We zullen laten zien dat het vinden van een toelaatbare stroom in netwerk N met ondergrenzen a en bovengrenzen b overeenkomt met het vinden van een maximale stroom in een geassocieerd netwerk $\bar{N} = (\bar{V},\bar{A})$ met bovengrenzen $\bar{b}$ en ondergrenzen 0, waarbij:

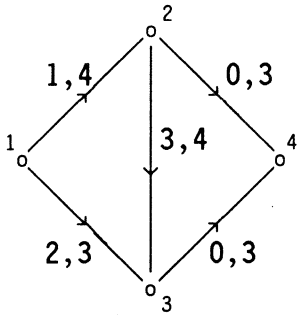
$\bar{V} = V \cup s \cup t$ (dus met twee extra knooppunten);

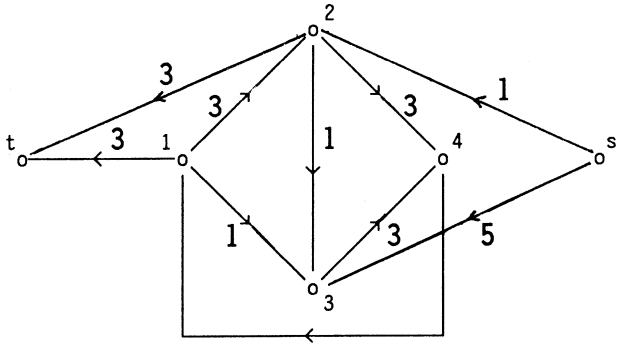
$\bar{A} = A \cup (n,1) \cup A_s \cup A_t$;

$A_s = \{(s,j) \mid j \in V \text{ met } \Sigma_i a_{ij} > 0\}$; $A_t = \{(i,t) \mid i \in V \text{ met } \Sigma_j a_{ij} > 0\}$;

$\bar{b}_{ij} = b_{ij} - a_{ij}$, $(i,j) \in A$; $\bar{b}_{sj} = \Sigma_i a_{ij}$, $(s,j) \in A_s$; $\bar{b}_{it} = \Sigma_j a_{ij}$, $(i,t) \in A_t$

Voorbeeld 4.7





Bij de pijlen staan de getallenparen (a_{ij},b_{ij}).

Een toelaatbare stroom $\bar{x}$ van s naar t in $\bar{N}$ heet een verzadigingsstroom als $\bar{x}_{sj} = \bar{b}_{sj}$ voor alle (s,j) en $x_{it} = \bar{b}_{it}$ voor alle (i,t), d.w.z. als de waarde van $\bar{x}$ gelijk is aan $\Sigma_i\Sigma_j\, a_{ij}$.

Omdat de ondergrenzen 0 zijn, kan met een van de methoden uit de vorige paragrafen een maximale stroom $\bar{x}$ in $\bar{N}$ worden bepaald. Hiermee is direct in te zien of $\bar{x}$ een verzadigingsstroom is. Of $\bar{x}$ al of niet een verzadigingsstroom is bepaalt het al of niet bestaan van een toelaatbare stroom in N, zoals de volgende stellingen aantonen.

Stelling 4.9

Als $\bar{x}$ een verzadigingsstroom is in $\bar{N}$, dan is x gedefiniëerd door $x_{ij} := \bar{x}_{ij} + a_{ij}$, $(i,j) \in A$, een toelaatbare stroom in $\bar{N}$ met waarde $\bar{x}_{n1}$.

Bewijs

Uit $0 \leq \bar{x}_{ij} \leq b_{ij} - a_{ij}$ volgt: $a_{ij} \leq x_{ij} \leq b_{ij}$ voor alle $(i,j) \in A$.

Voor $i \neq 1,n$ geldt:

$$\begin{aligned}\Sigma_{j\in V} x_{ij} - \Sigma_{j\in V} x_{ji} &= \Sigma_{j\in V} (\bar{x}_{ij} + a_{ij}) - \Sigma_{j\in V} (\bar{x}_{ji} + a_{ji}) \\ &= [\Sigma_{j\in V} \bar{x}_{ij} + \bar{b}_{it}] - [\Sigma_{j\in V} \bar{x}_{ji} + \bar{b}_{si}] \\ &= [\Sigma_{j\in V} \bar{x}_{ij} + \bar{x}_{it}] - [\Sigma_{j\in V} \bar{x}_{ji} + \bar{x}_{si}] \\ &= \Sigma_{j\in \bar{V}} \bar{x}_{ij} - \Sigma_{j\in \bar{V}} \bar{x}_{ji} = 0.\end{aligned}$$

Voor $i = 1$ geldt:

$$\begin{aligned}\Sigma_{j\in V} x_{1j} - \Sigma_{j\in V} x_{j1} &= \Sigma_{j\in V} (\bar{x}_{1j} + a_{1j}) - \Sigma_{j\in V} (\bar{x}_{j1} + a_{j1}) \\ &= [\Sigma_{j\in V} \bar{x}_{1j} + \bar{x}_{1t}] - [\Sigma_{j\in V} \bar{x}_{j1} + \bar{x}_{s1}] + \bar{x}_{n1} - \bar{x}_{n1} \\ &= \Sigma_{j\in \bar{V}} \bar{x}_{1j} - \Sigma_{j\in \bar{V}} \bar{x}_{j1} + \bar{x}_{n1} = \bar{x}_{n1}.\end{aligned}$$

Analoog geldt voor $i = n$: $\Sigma_{j\in V} x_{nj} - \Sigma_{j\in V} x_{jn} = -\bar{x}_{n1}$. □

Stelling 4.10

Als N een toelaatbare stroom x heeft, dan heeft $\bar{N}$ een verzadigingsstroom $\bar{x}$.

Bewijs

Neem: $\bar{x}_{ij} = x_{ij} - a_{ij}$ voor $(i,j) \in A$; $\bar{x}_{n1}$ = waarde stroom x;

$\bar{x}_{sj} = \Sigma_i a_{ij}$ voor $(s,j) \in A_s$; $\bar{x}_{it} = \Sigma_j a_{ij}$ voor $(i,t) \in A_t$.

Analoog aan Stelling 4.10 is direct in te zien dat:

$(i,j) \in \bar{A}$: $0 \leq \bar{x}_{ij} \leq \bar{b}_{ij}$;

$1 \leq i \leq n$: $\Sigma_{j\in \bar{V}} \bar{x}_{ij} - \Sigma_{j\in \bar{V}} \bar{x}_{ji} = 0$;

$i = s$: $\Sigma_{j\in \bar{V}} \bar{x}_{sj} = \Sigma_j \Sigma_i a_{ji} \geq 0$;

$i = t$: $\Sigma_{i\in \bar{V}} \bar{x}_{it} = \Sigma_i \Sigma_j a_{ij} \geq 0$.

Hieruit volgt dat $\bar{x}$ een verzadigingsstroom is. □

Indien we via $\bar{N}$ een toelaatbare stroom x voor N hebben gevonden, dan kan een maximale stroom worden gevonden met een kleine aanpassing van de methode van Ford en Fulkerson of van de DMKM methode. De aanpassing betreft het begrip achterwaartse pijl. In plaats van $x_{ij} > 0$ moet nu voor een achterwaartse pijl gelden dat $x_{ij} > a_{ij}$. Samenvattend luidt het algoritme om een maximale stroom in een netwerk met onder- en bovengrenzen te vinden als volgt.

Algoritme

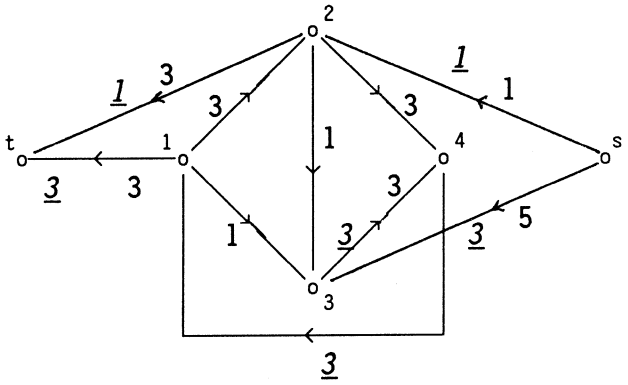
stap 1: Construeer het geassocieerde netwerk $\bar{N}$ en bepaal daarin een maximale stroom $\bar{x}$.

stap 2: Als $\bar{x}$ geen verzadigingsstroom is: er is geen toegelaten stroom (STOP).
Als $\bar{x}$ wel verzadigingsstroom is: ga naar 3.

stap 3: Bepaal een maximale stroom in N uitgaande van de toegelaten stroom
$x_{ij} := \bar{x}_{ij} + a_{ij}$, $(i,j) \in A$.

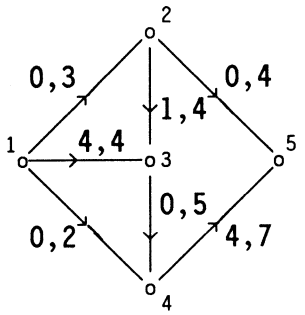
Voorbeeld 4.7 (vervolg)

In nevenstaand netwerk $\bar{N}$ is een maximale stroom aangegeven (de onderstreepte getallen). Dit is geen verzadigingsstroom. Het oorspronkelijke netwerk heeft dus geen toelaatbare stroom.



Opgave 4.16

Ga na of nevenstaand netwerk, waarin bij iedere pijl de onder- en bovengrens staat, een toelaatbare stroom heeft.
Zo ja, bepaal een maximale stroom.

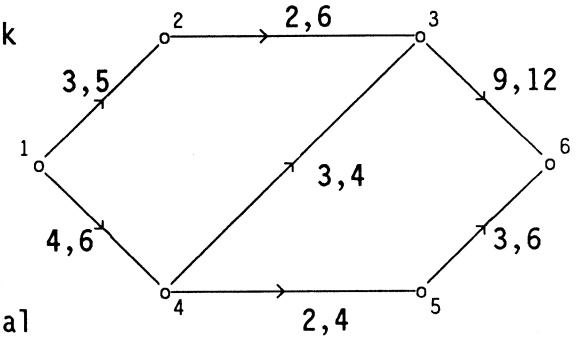


Opgave 4.17

Beschouw het hiernaast getekende netwerk met onder- en bovengrenzen.

a. Toon aan dat dit netwerk geen toelaatbare stroom heeft.

b. Verander de ondergrens van de pijl (3,6) in 5. Toon aan dat het netwerk een toelaatbare stroom heeft en bepaal een maximale stroom.



Opgave 4.18

Beschouw een netwerk N met onder- en bovengrenzen a_{ij} resp. b_{ij}, en veronderstel dat N een toelaatbare stroom heeft. Voor een (1,n)-snede (W,V-W) definiëren we de capaciteit c(W) door:

$$c(W) = \Sigma_{i\in W}\Sigma_{j\notin W}\, b_{ij} - \Sigma_{i\notin W}\Sigma_{j\in W}\, a_{ij}.$$

Toon het volgende aan:

De waarde van een maximale stroom is gelijk aan de capaciteit van een (1,n)-snede met minimale capaciteit.

Opgave 4.19

a. Beschijf een methode om een minimale stroom te bepalen in een netwerk met onder- en bovengrenzen.

b. Pas de methode toe op het netwerk uit opgave 4.17b.

c. Geeft de methode om een toelaatbare stroom te vinden altijd een minimale stroom ?

Opgave 4.20*

Beschouw een netwerk met onder- en bovengrenzen.

a. Formuleer het probleem om een maximale stroom te bepalen als een lineair programmeringsprobleem.

b. Stel het duale probleem op en geef er een interpretatie aan.

c. Bewijs met de dualiteitsstelling van de lineaire programmering dat de waarde van de maximale stroom gelijk is aan de capaciteit van de (1,n)-snede met de kleinste capaciteit (zie ook opgave 4.18).

d. Pas de onderdelen a,b en c aan voor het geval dat we een minimale toelaatbare stroom willen bepalen.

Opgave 4.21*

Zij N een netwerk met ondergrenzen a en bovengrenzen b zódanig dat positieve ondergrenzen alleen voorkomen bij pijlen (1,j) en (i,n). Knooppunt 1 heeft alleen uitgaande en knooppunt n alleen binnenkomende pijlen.

a. Bewijs dat N een toelaatbare stroom heeft d.e.s.d. als:

$\Sigma_{i\in W}\Sigma_{j\notin W}\, b_{ij} \geq \Sigma_{j\in W}\, a_{1j}$ voor iedere W met $1 \notin W$ en $n \notin W$;

$\Sigma_{i\in W}\Sigma_{j\notin W}\, b_{ij} \geq \Sigma_{i\notin W}\, a_{in}$ voor iedere W met $1 \in W$ en $n \in W$.

b. Bewijs dat N een toelaatbare stroom heeft d.e.s.d als er twee stromen x^1 en x^2 bestaan waarvoor geldt:

$0 \leq x^1_{ij} \leq b_{ij}$ en $0 \leq x^2_{ij} \leq b_{ij}$ voor alle $(i,j) \in A$;

$a_{1j} \leq x^1_{1j}$ voor alle $(1,j) \in A$; $a_{in} \leq x^2_{in}$ voor alle $(i,n) \in A$.

6. Minimale kostenstromen

Literatuur:
Hoofdstuk 7 in:
R.G.Busacker and T.L.Saaty: "Finite graphs and networks: an introduction with applications, McGraw-Hill, New York (1965).

In dit model hebben we een netwerk met ondergrenzen 0, capaciteiten b_{ij} en een kostenfunctie c_{ij}. Dit laatste houdt in dat per eenheid van vervoer over de pijl (i,j) kosten c_{ij} worden berekend. Bij een stroom x zijn de totale kosten

$$c(x) = \Sigma_i \Sigma_j \, c_{ij} x_{ij}.$$

We zullen een algoritme opstellen dat de volgende twee problemen oplost:
a. Bepaal een stroom met waarde w en met minimale kosten.
b. Bepaal een stroom met maximale waarde en minimale kosten.

De oplosmethode lijkt sterk op de methode van Ford en Fulkerson. In plaats van een groeiketen van v_1 naar v_n moet nu echter een kortste groeiketen van v_1 naar v_n worden gevonden t.a.v. de volgende lengtefunctie:

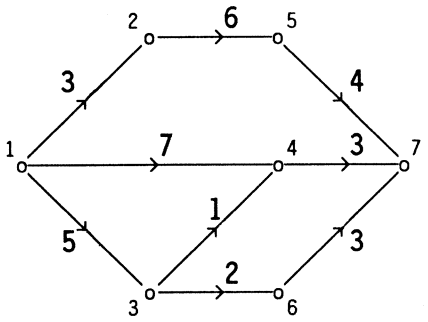
$$(4.14) \quad l_{ij} = \begin{cases} c_{ij} & \text{als (i,j) een voorwaartse pijl is} \\ -c_{ji} & \text{als (i,j) een achterwaartse pijl is} \end{cases}$$

We nemen aan dat voor iedere i en j hoogstens één van de twee pijlen (i,j) en (j,i) tot A behoort. We mogen ook wel aannemen dat voor iedere pijl $(i,j) \in A$ er een voorwaartse pijl (i,j) en ook een achterwaartse pijl (j,i) is. Als $x_{ij} = 0$ nemen we $l_{ji} = \infty$, en als $x_{ij} = b_{ij}$ nemen we $l_{ij} = \infty$.
Het algoritme is als volgt (k is de waarde van de stroom; indien we een maximale stroom met minimale kosten wensen, dan moet $w = \infty$ worden genomen).

stap 1: Start met $x = 0$, $c = 0$ en $k = 0$.
stap 2: a. Bepaal een kortste groeiketen P van v_1 naar v_n t.a.v. de lengtefunctie (4.14), en laat l(P) de lengte van dit pad zijn.
(Als er geen groeiketen is van v_1 naar v_n: x is een maximale kosten stroom met kosten c en waarde k en STOP).
b. Bepaal de maximale hoeveelheid Δ die over P kan worden vervoerd.
c. Als $k + \Delta \geq w$: $\Delta := w - k$.
stap 3: a. Pas x aan; $c := c + \Delta \cdot l(P)$; $k := k + \Delta$.
b. Als $k = w$: x een minimale kostenstroom met waarde w en STOP.
Anders: ga naar stap 2a.

Voorbeeld 4.8

Beschouw nevenstaand netwerk waarin de kosten zijn aangegeven. We nemen aan dat alle bovengrenzen 2 zijn. We zoeken een minmale kostenstroom met waarde 5.



Iteratie 1

x = 0; c = 0; k = 0.

Bij de start is de lengtefunctie gelijk aan de kosten c.

Het kortste pad van 1 naar 7 is: P = [1,3,4,7]; l(P) = 9; Δ = 2.

De nieuwe lengtefunctie staat in onderstaand netwerk.

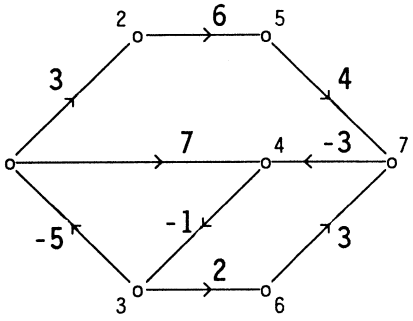
c = 2·9 = 18; k = 2.

Iteratie 2

Het korste pad van 1 naar 7 is: P = [1,4,3,6,].

l(P) = 11; Δ = 2; c = 18 + 2·11 = 40;

k = 2 + 2 = 4.



Iteratie 3

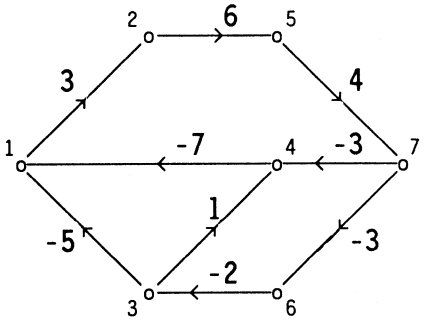
Het korste pad van 1 naar 7 is: P = [1,2,5,7].

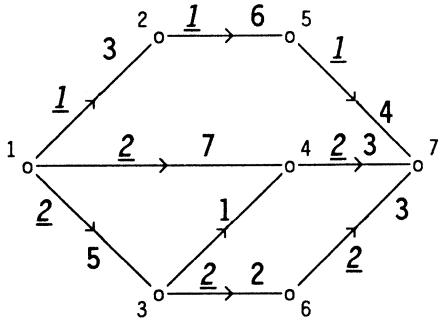
l(P) = 13; Δ = 2; Δ = 1; c = 40 + 1·13 = 53;

k = 4 + 1 = 5.

x is een minimale kostenstroom met waarde 5.

Deze stroom staat hieronder (onderstreept).





Hieronder staat de procedure om een minimale kostenstroom met waarde w te bepalen (als we w = ∞ nemen, dan wordt een maximale stroom met minimale kosten bepaald).

De procedure start met de stroom x = 0. De kosten zijn dan tevens de lengten. In CHAINSEARCH wordt een kortste pad van v_1 naar v_n bepaald. Hiervoor wordt het algoritme van Bellman en Ford gebruikt (versie 2). De lengte van v_1 naar v_i wordt bijgehouden in een arrray u[i].

In CHANGE worden de stroom en de lengte-functie aangepast. Zodra de stroom de waarde w heeft bereikt wordt STOP = 1 en eindigt het algoritme. Als de stroom maximaal is, dan wordt STOP = 2 en eindigt het algoritme eveneens.

We nemen aan dat de datastructuur wordt gegeven door de lijsten:

$L^+[i] = \{j \mid (i,j) \in A\}$ en $L^-[i] = \{j \mid (j,i) \in A\}$ voor i = 1,2,...,n.

```
procedure MINFLOW(w);
  var i,j,k,c,stop: integer;
      u: array [1:n] of real;
      PRED: array [1:n] of integer;
      x,l: array [1:n,1:n] of integer;
  procedure CHAINSEARCH;
    var ready: boolean;
        i,j,k: integer;
  begin u[1] := 0; PRED[1] := 0; k := 1;
        for j := 2 to n do if j ∈ L+[1] then
            begin u[j] := l[1,j]; PRED[j] := 1 end;
        repeat k := k+1; ready := true;
               for j := 2 to n do
                   begin for all i ∈ L+[j] ∪ L-[j] do
                             if u[i] + l[i,j] < u[j] then
                                begin ready := false; PRED[j] := i;
                                      u[j] := u[i] + l[i,j]
                                end
                   end
        until k = n-1 or ready;
        if u[n] = ∞ then stop := 2
  end CHAINSEARCH;
```

```
  procedure CHANGE;
    var Δ,p,q: integer;
  begin Δ := ∞; q := n; p := PRED[q];
        while p <> 0 do
              begin if q ∈ L+[p] then Δ := min(Δ, b[p,q] - x[p,q])
                    else begin Δ := min(Δ, x[q,p]) end;
                    q := p; p := PRED[q]
              end;
        Δ := min(Δ, w - k); k := k + Δ; if k = w then stop := 1;
        c := c + Δ × u[n];
        q := n; p := PRED[q];
        while p <> 0 do
              begin if q ∈ L+[p] then
                       begin x[p,q] := x[p,q] + Δ; l[q,p] := -c[p,q];
                             if x[p,q] = b[p,q] then l[p,q] := ∞
                       end
                    else begin x[q,p] := x[q,p] - Δ; l[q,p] := c[q,p];
                             if x[q,p] = 0 then l[p,q] := ∞
                         end;
                    q := p; p := PRED[q]
              end
  end CHANGE;

begin stop := 0; k := 0; c := 0;
      for i := 1 to n do
          for all j ∈ L+[i] do
              begin x[i,j] := 0; l[i,j] := c[i,j]; l[j,i] = ∞ end;
      repeat CHAINSEARCH;
             if u[n] < ∞ then CHANGE
      until  stop <>0
end MINFLOW.
```

We zullen nu ingaan op de correctheid van deze methode.
Laat N een netwerk zijn met daarin een (niet noodzakelijkerwijs toegelaten) stroom x. De gerichte graaf $N_x^* = (V^*,A^*)$ is als volgt gedefiniëerd:

V^*: dezelfde knooppunten als het netwerk (V).

A^*: x_{ij} pijlen van v_i naar v_j als $x_{ij} > 0$;

$-x_{ij}$ pijlen van v_j naar v_i als $x_{ij} < 0$.

Zij P een enkelvoudig pad of ronde in N_x^*. Hierbij hoort een <u>elementaire pad (of ronde)-stroom</u> x^P in N, gedefiniëerd door:

$$x_{ij}^P = \begin{cases} 1 \text{ als } (v_i,v_j) \in P \text{ en } x_{ij} > 0 \\ -1 \text{ als } (v_j,v_i) \in P \text{ en } x_{ij} < 0 \\ 0 \text{ als noch } (v_i,v_j) \text{ noch } (v_j,v_i) \text{ tot P behoort} \end{cases}$$

Het is eenvoudig in te zien x^P inderdaad een stroom is (d.w.z. dat behalve het begin- en eindpunt van P in ieder tussenpunt geldt "stroom in = stroom uit").

<u>Stelling 4.11</u>

Zij x een stroom met waarde w.

Dan geldt: $x = \Sigma_{i=1}^{w} y^i + \Sigma_{i=w+1}^{q} y^i$, met y^1 t/m y^w elementaire padstromen van v_1 naar v_n en y^{w+1} t/m y^q elementaire rondestromen.

<u>Bewijs</u>

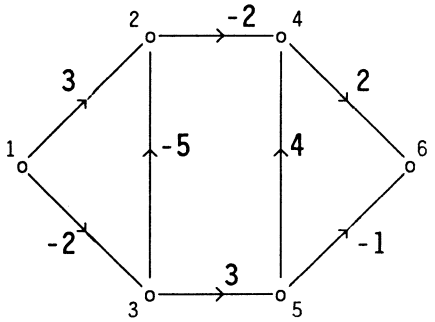
Beschouw N_x^*: $\delta^+(v_i) = \delta^-(v_i)$ voor alle $i \neq 1,n$;

$$\delta^+(v_1) - \delta^-(v_1) = \delta^-(v_n) - \delta^+(v_n) = w.$$

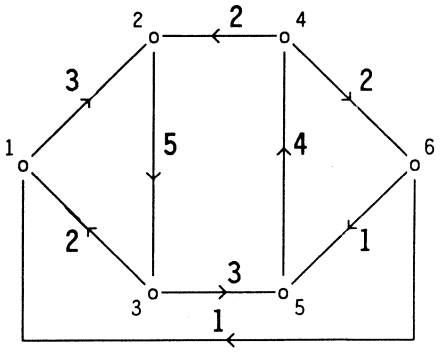
Door w pijlen van v_n naar v_1 toe te voegen ontstaat een Euler-graaf: de pijlen vormen dus één ronde. Haal de toegevoegde pijlen weer weg, dan ontstaan er w paden van v_1 naar v_n, plus eventueel een aantal ronden. Bij de paden horen elementaire padstromen, en bij de ronden elementaire rondestromen. □

<u>Voorbeeld 4.9</u>

Beschouw nevenstaande graaf met daarin aangegeven een stroom x met waarde 1. Dupliceer de pijlen overeenkomstig de waarde van de stroom en verander daarbij de richting als de stroom over een pijl negatief is. Voeg tevens de pijl (6,1) toe.



Alle pijlen één ronde, nl.
[1,2,3,1,2,3,1,2,3,5,4,2,3,5,4,2,3,5,4,6,5,4,6,1].
Door de pijl (6,1) weg te laten valt de ronde uiteen in:



ronde [1,2,3,1] komt twee keer voor;
ronde [2,3,4,3,2] komt twee keer voor;
ronde [5,4,6,5] komt één keer voor;
pad [1,2,3,5,4,6] komt één keer voor.

Stelling 4.12

Zij x een toelaatbare stroom met waarde k. Deze stroom induceert een lengte-functie l in de graaf met voorwaartse en achterwaartse pijlen.
Er geldt: x is een minimale kostenstroom met waarde k d.e.s.d. als de lengte van iedere ronde niet-negatief is.

Bewijs

⇒ Stel C is een enkelvoudige ronde met negatieve lengte. Dan bevat C geen pijl met lengte ∞, dus er kan een eenheid over C worden vervoerd, en laat y de daarbij bijbehorende stroom zijn. Dan is x+y ook een stroom met waarde k en kosten $c(x+y) = c(x) + \Sigma_i\Sigma_j\, c_{ij}y_{ij}$. Laat $A^+(C)$ de verz. van voorwaartse pijlen van C zijn, en $A^-(C)$ de verz. achterwaartse pijlen.

Als $(i,j) \in A^+(C)$, dan geldt: $l_{ij} = c_{ij}$ en $y_{ij} = +1$.

Als $(i,j) \in A^-(C)$, dan geldt: $l_{ij} = -c_{ji}$ en $y_{ji} = -1$.

Dus $\Sigma_i\Sigma_j\, c_{ij}y_{ij} = \Sigma_{(i,j)\in A^+(C)}\, c_{ij}y_{ij} + \Sigma_{(i,j)\in A^-(C)}\, c_{ji}y_{ji}$

$= \Sigma_{(i,j)\in A^+(C)}\, l_{ij} + \Sigma_{(i,j)\in A^-(C)}\, l_{ij} = l(C)$.

Hieruit volgt: $c(x+y) = c(x) + l(C) < c(x)$: x is geen minimale kostenstroom met waarde k: tegenspraak.

⇐ Veronderstel dat x′ een minimale kostenstroom is met waarde k. Dan is x′- x een stroom met waarde 0, en volgens Stelling 4.11 kunnen we schrijven:
$x' - x = \Sigma_{p=1}^{q}\, y^p$ met y^p een elementaire rondestroom, zeg bij C^p, $1 \le p \le q$.
Merk op dat dit ronden C^p zijn in $N^*_{x'-x}$.
Laat $I_1^p = \{(i,j) \in C^p \mid (x'-x)_{ij} > 0\}$ en $I_2^p = \{(j,i) \in C^p \mid (x'-x)_{ij} < 0\}$.
Als $(i,j) \in I_1^p$, dan geldt: $b_{ij} \ge x'_{ij} > x_{ij}$, dus (i,j) is voorwaartse pijl t.o.v. stroom x, $l_{ij} = c_{ij}$ en $y^p_{ij} = +1$.
Als $(j,i) \in I_2^p$, dan geldt: $0 \le x'_{ij} < x_{ij}$, dus (j,i) is achterwaartse pijl t.o.v. stroom x, $l_{ji} = -c_{ij}$ en $y^p_{ij} = -1$.
Alle pijlen van C^p komen dus ook (met eindige lengte) voor t.o.v. x:

$$c(y^p) = \Sigma_{(i,j)\in I_1^p}\, c_{ij}y^p_{ij} + \Sigma_{(j,i)\in I_2^p}\, c_{ij}y^p_{ij}$$

$$= \Sigma_{(i,j)\in I_1^p}\, l_{ij} + \Sigma_{(j,i)\in I_2^p}\, l_{ji} = l(C^p) \ge 0 \text{ voor alle } p.$$

Hieruit volgt: $c(x') = c(x) + \Sigma_{p=1}^{q}\, c(y^p) \ge c(x)$: x minimale kostenstroom. □

Stelling 4.13

Zij x een minimale kostenstroom met waarde k. Laat l de bij x behorende lengte-functie zijn, zij P het kortste pad t.o.v. l van v_1 naar v_n en laat Δ de hoeveelheid zijn die over P kan worden vervoerd.
Zij x^* de aanpassing van x m.b.t. P en Δ volgens het algoritme.
Dan is x^* een minimale kostenstroom met waarde $k + \Delta$.

Bewijs

Het is duidelijk dat x^* een toegelaten stroom is met waarde $k+\Delta$ en met kosten $c(x^*) = c(x) + \Delta \cdot l(P)$.
Veronderstel dat x' een minimale kostenstroom is met waarde $k + \Delta$. Dan is $x'-x$ een stroom met waarde Δ, en volgens Stelling 4.11 kunnen we schrijven:
$x' - x = \Sigma_{s=1}^{\Delta} y^s + \Sigma_{s=\Delta+1}^{q} y^s$ met y^1 t/m y^{Δ} elementaire padstromen bij paden P^1 t/m P^{Δ} van v_1 naar v_n en $y^{\Delta+1}$ t/m y^q elementaire rondestromen, zeg bij ronden C^s, $\Delta+1 \leq p \leq q$. Merk op dat dit paden en ronden zijn in $N^*_{x'-x}$.

Analoog aan het bewijs van Stelling 4.12 geldt ook hier:

$c(y^s) = l(P^s)$, $1 \leq s \leq \Delta$ en $c(y^s) = l(C^s)$, $\Delta+1 \leq s \leq q$.

Hieruit volgt:

$$c(x') = c(x) + \Sigma_{s=1}^{\Delta} l(P^s) + \Sigma_{s=\Delta+1}^{q} l(C^s)$$
$$\geq c(x) + \Delta \cdot l(P) = c(x^*),$$

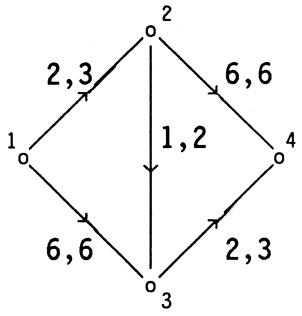
d.w.z. dat x^* een minimale kostenstroom met waarde $k + \Delta$ is. □

Gevolgen

1. Het algoritme om een minimale kostenstroom te bepalen is correct.
2. Aangezien de waarde van de stroom per iteratie met minstens één toeneemt, is het aantal iteraties $O(w)$.
   Per iteratie worden de procedures CHAINSEARCH en CHANGE uitgevoerd.
   CHAINSEARCH is de methode van Bellman en Ford en heeft complexiteit $O(n^3)$.
   In CHANGE worden de stroom en de lengte-functie aangepast over keten P.
   Aangezien P maximaal n knoopunten bevat is de complexiteit van CHANGE $O(n)$.
   (Met een iets aangepaste implementatie, zoals bij MAXFLOW, om in $O(1)$ na te gaan of een pijl (p,q) voorwaarts of achterwaarts is).
   De totale complexiteit is dus $O(w \cdot n^3)$.

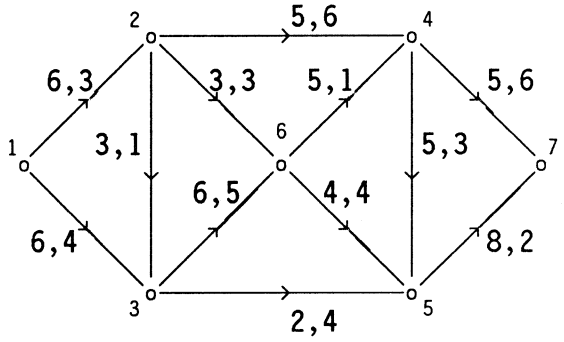
Opgave 4.22

Bepaal een maximale stroom met minimale kosten in nevenstaand netwerk.
Het getallenpaar bij de pijlen geeft b_{ij},c_{ij} aan.



Opgave 4.23

Bepaal een maximale stroom met minimale kosten in nevenstaand netwerk.
Het getallenpaar bij de pijlen geeft b_{ij},c_{ij} aan.



Opgave 4.24

Veronderstel dat we een minimale kostenstroom met waarde k hebben bepaald.
Ga na hoe een minimale kostenstroom met dezelfde waarde k kan worden gevonden (zonder het algoritme van vooraf aan opnieuw toe te passen) bij de volgende veranderingen:

a. Een extra pijl wordt toegevoegd.
b. De capaciteit van een pijl wordt verlaagd.
c. De kosten van een pijl worden verhoogd.

Opgave 4.25

Zij x een maximale stroom en (W,V-W) een minimale (1,n)-snede.
Toon aan dat x een minimale kostenstroom is d.e.s.d. als iedere ronde die geheel in W of geheel in V-W ligt een niet-negatieve lengte heeft t.o.v. de lengte-functie die door x wordt geïnduceerd.

Opgave 4.26

Beschouw het probleem om een maximale stroom met minimale kosten te bepalen in een netwerk met onder- en bovengrenzen.
Beschrijf een methode om een dergelijk probleem op te lossen.

Opgave 4.27*

Beschouw het volgende transportprobleem:

Er zijn m voorraadplaatsen en in plaats i is een voorraad a_i ($1 \leq i \leq m$). Daarnaast zijn er n bestemmingen en in bestemming j is een hoeveelheid b_j nodig ($1 \leq j \leq n$), waarbij $\Sigma_{i=1}^{m} a_i = \Sigma_{j=1}^{n} b_j$.
Verder is gegeven dat de transportkosten c_{ij} (per eenheid van vervoer) van voorraadplaats i naar bestemming j.
Gevraagd wordt een transportschema op te stellen met minmale kosten.

a. Formuleer dit transportprobleem als een minimale kosten probleem.
b. Construeer bij een minimale kosten probleem een transportprobleem zódat een toelaatbare stroom voor het minimale kosten probleem overeenkomt met een toelaatbaar transportschema met dezelfde kosten.

Opgave 4.28*

Beschouw het volgende probleem:

Een catering bedrijf heeft op n opeenvolgende dagen schone tafellakens nodig, zeg $a_1, a_2, \ldots, a_n$.
Het bedrijf kan nieuwe tafellakens kopen tegen een prijs van p guldens per stuk, maar het kan ook gebruikte tafellakens laten wassen.
Er is een snelle was die 1 dag duurt (een tafellaken gebruikt op dag i, is dan weer te gebruiken op dag i+2) en q gulden per tafellaken kost, en een langszame was van 2 dagen met kosten r gulden per stuk.
Er geldt: $p > q > r > 0$.
Aan het eind van iedere dag moet voor de gebruikte tafellakens het volgende worden beslist: hoeveel gaan naar de snelle was, hoeveel naar de langszame en hoeveel worden niet meer gebruikt.
Gevraagd wordt een strategie die de kosten minimaliseert.

a. Formuleer dit probleem als een minimale kosten probleem.
b. Los dit probleem op voor de volgende situatie:
$n = 4$; $a_1 = 3$, $a_2 = 6$, $a_3 = 2$, $a_4 = 7$; $p = 10$, $q = 6$ en $r = 2$.

Opgave 4.29*

a. Formuleer het probleem om een minimale kostenstroom met waarde w te vinden als een lineair programmeringsprobleem.
b. Stel het duale probleem op en geef de orthogonaliteitsrelaties.
c. Zij x een minimale kostenstroom met waarde w.
   Toon aan m.b.v. de orthogonaliteitsrelaties dat de lengte van iedere ronde niet-negatief is.
d. Zij x een toelaatbare basisoplossing van het LP-probleem zódanig dat $\{(v_i,v_j) \mid x_{ij}$ en y_{ij} in de basis$\}$ een voortbrengende gerichte boom met wortel v_1 is (zie paragraaf 2 van dit hoofdstuk).
   Veronderstel dat de lengte van iedere ronde t.a.v. de door x geïnduceerde lengte-functie niet negatief is.
   Toon aan dat x een minimale kostenstroom is.

Opgave 4.30*

Beschouw in het algoritme om een minimale kostenstroom te bepalen de volgende veranderingen in de lengte-functie l^k tijdens de k-de iteratie:

$$l^k_{ij} = \begin{cases} c_{ij} + \pi^k_i - \pi^k_j & \text{als (i,j) een voorwaartse pijl is} \\ -c_{ji} + \pi^k_i - \pi^k_j & \text{als (i,j) een achterwaartse pijl is,} \end{cases}$$

waarbij de getallen π^k_i als volgt recursief worden bepaald (i = 1,2,...,n)

$$\begin{cases} \pi^1_i = 0 \\ \pi^k_i = \pi^{k-1}_i + u^{k-1}_i \text{ met } u^{k-1}_i = \text{lengte kortste pad } v_1 \text{ naar } v_i \text{ in iteratie k-1} \end{cases}$$

a. Pas deze gewijzigde methode toe op het probleem uit opgave 4.22.
b. Toon aan dat een pad P van v_1 naar v_n een kortste pad is t.o.v. de oude lengte-functie d.e.s.d. als P een kortste pad is t.o.v. de nieuwe lengte-functie.
c. Toon aan dat $l^k_{ij} \geq 0$ voor alle i,j en k.
d. Beschrijf een methode met complexiteit $O(w \cdot n^2)$ om een minimale kostenstroom met waarde w te vinden.

7. Minimale kostenstroom met onder- en bovengrenzen

Literatuur

* Hoofdstuk 10 in:
  M.S.Bazaraa and J.J.Jarvis: "Linear programming and network flows", Wiley, New York (1977).
* Hoofdstuk 4 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds", Holt, Rinehart and Winston, New York (1976).

In deze paragraaf beschouwen we het probleem om een toelaatbare circulatiestroom te vinden met minimale kosten. Aan iedere pijl $(i,j) \in A$ zijn drie gehele getallen toegekend a_{ij}, b_{ij} en c_{ij}, resp. de ondergrens, de bovengrens en de kosten van de pijl. In LP-formulering luidt het probleem (ook nu wordt de geheeltalligheid niet expliciet geëist: in ieder hoekpunt is er automatisch aan voldaan).

$$(4.15) \qquad \min \left\{ \Sigma_i \Sigma_j c_{ij} x_{ij} \;\middle|\; \begin{array}{ll} \Sigma_j x_{ij} - \Sigma_j x_{ji} = 0 & \text{voor } i = 1,2,\ldots,n \\ a_{ij} \le x_{ij} \le b_{ij} & \text{voor alle } (i,j) \in A \end{array} \right\}.$$

Dit probleem is een generalisatie van de volgende reeds eerder besproken problemen:

a. Kortste pad van v_1 naar v_n: Neem $a_{n1} = b_{n1} = 1$; $a_{ij} = 0$, $b_{ij} = 1$ voor alle $(i,j) \neq (n,1)$; $c_{ij} = l_{ij}$ voor alle (i,j).
b. Maximale stroom probleem: Neem voor a_{ij} en b_{ij} de onder- en bovengrenzen; $c_{ij} = 0$ voor alle $(i,j) \neq (n,1)$; $c_{n1} = -1$.
c. Minimale kostenstroom met waarde w: Neem voor a_{ij} en b_{ij} de onder- en bovengrenzen en voor c_{ij} de kosten; $a_{n1} = b_{n1} = w$; $c_{n1} = 0$.

Het duale probllem van (4.15) luidt (u_i is de variabele bij de vergelijking stroom in = stroom uit in knooppunt i; v_{ij} behoort bij $x_{ij} \ge a_{ij}$ en w_{ij} bij $-x_{ij} \ge -b_{ij}$):

$$(4.16) \qquad \max \left\{ \Sigma_i \Sigma_j a_{ij} v_{ij} - \Sigma_i \Sigma_j b_{ij} w_{ij} \;\middle|\; \begin{array}{l} u_i - u_j + v_{ij} - w_{ij} = c_{ij} \text{ voor alle } (i,j) \in A \\ v_{ij}, w_{ij} \ge 0 \text{ voor alle } (i,j) \in A \end{array} \right\}$$

Opmerking

Als (u,v,w) een toegelaten oplossing is van (4.16), dan is ook $(u+\lambda, v+\mu, w+\mu)$ met λ een constante en μ een willekeurige vector (μ_{ij}) een oplossing mits $v+\mu$ en $w+\mu$ niet-negatief zijn. Omdat $b_{ij} \geq a_{ij}$ is een oplossing met μ_{ij} zo klein mogelijk dus het gunstigst.

Dit houdt in dat bij een gegeven u de waarden voor v en w vastliggen:

$$v_{ij} = \max(0,\ c_{ij} - u_i + u_j) \text{ en } w_{ij} = \max(0,\ -c_{ij} + u_i - u_j).$$

Probleem (4.16) is dus in feite een probllem met alleen u als variabelen.

Omdat we vaak werken met $-c_{ij} + u_i - u_j$ nemen we: $z_{ij} := -c_{ij} + u_i - u_j$.

Indien x een stroom is, dan volgt uit de theorie van de lineaire programmering dat x optimaal is voor (4.15) en (u,v,w) optimaal voor (4.16) als voor alle pijlen $(i,j) \in A$ geldt:

(4.17a) $v_{ij} \cdot (x_{ij} - a_{ij}) = 0;$

(4.17b) $w_{ij} \cdot (b_{ij} - x_{ij}) = 0;$

(4.17c) $a_{ij} \leq x_{ij} \leq b_{ij}.$

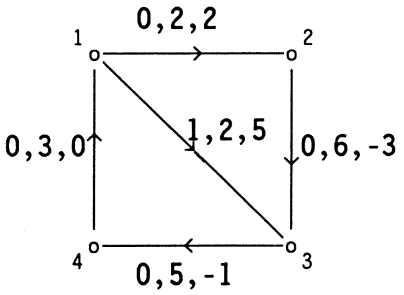
In termen van de variabelen x en z luiden deze optimaliteitsvoorwaarden:

(4.18a) als $z_{ij} < 0$, dan $x_{ij} = a_{ij}$;

(4.18b) als $z_{ij} > 0$, dan $x_{ij} = b_{ij}$;

(4.18c) $a_{ij} \leq x_{ij} \leq b_{ij}.$

Voorbeeld 4.10



Bij iedere pijl staat het drietal a_{ij}, b_{ij}, c_{ij}.

Neem x = 0 en u = 0.

Voor de pijl (1,2) geldt:

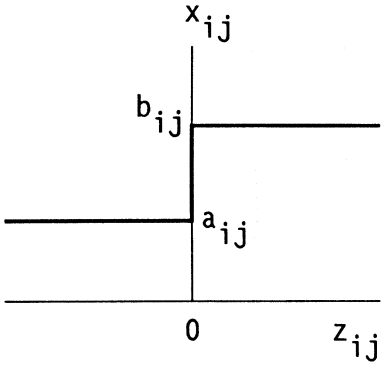
$z_{12} = -2$, dus deze pijl voldoet aan (4.18).

Voor de pijl (3,4) geldt:

$z_{34} = 1$: deze pijl voldoet niet aan (4.17).

Als een pijl voldoet aan (4.18), dan noemen we deze <u>in-orde</u>, en anders heet de pijl <u>niet-in-orde</u>. In onderstaande tabel staat voor ieder van de mogelijkheden voor x en z of de pijl in orde is of niet. Tevens staat dit in een (z,x)-diagram. Alleen op de dikke lijn is de pijl in orde.

	$z_{ij} < 0$	$z_{ij} = 0$	$z_{ij} > 0$
$x_{ij} > b_{ij}$	niet in orde	niet in orde	niet in orde
$x_{ij} = b_{ij}$	niet in orde	in orde	in orde
$a_{ij} < x_{ij} < b_{ij}$	niet in orde	in orde	niet in orde
$a_{ij} = x_{ij}$	in orde	in orde	niet in orde
$a_{ij} > x_{ij}$	niet in orde	niet in orde	niet in orde



Als maat voor hoe goed een pijl voldoet aan de optimaliteitsvoorwaarde (4.18) nemen we het zogenaamde <u>k-getal</u>. Dat is de afstand in vertikale richting die (z_{ij},x_{ij}) heeft tot de "in-orde lijn":

$$(4.19)\qquad k_{ij} = \begin{cases} |x_{ij} - a_{ij}| & \text{als } z_{ij} < 0 \text{ of } z_{ij} = 0 \text{ èn } x_{ij} < a_{ij} \\ |b_{ij} - x_{ij}| & \text{als } z_{ij} > 0 \text{ of } z_{ij} = 0 \text{ èn } x_{ij} > b_{ij} \\ 0 & \text{anders} \end{cases}$$

We zullen nu een methode gaan beschrijven waarbij in opeenvolgende iteraties de k-getallen nooit groter worden en na een eindig aantal iteraties minstens één k-getal kleiner wordt (of ontdekt wordt dat er geen toelaatbare oplossing bestaat). Dit impliceert dat het probleem in een eindig aantal iteraties wordt opgelost.

Tijdens een iteratiestap kiezen we een pijl (p,q) met $k_{pq} > 0$. Allereerst proberen we door x te veranderen k_{pq} kleiner te krijgen zonder andere k_{ij}'s te vergroten. We construeren daartoe een graaf G^* met dezelfde knooppunten als het netwerk en proberen in G^* zoveel mogelijk te vervoeren zódat k_{pq} kleiner wordt en de andere k_{ij}'s niet toenemen. Laat d_{ij} de hoeveelheid zijn die we over (i,j) vervoeren.

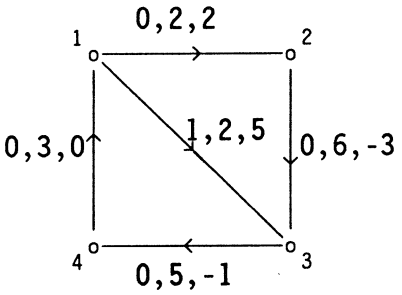
Dan is de constructie van G^* als volgt:

als $z_{ij} < 0$ en $x_{ij} < a_{ij}$: $(i,j) \in G^*$ en $d_{ij} = a_{ij} - x_{ij} = k_{ij}$;
als $z_{ij} < 0$ en $x_{ij} > a_{ij}$: $(j,i) \in G^*$ en $d_{ji} = x_{ij} - a_{ij} = k_{ij}$;
als $z_{ij} = 0$ en $x_{ij} < b_{ij}$: $(i,j) \in G^*$ en $d_{ij} = b_{ij} - x_{ij}$;
als $z_{ij} = 0$ en $x_{ij} > a_{ij}$: $(j,i) \in G^*$ en $d_{ji} = x_{ij} - a_{ij}$;
als $z_{ij} > 0$ en $x_{ij} < b_{ij}$: $(i,j) \in G^*$ en $d_{ij} = b_{ij} - x_{ij} = k_{ij}$;
als $z_{ij} > 0$ en $x_{ij} > b_{ij}$: $(j,i) \in G^*$ en $d_{ji} = x_{ij} - b_{ij} = k_{ij}$.

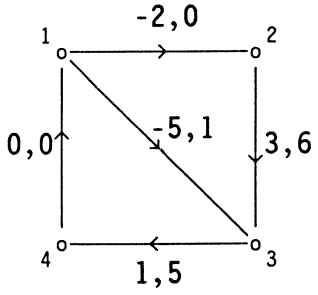
Om een stroom te houden mogen we alleen wat veranderen over een ronde. Indien (p,q) in een ronde C van G^* zit, dan vervoeren we daar zoveel mogelijk over. Dit doen we door in G^* vanuit q via het zijwaarts zoeken p te bereiken (of te ontdekken dat p niet bereikt kan worden). Analoog aan de methode van Ford en Fulkerson houden we bij wat er maximaal vervoerd kan worden.

Voorbeeld 4.10 (vervolg)
Start met x = 0 en u = 0.

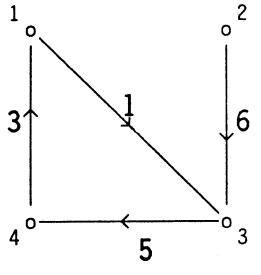
netwerk met a,b,c



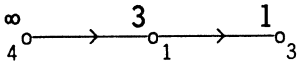
netwerk met z,k



de graaf G^* met d



Neem (p,q) = (3,4).



Over de ronde [4,1,3,4] kan $\Delta = 1$ worden vervoerd.

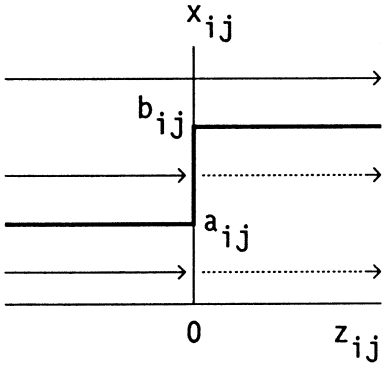
We herhalen de hierboven vermelde stap met (p,q) totdat $k_{pq} = 0$ of totdat er geen ronde in G^* is die (p,q) bevat. In het laatste geval gaan we u aanpassen. Deze aanpassing moet zo zijn dat geen enkel k-getal groter wordt en zodat op zeker moment een ronde met (p,q) ontstaat. Dan kan het eerste deel van de procedure weer worden toegepast.

Dit onderdeel van de methode gaat als volgt:
Zij $W = \{i \mid i \text{ is bereikbaar in } G^* \text{ vanuit } q\}$. Dan is dus $q \in W$ en $p \notin W$.
Beschouw $z_{ij} = -c_{ij} + u_i - u_j$. Als u_i en u_j binnen W met hetzelfde getal veranderen, dan verandert z_{ij} niet en blijven knooppunten die tot W behoren ook daarna in W zitten, want x verandert evenmin. De waarde van k_{ij} met $i,j \in W$ blijft ook hetzelfde. Dit geldt ook binnen V-W.
Laat nu (voor $\lambda > 0$)

$$u_i = \begin{cases} u_i + \lambda & \text{als } i \in W \\ u_i & \text{als } i \notin W \end{cases} \quad \Rightarrow \quad z_{ij} = \begin{cases} z_{ij} + \lambda & \text{voor } i \in W,\ j \notin W \\ z_{ij} - \lambda & \text{voor } i \notin W,\ j \in W \\ 0 & \text{anders} \end{cases}$$

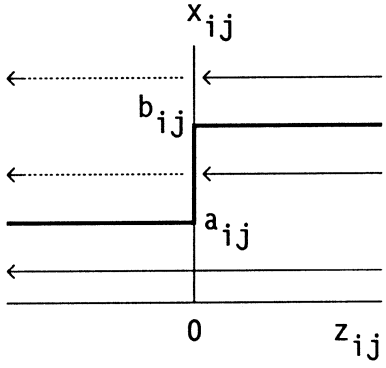
We gaan λ zó bepalen dat de k-getallen niet groter worden en λ zo groot mogelijk is.

a. $i \in W$, $j \notin W$:



k_{ij} neemt alleen toe als:
$z_{ij} < 0$, $x_{ij} < b_{ij}$ en $\lambda > -z_{ij}$.

b. $i \notin W$, j W:



k_{ij} neemt alleen toe als:
$z_{ij} > 0$, $x_{ij} > a_{ij}$ en $\lambda > z_{ij}$.

Daarom bekijken we (we nemen $x_{ij} = b_{ij}$ en $x_{ij} = a_{ij}$ mee i.v.m. convergentie-eigenschappen, zie het bewijs van Stelling 4.14).

$I_1 = \{(i,j) \mid i \in W,\ j \notin W,\ z_{ij} < 0 \text{ en } x_{ij} \leq b_{ij}\}$; $\lambda_1 = \min \{-z_{ij} \mid (i,j) \in I_1\}$;

$I_2 = \{(i,j) \mid i \notin W,\ j \in W,\ z_{ij} > 0 \text{ en } x_{ij} \geq a_{ij}\}$; $\lambda_2 = \min \{ z_{ij} \mid (i,j) \in I_2\}$;

$\lambda = \min\{\lambda_1, \lambda_2\}$. Als $I_1 = \emptyset$, dan is $\lambda_1 = \infty$ (analoog voor λ_2).

Stelling 4.14
Als $I_1 \cup I_2 = \emptyset$, dan heeft (4.15) geen toegelaten oplossing.
Bewijs
Omdat x een circulatiestroom is geldt: stroom uit W = Stroom in W, d.w.z.

(4.20) $\quad \Sigma_{i \in W} \Sigma_{j \notin W} x_{ij} = \Sigma_{i \notin W} \Sigma_{j \in W} x_{ij}$.

Voor iedere $i \in W$, $j \notin W$ is:

$$(4.21)\quad \begin{cases} \text{òf } z_{ij} \geq 0 \text{ en dan is } x_{ij} \geq b_{ij} \text{ (omdat } i \in W,\ j \notin W) \\ \text{òf } z_{ij} < 0 \text{ en dan is } x_{ij} > b_{ij} \text{ (omdat } I_1 = \varnothing) \end{cases}$$

Voor iedere $i \notin W$, $j \in W$ is:

$$(4.22)\quad \begin{cases} \text{òf } z_{ij} \leq 0 \text{ en dan is } x_{ij} \leq a_{ij} \text{ (omdat } i \notin W,\ j \in W) \\ \text{òf } z_{ij} > 0 \text{ en dan is } x_{ij} < a_{ij} \text{ (omdat } I_2 = \varnothing) \end{cases}$$

We zitten in de situatie dat $k_{pq} > 0$ en dat G^* geen ronde bevat die (p,q) bevat, d.w.z. dat $p \notin W$ en $q \in W$. Omdat $I_2 = \varnothing$ geldt: $x_{pq} \leq a_{pq}$.

Stel $x_{pq} = a_{pq}$. Uit $k_{pq} > 0$ volgt $z_{pq} > 0$, maar dan geldt $(p,q) \in I_2$: tegenspraak.

We kunnen nu schrijven op grond van (4.21), (4.22) en $x_{pq} < a_{pq}$:

$$(4.23)\quad 0 = \Sigma_{i\in W}\Sigma_{j\notin W}\, x_{ij} - \Sigma_{i\notin W}\Sigma_{j\in W}\, x_{ij} > \Sigma_{i\in W}\Sigma_{j\notin W}\, b_{ij} - \Sigma_{i\notin W}\Sigma_{j\in W}\, a_{ij}$$

Stel (4.15) heeft een toelaatbare stroom x^*. Dan geldt:

$$0 = \Sigma_{i\in W}\Sigma_{j\notin W}\, x^*_{ij} - \Sigma_{i\notin W}\Sigma_{j\in W}\, x^*_{ij} \leq \Sigma_{i\in W}\Sigma_{j\notin W}\, b_{ij} - \Sigma_{i\notin W}\Sigma_{j\in W}\, a_{ij} < 0:$$

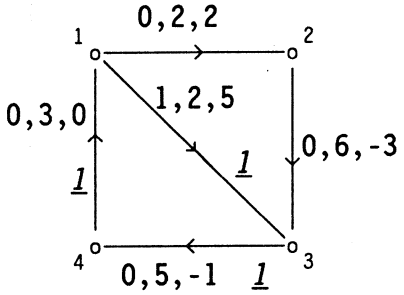
tegenspraak. Probleem (4.15) heeft dus geen toegelaten oplossing. □

Voorbeeld 4.10 (vervolg)

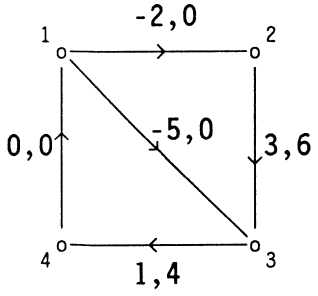
Hieronder staat het oorspronkelijke netwerk met de huidige stroom, het netwerk met de getallenparen z_{ij}, k_{ij} en de graaf G^*.

Stap 1

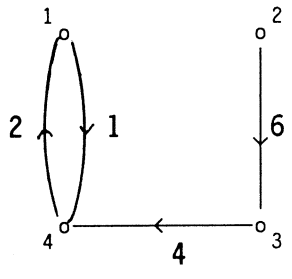
netwerk met a,b,c



netwerk met z,k



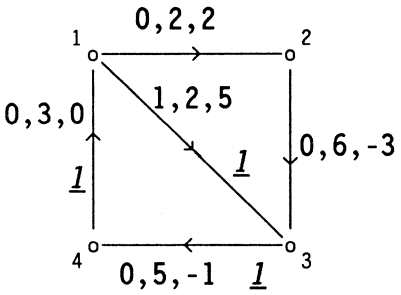
de graaf G^* met d



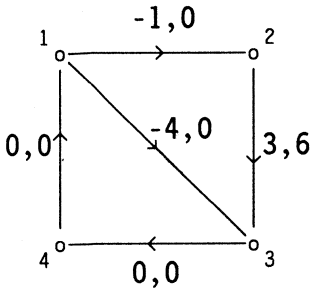
Neem $(p,q) = (3,4)$. $W = \{4,1\}$. $I_1 = \{(1,2),(1,3)\}$; $\lambda_1 = 2$; $I_2 = \{(3,4)\}$; $\lambda_2 = 1$; $\lambda = 1$; $z_{12} = -1$, $z_{13} = -4$, $z_{34} = 0$.

Stap 2

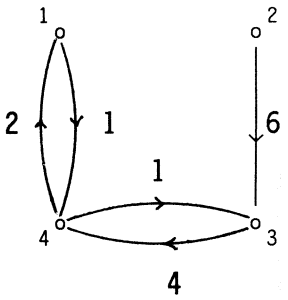
netwerk met a,b,c



netwerk met z,k



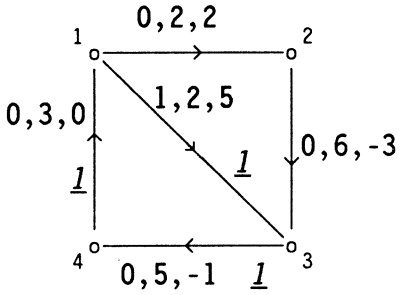
de graaf G^* met d



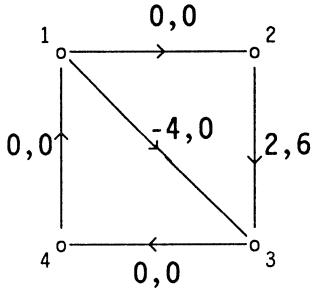
Neem (p,q) = (2,3). W = {3,4,1}; I_1 = {(1,2)}; λ_1 = 1; I_2 = {(2,3)}; λ_2 = 3; λ = 1. z_{12} = 0, z_{23} = 2.

Stap 3

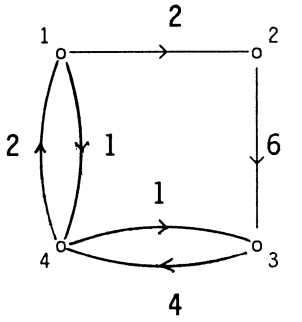
netwerk met a,b,c



netwerk met z,k



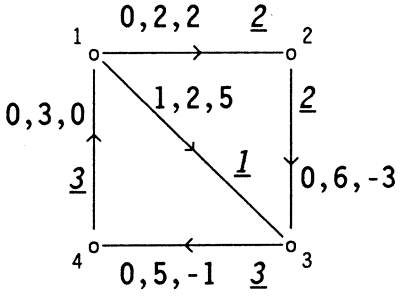
de graaf G^* met d



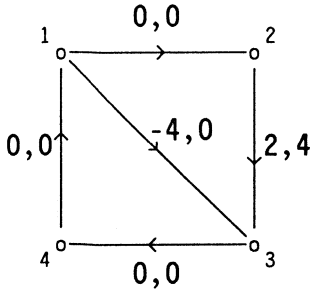
(p,q) = (2,3); G^* bevat de ronde [3,4,1,2]; Δ = 2.

Stap 4

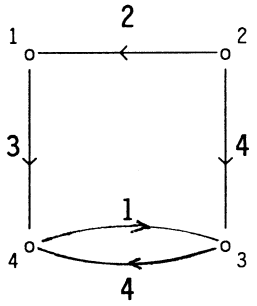
netwerk met a,b,c



netwerk met z,k



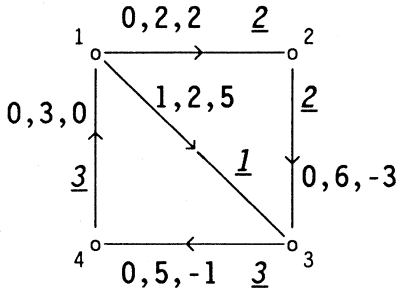
de graaf G^* met d



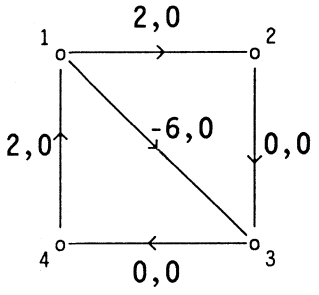
(p,q) = (2,3); W = {3,4}; I_1 = $\emptyset$; λ_1 = ∞; I_2 = {(2,3)}; λ_2 = 2; λ = 2. z_{41} = 2; z_{13} = -6; z_{23} = 0.

Stap 5

netwerk met a,b,c



netwerk met z,k



Alle k-getallen zijn = 0:
x is een optimale oplossing van (4.15).

Het algoritme kan als volgt worden samengevat.

stap 1: a. Start met een stroom en met een duale oplossing u (bv. $x = 0, u = 0$)
b. Bereken $z_{ij} = -c_{ij} + u_i - u_j$ voor alle (i,j).

stap 2: a. Bepaal k_{ij} volgens (4.19) voor alle (i,j).
b. Als $k_{ij} = 0$ voor alle (i,j): x is een optimale oplossing (STOP).
Anders: ga naar stap 3.

stap 3: a. Kies (p,q) zódat $k_{pq} > 0$ en zo mogelijk dezelfde keuze als de vorige keer.
b. Bepaal in G^* met het zijwaarts zoeken vanuit q of p bereikt kan worden en hoeveel kan worden vervoerd (Δ).
Zo ja; ga naar stap 4; zo neen: bepaal W en λ, en ga naar stap 5.

stap 4: Pas x met Δ aan en ga naar stap 2.

stap 5: a. Als $\lambda = \infty$: er is geen toelaatbare oplossing (STOP).
b. $z_{ij} := z_{ij} + \lambda$ voor $i \in W, j \notin W$; $z_{ij} := z_{ij} - \lambda$ voor $i \notin W, j \in W$.
c. ga naar stap 2.

Stelling 4.15

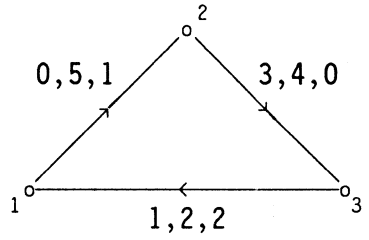
Het algoritme is eindig en correct.

Bewijs

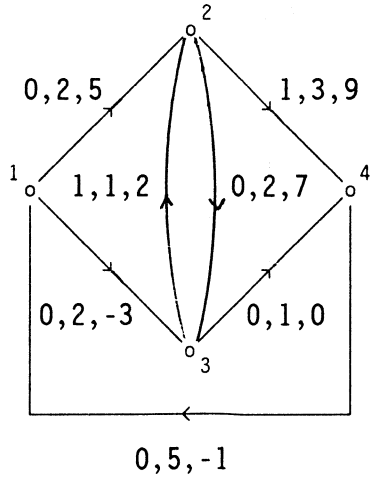
Het is voldoende om de eindigheid aan te tonen (het algoritme kan alleen stoppen in stap 2b en stap 5a en dan stopt het correct).
Tijdens het algoritme worden de k-getallen nooit groter; indien een ronde wordt gevonden, wordt in stap 4 de stroom aangepast en wordt vervolgens minstens één k-getal (nl. k_{pq}) kleiner.

Veronderstel dat geen ronde wordt gevonden en dat in stap 5 $\lambda < \infty$. Dan gebeurt het volgende: z_{ij} wordt groter voor $i \in W$, $j \notin W$ en z_{ij} wordt kleiner voor $i \notin W$, $j \in W$. Bovendien wordt minstens één $z_{ij} = 0$ (nl. degene die λ bepaalt) en de knooppunten van W blijven in W zitten.
Stel dat de methode niet eindig is. Dan blijft W uit dezelfde knooppunten bestaan. Omdat minstens één $z_{ij} = 0$ wordt, wordt $I_1 \cup I_2$ kleiner.
Dit impliceert dat het algoritme eindigt met $I_1 \cup I_2 = \emptyset$. □

Opgave 4.31
Los het volgende minimale kosten circulatie probleem op. De getallen bij de pijlen geven a_{ij}, b_{ij}, c_{ij} aan. Start met $x = 0$ en $u = 0$.



Opgave 4.32
Los het volgende minimale kosten circulatie probleem op. De getallen bij de pijlen geven a_{ij}, b_{ij}, c_{ij} aan. Start met $x = 0$ en $u = 0$.



Opgave 4.33
Beschouw probleem (4.15). Toon aan dat er een toelaatbare oplossing van dit probleem bestaat dan en slechts dan als voor iedere (W,V-W) geldt:

$$\Sigma_{i \in W} \Sigma_{j \notin W} b_{ij} \geq \Sigma_{i \notin W} \Sigma_{j \in W} a_{ij}.$$

V. KOPPELINGEN IN BIPARTIETE GRAFEN

1. Inleiding

Literatuur

* Hoofdstuk 5 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds", Holt, Rinehart and Winston, New York (1976).
* Hoofdstuk 10 in:
  C.H.Papadimitriou and K.Steiglitz: "Combinatorial optimization: algorithms and complexity", Prentice Hall, Englwood Cliffs (1982).
* Hoofdstuk 8 in:
  M.N.S.Swamy and K.Thulasiraman: "Graphs, networks and algorithms", Wiley, New York (1981).

Zij G = (V,E) een niet-gerichte normale graaf.
$M \subseteq E$ heet een koppeling als er geen twee takken van M aan elkaar grenzen. Een koppeling M heet maximaal als er geen koppeling M' bestaat met #M' > #M.
Omdat ieder tak van de koppeling twee knooppunten bindt en ieder knooppunt bij hoogstens één tak van de koppeling hoort, kan een koppeling nooit meer dan n/2 elementen bevatten. Een koppeling die n/2 elementen bevat heet een volmaakte koppeling. Als (i,j) een tak van de koppeling is, dan heten v_i en v_j gebonden knooppunten en zijn ze elkaars partner. Knooppunten die niet gebonden zijn heten vrij.

Een keten C heet een wisselketen m.b.t. een koppeling M als de takken van C afwisselend wel en niet tot de koppeling behoren. Als de keten gesloten is, dan spreken we van een wisselkring (deze bevat dan een even aantal takken waarvan de helft tot de koppeling behoort).

Een wisselketen heet een groeiketen als de eindpunten vrij zijn. Een groeiketen heeft dus een oneven aantal, zeg 2p+1, takken waarvan er p tot M behoren. Bij een groeiketen kunnen we $M \oplus C := M \cup C - M \cap C$ beschouwen, dit zijn de takken van C die niet tot M behoren en buiten C wat wel tot M behoort. $M \oplus C$ een weer een koppeling, maar met één tak meer dan M.

Een wisselboom T m.b.t. een koppeling M is een boom waarvoor geldt:

(i) Eén knooppunt is vrij, zeg v_0, en heet de wortel van de boom;

(ii) Alle keten van T die in de wortel v_0 beginnen zijn wisselketens;

(iii) Alle ketens van v_0 tot een eindpunt van T hebben een even aantal takken.

Omdat een boom een bipartiete graaf is, kunnen de knooppunten van T in twee klassen V_1 en V_2 worden verdeeld zódat de takken van T tussen V_1 en V_2 lopen. Veronderstel dat de wortel v_0 in V_1 zit, dan behoren de eindpunten van T eveneens tot V_1. De knooppunten van V_1 heten de buitenpunten, die van V_2 de binnenpunten.

Voorbeeld 5.1

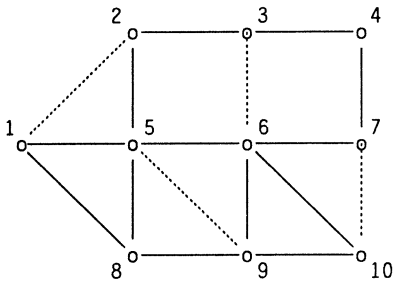
In nevenstaande graaf zijn de takken van de koppeling M gestippeld aangegeven.

C = [8,1,2,3,6,10,7,4] is een groeiketen.

M ⊕ C = {(1,8),(2,3),(6,10),(7,4),(5,9)}

is een volmaakte koppeling.

De takken {(8,1),(1,2),(2,3),(3,6),(8,5),(5,9)} vormen een wisselboom met wortel v_8.



Stelling 5.1

Een koppeling M is maximaal d.e.s.d. als er geen groeiketen m.b.t. M is.

Bewijs

⇒ Het is duidelijk dat als er groeiketen is de koppeling via M ⊕ C kan worden uitgebreid.

⇐ Het is voldoende om aan te tonen dat als M niet maximaal is, er een groeiketen bestaat. Als M niet maximaal is, dan is er een koppeling M′ met #M′ > #M. Beschouw de deelgraaf G′ = (V, M ⊕ M′). Ieder knooppunt behoort bij hoogstens één tak van M′ en bij hoogstens één tak van M: de graad in G′ is hoogstens 2. De componenten van G′ zijn dus: geïsoleerde knooppunten, wisselketen waarvan de takken afwisselend tot M′ en M behoren en wisselkringen met evenveel takken van M als van M′. Omdat #M′ > #M, is er een wisselketen die begint èn eindigt met een tak van M′. Dit is een groeiketen t.o.v. M. □

Gevolg

Zij M een koppeling, dan bestaat er een maximale koppeling M^* die alle knoopunten bindt die ook gebonden zijn door M.

Bewijs

Als M niet maximaal is, dan is er een groeiketen m.b.t. M. Dit geeft een koppeling $M' = M \oplus C$ met één tak meer dan M en waarin gebonden knooppunten gebonden blijven. Op deze wijze kunnen we doorgaan totdat een maximale koppeling M^* is verkregen. □

Veronderstel dat de graaf G bipartiet is, d.w.z. $G = (V_1 \cup V_2, E)$ met iedere $e \in E$ één eindpunt in V_1 en één eindpunt in V_2. Laat $V_1 = \{v_1, v_2, \ldots, v_p\}$ en $V_2 = \{w_1, w_2, \ldots, w_q\}$.

Het probleem om een maximale koppeling te vinden kan geformuleerd worden als het volgende LP-probleem (de sommaties over (i,j) met $(i,j) \in E$):

$$(5.1) \qquad \max \left\{ \Sigma_i \Sigma_j \, x_{ij} \;\middle|\; \begin{array}{l} \Sigma_j \, x_{ij} \leq 1, \; i = 1,2,\ldots,p \\ \Sigma_i \, x_{ij} \leq 1, \; j = 1,2,\ldots,q \\ x_{ij} \geq 0, \text{ voor alle i en j} \end{array} \right\}$$

De coëfficiëtenmatrix van (5.1) is de incidentiematrix van de graaf G. Omdat deze totaal unimodulair is (Stelling 1.3) zijn de hoekpunten van (5.1) geheeltallig. Iedere basisoplossing heeft dus een (0,1)-oplossing.

Stelling 5.2

Er is een één-éénduidig verband tussen de hoekpunten van (5.1) en de koppelingen in G zódat de waarde van de doelfunctie het aantal elementen in de koppeling is.

Bewijs

Zij x een hoekpunt van (5.1). x is dus een (0,1)-oplossing. Uit de beperkingen van (5.1) volgt dat voor iedere i er hooogstens één $x_{i1}, x_{i2}, \ldots, x_{iq}$ positief is; evenzo voor iedere j maximaal één $x_{1j}, x_{2j}, \ldots, x_{pj}$ positief.

Laat $M := \{(i,j) \mid x_{ij} = 1\}$, dan is M dus een koppeling en $\#M = \Sigma_i \Sigma_j \, x_{ij}$.

Omgekeerd, zij M een koppeling. Meen $x_{ij} = 1$ als $(i,j) \in M$ en anders $x_{ij} = 0$. Stel x is geen hoekpunt: $x = \lambda x^1 + (1-\lambda)x^2$ met x^1 en x^2 toelaatbaar voor (5.1) en $0 < \lambda < 1$. Dan geldt (omdat ook alle componenten tussen 0 en 1 liggen):

als $(i,j) \in M$: $1 = x_{ij} = \lambda x^1_{ij} + (1-\lambda)x^2_{ij} \leq 1 \Rightarrow x_{ij} = x^1_{ij} = x^2_{ij} = 1.$

als $(i,j) \notin M$: $0 = x_{ij} = \lambda x^1_{ij} + (1-\lambda)x^2_{ij} \geq 0 \Rightarrow x_{ij} = x^1_{ij} = x^2_{ij} = 0.$

Dus is $x = x^1 = x^2$, d.w.z. x is wel een hoekpunt. □

Voorbeeld 5.2

Beschouw een nxn matrix $A = (a_{ij})$ met niet-negatieve coëfficiënten en zodanig dat iedere rij en kolomsom 1 is. Een dergelijke matrix wordt een dubbel stochastische matrix genoemd.

Een dubbel stochastische matrix heet een permutatie matrix als alle positieve elementen 1 zijn, d.w.z. dat iedere rij en iedere kolom procies één 1 bevat.

Een dubbel stochastische matrix voldoet aan het stelstel:

$$\begin{cases} \Sigma_{j=1}^{n} x_{ij} = 1, \ i = 1,2,\ldots,n \\ \Sigma_{i=1}^{n} x_{ij} = 1, \ j = 1,2,\ldots,n \\ x_{ij} \geq 0, \text{ voor alle i en j} \end{cases}$$

Omdat de matrix van dit stelsel totaal unimodulair is (het is de incidentie matrix van de volledige bipartiete graaf $K_{n,n}$) zijn de extreme punten van dit stelsel geheeltallig, d.w.z. dat zij corresponderen met permutatiematrices.

Als gevolg hiervan geldt:

Stelling 5.3

Iedere dubbelstochastische matrix is een convexe combinatie van permutatie matrices.

Beschouw het duale probleem van (5.1):

$$(5.2) \qquad \min \left\{ \Sigma_{i=1}^{p} s_i + \Sigma_{j=1}^{q} t_j \;\middle|\; \begin{array}{l} s_i + t_j \geq 1 \quad \text{voor alle } (i,j) \in E \\ s_i, t_j \geq 0, \ 1 \leq i \leq p, \ 1 \leq j \leq q \end{array} \right\}.$$

$W_1 \cup W_2$ met $W_1 \subseteq V_1$ en $W_2 \subseteq V_2$ heet een takkenbedekking als iedere $e \in E$ minstens één eindpunt in $W_1 \cup W_2$ heeft. Een takkenbedekking $W_1 \cup W_2$ heet minimaal als er geen takkenbedekking $W_1' \cup W_2'$ is met $|W_1' \cup W_2'| < |W_1 \cup W_2|$.

Stelling 5.4
Er is een één-éénduidig verband tussen de (0,1)-oplossingen (s,t) van (5.2) en de takkenbedekkingen van G zódat de waarde van de doelfunctie het aantal elementen in de takkenbedekking is.
Bewijs
Zij (s,t) een (0,1)-oplossing van (5.2).
Neem $W_1 = \{i \in V_1 \mid s_i = 1\}$ en $W_2 = \{j \in V_2 \mid t_j = 1\}$. Dan is dit een takkenbedekking, nl. voor iedere $(i,j) \in A$ is $s_i + t_j \geq 1$, dus v_i of w_j in $W_1 \cup W_2$.
Tevens is $\Sigma_{i=1}^{p} s_i + \Sigma_{j=1}^{q} t_j = |W_1 \cup W_2|$.
Omgekeerd, zij $W_1 \cup W_2$ een takkenbedekking.

Neem $s_i = \begin{cases} 1 \text{ als } i \in W_1 \\ 0 \text{ als } i \notin W_1 \end{cases}$ en $t_j = \begin{cases} 1 \text{ als } j \in W_2 \\ 0 \text{ als } j \notin W_2. \end{cases}$

Dan is $\Sigma_{i=1}^{p} s_i + \Sigma_{j=1}^{q} t_j = |W_1 \cup W_2|$, en omdat $W_1 \cup W_2$ een takkenbedekking is, geldt: $s_i + t_j \geq 1$. □

Gevolg (Stelling van König)
Het aantal elementen in een minimale takkenbedekking is gelijk aan het aantal elementen in een maximale koppeling.
Bewijs
Zij (s,t) een optimaal hoekpunt van (5.2). Dan is (s,t) geheeltallig en dus een minimale (0,1)-oplossing. Deze corresponddert met een minimale takkenbedekking.
Omdat volgens de dualiteitsstelling van de lineaire programmering de optimale waarden van de duale LP-problemen (5.1) en (5.2) aan elkaar gelijk zijn, geldt:
aantal elementen maximale koppeling = waarde optimum (5.1) = waarde optimum (5.2) = aantal elementen minimale takkenbedekking. □

Opgave 5.1

Is de volgende uitspraak waar of niet waar?
"Voor iedere koppeling M_1 is er een maximale koppeling M_2 met $M_1 \subseteq M_2$."

Opgave 5.2

Zij G = (V,E) een willekeurige graaf met n knooppunten.
$F \subseteq E$ heet een knooppuntenbedekking als ieder knooppunt het eindpunt is van minstens één tak van F.

a. Geef aan hoe uit een maximale koppeling M een knooppuntenbedekking F' kan worden gevonden met n - |M| elementen.
b. Geef aan hoe uit een minimale knooppuntenbedekking F een koppeling M' kan worden gevonden met n - |F| elementen.
c. Toon aan dat de in a en b geconstrueerde knooppuntenbedekking F' en koppeling M' minimaal resp. maximaal zijn.

Opgave 5.3

Zij G = (V,E) een willekeurige graaf.
Zij $\alpha(G)$ het maximum aantal knooppunten van G zódat geen enkel tweetal van G door een tak is verbonden.
Laat F een minimale knooppuntenbedekking zijn (zie Opgave 5.2).

a. Toon aan dat $\alpha(G) \leq |F|$.
b. Geef een voorbeeld met $\alpha(G) < |F|$ en een voorbeeld met $\alpha(G) = |F|$.

Opgave 5.4

Zij A een nxn (0,1)-matrix met in iedere rij en in iedere kolom precies k 1'en.
Toon aan dat $A = P_1 + P_2 + \cdots + P_k$ met P_i een permutatie matrix ($1 \leq i \leq k$).

Opgave 5.5

Zij M_1 en M_2 twee koppelingen in een bipartiete graaf $G = (V_1 \cup V_2, E)$.
Toon de volgende bewering aan:
Er bestaat een koppeling $M \subseteq M_1 \cup M_2$ zódat M de knooppunten van V_1 bindt die t.o.v. M_1 gebonden zijn en de knooppunten van V_2 die t.o.v. M_2 gebonden zijn.

2. Equivalente combinatorische problemen

Literatuur

* Hoofdstuk 5 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds",
  Holt, Rinehart and Winston, New York (1976).
* P.F.Reichmeider: "The equivalence of some combinatorial matching theorems",
  Polyganal, Washington (1984).
* Hoofdstuk 8 in:
  M.N.S.Swamy and K.Thulasiraman: "Graphs, networks and algorithms",
  Wiley, New York (1981).

We zullen in deze paragraaf de onderlinge relatie van een aantal combinatorische resultaten bespreken. De eerste vier resultaten (A t/m D) zijn equivalent, de volgende drie (E t/m G) eveneens; De eerste vier resultaten zijn een gevolg van de laatste drie, en alle resultaten zijn een gevolg van de dualiteitsstelling van de lineaire programmering (resultaat H).

RESULTAAT A (STELLING VAN KONIG):
Zij G een bipartiete graaf.
Het aantal elementen in een minimale takkenbedekking is gelijk aan het aantal elementen in een maximale koppeling.

Zij A een m bij n matrix waarvan de elementen 0 of 1 zijn. Onafhankelijke 1'en zijn 1'en die in verschillende rijen en kolommen staan (zie dictaat Discrete wiskunde, paragraaf 19).

RESULTAAT B (STELLING VAN KONIG-EGERVARY):
Zij A een m bij n matrix waarvan de elementen 0 of 1 zijn.
Het minimum aantal rijen en kolommen dat alle 1'en bevat is gelijk aan het maximum aantal onafhankelijke 1'en.

RESULTAAT C (STELLING VAN HALL):
Zij E een eindige niet-lege verzameling en $\mathcal{S} = (S_1, S_2, \ldots, S_n)$ een collectie van deelverz. van E.
$\mathcal{S}$ is een transversaal d.e.s.d. als de vereniging van ieder k-tal S_i's minstens k elementen bevat voor $k = 1,2,\ldots,n$.

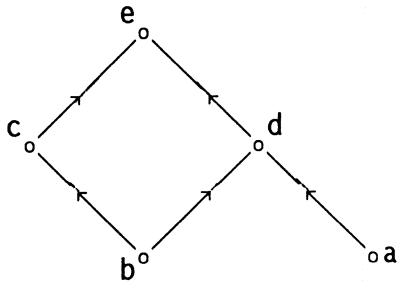
Het volgende resultaat betreft partiëel geordende verzamelingen. Het is op het eerste gezicht wat verrassend dat een dergelijk probleem met de andere equivalent is.

Zij X een eindige verzameling met daarop een relatie $\leq$. Het tweetal $(X,\leq)$ heet een <u>partieel geordende verzameling</u> (afgekort: p.g.v.) als de volgende eigenschappen gelden:

(i) $x \leq x$ voor alle $x \in X$ (reflexiviteit);
(ii) $x \leq y$ en $y \leq x$ impliceert $x = y$ voor alle $x,y \in X$ (antisymmetrie);
(iii) $x \leq y$ en $y \leq z$ impliceert $x \leq z$ voor alle $x,y,z \in X$ (transitiviteit).

Twee elementen x en y zodat $x \leq y$ of $y \leq x$ heten <u>vergelijkbaar;</u> is dit niet het geval dan heten ze <u>onvergelijkbaar</u>. Als $x \leq y$ en $x \neq y$, dan is $x < y$.
Als $x < z$ en er is geen y zodat $x < y < z$ dan heet x de <u>voorganger</u> van z.
Een element x heet <u>kleinste element</u> van X als $x \leq y$ voor alle $y \in X$. Een p.g.v. hoeft geen kleinste element te bevatten. Een element x heet <u>minimaal</u> als er geen $y \neq x$ bestaat met $y \leq x$. Bij een eindige X is er altijd een minimaal element. Een kleinste element is minimaal, maar het omgekeerde hoeft niet waar te zijn. Op analoge wijze definiëren we het <u>grootste</u> en een <u>maximaal</u> element.

<u>Voorbeeld 5.2</u>
Zij E = {a,b,c,d,e}.
De elementen van E tekenen we als knooppunten. Als $x < y$, dan ligt x lager dan y. Vervolgens tekenen we een pijl van ieder element naar zijn voorganger. Dan is in de figuur te zien of $x \leq y$, namelijk als er een pad omhoog is van x naar y. Er is geen kleinste element, maar wel twee minimale, nl. a en b. e is het grootste en dus ook maximaal element.



$b \leq e$, want $b \leq c$ en $c \leq e$.

Een verz. elementen $Y \subseteq X$ zodat ieder tweetal daaruit vergelijkbaar is heet een keten. #Y is de lengte van de keten. Een keten Y is een maximale keten als er geen keten $Z \neq Y$ bestaat met $Y \subset Z$. Een verz. ketens bedekt de p.g.v. als ieder element van X in minstens één keten zit.
Een verz. elementen $A \subseteq X$ waarvan geen enkel tweetal vergelijkbaar is noemen we een anti-keten met #A de lengte ervan. De begrippen maximaal en bedekken worden analoog gedefiniëerd.

Voorbeeld 5.2 (vervolg)
{b,c,e} is een maximale keten; de ketens {b,c,e} en {a,d} bedekken de p.g.v..
{c,d} is een maximale anti-keten en de anti-ketens {c,d}, {a,b} en {e} bedekken X.

Merk op dat iedere keten hoogstens één element gemeen heeft met iedere anti-keten. Dus de maximale lengte van een anti-keten is hoogstens het minimale aantal ketens dat X bedekt. De Stelling van Dilworth zegt dat er een gelijkheid is.

RESULTAAT D (STELLING VAN DILWORTH):
Als $(X,\leq)$ een eindige partiëel geordende verzameling is, dan is het maximum aantal elementen in een anti-keten is gelijk aan het minimum aantal ketens dat X kan bedekken.

Het volgende resultaat betreft een willekeurige gerichte graaf $G = (V,A)$.
We bekijken het aantal paden tussen twee niet aangrenzende knooppunten, zeg s en t. Paden heten onafhankelijk als er geen gemeenschappelijke pijlen in voorkomen.
Een $F \subseteq A$ met heet een (s,t)-splitsende pijlenverz. als in $G' = (V,A-F)$ geen pad van s naar t bestaat. Een splitsende pijlenverz. F_1 heet minimaal als er geen splitsende pijlenverz. F_2 bestaat met $\#F_2 < \#F_1$.

RESULTAAT E (STELLING VAN MENGER):
Zij s en t twee niet aangrenzende knooppunten van een gerichte graaf.
Het maximum aantal onafhankelijke paden van s naar t is gelijk aan het aantal elementen in een minimale (s,t)-splitsende pijlenverzameling.

RESULTAAT F (STELLING VAN FORD EN FULKERSON):
Zij N = (V,A) een netwerk met geheeltallige capaciteiten b.
Dan is de maximale stroom geheeltallig en heeft een waarde die gelijk is een de capaciteit van een (1,n)-snede met minimale capaciteit.

RESULTAAT G (STELLING VAN HOFFMAN):
Zij N = (V,A) een netwerk met onder- en bovengrenzen a resp. b.
N heeft een toelaatbare circulatiestroom d.e.s.d. als

$$\Sigma_{i\in W}\Sigma_{j\notin W}\, b_{ij} \geq \Sigma_{i\notin W}\Sigma_{j\in W}\, a_{ij} \text{ voor alle } W \subseteq V.$$

RESULTAAT H (DUALITEITSTELLING LINEAIRE PROGRAMMERING)
Beschouw een eindig lineair programmeringsprobleem en het duale probleem.
Dan zijn de optima (maximum resp. minimum) van beide problemen aan elkaar gelijk.

De acht combinatorische resultaten A t/m H zijn alle waar. Een aantal is reeds eerder bewezen:

resultaat A: Gevolg van Stelling 5.4 in dit dictaat;
resultaat B: zie Stelling 19.2 in dictaat Discrete Wiskunde;
resultaat C: zie Stelling 18.1 in dictaat Discrete Wiskunde;
resultaat F: Gevolg 4.2 en Stelling 4.3 in dit dictaat;
resultaat G: zie Opgave 4.33 in dit dictaat;
resultaat H: zie Stelling 14.29 in dictaat Discrete Wiskunde

Stelling 5.5
De resultaten A, B, C en D zijn equivalent.
Bewijs

A $\Rightarrow$ B:
Zij A een m bij n matrix waarvan de elementen 0 of 1 zijn.
Construeer de volgende bipartiete graaf $G = (V_1 \cup V_2, E)$:
$V_1 = \{1,2,\ldots,m\}$, $V_2 = \{1,2,\ldots,n\}$ en $E = \{(i,j) \mid a_{ij} = 1\}$.
Onafhankelijke 1'en van A corresponderen met een koppeling in G en rijen en kolommen van A die alle 1'en bevatten corresponderen met een takkenbedekking van G.

Omdat volgens resultaat A het aantal elementen in een minimale takken-bedekking is gelijk aan het aantal elementen in een maximale koppeling, geldt: het minimum aantal rijen en kolommen dat alle 1'en bevat is gelijk aan het maximum aantal onafhankelijke 1'en.

B $\Rightarrow$ D:

Zij $(X,\leq)$ een eindige partiëel geordende verzameling met $X = \{1,2,\ldots,n\}$. Beschouw een willekeurige ketenbedekking $C_1,C_2,\ldots,C_p$ en een willekeurige anti-keten C^* met q elementen. Dan is $q \leq p$, omdat ieder element van C^* uit een andere C_i komt (immers elementen binnen een C_i zijn vergelijkbaar). We moeten dus bewijzen dat er een ketenbedekking $C_1,C_2,\ldots,C_p$ is en een anti-keten C^* met $|C^*| = p$ (dan is het aantal ketens van de bedekking minimaal en de anti-keten maximaal).
Construeer de volgende n bij n (0,1)-matrix A: $a_{ij} = 1$ d.e.s.d. als $i < j$. Beschouw een keten $C = \{i_1,i_2,\ldots,i_k\}$ met $i_1 < i_2 < \cdots < i_k$, dan zijn de 1'en in de posities (i_1,i_2), $(i_2,i_3),\ldots,(i_{k-1},i_k)$ onafhankelijk, want ze staan in verschillende rijen en kolommen. Zo ook als $C_1,C_2,\ldots,C_p$ een disjuncte verz. ketens is die X bedekt, dan zijn de daarbij behorende 1'en onafhankelijk (we mogen ons wel beperken tot disjuncte ketens, want laat anders een dubbel element weg uit een keten; het aantal ketens verandert daardoor niet).

Voorbeeld 5.2 (vervolg)

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$C_1 = \{b,c,e\}$, $C_2 = \{a,d\}$ is een disjunte ketenbedekking van X. De bijbehorende 1'en staan op de plaatsen (2,3), (3,5) en (1,4) en zijn onafhankelijk.

Omgekeerd, beschouw een onafhankelijke verz. 1'en van de bij $(X,\leq)$ behorende matrix A, zeg $(i_1,j_1),(i_2,j_2),\ldots,(i_k,j_k)$. Deze corresponderen met een disjunte ketenbedekking, waarbij een i die niet in een rij of kolom voorkomt als keten $\{i\}$ wordt opgenomen. Dus als de onafhankelijke verz. n-p 1'en bevat, dan heeft de bedekking p ketens. Hieruit volgt:
Als er maximaal n-p onafhankelijke 1'en zijn, dan heeft een minimale ketenbedekking p ketens.

Voorbeeld 5.2 (vervolg)

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Neem de onafhankelijke 1'en op de plaatsen (2,3) en (1,4). Dit geeft de ketens: $C_1 = \{b,c\}$, $C_2 = \{a,d\}$ en $C_3 = \{e\}$.

Als er maximaal n-p onafhankelijke 1'en zijn, dan is volgens resultaat B het minimum aantal rijen en kolommen dat alle enen bevat eveneens n-p. Beschouw nu een minimum aantal van n-p rijen en kolommen die tezamen alle 1'en bevatten.
Laat J de verz. van de kolommen zijn en I de verz. van de rijen waarin 1'en voorkomen die niet in de kolommen van J zitten: $i \in I$ d.e.s.d. als $i < j$ voor zekere $j \notin J$.
Nu geldt: $I \cap J = \varnothing$ (immers: stel $k \in I \cap J$, dan er een $j \notin J$ met $k < j$ en een $i \notin I$ met $i < k$; dus $i < j$, d.w.z. $a_{ij} = 1$ en noch rij i noch kolom j is doorgestreept).
Nu is $\{1,2,\ldots,n\} \setminus (I \cup J)$ een anti-keten C^* van $(X,\leq)$, immers:
Stel niet, dan is $i < j$ voor zekere $i,j \notin I \cup J$: $a_{ij} = 1$ en dit element is niet doorgestreept: tegenspraak.
Tevens geldt: $|C^*| = n - |I \cup J| = n - [|I| + |J|] = n - (n-p) = p$
We hebben nu aangetoond:
p = minimum aantal ketens dat X kan bedekken = $|C^*|$.

Voorbeeld 5.2 (vervolg)

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

J = {kolom 4 en 5} en I = {rij 2} vormen een minimale bedekking van alle 1'en. De bijbehorende anti-keten is: $C^* = \{a,c\}$.

D $\Rightarrow$ C:
Beschouw een eindige verz. $E = \{1,2,\ldots,m\}$ en een collectie $S_1,S_2,\ldots,S_n$ van deelverz. van E, met $n \leq m$.
Als $\mathcal{S} = (S_1,S_2,\ldots,S_n)$ een transversaal is, dan heeft de vereniging van ieder k-tal S_i's minstens k elementen, voor iedere $k = 1,2,\ldots,n$.
Het omgekeerde moet nu worden aangetoond.

Laat $X = X_1 \cup X_2$ met $X_1 = \{1,2,\ldots,m\}$ en $X_2 = \{m+1,m+1,\ldots,m+n\}$.
Neem de volgende partiële ordening: $i \leq j$ als $i \leq m$, $j > m$ en $i \in S_{j-m}$.
Dus een keten bestaat uit 1 of 2 elementen.
Stel ieder tweetal van $i_1,i_2,\ldots,i_p \in X_1$ en $m+j_1,m+j_2,\ldots,m+j_q \in X_2$ is onvergelijkbaar, d.w.z. $i_k \notin S_{j_l}$ voor $1 \leq k \leq p$ en $1 \leq l \leq q$.
Omdat de vereniging van ieder q-tal S_i's minstens q elementen bevat, geldt: $p \leq m-q$, d.w.z. $p + q \leq m$. Hieruit volgt dat een anti-keten maximaal m elementen bevat. Dus $C^* = \{1,2,\ldots,m\}$ is een maximale anti-keten.
Uit resultaat D volgt dat een minimale ketenbedekking m ketens bevat. Omdat iedere keten 1 of 2 elementen bevat is een minimale ketenbedekking van de vorm: $C_1 = (i_1,m+j_1)$, $C_2 = (i_2,m+j_2)$, ... $C_n = (i_n,m+j_n)$, $C_{n+1} = \{i_{n+1}\}$,
$C_{n+2} = \{i_{n+2}\}$, ..., $C_m = \{i_m\}$.
Nu is $\{i_1 \in S_{j_1}, i_2 \in S_{j_2}, \ldots, i_n \in S_{j_n}\}$ een stelsel representanten,
d.w.z $\mathcal{S} = (S_1,S_2,\ldots,S_n)$ is een transversaal.

C $\Rightarrow$ A:
Zij $G = (V_1 \cup V_2,E)$ een bipartiete graaf. Zij $W_1 \cup W_2$ een takkenbedekking en M een koppeling. Iedere tak van M heeft minstens één eindpunt in $W_1 \cup W_2$, en verschillende takken van M corresponderen met verschillende eindpunten: $|W_1 \cup W_2| \geq |M|$. We moeten dus nog bewijzen dat er een $W_1 \cup W_2$ en M bestaan zódanig dat $|W_1 \cup W_2| = |M|$.
Stel $W_1 \cup W_2$ is een minimale takkenbedekking met $|W_1| = p$ en $|W_2| = q$.
Veronderstel dat $W_1 = \{1,2,\ldots,p\}$ en $W_2 = \{p+1,p+2,\ldots,q\}$.

$$\text{Laat } S_i = \begin{cases} \{j \in V_2 - W_2 \mid (i,j) \in E\}, & i = 1,2,\ldots,p \\ \{j \in V_1 - W_1 \mid (j,i) \in E\}, & i = p+1,p+2,\ldots,p+q. \end{cases}$$

De vereniging van een k-tal S_i's bevat minstens k elementen, immers:
Stel dat het niet waar is. De takken van $S_i, 1 \leq i \leq p$, en $S_i, p+1 \leq i \leq p+q$ hebben geen gemeenschappelijke eindpunten; er is dus een j-tal binnen de eerste of tweede groep waarvan de vereniging minder dan j elementen bevat; dan kan een kleinere bedekking worden gevonden door de hoogstens j-1 eindpunten te nemen i.p.v. de j eindpunten van W_1 of W_2.
Volgens resultaat C is $(S_1,S_2,\ldots,S_{p+q})$ een transversaal met corresponderende representanten, zeg $w_1,w_2,\ldots,w_{p+q}$. Nu vormen de takken (i,w_i), $1 \leq i \leq p$ en (w_i,i), $p+1 \leq i \leq p+q$, een koppeling M met $|M| = |W_1 \cup W_2|$. □

Stelling 5.6

De resultaten E, F en G zijn equivalent.

Bewijs

E ⇒ F:

Zij N = (V,A) een netwerk met capaciteiten b.
Vervang iedere pijl (i,j) met capaciteit b_{ij} door b_{ij} pijlen van i naar j.
Zij N′ = (V,A′) de aldus geconstrueerde graaf. Volgens resultaat E is het maximum aantal onafhankelijke paden van 1 naar n gelijk aan het aantal elementen in een minimale (1,n)-splitsende pijlenverz..
Ieder onafhankelijk pad induceert een stroom met waarde 1: het maximum aantal onafhankelijke paden van 1 naar n is gelijk aan de waarde van de maximale stroom.
Indien een pijl van i naar j in een minimale (1,n)-splitsende pijlenverz. zit, dan zitten alle b_{ij} pijlen van i naar j erin (anders is i niet van j gescheiden en is het beter om de pijl (i,j) niet op te nemen in deze verz.).
Het aantal elementen in een minimale (1,n)-splitsende pijlenverz. is dus de capaciteit van een aantal pijlen in N zódat door het weglaten ervan n niet meer bereikbaar is vanuit 1: dit is de capaciteit c(W) met voor W ⊆ V de knooppunten die vanuit 1 bereikbaar zijn.
We hebben nu bewezen: de waarde van de maximale stroom is gelijk aan c(W) voor een (1,n)-snede (W,V-W).
Omdat altijd geldt c(W) ≥ waarde van een stroom, is resultaat F aangetoond.

F ⇒ G: (zie ook Opgave 4.33)

Zij x een toelaatbare stroom en W een willekeurige deelverz. van V.

Dan geldt:

$$0 = \text{stroom uit } W - \text{stroom in } W = \Sigma_{i\in W}\Sigma_{j\notin W}\, x_{ij} - \Sigma_{i\notin W}\Sigma_{j\in W}\, x_{ij}$$

$$\leq \Sigma_{i\in W}\Sigma_{j\notin W}\, b_{ij} - \Sigma_{i\notin W}\Sigma_{j\in W}\, a_{ij}.$$

Veronderstel vervolgens dat $\Sigma_{i\in W}\Sigma_{j\notin W}\, b_{ij} \geq \Sigma_{i\notin W}\Sigma_{j\in W}\, a_{ij}$ voor iedere W ⊆ V.
In paragraaf 5 van hoofdstuk 4 hebben we gezien dat er een toelaatbare stroom bestaat d.e.s.d als het hulpnetwerk $\bar{N}$ een stroom heeft met waarde $\Sigma_i\Sigma_j\, a_{ij}$. Volgens resultaat F is dit het geval als iedere (s,t)-snede een capaciteit van minstens $\Sigma_i\Sigma_j\, a_{ij}$ heeft.

Neem een willekeurige (s,t)-snede in $\bar{N}$: $(\bar{W}, V \cup S \cup t - \bar{W})$. Laat $W = \bar{W} - s$.

$$\begin{aligned} c(\bar{W}) &= c(s \rightarrow V - W) + c(W \rightarrow V - W) + c(W \rightarrow t) \\ &= \Sigma_{j \notin W} \Sigma_i \, a_{ij} + \Sigma_{i \in W} \Sigma_{j \notin W} (b_{ij} - a_{ij}) + \Sigma_{i \in W} \Sigma_j \, a_{ij} \\ &= \Sigma_{i \in W} \Sigma_{j \notin W} \, b_{ij} + \Sigma_{i \notin W} \Sigma_{j \notin W} \, a_{ij} + \Sigma_{i \in W} \Sigma_j \, a_{ij} \\ &= \Sigma_{i \in W} \Sigma_{j \notin W} \, b_{ij} + \Sigma_i \Sigma_j \, a_{ij} - \Sigma_{i \notin W} \Sigma_{j \in W} \, a_{ij} \\ &\geq \Sigma_i \Sigma_j \, a_{ij}, \end{aligned}$$

de laatste ongelijkheid omdat $\Sigma_{i \in W} \Sigma_{j \notin W} \, b_{ij} \geq \Sigma_{i \notin W} \Sigma_{j \in W} \, a_{ij}$.

G ⇒ F:

Het is direct duidelijk dat de capaciteit van iedere snede minstens zo groot moet zijn als de waarde van van de stroom (deze moet er immers door kunnen). Het is dus voldoende om te bewijzen dat er een snede en een stroom bestaan zódat de waarde van de stroom gelijk is aan de capaciteit van de snede.
Voeg aan het netwerk de pijl (n,1) toe met $a_{n1} = b_{n1}$ = waarde van de capaciteit van de snede met de kleinste capaciteit. We moeten nu aantonen dat er een toelaatbare circulatiestroom is.
Volgens resultaat G is dit waar als $\Sigma_{i \in W} \Sigma_{j \notin W} \, b_{ij} - \Sigma_{i \notin W} \Sigma_{j \in W} \, a_{ij} \geq 0$ voor alle $W \subseteq V$. Alle $a_{ij} = 0$ voor $(i,j) \neq (n,1)$.
Dus de ongelijkheid moet alleen worden gecontroleerd voor $n \notin W$, $1 \in W$:
$\Sigma_{i \in W} \Sigma_{j \notin W} \, b_{ij} = c(W) \geq$ waarde kleinste capaciteit $= a_{n1} = \Sigma_{i \notin W} \Sigma_{j \in W} \, a_{ij}$.

F ⇒ E:

Beschouw een gerichte graaf G = (V,A) met twee niet aangrenzende knooppunten s en t.
Omdat ieder onafhankelijk pad van s naar t een verschillende pijl van een (s,t)-splitsende pijlenverz. bevat geldt: het maximum aantal onafhankelijke paden van s naar t is minstens het aantal elementen in een minimale (s,t)-splitsende pijlenverz..
Geef iedere pijl de capaciteit 1. Dan is het maximum aantal onafhankelijke paden gelijk aan de maximale stroom van s naar t.
Volgens resultaat F is dit ook gelijk aan de capaciteit van een (s,t)-snede (W,V-W). Neem nu de pijlensnede bestaande uit alle pijlen van W naar V-W.
Omdat alle capaciteiten 1 zijn is c(W) = aantal pijlen in deze pijlensnede.
Hieruit volgt resultaat E. □

Stelling 5.7

Resultaat F impliceert resultaat A.

Bewijs

Zij $G = (V_1 \cup V_2, E)$ een bipartiete graaf.

Voeg een extra knooppunt s toe met pijlen (s,v) voor iedere $v \in V_1$, richt de takken van V_1 naar V_2 en voeg ook knooppunt t toe met pijlen (w,t) voor alle $w \in V_2$. Dit geeft een netwerk $N = (V_1 \cup V_2 \cup s \cup t, A)$ en neem de capaciteiten b_{sv} en b_{wt} gelijk aan 1 en de andere capaciteiten ∞.

Een stroom met waarde w correspondeert met onafhankelijke paden van s naar t, welke weer corresponderen met een koppeling bestaande uit w takkken tussen V_1 en V_2.

Een (s,t)-snede met eindige capaciteit correspondeert met s, $W_1 \in V_1$ en $W_2 \in V_2$. Omdat de capaciteit eindig is, zijn er geen pijlen van W_1 naar $V_2 - W_2$. De capaciteit van deze snede is dus $|V_1 - W_1| + |W_2|$.

$(V_1 - W_1) \cup W_2$ is ook een takkenbedekking van G (stel niet, dan zou en een pijl zijn met beginpunt in W_1 en eindpunt in $V_2 - W_2$).

Omdat volgens resultaat F de waarde van een maximale stroom gelijk is aan de minimale capaciteit van een (s,t)-snede, is het aantal elementen in een maximale koppeling gelijk aan het aantal elementen in een minimale takken-bedekking.

Gevolg

Ieder der resultaten E, F en G impliceert ieder der resultaten A, B, C en D.

Stelling 5.8

Resultaat H impliceert resultaat F.

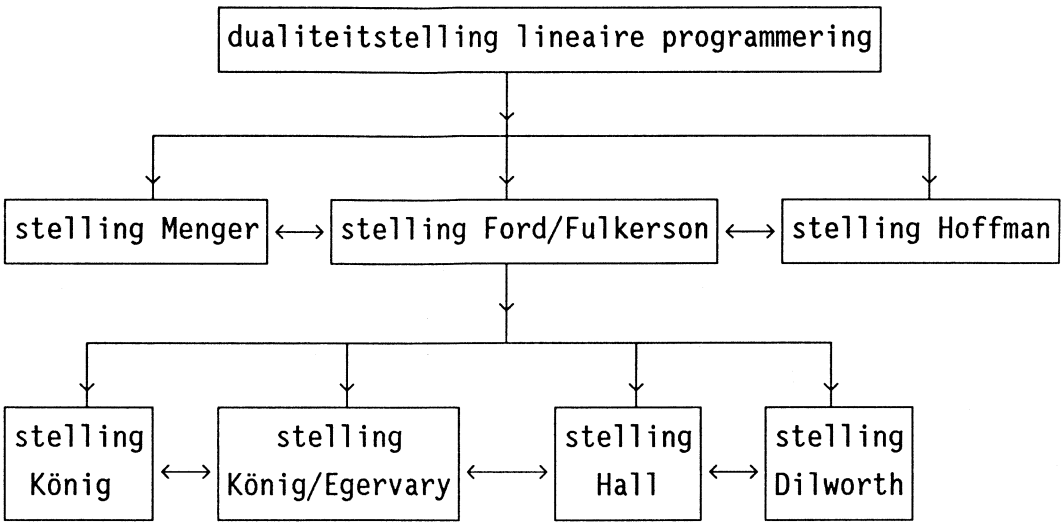
Bewijs

Dit is reeds gedaan in paragraaf 1 van hoofdstuk 4. □

Gevolg

Resultaat H impliceert ieder der resultaten A, B, C, D, E, F en G.

Schematisch hebben we dus het volgende aangetoond:



Opgave 5.6

Zij $G = (V_1 \cup V_2, E)$ een bipartiete graaf.

Laat $\sigma(G) = \max_{W_1 \subseteq V_1}[|W_1| - |\Gamma W_1|]$, met $\Gamma W_1 = \{j \in V_2 | (i,j) \in E \text{ voor een } i \in W_1\}$

Toon aan dat het aantal takkken in een maximale koppeling gelijk is aan:

$|V_1| - \sigma(G)$.

Opgave 5.7

Zij $G = (V_1 \cup V_2, E)$ een bipartiete graaf.

Veronderstel dat $\min_{v \in V_1} \delta(v) \geq \max_{w \in V_2} \delta(w)$, d.w.z. ieder knooppunt van V_1 heeft een graad die minstens zo groot is als die van ieder knooppunt van V_2.

Toon aan dat er een maximale koppeling is die alle knooppunten van V_1 bindt.

(Hint: Gebruik Opgave 5.6)

Opgave 5.8

Zij $G = (V_1 \cup V_2, E)$ een bipartiete graaf, en laat $k = \max_{v \in V_1 \cup V_2} \delta(v)$.

a. Toon aan dat er een koppeling bestaat die alle knooppunten met graad k bindt.

b. Toon aan dat er disjuncte koppelingen $M_1, M_2, \ldots, M_k$ bestaan zodat $E = M_1 \cup M_2 \cup \cdots \cup M_k$.

(Hint: Gebruik de Opgaven 5.7 en 5.5).

Opgave 5.9

Toon rechtstreeks het volgende aan:

a. Resultaat A volgt uit resultaat B.

b. De resultaten B en C zijn equivalent.

c. Resultaat C volgt uit resultaat A.

Opgave 5.10

Toon rechtstreeks het volgende aan:

a. De resultaten A en D zijn equivalent.

b. Resultaat B volgt uit resultaat F.

Opgave 5.11*

Bewijs resultaat D rechtstreeks.

3. Maximale koppeling

Literatuur

* J.E.Hopcroft and R.M.Karp: "A $n^{5/2}$ algorithm for maximum matching in bipartite graphs", Journal SIAM on Computing 2 pp. 225-231 (1973).
* Hoofdstuk 5 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds", Holt, Rinehart and Winston, New York (1976).
* Hoofdstuk 10 in:
  C.H.Papadimitriou and K.Steiglitz: "Combinatorial optimization: algorithms and complexity", Prentice Hall, Englwood Cliffs (1982).
* Paragraaf 5 van hoofdstuk 8 in:
  M.N.S.Swamy and K.Thulasiraman: "Graphs, networks and algorithms", Wiley, New York (1981).

In deze paragraaf zullen we een algoritme bespreken om een maximale koppeling in een bipartiete graaf te vinden. Er is een direct verband tussen dit probleem en het maximale stroom probleem, zoals in Stelling 5.7 is afgeleid.

Zij $G = (V_1 \cup V_2, E)$ een bipartiete graaf.
Voeg een extra knooppunt s toe met pijlen (s,v) voor iedere $v \in V_1$, richt de takken van V_1 naar V_2 en voeg ook knooppunt t toe met pijlen (w,t) voor alle $w \in V_2$. Dit geeft een netwerk $N = (V_1 \cup V_2 \cup s \cup t, A)$ en neem de capaciteiten b_{sv} en b_{wt} gelijk aan 1 en de andere capaciteiten ∞.

Een maximale stroom x correspondeert met een maximaal aantal onafhankelijke paden van s naar t, welke weer corresponderen met een maximale koppeling M.
Een minimale (s,t)-snede $(s \cup W_1 \cup W_2, [V_1\text{-}W_1] \cup [V_2\text{-}W_2] \cup t)$ correspondeert met een minimale takkenbedekking $(V_1\text{-}W_1) \cup W_2$, waarbij $|M| = |(V_1\text{-}W_1) \cup W_2|$.

We kunnen dus de resultaten van Hoofdstuk 4 toepassen en bijv. het DMKM algoritme met complexiteit $O(n^3)$ toepassen. Dit kan echter nog worden verbeterd tot een $O(n^{5/2})$ algoritme (zie ook Opgave 4.15). Dit resultaat is afkomstig Hopcroft en Karp, en is ook weer gebaseerd op het zijwaarts zoeken.

Een groeiketen m.b.t. een koppeling M komt overeen met een groeiketen in het netwerk m.b.t. de bijbehorende stroom x. In de DMKM methode worden groeiketens

met zo min mogelijk pijlen bekeken en i.v.m. de capaciteiten 1 hebben deze groeiketens geen gemeenschappelijke knooppunten (behalve s en t). Over al deze groeiketens wordt een hoeveelheid 1 vervoerd. Dit gebeurt in één iteratie en de complexiteit van zo'n iteratie is $O(n^2)$ (zie het bewijs van Stelling 4.8).

De volgende stelling laat zien dat het aantal iteraties hoogstens $2\sqrt{p} + 1$ is, waarbij p de het aantal elementen van een maximale koppeling is. Omdat $p = O(n)$ heeft de DMKM methode, toegepast op het koppelingsprobleem, complexiteit $O(n^{5/2})$. Voordat we deze stelling gaan bewijzen, eerst enkele lemma's.

Lemma 5.1

Zij M_1 en M_2 twee koppelingen met $m_1 = |M_1| > m_2 = |M_2|$, dan zijn er minstens $m_1 - m_2$ groeiketens m.b.t. M_2 in $M_1 \oplus M_2$ zonder gemeenschappelijke knooppunten.

Bewijs

Beschouw de componenten van $M_1 \oplus M_2$: dit zijn kringen en ketens waarin afwisselend takken van $M_1 - M_2$ en van $M_2 - M_1$ voorkomen. Omdat in een kring evenveel takken van $M_1 - M_2$ als van $M_2 - M_1$ voorkomen, zijn er dus minstens $m_1 - m_2$ ketens die beginnen en eindigen met een tak van $M_1 - M_2$. Dit zijn $m_1 - m_2$ groeiketens m.b.t. M_2 zonder gemeenschappelijke knooppunten. □

Lemma 5.2

Zij M_1 en M_2 twee koppelingen met $m_1 = |M_1| > m_2 = |M_2|$, dan bestaat er een groeiketen m.b.t. M_2 met hoogstens $(m_1 + m_2)/(m_1 - m_2)$ takken.

Bewijs

Volgens Lemma 5.1 bestaan er $m_1 - m_2$ groeiketens m.b.t. M_2 in $M_1 \oplus M_2$ zonder gemeenschappelijke knooppunten, dus zeker zonder gemeenschappelijke takken.
In totaal zijn er in $M_1 \oplus M_2$ hoogstens $m_1 + m_2$ takken. Er is dus minstens één groeiketen met maximaal $(m_1 + m_2)/(m_1 - m_2)$ takken. □

Lemma 5.3

Zij M een koppeling, C een kortste groeiketen m.b.t. M en C' een willekeurige groeiketen m.b.t. $M' := M \oplus C$. Dan geldt: $|C'| \geq |C| + 2 \cdot |C' \cap C|$.

Bewijs

$M'' := M' \oplus C'$ is weer een koppeling met $|M''| = |M| + 2$. Dus $M'' \oplus M$ bevat volgens Lemma 5.1 twee disjuncte groeiketens C_1 en C_2 m.b.t. M. Omdat C de kortste groeiketen m.b.t. M is geldt: $|C_1| \geq |C|$ en $|C_2| \geq |C|$.
$M'' \oplus M = C \oplus C'$, dus $|C \oplus C'| \geq |C_1| + |C_2| \geq 2 \cdot |C|$. We kunnen schrijven:
$2 \cdot |C| \leq |C \oplus C'| \leq |C| + |C'| - 2 \cdot |C \cap C'| \rightarrow |C'| \geq |C| + 2 \cdot |C' \cap C|$. □

Lemma 5.4

Veronderstel dat we, startend met $M = \emptyset$, een rij $M_1, M_2, \ldots, M_i, \ldots$ bepalen met $M_{i+1} = M_i \oplus C_i$, waarbij C_i een kortste groeiketen is m.b.t. M_i. Dan geldt:

a. $|C_{i+1}| \geq |C_i|$, $i = 1,2,\ldots$.

b. Voor alle i en j met $|C_i| = |C_j|$ hebben C_i en C_j geen gemeenschappelijke knooppunten.

Bewijs

a. Volgens Lemma 5.3 geldt: $|C_{i+1}| \geq |C_i| + |C_{i+1} \cap C_i| \geq |C_i|$, $i = 1,2,\ldots$.

b. Veronderstel dat de bewering niet waar is, en laat $j > i$. Dan bestaan er getallen k en l zódat $i \leq k < l \leq j$ en C_k en C_l hebben wel gemeenschappelijke knooppunten, maar voor iedere r met $k < r < l$ is C_r wel disjunct met C_k en C_l.
Dan is C_l een groeiketen m.b.t. $M_k \oplus C_k$. Volgens Lemma 5.3 geldt dan: $|C_l| \geq |C_k| + |C_k \cap C_l|$. Omdat $|C_l| = |C_k|$, is $C_k \cap C_l = \emptyset$, d.w.z. C_l en C_k hebben geen takken gemeen, maar wel een knooppunt, zeg v, en $v \in M_k \oplus C_k$. Maar dan hebben C_k en C_l wel een gemeenschappelijke tak, nl. de tak van $M_k \oplus C_k$ incident met v: tegenspraak. □

Stelling 5.9

Zij p het aantal takken van een maximale koppeling, dan is het aantal iteraties hoogstens $2 \cdot \sqrt{p} + 1$.

Bewijs

Volgens Lemma 5.4 is het voldoende om aan te tonen dat in de rij $|C_1|, |C_2|, \ldots, |C_i|, \ldots, |C_p|$ hoogstens $2 \cdot \sqrt{p} + 1$ verschillende getallen voorkomen (dezelfde getallen corresponderen met disjuncte groeiketens en zitten in één iteratie).
We merken allereerst op dat de getallen $|C_i|$ oneven zijn voor iedere i.
Laat r het eerste gehele getal $\geq \sqrt{p}$ zijn, en $q = p - r$. Omdat $|M_q| = q$, is volgens Lemma 5.2: $|C_q| \leq (p + q)/(p - q) = (2p - r)/r = 2p/r - 1$.
Dus voor ieder $1 \leq i \leq q$ is $|C_i|$ één van de oneven getallen $\leq 2p/r - 1$: dit zijn er hoogstens p/r en $p/r \leq \sqrt{p}$.
In de getallen $p - q$ getallen $|C_{q+1}|, |C_{q+2}|, \ldots, |C_p|$ komen hoogstens $p - q = r$ verschillende getallen voor en $r \leq \sqrt{p} + 1$.
Dus in totaal hoogstens $2 \cdot \sqrt{p} + 1$ verschillende getallen. □

Bovenstaande resultaten leiden tot het volgende algoritme.

Algoritme (versie 1)

stap 1: Start met $M = \emptyset$.

stap 2: a. Als er geen groeiketen is: M is een maximale koppeling (STOP).
b. Bepaal met het zijwaarts zoeken disjuncte groeiketens m.b.t. M van minimale lengte, zeg $C_1, C_2, \ldots, C_k$.

stap 3: a. $M := M \oplus C_1 \oplus C_2 \oplus \cdots \oplus C_k$.
b. Ga naar stap 2.

We zullen het algoritme nader uitwerken. Bij het opstellen van wisselketens (om een groeiketen te vinden) gaan we heen en weer tussen V_1 en V_2. We beginnen in de vrije knooppunten van V_1, zeg dat deze de verz. W_1 vormen. Bereiken we vanuit W_1 een vrij knooppunt, dan is er een groeiketen gevonden; wordt een gebonden knooppunt $w_2 \in V_2$ bereikt, dan hoort hier één element van de koppeling bij, d.w.z. een unieke $w_3 \in V_1$ met $(w_2, w_3) \in M$. De wisselketen is dan $[w_1, w_2, w_3]$.

Net zoals in de DMKM methode is het netwerk gelaagd. Dit komt tot uitdrukking doordat we w_1 in W_1, w_2 in W_2 en w_3 in W_3 stoppen. Vanuit W_3 kijken we weer welke nieuwe knooppunten van V_2 bereikt kunnen worden en deze komen in W_4.
Zo gaan we door totdat voor zekere k òf $W_{2k} = \emptyset$ (er bestaat dan geen groeiketen en de koppeling is maximaal) òf W_{2k} bevat een vrij knooppunt (dan is een kortste groeiketen gevonden).

Als we ontdekken dat de koppeling maximaal is, dan hebben we eveneens een minimale takkenbedekking gevonden: de knooppunten van V_1 die niet bereikt zijn en de knooppunten van V_2 die wel bereikt zijn.
Als de kortste groeiketen is gevonden, dan gaan we in het gelaagde netwerk met knooppuntenverz. $W_1, W_2, \ldots, W_{2k}$ als volgt kortste groeiketens bepalen. Voor ieder vrij knooppunt van W_{2k} lopen we via een voorganger terug totdat we in een vrij knooppunt van W_1 zijn aangekomen. Dan is een groeiketen C gevonden en op C verwisselen we de takken van M met die van $M \oplus C$, waardoor de koppeling één tak groter wordt. Deze groeiketens moeten disjunct zijn: een knooppunt dat bezocht is, mag nooit meer in een groeiketen worden opgenomen. Dit moet dus worden bijgehouden.

De organisatie is als volgt:
Met LABEL houden we bij in welke W_k een knooppunt zit:
  LABEL[i] = k $\Leftrightarrow$ $v_i \in W_k$.
In PRED[i] houden we de voorgangers van knooppunt i bij.
Met een boolean CHAIN houden we bij of een knooppunt in een groeiketen mag worden opgenomen:
  CHAIN[i] = true $\Leftrightarrow$ i mag worden opgenomen.
De koppeling geven we aan met MATE:
  MATE[i] = j $\Leftrightarrow$ (i,j) $\in$ M en MATE[i] = 0 $\Leftrightarrow$ i vrij knooppunt.
De vrije eindpunten van groeiketens worden in de verz. X gestopt.
Met de boolean EMPTY houden we bij of de koppeling maximaal is:
  EMPTY = true $\Leftrightarrow$ koppeling is maximaal.
Als een groeiketen wordt gevonden, dan bergen we deze op in PATH:
  PATH[i] = het i-de knooppunt (achterwaarts gerekend) van de groeiketen.
We nemen aan dat de verbindingen tussen V_1 en V_2 via structuurlijsten bekend zijn:
  $L[i] = \{j \in V_2 \mid (i,j) \in E\}$, $i \in V_1$.

Algoritme (versie 2)

stap 1: Start met MATE[i] = 0 voor iedere $i \in V_1 \cup V_2$.

stap 2: (Initialisatie iteratiestap)
  a. Voor i = 1,2,...,n:
    LABEL[i] := 0; PRED[i] := $\emptyset$; CHAIN[i] := false;
    Als $i \in V_1$ èn MATE[i] = 0:
      LABEL[i] := 1; CHAIN[i] := true.
  b. k := 1; X := $\emptyset$.

stap 3: (Onderzoek W_k voor oneven k)
  Voor iedere i met LABEL[i] = k:
  Voor iedere $j \in L[i]$ met $MATE[j] \neq i \wedge (LABEL[j] = 0 \vee LABEL[j] = k+1)$
    LABEL[j] := k+1; PRED[j] := PRED[j] $\cup$ {i}; CHAIN[j] := true.

stap 4: (onderzoek W_k voor even k)
  a. k := k+1; EMPTY := true.
  b. Voor iedere i met LABEL[i] = k:
    EMPTY := false;
    Als MATE[i] = 0: X := X $\cup$ {i};
    Anders: j := MATE[i]; LABEL[j] := k+1;
      PRED[j] := {i}; CHAIN[j] := true.

c. Als EMPTY = true: ga naar stap 7;
Anders: als X ≠ ∅: ga naar stap 5;
anders: k := k+1 en ga naar stap 3.

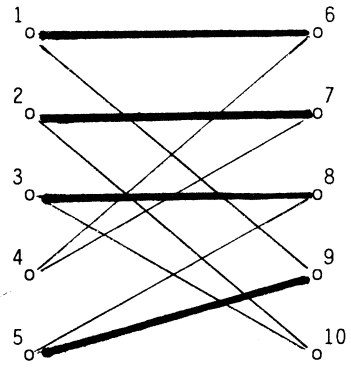
stap 5: (Groeiketens bepalen en koppeling aanpassen)
Voor iedere i ∈ X doe totdat CHAIN[i] = false:
a. q := i; PATH[1] := i; l := 2.
b. Voor iedere p ∈ PRED[q]:
Als CHAIN[p] = false: PRED[q] := PRED[q] - p.
c. Als PRED[q] = ∅: CHAIN[i] := false;
Anders: Neem p ∈ PRED[q]; PATH[l] := p; l := l + 1;
CHAIN[p] := false;
Als LABEL[p] = 1: ga naar stap 5d;
Anders: q := p en ga naar stap 5b.
d. Als CHAIN[i] = true:
Voor j = 1,2,...,(l-1)/2:
MATE[PATH[2j-1]] := PATH[2j];MATE[PATH[2j]] := PATH[2j-1].

stap 6: (Volgende iteratie)
Ga naar stap 2.

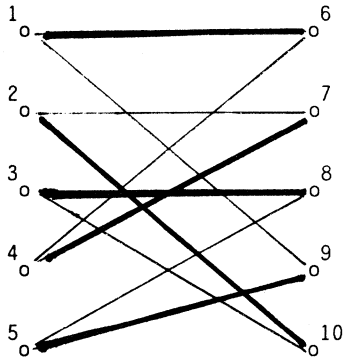
stap 7: (Er is geen groeiketen)
a. $W_1 = \{i \in V_1 |$ LABEL[i] = 0$\}$; $W_2 = \{j \in V_2 |$LABEL[j] ≠ 0$\}$.
b. MATE geeft een maximale koppeling en $W_1 \cup W_2$ een minimale takken-takkenbedekking (STOP).

Voorbeeld 5.3

Iteratie 1

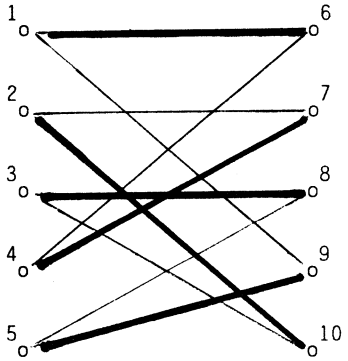


LABEL[i] = 1 voor i = 1,2,3,4 en 5;
LABEL[i] = 2 voor i = 6,7,8,9 en 10.
PRED[i] = ∅ voor i = 1,2,3,4 en 5;
PRED[6] = {1,4}; PRED[7] = {2,4}; PRED[8] = {3,5};
PRED[9] = {1,5}; PRED[10] = {2,3}.
X = {6,7,8,9,10}.
We vinden de volgende groeiketens:
6 ← 1: MATE[6] = 1; MATE[1] = 6. 7 ← 2: MATE[7] = 2; MATE[2] = 7.
8 ← 3: MATE[8] = 3; MATE[3] = 8. 9 ← 5: MATE[9] = 5; MATE[5] = 9.
Vanuit 10 wordt geen groeiketen gevonden, want de voorgangers 2 en 3 zijn al



Iteratie 2

LABEL[1] = 0 voor i = 1,2,...,10;
LABEL[4] = 1;
LABEL[6] = 2; PRED[6] = {4};
LABEL[7] = 2; PRED[7] = {4};
LABEL[1] = 3; PRED[1] = {6};
LABEL[2] = 3; PRED[2] = {7};
LABEL[9] = 4; PRED[9] = {1};
LABEL[10] = 4; PRED[10] = {2};
LABEL[5] = 5; PRED[5] = {9}.
X = {10}.
We vinden de volgende groeiketen:
10 $\leftarrow$ 2: MATE[10] = 2; MATE[2] = 10. 7 $\leftarrow$ 4: MATE[7] = 4; MATE[2] = 4.



Iteratie 3

LABEL[1] = 0 voor i = 1,2,...,10;
EMPTY = true:
W_1 = {1,2,3,4,5}; W_2 = $\emptyset$;
$W_1 \cup W_2$ is minimale takkenbedekking.
De maximale kopppeling wordt gegeven door:
MATE[1] = 6; MATE[2] = 10; MATE[3] = 8;
MATE[4] = 7; MATE[5] = 9.

Stelling 5.10

Het algoritme bepaalt met complexiteit $O(n^{5/2})$ een maximale koppeling en een minimale takkenbedekking.

Bewijs

Het algoritme stopt als er geen groeiketen meer is en volgens Stelling 5.1 is de koppeling dan maximaal.
Beschouw vervolgens de geconstrueerde $W_1 \cup W_2$. Dit is een takkenbedekking, immers: Stel $(i,j) \in E$, $i \notin W_1$ en $j \notin W_2$. Dan heeft $i \in V_1$ een LABEL[i] $\neq$ 0, d.w.z. er is een wisselketen, beginnend in een vrij knooppunt van V_1 en eindigend via een tak van M in knooppunt i. Omdat $j \notin W_2$ is LABEL[j] = 0: j zit niet in een groeiketen. Maar dan zou vanuit i via de tak (i,j) j een LABEL[j] $\neq$ 0 hebben gekregen.

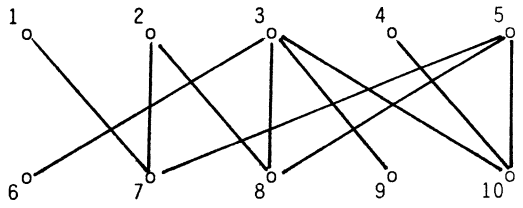
De takkenbedekking $W_1 \cup W_2$ heeft $|M|$ elementen, dus is minimaal,
immers: Veronderstel dat er in V_1 r vrije knooppunten zijn. Laat vanuit het i-de vrije knooppunt een wisselboom met t_i takken bestaan en laat de bomen vanuit verschillende vrije knooppunten geen gemeenschappelijke knooppunten hebben. Behalve het vrije knooppunt zijn alle andere gebonden door tak van M die loopt van een knooppunt van V_2 naar een knooppunt van V_1: in deze boom zitten $\frac{1}{2}t_i + 1$ knooppunten van V_1 en $\frac{1}{2}t_i$ knooppunten van V_2. Dus geldt:

$$|M| = |V_1| - r = |V_1| - \Sigma_{i=1}^{r} (\tfrac{1}{2}t_i + 1) + \Sigma_{i=1}^{r} \tfrac{1}{2}t_i = |W_1| + |W_2|.$$

Voor de complexiteit geldt het volgende:
Er zijn maximaal $2 \cdot \sqrt{p} + 1$ iteraties, met $p = |M|$ (Stelling 5.9), dus $O(n^{1/2})$ iteraties. Per iteratie is de complexiteit $O(n^2)$, omdat in feite dezelfde techniek als bij de DMKM methode wordt toegepast (zie Stelling 4.8). □

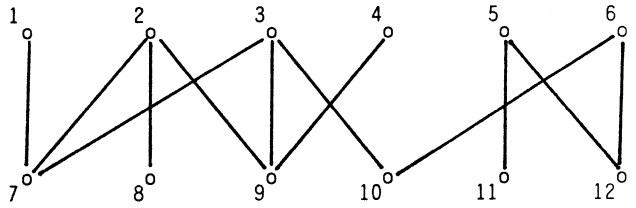
Opgave 5.12

Bepaal een maximale koppeling en een minimale takkenbedekking voor de hiernaast getekende graaf.



Opgave 5.13

Bepaal een maximale koppeling en een minimale takkenbedekking voor hiernaast getekende graaf.



4. Maximaal gewogen koppeling

Literatuur

* Paragraaf 8 van hoofdstuk 5 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds", Holt, Rinehart and Winston, New York (1976).
* Paragraaf 2 van hoofdstuk 11 in:
  C.H.Papadimitriou and K.Steiglitz: "Combinatorial optimization: algorithms and complexity", Prentice Hall, Englwood Cliffs (1982).

Ook in deze paragraaf nemen we aan dat $G = (V_1 \cup V_2, E)$ een bipartiete graaf is. Bovendien veronderstellen we dat aan een iedere tak (i,j) een gewicht w_{ij} is toegekend. We zijn geïnteresseerd in het vinden van een koppeling met een maximaal gewicht. Zo'n koppeling heet een maximaal gewogen koppeling. Dit is een generalisatie van het maximale koppelingsprobleem uit de vorige paragraaf (neem $w_{ij} = 1$ voor alle (i,j)). Omdat takken met een negatief gewicht toch nooit gekozen zullen worden, kunnen we wel aannemen dat $w_{ij} \geq 0$ voor alle (i,j).

Het gewogen koppelingsprobleem is equivalent met het toewijzingsprobleem (zie het dictaat Operations Research Technieken en ook opgave 5.14). We zullen voor het gewogen koppelingsprobleem de Hongaarse methode bespreken, zoals bij Operations Research Technieken is gedaan voor het toewijzingsprobleem. Deze methode is gebaseerd op eigenschappen van de lineaire programmering, en staat in nauwe relatie tot de methode voor het minimale kostenprobleem met onder- en bovengrenzen (paragraaf 7 van hoofdstuk 4).

De lineaire programmeringsformulering van het gewogen koppelingsprobleem is:

$$(5.3) \qquad \max \left\{ \Sigma_i \Sigma_j \, w_{ij} x_{ij} \;\middle|\; \begin{array}{l} \Sigma_j \, x_{ij} \leq 1, \; i = 1,2,\ldots,p \\ \Sigma_i \, x_{ij} \leq 1, \; j = 1,2,\ldots,q \\ x_{ij} \geq 0, \text{ voor alle } i \text{ en } j \end{array} \right\}$$

We hebben in de vorige paragraaf gezien dat de matrix van (5.3) totaal unimodulair is, en dat er een één-éénduidig verband bestaat tussen de hoekpunten van (5.3) en de koppelingen van de graaf, waarbij de waarde van de doelfunctie gelijk is aan het gewicht van de koppeling. Dit verband wordt gegeven door: $(i,j) \in M \Leftrightarrow x_{ij} = 1$.

Beschouw het duale probleem van (5.3):

$$(5.4) \qquad \min \left\{ \Sigma_{i=1}^{p} s_i + \Sigma_{j=1}^{q} t_j \;\middle|\; \begin{array}{l} s_i + t_j \geq w_{ij} \text{ voor alle } (i,j) \in E \\ s_i, t_j \geq 0,\ 1 \leq i \leq p,\ 1 \leq j \leq q \end{array} \right\}.$$

De Hongaarse methode werkt als volgt:

In iedere iteratie hebben we een (0,1)-vector x en een (s,t) zódanig dat:

(5.5a) x is een toelaatbare (0,1)-oplossing van (5.3), d.w.z. x correspondeert met een koppeling M.

(5.5b) (s,t) is een toelaatbare oplossing van (5.4).

(5.5c) Als $x_{ij} > 0$, dan is $s_i + t_j = w_{ij}$.

(5.5d) Als $t_j > 0$, dan is $\Sigma_{i=1}^{p} x_{ij} = 1$.

Uit de theorie van de lineaire programmering volgt dat als bovendien voor alle $i \in V_1$ geldt:

(5.5e) Als $s_i > 0$, dan is $\Sigma_{j=1}^{q} x_{ij} = 1$,

dan zijn x en (s,t) optimaal voor (5.3) resp. (5.4).

We starten met $x = 0$, $t = 0$ en $s_i = \max_{k,l} w_{kl}$ voor $i = 1,2,\ldots,p$.
Deze x en (s,t) voldoen aan (5.5a) t/m (5.5d). We gaan in een aantal iteraties zorgen dat ook aan (5.5e) wordt voldaan, terwijl (5.5a) t/m (5.5d) steeds blijven gelden. Een algemene iteratiestap ziet er als volgt uit (voor de vrije knooppunten $i \in V_1$ zal steeds gelden: $s_i = \min_k \{s_k \mid k \in V_1\} > 0$).

Beschouw de deelgraaf G' van de oorspronkelijke graaf met dezelfde knooppuntenverz. $V_1 \cup V_2$ en met als takken $E' := \{(i,j) \mid s_i + t_j = w_{ij}\}$.
Uit voorwaarde (5.5c) volgt dat $M \subseteq E'$.

Vervolgens gaan we in deze deelgraaf disjuncte groeiketens van minimale lengte zoeken (volgens het algoritme van de vorige paragraaf).
Als er groeiketens C_i, $i = 1,2,\ldots,k$ worden gevonden: breid M uit volgens $M_0 := M$ en $M_i := M_{i-1} \oplus C_i$, $i = 1,2,\ldots,k$.
Een oplossing x verandert door het toevoegen van groeiketen C als volgt:

$$x_{ij} := \begin{cases} x_{ij} & \text{als } (i,j) \notin C \\ 1 - x_{ij} & \text{als } (i,j) \in C \end{cases}$$

Omdat s en t niet veranderen en alleen takken (i,j) met $s_i + t_j = w_{ij}$ worden gebruikt, is het direct in te zien dat weer aan de voorwaarden (5.5a) t/m (5.5d) is voldaan, terwijl nu ook aan (5.5e) wordt voldaan door het beginpunt van de groeiketens C_i.
Ook heeft een nieuwe koppeling een groter gewicht: stel $C = [i_1,j_1,\ldots,i_k,j_k]$, dan is:

$$\begin{aligned}\text{gewicht}(M \oplus C) - \text{gewicht}(M) &= (w_{i_1j_1} + \cdots + w_{i_kj_k}) - (w_{i_2j_1} + \cdots + w_{i_kj_{k-1}}) \\ &= (s_{i_1} + t_{j_1} + \cdots + s_{i_k} + t_{j_k}) - (s_{i_2} + t_{j_1} + \cdots + s_{i_k} + t_{j_{k-1}}) \\ &= s_{i_1} + t_{j_k} \geq s_{i_1} > 0.\end{aligned}$$

Veronderstel vervolgens dat er geen groeiketen wordt gevonden.

We onderscheiden de volgende gevallen:

a. V_1 heeft geen vrije knooppunten:
   $\Sigma_{j=1}^{q} x_{ij} = 1$, $1 \leq i \leq p$: aan (5.5e) is voldaan en M is een maximale gewogen koppeling.

b. $s_i = 0$ voor alle vrije knooppunten van V_1: er is eveneens aan (5.5e) voldaan en M is dus een maximale gewogen koppeling.

c. Er is een vrij knooppunt $i \in V_1$ met $s_i > 0$ (alle vrije knooppunten van V_1 hebben dan deze s-waarde):

   Aan het einde van de procedure om een groeiketen te bepalen hebben we:

   $W_1 \cup W_2$, de bereikbare knooppunten van $V_1 \cup V_2$. Het volgende wordt berekend:

   * $\pi_j := \min_{i \in W_1} \{s_i + t_j - w_{ij}\}$, $j \notin W_2$.
   * $\delta_1 := \min_{j \notin W_2} \pi_j$; $\delta_2 := \min_{i \in W_1} s_i$; $\delta := \min(\delta_1,\delta_2)$.
   * $s_i := s_i - \delta$ voor alle $i \in W_1$;
     $t_j := t_j + \delta$ voor alle $j \in W_2$.

Deze nieuwe oplossing (s,t) voldoet met x weer aan (5.5a) t/m (5.5d) en in de nieuwe G' zijn meer knooppunten bereikbaar dan in de vorige, dus na een eindig aantal iteraties wordt een groeiketen gevonden of wordt ontdekt dat de koppeling maximaal is, immers:

(5.5a): Deze eigenschap geldt, want x is niet veranderd.

(5.5b): Stel dat dit niet meer geldt, d.w.z. $i \in W_1$ en $j \notin W_2$. Maar dan is de nieuwe waarde $s_i + t_j$ = oude waarde $s_i + t_j - \delta \geq w_{ij}$, vanwege $\delta \leq \delta_1 \leq \pi_j \leq s_i + t_j - w_{ij}$ voor $i \in W_1$ en $j \notin W_2$.

(5.5c): Als $x_{ij} > 0$, dan is $(i,j) \in M$, zodat òf $i \in W_1$, $j \in W_2$ òf $i \notin W_1$, $j \notin W_2$: in beide gevallen geldt: $s_i + t_j = w_{ij}$.

(5.5d): Als $t_j > 0$: stel t_j was 0; dan is $j \in W_2$, terwijl er geen groeiketen wordt gevonden: j is gebonden, d.w.z. $\Sigma_{i=1}^{p} x_{ij} = 1$.

Als $\delta = \delta_2$: s_i wordt 0 voor alle vrije knooppunten: M is een maximale gewogen koppeling.

Als $\delta = \delta_1 < \delta_2$: er is een $i \in W_1$ en een $j \notin W_2$ waarvoor $s_i + t_j := w_{ij}$ en in de vorige iteratie $s_i + t_j > w_{ij}$: G′ wordt uitgebreid met (i,j) en j wordt hiermee bereikbaar.
Als $i \notin W_1$, $j \in W_2$: $s_i + t_j$ wordt groter, dus als (i,j) in G′ zat, dan verdwijnt deze tak uit G′; echter, omdat $j \in W_2$, $i \notin W_1$ zit (i,j) niet in M en komt (i,j) niet voor in een groeiketen m.b.t. M.
Vervolgens bepalen we de nieuwe $W_1 \cup W_2$ en herhalen we de berekening van de π's, de δ's, de s- en t-waarden, etc.

Samengevat ziet het algoritme er als volgt uit.

<u>stap 1</u>: (Initialisatie)
Start met $M = \emptyset$; $s_i := \max_{k,l} w_{kl}$, $i \in V_1$; $t_j = 0$, $j \in V_2$.
$E' := \{(i,j) \mid s_i + t_j = w_{ij}\}$; $G' := (V_1 \cup V_2, E')$.

<u>stap 2:</u> (Bepaling groeiketen)
a. Bepaal kortste onafhankelijke groeiketens in G′ (stappen 2 t/m 5 van het algoritme uit de vorige paragraaf).
b. Als er groeiketens $C_1, C_2, \ldots, C_k$ worden gevonden: ga naar stap 3;
Als er geen groeiketen bestaat: bepaal de bereikbare knooppunten W_1 en W_2, en ga naar stap 4.

<u>stap 3</u>: (Koppeling aanpassen)
a. Voor $i = 1,2,\ldots,k$: $M := M \oplus C_i$.
b. Ga naar stap 2.

<u>stap 4</u>: (Optimaliteitscontrole)
Als er geen vrije knooppunten in V_1 zijn: M is een maximale gewogen koppeling (STOP).
Anders: ga naar stap 5.

stap 5: (Aanpassen graaf G')

a. $\pi_j := \min_i \{s_i + t_j - w_{ij} \mid i \in W_1\}$, $j \notin W_2$.

b. $\delta_1 := \min_{j \notin W_2} \pi_j$; $\delta_2 := \min_{i \in W_1} s_i$.

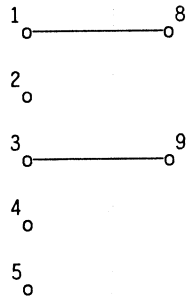
c. Als $\delta_2 \leq \delta_1$: M is een maximale gewogen koppeling (STOP).

Anders: Laat $F' := \{(i,j) \mid i \in W_1, j \notin W_2, s_i + t_j - w_{ij} = \delta_1\}$;
Laat $F'' := \{(i,j) \mid i \notin W_1, j \in W_2, s_i + t_j - w_{ij} = 0\}$;
$s_i := s_i - \delta_1$, $i \in W_1$; $t_j := t_j + \delta_1$, $j \in W_2$;
Voeg F' aan G' toe, verwijder F'' uit G' en bepaal de nieuwe W_1 en W_2 door vanuit de takken van F' verder te lopen (als er groeiketens $C_1,\ldots,C_k$ bestaan: ga naar 3);
ga naar 5a.

Voorbeeld 5.4

We gaan een maximale gewogen koppeling bepalen voor de volledige bipartiete graaf $K_{5,5}$.
De gewichten w_{ij} staan in nevenstaande 5x5 matrix.
De knooppunten van V_1 nummeren we met 1 t/m 5, die van V_2 met 6 t/m 10.

$$W = \begin{pmatrix} 3 & 8 & 9 & 1 & 6 \\ 1 & 4 & 1 & 5 & 5 \\ 7 & 2 & 7 & 9 & 2 \\ 3 & 1 & 6 & 8 & 8 \\ 2 & 6 & 3 & 6 & 2 \end{pmatrix}$$

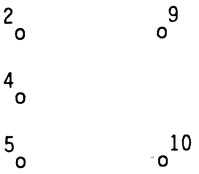


Iteratie 1

$M = \emptyset$; $s_i = 9$, $1 \leq i \leq 5$; $t_j = 0$, $6 \leq j \leq 10$.
$E' = \{(1,8),(3,9)\}$.
Onafhankelijke groeiketens: $C_1 = [1,8]$; $C_2 = [3,9]$.
$M = \{(1,8),(3,9)\}$.

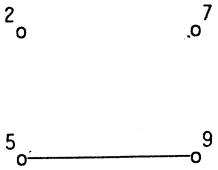
Iteratie 2

Geen groeiketen in G'; $W_1 = \{2,4,5\}$; $W_2 = \emptyset$.
$\pi_6 = 6$; $\pi_7 = 3$; $\pi_8 = 3$; $\pi_9 = 1$; $\pi_{10} = 1$; $\delta_1 = 1$; $\delta_2 = 9$.
$F' = \{(4,9),(4,10)\}$; $F'' = \emptyset$; $s_2 = s_4 = s_5 = 8$.
Er is een groeiketen: $C_1 = [4,10]$.
$M = \{(1,8),(3,9),(4,10)\}$.

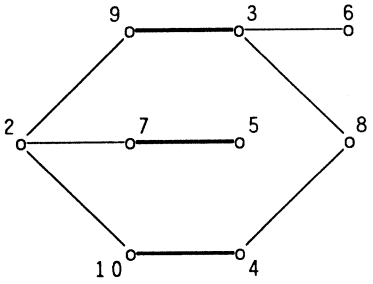


Iteratie 3

$E' = \{(1,8),(3,9),(4,9),(4,10)\}$.
Geen groeiketen in G'; $W_1 = \{2,5\}$; $W_2 = \emptyset$.
$\pi_6 = 6$; $\pi_7 = 2$; $\pi_8 = 5$; $\pi_9 = 2$; $\pi_{10} = 3$; $\delta_1 = 2$; $\delta_2 = 8$.
$F' = \{(5,7),(5,9)\}$; $F'' = \emptyset$; $s_2 = s_5 = 6$.
Er is een groeiketen: $C_1 = [5,7]$.
$M = \{(1,8),(3,9),(4,10),(5,7)\}$.



Iteratie 4



$E' = \{(1,8),(3,9),(4,9),(4,10),(5,7),(5,9)\}$.
Geen groeiketen in G'; $W_1 = \{2\}$; $W_2 = \emptyset$.
$\pi_6 = 5$; $\pi_7 = 2$; $\pi_8 = 5$; $\pi_9 = 1$; $\pi_{10} = 1$.
$\delta_1 = 1$; $\delta_2 = 6$.
$F' = \{(2,9),(2,10)\}$; $F'' = \emptyset$; $s_2 = 5$.
$W_1 = \{2,3,4\}$; $W_2 = \{9,10\}$.
$\pi_6 = 2$; $\pi_7 = 1$; $\pi_8 = 2$; $\delta_1 = 1$; $\delta_2 = 5$.
$F' = \{(2,7)\}$; $F'' := \{(5,9)\}$; $s_2 = 4$; $s_3 = 8$; $s_4 = 7$; $t_9 = t_{10} = 1$.
$W_1 = \{2,3,4,5\}$; $W_2 = \{7,9,10\}$. $\pi_6 = 1$; $\pi_8 = 1$; $\delta_1 = 1$; $\delta_2 = 4$.
$F' = \{(3,6),(3,8),(4,8)\}$; $F'' = \emptyset$.
$s_2 = 3$; $s_3 = 7$; $s_4 = 6$; $s_5 = 5$; $t_7 = 1$; $t_9 = t_{10} = 2$.
Er is een groeiketen in G': $C_1 = [2,9,3,6]$.
$M = \{(1,8),(2,9),(4,10),(5,7),(3,6)\}$.

Iteratie 5

$E' = \{(1,8),(3,9),(4,9),(4,10),(5,7),(2,9),(2,10),(2,7),(3,6),(3,8),(4,8)\}$.
M is een maximaal gewogen koppeling.

Stelling 5.10

Het algoritme voor het vinden van een maximale gewogen koppeling is correct en heeft, bij een geschikte implementatie, complexiteit $O(n^3)$.

Bewijs

Uit de hierboven gegeven beschrijving van het algoritme volgt de correctheid. De koppeling M bevat maximaal n takken. Er zijn dus hoogstens n iteraties en het is dus voldoende om aan te tonen dat de hoeveelheid werk per iteratie $O(n^2)$ is.
Het bepalen aan het begin van een iteratie van wisselketens is $O(n^2)$, omdat dit volgens de DMKM methode gebeurt. Indien er geen groeiketens worden gevonden, moet extra werk worden verricht. Beschouw de hoeveelheid werk verbonden met het bereikbaar worden van één nieuw knooppunt (dit kan $O(n)$ keer gebeuren). We berekenen niet alle grootheden geheel opnieuw, maar passen de huidige waarden aan. Voor het aanpassen van de π's, δ's, s- en t-getallen, en het uitbreiden van de wisselketens geeft dit $O(n)$ werk.
Hiermee is aangetoond dat - bij een geschikte implementatie - de totale complexiteit $O(n^3)$ is. □

Opgave 5.14

Beschouw het toewijzingsprobleem:

In de volledige bipartiete graaf $K_{n,n}$ zijn aan iedere tak kosten toegekend, c_{ij} voor tak (i,j). Bepaal een volmaakte koppeling met minimale kosten.

Toon aan dat het toewijzingsprobleem equivalent is met het gewogen koppelingsprobleem.

Opgave 5.15

Bepaal een maximale gewogen koppeling voor de volledige bipartiete graaf $K_{5,5}$.

De gewichten w_{ij} staan in nevenstaande 5x5 matrix.

$$W = \begin{pmatrix} 5 & 6 & 4 & 8 & 5 \\ 8 & 8 & 6 & 3 & 7 \\ 9 & 5 & 6 & 4 & 9 \\ 6 & 6 & 5 & 8 & 5 \\ 0 & 8 & 7 & 8 & 7 \end{pmatrix}$$

Opgave 5.16

Toon aan dat een maximale gewogen koppeling M geen maximale koppeling hoeft te zijn, d.w.z. er kan een koppeling M' bestaan met $|M'| > |M|$.

5. Gilmore-Gomory en Gale-Shapley koppelingen.

Literatuur

* Paragraaf 9 en paragraaf 10 van hoofdstuk 5 in:
  E.L.Lawler: "Combinatorial optimization: networks and matroïds",
  Holt, Rinehart and Winston, New York (1976).

Beschouw de volledige bipartiete graaf $K_{n,n}$ met knooppuntenverz. $V_1 \cup V_2$, en met gewichten w_{ij} voor ieder tweetal (i,j). Laat $V_1 = \{v_1, v_2, \ldots, v_n\}$ en $V_2 = \{w_1, w_2, \ldots, w_n\}$.
Veronderstel dat bij iedere $v_i \in V_1$ een reëel getal α_i en bij iedere $w_j \in V_2$ een reëel getal β_j behoort, en dat de knooppunten zó genummerd zijn dat:
$\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_n$ en $\beta_1 \geq \beta_2 \geq \cdots \geq \beta_n$.

We spreken van een Gilmore-Gomory graaf als de gewichten w_{ij} voldoen aan:

$$(5.6) \qquad w_{ij} = \begin{cases} \int_{\alpha_i}^{\beta_j} f(y)dy & \text{als } \beta_j \geq \alpha_i \\ \int_{\beta_j}^{\alpha_i} g(y)dy & \text{als } \beta_j < \alpha_i \ , \end{cases}$$

waarbij f(y) en g(y) gegeven functies zijn met $f(y) + g(y) \geq 0$.

Stelling 5.11

(i) De koppeling $M_1 = \{(v_i, w_i),\ 1 \leq i \leq n\}$ is een volmaakte koppeling met minimaal gewicht;

(ii) De koppeling $M_2 = \{(v_i, w_{n+1-i}),\ 1 \leq i \leq n\}$ is een volmaakte koppeling met maximaal gewicht.

(Opmerking: dergelijke "niet gekruiste" resp. "maximaal gekruiste" koppelingen heten Gilmore-Gomory koppelingen).

Bewijs

(i) Zij $M \neq M_1$ een willekeurige volmaakte koppeling. Dan zijn er dus in M takken (v_i, w_j) en (v_k, w_l) met $i < k$ en $j > l$.
Beschouw $M' := M \cup (v_k, w_j) \cup (v_i, w_l) - (v_i, w_j) - (v_k, w_l)$.
Laat $\Delta w :=$ gewicht(M') - gewicht(M), dan zullen we aantonen dat $\Delta w \leq 0$.

Als $\alpha_k \geq \beta_l$: $\Delta w = \int_{\beta_j}^{\alpha_k} g(y)dy + \int_{\beta_l}^{\alpha_i} g(y)dy - \int_{\beta_j}^{\alpha_i} g(y)dy - \int_{\beta_l}^{\alpha_k} g(y)dy = 0.$

Als $\beta_j \leq \alpha_k \leq \beta_l \leq \alpha_i$:

$$\Delta w = \int_{\beta_j}^{\alpha_k} g(y)dy + \int_{\beta_l}^{\alpha_i} g(y)dy - \int_{\beta_j}^{\alpha_i} g(y)dy - \int_{\alpha_k}^{\beta_l} f(y)dy$$

$$= - \int_{\alpha_k}^{\beta_l} \{g(y) + f(y)\}dy \leq 0.$$

Als $\alpha_k \leq \beta_j \leq \beta_l \leq \alpha_i$:

$$\Delta w = \int_{\alpha_k}^{\beta_j} f(y)dy + \int_{\beta_l}^{\alpha_i} g(y)dy - \int_{\beta_j}^{\alpha_i} g(y)dy - \int_{\alpha_k}^{\beta_l} f(y)dy$$

$$= - \int_{\beta_j}^{\beta_l} \{g(y) + f(y)\}dy \leq 0.$$

Als $\beta_j \leq \alpha_k \leq \alpha_i \leq \beta_l$:

$$\Delta w = \int_{\beta_j}^{\alpha_k} g(y)dy + \int_{\alpha_i}^{\beta_l} f(y)dy - \int_{\beta_j}^{\alpha_i} g(y)dy - \int_{\alpha_k}^{\beta_l} f(y)dy$$

$$= - \int_{\alpha_k}^{\alpha_i} \{g(y) + f(y)\}dy \leq 0.$$

Als $\alpha_k \leq \beta_j \leq \alpha_i \leq \beta_l$:

$$\Delta w = \int_{\alpha_k}^{\beta_j} f(y)dy + \int_{\alpha_i}^{\beta_l} f(y)dy - \int_{\beta_j}^{\alpha_i} g(y)dy - \int_{\alpha_k}^{\beta_l} f(y)dy$$

$$= - \int_{\beta_j}^{\alpha_i} \{g(y) + f(y)\}dy \leq 0.$$

Als $\alpha_k \leq \alpha_i \leq \beta_j \leq \beta_l$:

$$\Delta w = \int_{\alpha_k}^{\beta_j} f(y)dy + \int_{\alpha_i}^{\beta_l} f(y)dy - \int_{\alpha_i}^{\beta_j} f(y)dy - \int_{\alpha_k}^{\beta_l} f(y)dy = 0.$$

Aldus zien we dat een koppeling zonder gekruiste takken, dus M_1, een minimaal gewicht heeft.

(ii) Dit volgt op dezelfde manier: niet gekruiste takken vervangen door gekruiste takken maakt het gewicht niet kleiner. Het maximale gewicht wordt dus verkregen door zoveel mogelijk gekruiste takken, d.w.z. M_2. □

Beschouw een groep personen bestaande uit n mannen en n vrouwen. Iedere man rangschikt de vrouwen naar zijn voorkeur, en evenzo rangschikt iedere vrouw de mannen naar haar voorkeur.

Een volmaakte koppeling heet instabiel als er een man en een vrouw zijn die niet aan elkaar gekoppeld zijn, maar die beiden elkaar verkiezen boven hun huidige partners. Een stabiele koppeling (d.w.z. niet instabiel) heet een Gale-Shapley koppeling als iedere man in iedere andere stabiele koppeling niet beter af is, d.w.z. niet gekoppeld is aan een vrouw die de voorkeur heeft boven zijn huidige partner.

We zullen hieronder een constructief bewijs (algoritme) geven voor de existentie van een Gale-Shapley koppeling bij iedere gegeven rangschikking. Het algoritme werkt als volgt:

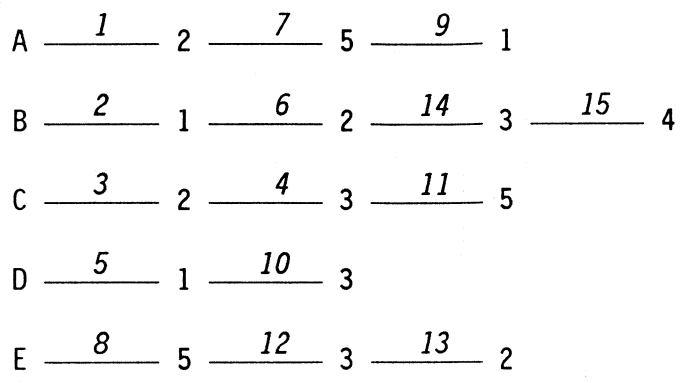
Voor i = 1,2,...,n:
  Koppel de i-de man als volgt:
    a. Geef man i de vrouw van zijn volgende voorkeur (begin met hoogste voorvoorkeur), zeg vrouw j.
    b. Als vrouw j man i accepteert (d.w.z. als ze nog niet aan een man was gekoppeld of als man i haar meer bevalt dan haar huidige man):
        * koppel man i en vrouw j;
        * als vrouw j haar huidige partner, zeg man k, verlaat:
          probeer man k te koppelen volgens deze zelfde procedure.
      Als vrouw j man i niet accepteert (d.w.z. als ze aan een man is gekoppeld die haar voorkeur heeft boven man i): ga naar onderdeel a.

Voorbeeld 5.5
Beschouw onderstaande voorkeursmatrices, M voor de mannen en V voor de vrouwen

$$M = \begin{pmatrix} 2 & 5 & 1 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 5 & 4 & 1 \\ 1 & 3 & 2 & 4 & 5 \\ 5 & 3 & 2 & 1 & 4 \end{pmatrix} \quad \text{en} \quad V = \begin{pmatrix} 2 & 4 & 5 & 3 & 1 \\ 4 & 3 & 5 & 1 & 2 \\ 1 & 3 & 4 & 2 & 5 \\ 4 & 2 & 1 & 3 & 5 \\ 5 & 2 & 3 & 1 & 4 \end{pmatrix}$$

De betekenis hiervan is als volgt: $m_{ij} = k \Leftrightarrow$ vrouw k komt bij man i op de j-de plaats; de voorkeur (in afnemende volgorde) van man 1 is: vrouw 2, vrouw 5, vrouw 1, vrouw 3 en vrouw 4.
Voor de vrouwen geldt: $v_{ij} = k \Leftrightarrow$ vrouw i geeft man j voorkeur k; de voorkeur (in afnemende volgorde van vrouw 1 is: man 5, man 1, man 4, man 2 en man 3.

Schematisch gaat het algoritme als volgt (de mannen noemen we A t/m E, de vrouwen 1 t/m 5; het getal bij de verbindingen is het nummer van de toewijzing).



Een Gale-Shapley koppeling is:

man A ↔ vrouw 1

man B ↔ vrouw 4

man C ↔ vrouw 5

man D ↔ vrouw 3

man E ↔ vrouw 2

We zullen nu een programma geven voor dit algoritme. Hierin gebruiken we de volgende parameters:
MATE[j] = i ⇔ als vrouw j aan man i is gekoppeld;
NEXT[i] geeft de volgende keuze van man i aan.
Procedure partner(i) koppelt man i.

```
GALE-SHAPLEY(M,V);

var i,j,k: integer;
     MATE: array[1:n] of integer;
     NEXT: array[1:n] of integer;

    procedure partner(i);
    var k: integer;
    begin NEXT[i] := NEXT[i] + 1; j := M[i,NEXT[i]];
          if   MATE[j] = 0 then MATE[j] := i
          else begin if   V[j,i] < V[j,MATE[j]] then
                          begin k := MATE[j]; MATE[j] := i; partner(k) end
                     else partner(i)
               end
    end partner;

begin for j := 1 to n do MATE[j]:= NEXT[j] := 0;
      for i := 1 to n do partner(i)
end GALE-SHAPLEY.
```

Stelling 5.12

Het Gale-Shapley algoritme is correct en heeft complexiteit $\mathcal{O}(n^2)$.

Bewijs

De mannen krijgen de vrouwen in afnemende voorkeur toegewezen, en de vrouwen krijgen de mannen in toenemende volgorde toegewezen: een verbroken koppeling wordt dus nooit meer hersteld.

Het algoritme eindigt als alle mannen een partner hebben toegewezen gekregen (dan hebben ook alle vrouwen een partner). Een man die gekoppeld is aan de vrouw van zijn laagste voorkeur houdt deze definitief. Het algoritme is dus eindig en heeft maximaal n^2 stappen.

De verkregen koppeling is stabiel, immers:

Stel man i en vrouw j zijn niet aan elkaar gekoppeld, maar hebben liever elkaar dan hun huidige partners. Man i was op zeker moment aan vrouw j gekoppeld, wat in strijd is met de eigenschap dat de vrouwen er steeds "beter op worden".

Een vrouw heet een mogelijke kandidaat voor een man als er een stabiele koppeling bestaat waarin ze gekoppeld zijn. Het is nu voldoende te bewijzen dat een mogelijke kandidaat nooit meer wordt verlaten, want dan krijgt iedere man de beste mogelijke kandidaat.

Veronderstel dat tijdens het algoritme man i aan vrouw j wordt gekoppeld, vrouw j een mogelijke kandidaat voor man i is en dit (het koppelen van een mogelijke kandidaat) voor de eerste keer gebeurt tijdens het algoritme.

Stel dat het koppel (i,j) wel wordt verbroken, zeg vrouw j neemt man k, d.w.z. vrouw j verkiest man k boven man i.

Omdat vrouw j een mogelijke kandidaat voor man i is, is er een stabiele koppeling M waarin (i,j) een koppel is. Stel man k is in deze koppeling M gekoppeld aan vrouw l, dus vrouw l is een mogelijke kandidaat voor man k.

Omdat tijdens het algoritme (i,j) het eerste mogelijke koppel is dat wordt gevormd, was man k tijdens het algoritme nog nooit aan vrouw l gekoppeld toen hij aan j werd gekoppeld: man k verkiest vrouw j boven vrouw l.

Dus man k en vrouw j verkiezen elkaar boven hun partners in M: M niet stabiel, wat een tegenspraak geeft.

Geen enkel mogelijk koppel wordt dus ooit verbroken: de verkregen koppeling is een Gale-Shapley koppeling.

Wat de complexiteit betreft geldt het volgende:

* maximaal n^2 aanroepen van de procedure partner;
* iedere aanroep van partner heeft complexiteit $\mathcal{O}(1)$, afgezien van andere geneste aanroepen van partner.

De totale complexiteit is dus $\mathcal{O}(n^2)$. □

Opgave 5.17

Een ski-leraar heeft n paar ski's voor n skiërs. Hij wil iedere skiër een paar ski's geven waarvan de lengte zo min mogelijk afwijkt van de lengte van de skiër zelf.
Hoe kan de leraar het beste de ski's verdelen zódat de som van de verschillen (in absolute waarde) tussen de lengte van de ski's en de lengte van de skiërs minimaal is?

Opgave 5.18

Bepaal een Gale-Shapley koppeling voor onderstaande voorkeursmatrices.

$$M = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 3 & 2 \\ 2 & 1 & 3 & 4 \\ 4 & 2 & 3 & 1 \end{pmatrix} \quad \text{en} \quad V = \begin{pmatrix} 3 & 4 & 2 & 1 \\ 3 & 1 & 4 & 2 \\ 2 & 3 & 4 & 1 \\ 3 & 2 & 1 & 4 \end{pmatrix}$$

Opgave 5.19

Veronderstel dat de voorkeursmatrix van de vrouwen net zo is opgesteld als die van de mannen, d.w.z. $v_{ij} = k \Leftrightarrow$ man k komt bij vrouw i op de j-de plaats.
Geef een procedure met complexiteit $O(n^2)$ om deze V-matrix om te zetten in de gewenste vorm, d.w.z. in een matrix V' met $v'_{ij} = k \Leftrightarrow$ man j komt bij vrouw i op de k-de plaats.

Opgave 5.20

Het begrip Gale-Shapley koppeling is een optimaliteitscriterium vanuit het mannen standpunt; we kunnen het ook wel mannen-optimaal noemen.
Analoog heet een stabiele koppeling vrouwen-optimaal als iedere vrouw in iedere andere stabiele koppeling niet beter af is.

a. Stel een algoritme op om een vrouwen-optimale koppeling te bepalen.
b. Pas dit algoritme toe op de voorkeursmatrices uit Opgave 5.18.
c. Toon aan dat als een koppeling zowel mannen- als vrouwen-optimaal is deze koppeling de enige stabiele koppeling is.

Opgave 5.21

Bestaat er voor n = 4 een Gale-Shapley koppeling waarin alle mannen en alle vrouwen aan hun tweede keuze zijn gekoppeld? Verklaar uw antwoord.

Opgave 5.22

Bestaat er een Gale-Shapley koppeling waarin twee mannen beiden hun minst gewenste partner krijgen? Verklaar uw antwoord.

Opgave 5.23

Vier mannen willen een herendubbel (tennis) gaan spelen, d.w.z. dat er twee koppels van elk twee mannen moet woren gevormd.
Iedere man gefft met de cijfers 1,2 en 3 zijn voorkeur voor de drie mogelijke partners aan.
Een indeling heet goed als er geen twee mannen zijn die beiden liever met elkaar dubbelen dan met hun huidige partner.
Bestaat er altijd een goede indeling? Verklaar uw antwoord.