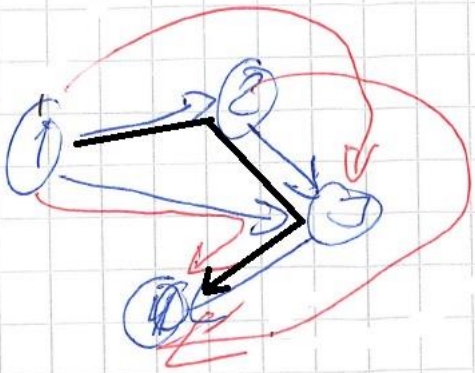
Johdatus verkkoteoriaan luento 3.4.18

Matriisioperaatio suunnatuissa verkoissa

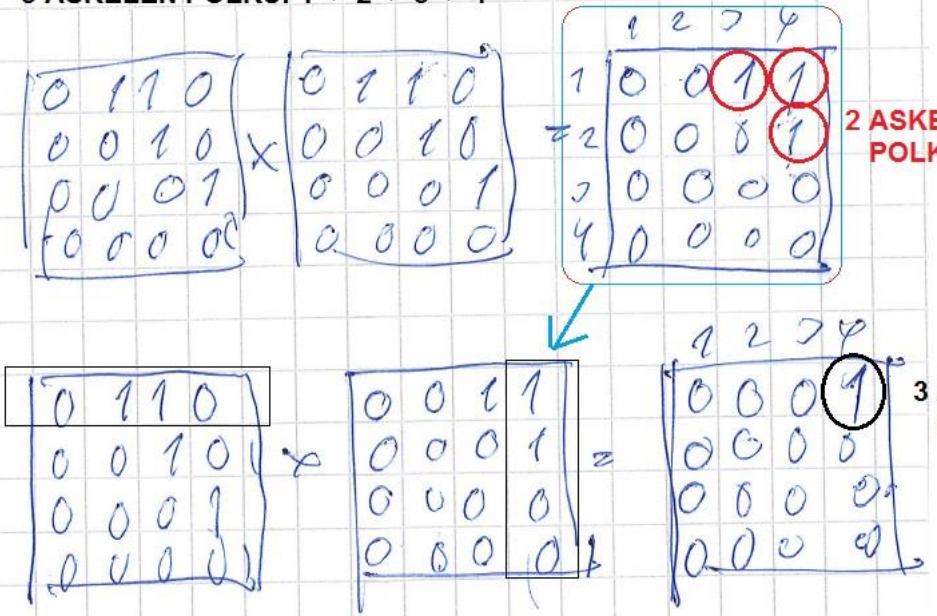
MATRIISIOPERAATIO SUUNNATUSSA VERKOSSA



	1	2	3	4
1	0	1	1	0
2	0	0	1	0
3	0	0	0	1
4	0	0	0	0

2 ASKELEN POLUT: 1 -> 2 -> 3, 1 -> 3-> 4, 2 -> 3-> 4

3 ASKELEN POLKU: 1 -> 2 -> 3 -> 4



LASKETAAN:

$$\begin{bmatrix} 0 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = 0*1 + 1*1 + 1*0 + 0*0 = 1$$

Taustoitusta verkkoteorian ulkopuolelta ennen kuljetusalgoritmia …

LP-ongelma yleisesti

LP = linear programming = lineaarinen ohjelmointi = lineaarinen optimointi. LP-malliin kuuluu maksimoitava tai minimoitava **kohdefunktio eli objektifunktio**, sekä kohdefunktion muuttujia koskevat **rajoite-epäyhtälöt**. Sekä kohdefunktio että rajoitteet ovat lineaarisia, mikä tarkoittaa sitä että esimerkiksi tavallisessa xy-koordinaatistossa ne ovat suoria, kolmiulotteisessa xyz-koordinaatistossa tasopintoja jne...

Rajoitteiden erottamaa aluetta kutsutaan nimellä **käypä alue**. Mikäli käypä alue on olemassa niin se on aina **konveksi**, mikä tarkoittaa sitä että jos kuviteltu kulkija liikkuu käyvän alueen reunalla niin hän “laskeutuu” jatkuvasti alaspäin. Siis jos käypä alue sijaitsee tasolla ja se kierretään myötäpäivään niin missään vaiheessa ei käännytä vasemmalle (ks. kuva). Konveksi alue voidaan määritellä myös siten että jos kahden sen sisälle asetetun pisteen välille piirretään suora jana niin tämä pysyy koko ajan konveksilla alueella.

LP-ongelmalle haettava **optimiratkaisu** löytyy käyvän alueen reunalta, paikasta jossa kohdefunktion maksimi (tai minimi) saavutetaan.

LP-ongelma on **standardimuodossa**:

$\text{Max } z = cx$

$Ax \leq b$

$x \geq 0$

ja **kanonisessa muodossa**:

$\text{Max } z = cx$

$Ax = b$

$x \geq 0$

Simplex

Simplex-menetelmän perusajatuksena on kulkea pitkin käyvän alueen reunaa alkaen yhdestä kulmapisteestä (eli kantaratkaisusta) ja jatkaen sitten johonkin sellaiseen naapurikulmaan, joka on lähempänä optimia. Näin kuljetaan läpi reitti, jonka varrella sivuutetaan osa kantaratkaisuista ennen kuin saavutetaan se kulma, jossa optimaalinen kantaratkaisu sijaitsee.

Menetelmän nimi ei kuvaa sen “yksinkertaisuutta” vaan juontaa juurensa käyvän alueen muodosta. Käypä aluehan on konveksi monikulmio jos muuttujia on kaksi ja monitahokas jos muuttujia on kolme. Mikäli muuttujia on m kpl niin se on m-ulotteinen “monitahokas”, jonka reunalla Simplexissä liikutaan. Tällaista kappaletta kutsutaan matematiikassa simpleksiksi siten, että monikulmio on 2-simpleksi, tavallinen monitahokas 3-simpleksi jne.

Itse Simplex-algoritmi on peräisin vuodelta 1947, jolloin George Dantzig sen kehitti.

Kuljetusongelma

KULJETUSONGELMAssa on joukko tavaran tuottajia (m kpl) ja joukko tavaran hankkijoita (n kpl). Kukin tuottaja voi toimittaa vain tietyn määrän tavaraa (ai, i=1..m) ja kukin hankkija tarvitsee tietyn määrän tavaraa (bj, j=1..n). Kuljetuskustannuksia tuottaja/hankkija-parin välillä merkitään cij. Tarkoituksena on suunnitella toimitetun tavaramäärän (xij) kuljetuksen jakautuminen tuottajilta hankkijoille siten että kustannukset tulevat minimoitua kun samalla huomioidaan annetut rajoitukset. Kuljetusongelmaa voidaan kuvata kaksijakoisella suunnatulla verkolla (koostuu kahdesta solmujoukosta vailla sisäisiä välejä, *kaksijakoinen=bipartite*, *suunnattu verkko=digraph*). Kuljetusongelma voidaan esittää LP-ongelmana allaolevassa muodossa:

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

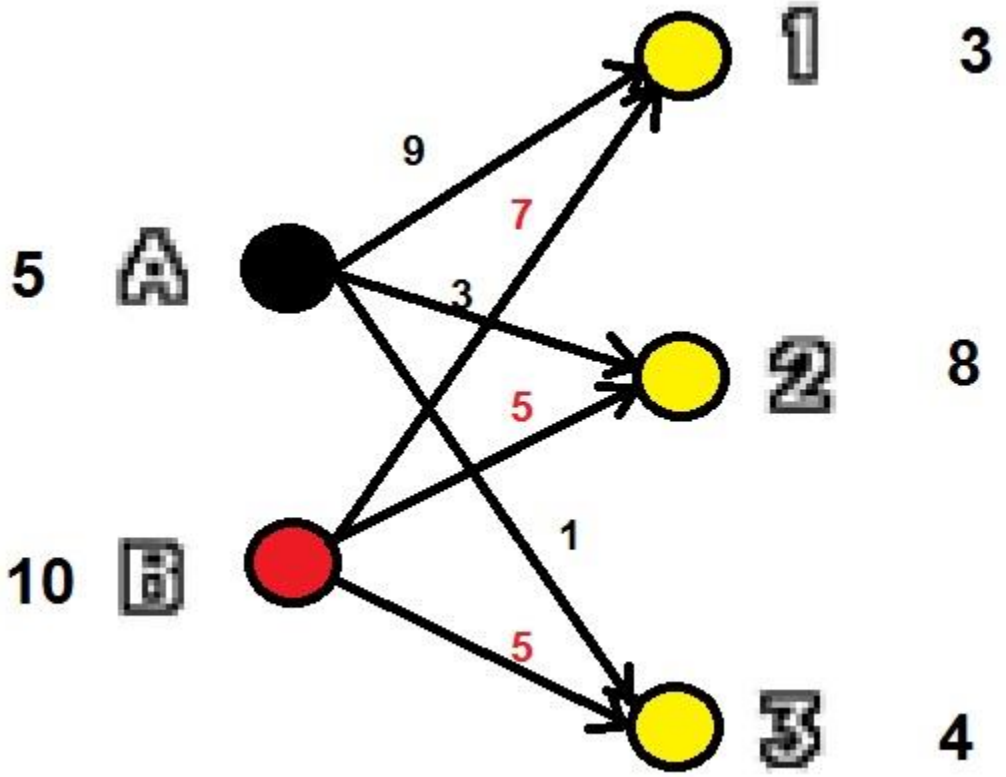
$$\sum_{i=1}^{m} x_{ij} = a_i \quad \text{kun } i = 1 \ldots m$$

$$\sum_{j=1}^{n} x_{ij} = b_j \quad \text{kun } j = 1 \ldots n$$

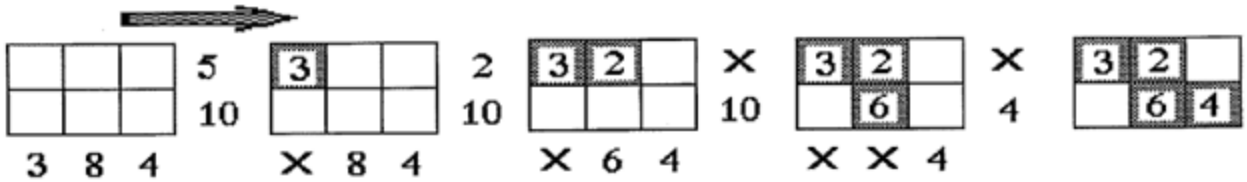
$$x_{ij} \geq 0$$

Kuljetusongelma voidaan ratkaista mm. siten että muodostetaan kustannusmatriisi, josta haetaan jokin alkukantaratkaisu, jossa on aina m+n-1 alkiota. Alkukantaratkaisun pohjalta lähdetään hakemaan optimiratkaisua kuljetusalgoritmin avulla. Samalla tehtävällä voi olla useitakin optimiratkaisuja. Kantaratkaisun hakemiseen voidaan käyttää mm. seuraavia menetelmiä: luoteiskulmamenetelmä, minimikustannusmenetelmä tai Vogelin menetelmä. Näistä ensin mainittu on vaivattomin, mutta antaa yleensä kehnoimmat alkuarvot. Paras tulos saadaan Vogelin menetelmällä, joka on työläin mutta palkitsee vähäisemmillä iteraatioilla itse kuljetusalgoritmissa.

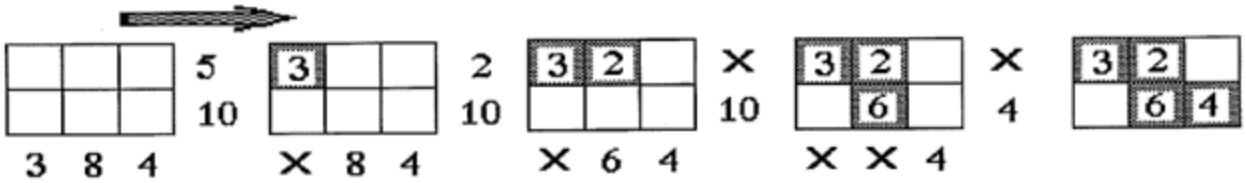
Esimerkki: A tuottaa tavaraa 5 yksikköä ja B 10 yksikköä. Hankkijat 1, 2 ja 3 tarvitsevat k.o. tavaraa 3, 8 ja 4 yksikköä. Kuljetuskustannukset ovat euroina: A1=9, A2=3, A3=1, B1=7, B2=5, B3=5. Haetaan ohessa alkukantaratkaisut luoteiskulmamenetelmää ja Vogelin menetelmää käyttäen.


1
3
9
7
5
A
3
2
8
5
1
10
B
5
3
4

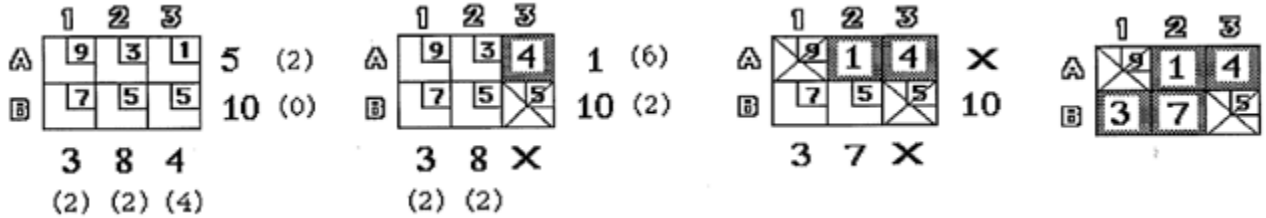
1° Luoteiskulmamenetelmä (lähdetään “luoteiskulmasta” ja sijoitetaan ruutuun suurin mahdollinen tavaramäärä, poistetaan yksi rivi tai sarake):



1° Luoteiskulmamenetelmä (lähdetään “luoteiskulmasta” ja sijoitetaan ruutuun suurin mahdollinen tavaramäärä, poistetaan yksi rivi tai sarake):



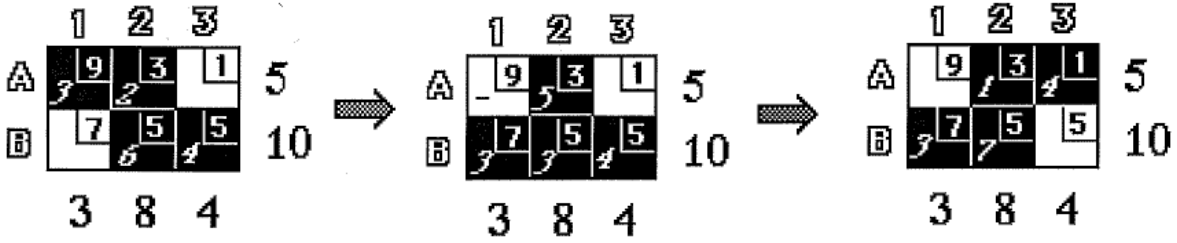
2° Vogelin menetelmä (lasketaan kullekin riville ja sarakkeelle sakkoluku - kahden pienimmän arvon erotus - ja valitaan suurimman sakkoluvun riviltä/sarakkeelta pienimmän kustannuksen ruutu, sijoitetaan ruutuun suurin mahdollinen tavaramäärä):



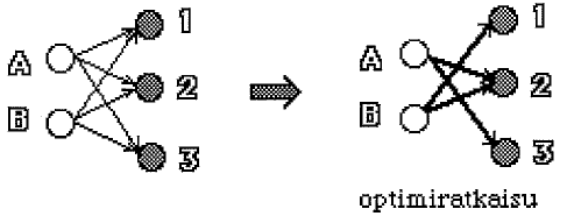
Lopputulokset:
Luoteiskulmamenetelmällä 3*9+2*3+6*5+4*5 = 83
Vogelin menetelmällä 3*7+7*5+1*3+4*1 = 63

Luoteiskulmamenetelmällä saatiin kantaratkaisu, jonka ulkopuolelle jäivät ruudut A3 ja B1. Kuljetusalgoritmissa lasketaan sakot näille kantaratkaisun ulkopuolelle jääneille ruuduille, joita esimerkkitapauksessa on vain kaksi. Sakon arvo saadaan kiertämällä ruutua vastaava polku (ks. sivun alin kuva) kannassa ja vuorotellen summaamalla ja vähentämällä kustannukset. Polku kierretään liikkumalla vuoroin pysty- ja vaakasuoraan, kiertosuunta ei vaikuta tulokseen. Saadaan: S(A3) = A3 - B3 + B2 - A2 = 1 - 5 + 5 - 3 = -2 ja S(B1) = B1 - B2 + A2 - A1 = 7 - 5 + 3 - 9 = -4. Algoritmi pysähtyy jos yksikään lasketuista sakoista ei ole negatiivinen. Mikäli negatiivisia sakkolukuja löytyy niin valitaan kantaan itseisarvoltaan suurimman arvon omaava ruutu; tässä tapauksessa siis B1. Kuormitus uudelle kantaruudulle saadaan kiertämällä sama polku jolta sakko laskettiin ja siirtämällä polulta kyseiseen kantaruutuun suurin mahdollinen kuorma. Esimerkkitapauksessa se on min{3,6} = 3. Kun polku kierretään ympäri niin siirretään kuormia ruudusta toiseen siten että lopputilanteessa on tasapaino jälleen saavutettu eli rivi ja sarakesummat täsmäävät. Siis seuraavasti: B1 ↵ +3 ⇒ 3, A1 ↵ -3 ⇒ 0, A2 ↵ +3 ⇒ 5, B2 ↵ -3 ⇒ 3. Se ruutu, jonka kuormitus nollautuu (A1) poistuu kannasta. Jos useamman ruudun kuorma tulee nollaksi niin kannasta poistetaan ruutu, jolla on korkeimmat kustannukset.

Luoteiskulmamenetelmällä saatiin kantaratkaisu, jonka ulkopuolelle jäivät ruudut A3 ja B1. Kuljetusalgoritmissa lasketaan sakot näille kantaratkaisun ulkopuolelle jääneille ruuduille, joita esimerkkitapauksessa on vain kaksi. Sakon arvo saadaan kiertämällä ruutua vastaava polku (ks. sivun alin kuva) kannassa ja vuorotellen summaamalla ja vähentämällä kustannukset. Polku kierretään liikkumalla vuoroin pysty- ja vaakasuoraan, kiertosuunta ei vaikuta tulokseen. Saadaan: S(A3) = A3 - B3 + B2 - A2 = 1 - 5 + 5 - 3 = -2 ja S(B1) = B1 - B2 + A2 - A1 = 7 - 5 + 3 - 9 = -4. Algoritmi pysähtyy jos yksikään lasketuista sakoista ei ole negatiivinen. Mikäli negatiivisia sakkolukuja löytyy niin valitaan kantaan itseisarvoltaan suurimman arvon omaava ruutu; tässä tapauksessa siis B1. Kuormitus uudelle kantaruudulle saadaan kiertämällä sama polku jolta sakko laskettiin ja siirtämällä polulta kyseiseen kantaruutuun suurin mahdollinen kuorma. Esimerkkitapauksessa se on min{3,6} = 3. Kun polku kierretään ympäri niin siirretään kuormia ruudusta toiseen siten että lopputilanteessa on tasapaino jälleen saavutettu eli rivi ja sarakesummat täsmäävät. Siis seuraavasti: B1 ↵ +3 ⇒ 3, A1 ↵ -3 ⇒ 0, A2 ↵ +3 ⇒ 5, B2 ↵ -3 ⇒ 3. Se ruutu, jonka kuormitus nollautuu (A1) poistuu kannasta. Jos useamman ruudun kuorma tulee nollaksi niin kannasta poistetaan ruutu, jolla on korkeimmat kustannukset.



Lasketaan uuden kantaratkaisun ulkopuolelle jääneiden ruutujen sakot: S(A1) = 9 - 3 + 5 - 7 = 4 ja S(A3) = 1 - 5 + 5 - 3 = -2. Kantaan otetaan siis ruutu A3, johon siirretään kuormaa 4 yksikköä. Siis: A3 ↵ +4 ⇒ 4, B3 ↵ -4 ⇒ 0, B2 ↵ +4 ⇒ 7, A2 ↵ -4 ⇒ 1, joten kannasta poistuu B3. Ulkopuolelle jääneet ruudut ovat nyt A1 ja B3, joiden sakot, S(A1) = 9 - 7 + 5 - 3 = 4 ja S(B3) = 5 - 5 + 3 - 1 = 2 ovat molemmat positiivisia. Näiden algoritmi päättyy ja saatu ratkaisu on optimi. Huomataan, että se on sama kuin Vogelin menetelmällä haettu alkukantaratkaisu. Ratkaisua voidaan kuvata alla olevalla kaksijakoisella suunnatulla verkolla.



Sijoitusongelma

LP – ongelman perusmuodot (vas.), kuljetusongelma ja sijoitusongelma (oik. ylh. ja alh.)
A on yhtälöryhmän kertoimet antava matriisi.

Primaaliongelma:

$\text{Max } z = c^T x$
$Ax = b,$
$x = 0$

Duaaliongelma:

$\text{Min } z = b^T y$
$A^T y = c,$
$y = 0$

$$\text{Min } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\sum_{i=1}^{m} x_{ij} = a_i \quad \text{kun } i = 1 \ldots m$$

$$\sum_{j=1}^{n} x_{ij} = b_j \quad \text{kun } j = 1 \ldots n$$

$$x_{ij} \geq 0$$

$$\text{Min } z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{kun } i = 1 \ldots n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{kun } j = 1 \ldots n$$

$$x_{ij} = 0 \text{ tai } 1$$

SIJOITUSONGELMAssa on kaksi yhtä suurta joukkoa ja kummankin joukon jokaiselle alkiolle on haettava vastinalkio toisesta joukosta. On esimerkiksi jaettava k kpl tehtäviä k:n henkilön kesken s.e. kullekin tulee täsmälleen yksi (sopivin) tehtävä. Sijoitusongelma on *kuljetusongelman erikoistapaus*. Siis sellainen kuljetusongelma, jossa tuottajia ja hankkijoita on yhtä paljon ja jossa kukin tuottaja vie optimiratkaisussa tavaraa ainoastaan täsmälleen yhdelle hankkijalle, joten optimiratkaisussa on alkioita n kpl (kuljetusongelmassahan määrä oli 2n-1 kun m=n). Mikäli tuottajien ja hankkijoiden määrä ei ole sama niin lisätään varjomuuttujia (dummy person), joille kustannuskerroin (cij) on nolla. Sijoitusongelma voidaan esittää LP-ongelmana alla olevassa muodossa (vertaa vastaavaan kuljetusongelman esitykseen):

$$\text{Min } z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \text{kun } i = 1 \ldots n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{kun } j = 1 \ldots n$$

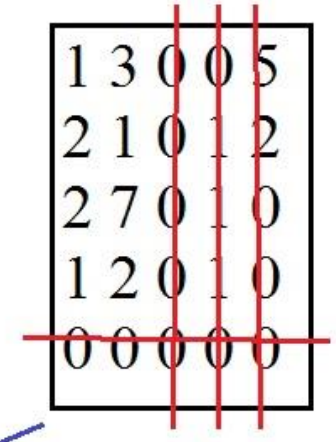
$$x_{ij} = 0 \text{ tai } 1$$

Sijoitusongelma voitaisiin kuljetusongelman erikoistapauksena myös ratkaista sellaisena, mikä kuitenkin on turhan työlästä. On syytä muistaa, että tuolloin saadaan 2n-1 ratkaisua, joista n valitaan (loput n-1 alkiota ovat aina nollia). Sijoitusongelman ratkaisemiseksi on kehitetty ns. unkarilainen menetelmä, joka sisältää seuraavat vaiheet: 1° ongelmasta muodostetun nxn-matriisin kultakin riviltä vähennetään sen riviminimi, 2° sen jälkeen kultakin sarakkeelta vähennetään sarakeminimi, 3° yliviivataan kaikki 0-alkiot siten että viivoja on mahdollisimman vähän (aluksi eniten nollia sisältävät rivit/sarakkeet) 4° jos viivojen lukumäärä on suurempi tai yhtä suuri kuin n niin mennään viimeiseen askeleeseen (6°), 5° etsitään pienin alkio (merk. L), jota ei ole peitetty viivalla ja *vähennetään se kaikista peittämättömien rivien alkioista* ja *lisätään kaikkien peitettyjen sarakkeiden alkioihin* ja palataan askeleeseen 3°, 6° valitaan ratkaisuun tulevat nolla-alkiot siten että kultakin riviltä ja sarakkeelta tulee täsmälleen yksi alkio (huom! ratkaisu ei ole aina yksiselitteinen).

	Vi	Vo	T	U	A
Gina	3	5	2	2	7
Nina	3	2	1	2	3
Tina	4	9	2	5	3
Xena	5	6	4	5	4

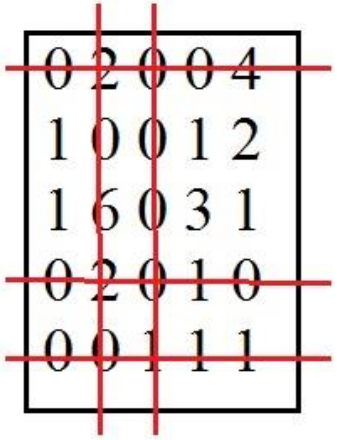
	Vi	Vo	T	U	A	
Gina	3	5	2	2	7	2
Nina	3	2	1	2	3	1
Tina	4	9	2	5	3	2
Xena	5	6	4	5	4	4
dummy	0	0	0	0	0	0

vaiheet 2 ja 3



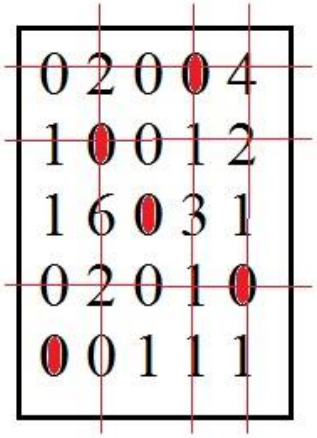
viivoja 4 < 5

vaiheet 5, 3 ja 4



viivoja 5 = 5

vaihe 6

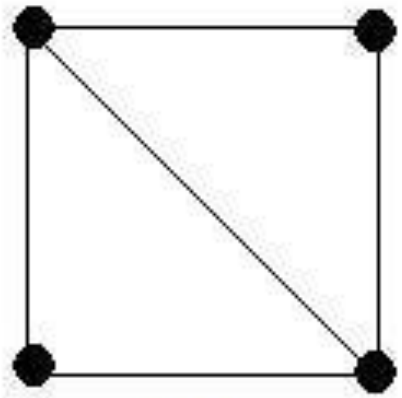


Ratkaisu:

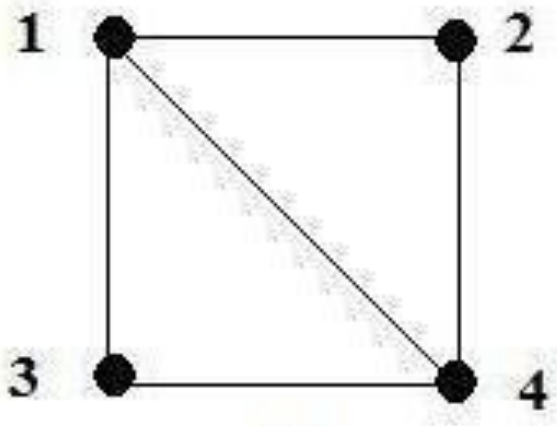
Gina - U
Nina - Vo
Tina - T
Xena - A

Verkon lukuarvosta (extra-asiaa)

G



(a)



(b)

Yhteysmatriisi:

$$\mathbf{A} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

Ylädiagonaalimatriisi 1 1 1

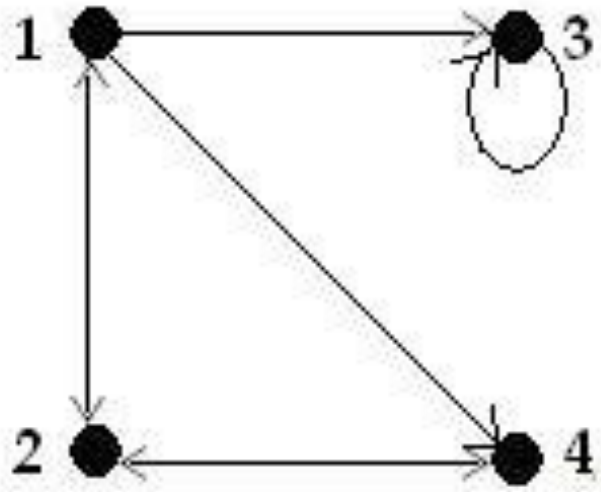
0 1

1

Binäärinumeroksi kirjoitettuna se on111011, eli #H(G) 59 desimaalilukuna.

Suunnatun verkon numero

Suunnattu verkko silmukoilla



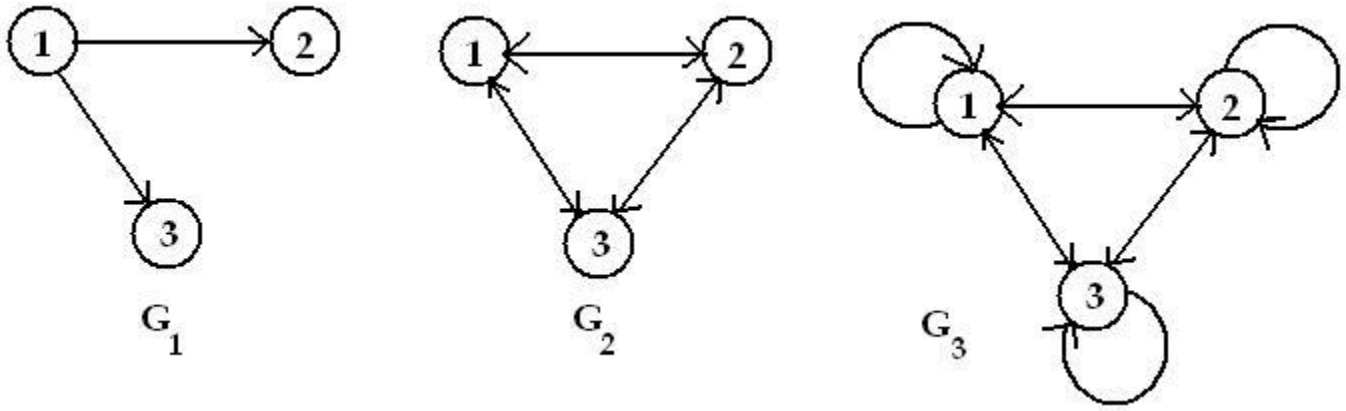
G

	1	2	3	4
1	0	1	1	1
2	1	0	0	1
3	0	0	1	0
4	0	1	0	0

Binäärinumero on 111100100100100

=> Desimaalinumero on 31012 = #K(G).

Esimerkki 1:

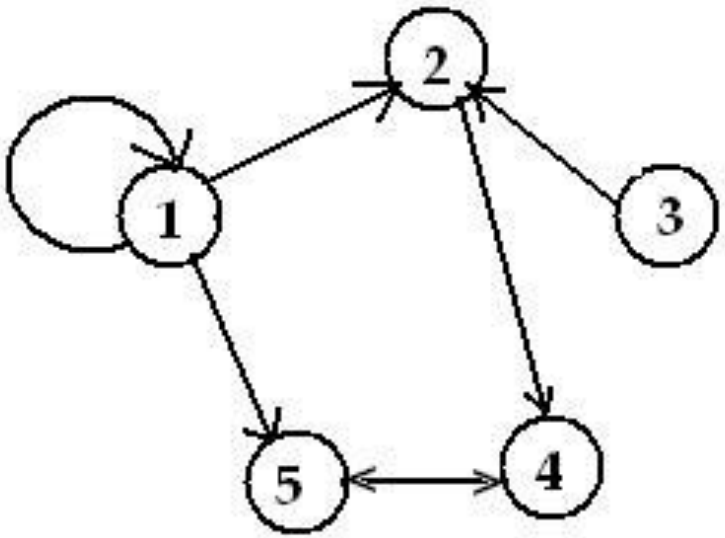


#K(G1) = 11000000 = 192

#K(G2) = 11101110= 238

#K(G3) = 11111111 = 511

Esimerkki 2:



H

	1	2	3	4	5
1	1	1	0	0	1
2	0	0	0	1	0
3	0	1	0	0	0
4	0	0	0	0	1
5	0	0	0	1	0

#(H)= 1100100010010000000100010

= 26288162