

# Kisakoodarin käsikirja

Antti Laaksonen

4. joulukuuta 2016



# Sisältö

<b>Alkusanat</b>	<b>ix</b>
<b>I Perusasiat</b>	<b>1</b>
<b>1 Johdanto</b>	<b>3</b>
1.1 Ohjelmointikielet . . . . .	3
1.2 Syöte ja tuloste . . . . .	4
1.3 Lukujen käsittely . . . . .	6
1.4 Koodin lyhentäminen . . . . .	8
1.5 Virheen etsiminen . . . . .	10
1.6 Matematiikka . . . . .	11
<b>2 Aikavaativuus</b>	<b>15</b>
2.1 Laskusäännöt . . . . .	15
2.2 Vaativuusluokkia . . . . .	18
2.3 Tehokkuuden arviointi . . . . .	19
2.4 Suurin alitaulukko . . . . .	20
<b>3 Järjestäminen</b>	<b>23</b>
3.1 Järjestämisen teoriaa . . . . .	23
3.2 Järjestäminen C++:ssa . . . . .	27
3.3 Binäärihaku . . . . .	29
<b>4 Tietorakenteet</b>	<b>33</b>
4.1 Dynaaminen taulukko . . . . .	33
4.2 Joukkorakenne . . . . .	35
4.3 Hakemisto . . . . .	36
4.4 Iteraattorit ja välit . . . . .	37
4.5 Muita tietorakenteita . . . . .	39
4.6 Vertailu järjestämiseen . . . . .	41
<b>5 Täydellinen haku</b>	<b>43</b>
5.1 Osajoukkojen läpikäynti . . . . .	43
5.2 Permutaatioiden läpikäynti . . . . .	45
5.3 Peruuttava haku . . . . .	46
5.4 Haun optimointi . . . . .	47
5.5 Puolivälihaku . . . . .	49

<b>6</b>	<b>Ahneet algoritmit</b>	<b>51</b>
6.1	Kolikkotehtävä . . . . .	51
6.2	Aikataulutus . . . . .	52
6.3	Tehtävät ja deadlinet . . . . .	54
6.4	Keskiluvut . . . . .	55
6.5	Huffmanin koodaus . . . . .	56
<b>7</b>	<b>Dynaaminen ohjelmointi</b>	<b>59</b>
7.1	Kolikkotehtävä . . . . .	59
7.2	Pisin nouseva alijono . . . . .	64
7.3	Reitinhaku ruudukossa . . . . .	65
7.4	Repunpakkaus . . . . .	66
7.5	Editointietäisyys . . . . .	68
7.6	Laatoitukset . . . . .	69
<b>8</b>	<b>Tasoitettu analyysi</b>	<b>71</b>
8.1	Kaksi osoitinta . . . . .	71
8.2	Lähin pienempi edeltäjä . . . . .	74
8.3	Liukuvan ikkunan minimi . . . . .	75
<b>9</b>	<b>Välikyselyt</b>	<b>77</b>
9.1	Staattiset kyselyt . . . . .	77
9.2	Binääri-indeksipuu . . . . .	80
9.3	Segmenttipuu . . . . .	83
9.4	Lisäteknikoita . . . . .	87
<b>10</b>	<b>Bittien käsittely</b>	<b>89</b>
10.1	Luvun bittiesitys . . . . .	89
10.2	Bittioperaatiot . . . . .	90
10.3	Joukon bittiesitys . . . . .	92
10.4	Dynaaminen ohjelmointi . . . . .	94
<b>II</b>	<b>Verkkoalgoritmit</b>	<b>97</b>
<b>11</b>	<b>Verkkojen perusteet</b>	<b>99</b>
11.1	Käsitteitä . . . . .	99
11.2	Verkko muistissa . . . . .	103
<b>12</b>	<b>Verkon läpikäynti</b>	<b>107</b>
12.1	Syvyyshaku . . . . .	107
12.2	Leveyshaku . . . . .	109
12.3	Sovelluksia . . . . .	111
<b>13</b>	<b>Lyhimmät polut</b>	<b>113</b>
13.1	Bellman-Fordin algoritmi . . . . .	113
13.2	Dijkstran algoritmi . . . . .	116
13.3	Floyd–Warshallin algoritmi . . . . .	119

<b>14 Puiden käsittely</b>	<b>123</b>
14.1 Puun läpikäynti . . . . .	124
14.2 Lämpimittä . . . . .	125
14.3 Solmujen etäisyydet . . . . .	126
14.4 Binaäripuut . . . . .	127
<b>15 Virittävät puut</b>	<b>129</b>
15.1 Kruskalin algoritmi . . . . .	130
15.2 Union-find-rakenne . . . . .	132
15.3 Primin algoritmi . . . . .	134
<b>16 Suunnatut verkot</b>	<b>137</b>
16.1 Topologinen järjestys . . . . .	137
16.2 Dynaaminen ohjelmointi . . . . .	139
16.3 Tehokas eteneminen . . . . .	142
16.4 Syklin tunnistaminen . . . . .	143
<b>17 Vahvasti yhtenäisyys</b>	<b>145</b>
17.1 Kosarajun algoritmi . . . . .	146
17.2 2SAT-ongelma . . . . .	148
<b>18 Puukyselyt</b>	<b>151</b>
18.1 Tehokas nouseminen . . . . .	151
18.2 Solmutaulukko . . . . .	152
18.3 Alin yhteinen esivanhempi . . . . .	155
<b>19 Polut ja kierrokset</b>	<b>159</b>
19.1 Eulerin polku . . . . .	159
19.2 Hamiltonin polku . . . . .	163
19.3 De Bruijnin jono . . . . .	165
19.4 Ratsun kierros . . . . .	166
<b>20 Virtauslaskenta</b>	<b>167</b>
20.1 Ford-Fulkersonin algoritmi . . . . .	168
20.2 Rinnakkaiset polut . . . . .	172
20.3 Maksimiparitus . . . . .	173
20.4 Polkupeitteet . . . . .	176
<b>III Lisäaiheita</b>	<b>179</b>
<b>21 Lukuteoria</b>	<b>181</b>
21.1 Alkuluvut ja tekijät . . . . .	181
21.2 Modulolaskenta . . . . .	185
21.3 Yhtälönratkaisu . . . . .	188
21.4 Muita tuloksia . . . . .	189

<b>22 Kombinatoriikka</b>	<b>193</b>
22.1 Binomikerroin . . . . .	194
22.2 Catalanin luvut . . . . .	196
22.3 Inklusio-eksklusio . . . . .	198
22.4 Burnsiden lemma . . . . .	200
22.5 Cayleyn kaava . . . . .	201
<b>23 Matriisit</b>	<b>203</b>
23.1 Laskutoimitukset . . . . .	203
23.2 Lineaariset rekursioyhtälöt . . . . .	206
23.3 Verkkojen käsittely . . . . .	208
<b>24 Todennäköisyys</b>	<b>211</b>
24.1 Tapahtumat . . . . .	212
24.2 Satunnaismuuttuja . . . . .	214
24.3 Markovin ketju . . . . .	216
24.4 Satunnaisalgoritmit . . . . .	217
<b>25 Peliteoria</b>	<b>221</b>
25.1 Pelin tilat . . . . .	221
25.2 Nim-peli . . . . .	223
25.3 Sprague–Grundyn lause . . . . .	224
<b>26 Merkkijonoalgoritmit</b>	<b>229</b>
26.1 Trie-rakenne . . . . .	230
26.2 Merkkijonohajautus . . . . .	231
26.3 Z-algoritmi . . . . .	234
<b>27 Neliöjuorialgoritmit</b>	<b>239</b>
27.1 Eräkäsittely . . . . .	240
27.2 Tapauskäsittely . . . . .	241
27.3 Mo’n algoritmi . . . . .	241
<b>28 Lisää segmenttipuusta</b>	<b>243</b>
28.1 Laiska eteneminen . . . . .	244
28.2 Dynaaminen toteutus . . . . .	247
28.3 Tietorakenteet . . . . .	249
28.4 Kaksiulotteisuus . . . . .	250
<b>29 Geometria</b>	<b>253</b>
29.1 Kompleksiluvut . . . . .	254
29.2 Pisteet ja suorat . . . . .	256
29.3 Monikulmion pinta-ala . . . . .	259
29.4 Etäisyysmitat . . . . .	260

<b>30 Pyyhkäisyviiva</b>	<b>263</b>
30.1 Janojen leikkauspisteet . . . . .	264
30.2 Lähin pistepari . . . . .	265
30.3 Konvekssi peite . . . . .	266





# **Osa I**

## **Perusasiat**



# **Osa II**

## **Verkkoalgoritmit**



## Luku 20

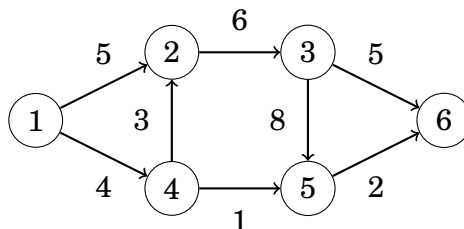
# Virtauslaskenta

Virtauslaskennan keskeiset ongelmat ovat:

- **Maksimivirtauksen etsiminen:** Kuinka paljon virtausta on mahdollista kuljettaa verkon alkusolmusta loppusolmuun kaaria pitkin?
- **Maksimileikkauksen etsiminen:** Mikä on yhteispainoltaan pienin joukko kaaria, joiden poistaminen erottaa alkusolmun loppusolmusta?

Osoittautuu, että nämä ongelmat vastaavat toisiaan ja ne on mahdollista ratkaista samanaikaisesti toistensa avulla.

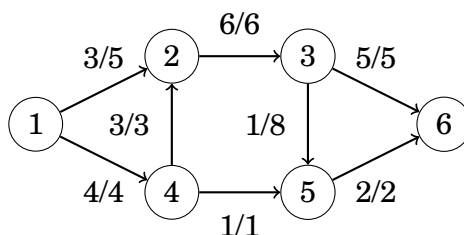
Oletamme, että annettuna on suunnattu, painotettu verkko, jossa on valittu tietty alkusolmu ja loppusolmu. Tarkastellaan esimerkkinä seuraavaa verkkoa, jossa solmu 1 on alkusolmu ja solmu 6 on loppusolmu:



### Maksimivirtaus

Verkossa oleva **virtaus** lähtee liikkeelle alkusolmusta ja päättyy loppusolmuun. Kunkin kaaren paino on kapasiteetti, joka ilmaisee, kuinka paljon virtausta kaaren kautta voi kulkea. Kaikissa solmuissa alkusolmu- ja loppusolmu lukuun ottamatta tulevan ja lähtevän virtauksen on oltava yhtä suuri.

**Maksimivirtaus** on suurin mahdollinen virtaus verkossa. Esimerkkiverkossa maksimivirtauksen suuruus on 7:



Merkintä  $v/k$  kaareissa tarkoittaa, että kaareissa kulkee virtausta  $v$  ja kaaren kapasiteetti on  $k$ . Virtauksen suuruus on 7, koska alkusolmusta lähtevä virtaus on  $3 + 4 = 7$  ja loppusolmuun saapuva virtaus on  $5 + 2 = 7$ .

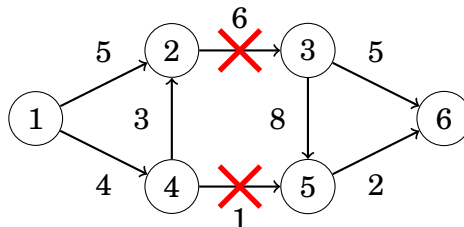
Huomaa, että jokaisessa välisolmussa tulevan ja lähtevän virtauksen määrä on sama. Esimerkiksi solmuun 2 tulee virtausta  $3 + 3 = 6$  yksikköä solmuista 1 ja 4 ja siitä lähtee virtausta 6 yksikköä solmuun 3.

Virtaus 7 on verkon maksimivirtaus, koska verkon rakenteesta johtuen ei ole tapaa kuljettaa enempää virtausta verkossa.

## Minimileikkaus

**Leikkaus** jakaa verkon solmut kahteen osaan niin, että alkusolmu ja loppusolmu ovat eri osissa. Leikkauksen paino on niiden kaarten yhteispaino, jotka kulkevat alkuosasta loppuosaan.

**Minimileikkaus** on leikkaus, jonka paino on pienin mahdollinen. Esimerkiverkossa minimileikkaus on painoltaan 7:



Tässä leikkauksessa alkuosassa ovat solmut  $\{1, 2, 4\}$  ja loppuosassa ovat solmut  $\{3, 5, 6\}$ . Alkuosasta loppuosaan kulkevat kaaret  $2 \rightarrow 3$  ja  $4 \rightarrow 5$ , joiden yhteispaino on  $6 + 1 = 7$ .

Ei ole sattumaa, että yllä olevassa verkossa sekä maksimivirtauksen suuruus että minimileikkauksen paino on 7. Virtauslaskennan keskeinen tulos on, että verkon maksimivirtaus ja minimileikkaus ovat *aina* yhtä suuret, eli käsitteet kuvaavat saman asian kahta eri puolta.

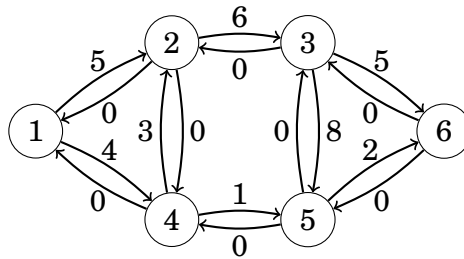
Seuraavaksi tutustumme Ford-Fulkersonin algoritmiin, jolla voi etsiä verkon maksimivirtauksen ja minimileikkauksen. Algoritmi auttaa myös ymmärtämään, *miksi* maksimivirtaus ja minimileikkaus ovat yhtä suuret.

## 20.1 Ford-Fulkersonin algoritmi

**Ford-Fulkersonin algoritmi** etsii verkon maksimivirtauksen. Algoritmin idea on aloittaa tilanteesta, jossa virtaus on 0, ja etsiä sitten verkosta polkuja, jotka tuottavat siihen lisää virtausta. Kun mitään polkua ei enää pysty muodostamaan, maksimivirtaus on valmis.

Algoritmi käsittelee verkkoa muodossa, jossa jokaiselle kaarelle on vastakkaiseen suuntaan kulkeva pari. Kaaren paino kuvastaa, miten paljon lisää virtausta sen kautta pystyy vielä kulkemaan. Aluksi alkuperäisen verkon kaarilla on painona niiden kapasiteetti ja käänteisillä kaarilla on painona 0.

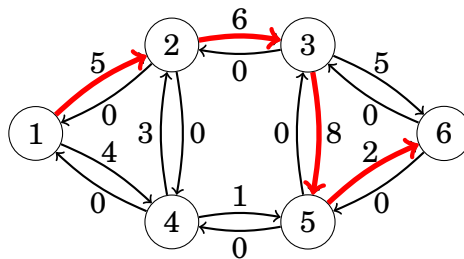
Esimerkkiverkosta syntyy seuraava verkko:



### Algoritmin toiminta

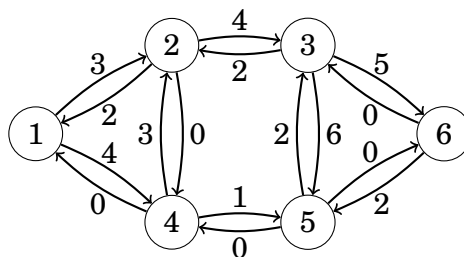
Ford-Fulkersonin algoritmi etsii verkosta joka vaiheessa polun, joka alkaa alkusolmusta, päättyy loppusolmuun ja jossa jokaisen kaaren paino on positiivinen. Jos vaihtoehtoja on useita, mikä tahansa valinta kelpaa.

Esimerkkiverkossa voimme valita vaikkapa seuraavan polun:



Polun valinnan jälkeen virtaus lisääntyy  $x$  yksikköä, jossa  $x$  on pienin kaaren kapasiteetti polulla. Samalla jokaisen polulla olevan kaaren kapasiteetti vähenee  $x$ :llä ja jokaisen käänteisen kaaren kapasiteetti kasvaa  $x$ :llä.

Yllä valitussa polussa kaarten kapasiteetit ovat 5, 6, 8 ja 2. Pienin kapasiteetti on 2, joten virtaus kasvaa 2:lla ja verkko muuttuu seuraavasti:



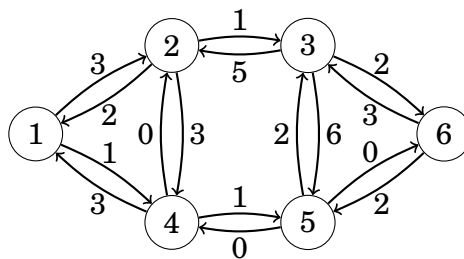
Muutoksessa on ideana, että virtauksen lisääminen vähentää polkuun kuuluvien kaarten kykyä välittää virtausta. Toisaalta virtausta on mahdollista peruuttaa myöhemmin käyttämällä käänteisiä kaaria, jos osoittautuu, että virtausta on järkevää reitittää verkossa toisella tavalla.

Algoritmi kasvattaa virtausta niin kauan, kuin verkossa on olemassa polku alkusolmusta loppusolmuun positiivisia kaaria pitkin. Tässä tapauksessa voimme valita seuraavan polun vaikkapa näin:

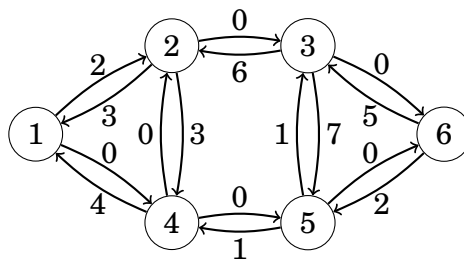


Tämän polun pienin kapasiteetti on 3, joten polku kasvattaa virtausta 3:lla ja kokonaisvirtaus polun käsittelyn jälkeen on 5.

Nyt verkko muuttuu seuraavasti:



Maksimivirtaus tulee valmiiksi lisäämällä virtausta vielä polkujen  $1 \rightarrow 2 \rightarrow 3 \rightarrow 6$  ja  $1 \rightarrow 4 \rightarrow 5 \rightarrow 3 \rightarrow 6$  avulla. Molemmat polut tuottavat 1 yksikön lisää virtausta, ja lopullinen verkko on seuraava:



Nyt virtausta ei pysty enää kasvattamaan, koska verkossa ei ole mitään polkua alkusolmusta loppusolmuun, jossa jokaisen kaaren paino olisi positiivinen. Niinpä algoritmi pysähtyy ja verkon maksimivirtaus on 7.

### Polun valinta

Ford-Fulkersonin algoritmi ei ota kantaa siihen, millä tavoin virtausta kasvattava polku valitaan verkossa. Valintatavasta riippumatta algoritmi pysähtyy ja tuottaa maksimivirtauksen ennemmin tai myöhemmin, mutta polun valinnalla on vaikutusta algoritmin tehokkuuteen.

Yksinkertainen tapa on valita virtausta kasvattava polku syvyysshaulla. Tämä toimii usein hyvin, mutta pahin tapaus on, että jokainen polku kasvattaa virtausta vain 1:llä ja algoritmi toimii hitaasti. Seuraavaksi käymme läpi kaksi tapaa polun valintaan, jotka estävät tämän ilmiön.



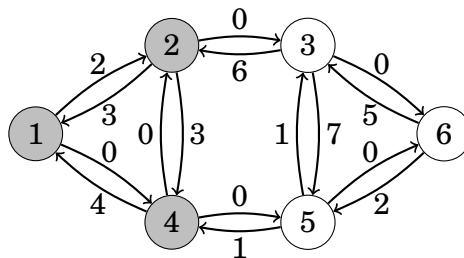
**Edmonds-Karpin algoritmi** on Ford-Fulkersonin algoritmin toteutus, jossa virtausta kasvattava polku valitaan aina niin, että siinä on mahdollisimman vähän kaaria. Tämä onnistuu etsimällä polku syvyysshaun sijasta leveyshaulla. Osoittautuu, että tämä varmistaa virtauksen kasvamisen nopeasti ja maksimivirtauksen etsiminen vie aikaa  $O(m^2n)$ .

**Skaalaava algoritmi** asettaa minimiarvon, joka on ensin alkusolmusta lähtevien kaarten kapasiteettien summa  $c$ . Joka vaiheessa verkosta etsitään syvyysshaulla polku, jonka jokaisen kaaren kapasiteetti on vähintään minimiarvo. Aina jos kelpollista polkua ei löydy, minimiarvo jaetaan 2:lla, kunnes lopuksi minimiarvo on 1. Algoritmin aikavaativuus on  $O(m^2 \log c)$ .

Käytännössä skaalaava algoritmi on mukavampi koodattava, koska siinä riittää etsiä polku syvyysshaulla. Molemmat algoritmit ovat yleensä aina riittävän nopeita ohjelmointikisoissa esiintyviin tehtäviin.

## Minimileikkaus

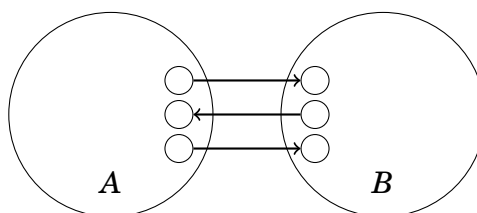
Osoittautuu, että kun Ford-Fulkersonin algoritmi on saanut valmiiksi maksimivirtauksen, se on tuottanut samalla minimileikkauksen. Olkoon  $A$  niiden solmujen joukko, joihin verkossa pääsee alkusolmusta positiivisia kaaria pitkin. Esimerkkiverkossa  $A$  sisältää solmut 1, 2 ja 4:



Nyt minimileikkauksen muodostavat ne alkuperäisen verkon kaaret, jotka kulkevat joukosta  $A$  joukon  $A$  ulkopuolelle ja joiden kapasiteetti on täysin käytetty maksimivirtauksessa. Tässä verkossa kyseiset kaaret ovat  $2 \rightarrow 3$  ja  $4 \rightarrow 5$ , jotka tuottavat minimileikkauksen  $6 + 1 = 7$ .

Miksi sitten algoritmin tuottama virtaus ja leikkaus ovat varmasti maksimivirtaus ja minimileikkaus? Syynä tähän on, että virtauksen suuruus on *aina* enintään yhtä suuri kuin leikkauksen paino. Niinpä kun virtaus ja leikkaus ovat yhtä suuret, ne ovat varmasti maksimivirtaus ja minimileikkaus.

Tarkastellaan mitä tahansa verkon leikkausta, jossa alkusolmu kuuluu osaan  $A$ , loppusolmu kuuluu osaan  $B$  ja osien välillä kulkee kaaria:



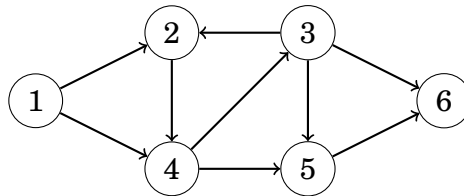
Leikkauksen paino on niiden kaarten painojen summa, jotka kulkevat osasta  $A$  osaan  $B$ . Tämä on yläraja sille, kuinka suuri verkossa oleva virtaus voi olla, koska virtauksen täytyy edetä osasta  $A$  osaan  $B$ . Niinpä maksimivirtaus on pienempi tai yhtä suuri kuin mikä tahansa verkon leikkaus.

Toisaalta Ford-Fulkersonin algoritmi tuottaa virtauksen, joka on tarkalleen yhtä suuri kuin verkossa oleva leikkaus. Niinpä tämän virtauksen on oltava maksimivirtaus ja vastaavasti leikkauksen on oltava minimileikkaus.

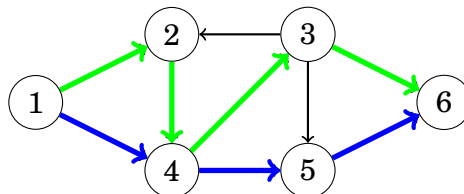
## 20.2 Rinnakkaiset polut

Ensimmäisenä virtauslaskennan sovelluksena tarkastelemme tehtävää, jossa tavoitteena on muodostaa mahdollisimman monta rinnakkaista polkua verkon alkusolmusta loppusolmuun. Vaatimuksena on, että jokainen verkon kaari esiintyy enintään yhdellä polulla.

Esimerkiksi verkossa



pystyy muodostamaan kaksi rinnakkaista polkua solmusta 1 solmuun 6. Tämä toteutuu valitsemalla polut  $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 6$  ja  $1 \rightarrow 4 \rightarrow 5 \rightarrow 6$ :



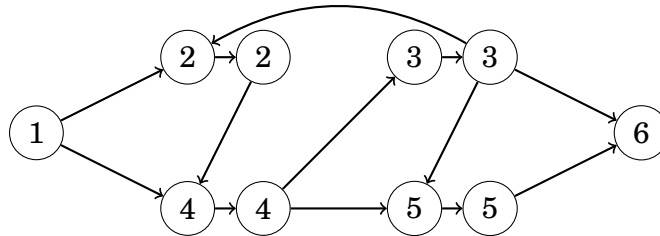
Osoittautuu, että suurin rinnakkaisten polkujen määrä on yhtä suuri kuin maksimivirtaus verkossa, jossa jokaisen kaaren kapasiteetti on 1. Kun maksimivirtaus on muodostettu, rinnakkaiset polut voi löytää ahneesti etsimällä alkusolmusta loppusolmuun kulkevia polkuja.

Tarkastellaan sitten tehtävän muunnelmää, jossa jokainen solmu (alku- ja loppusolmuja lukuun ottamatta) saa esiintyä enintään yhdellä polulla. Tämän rajoituksen seurauksena äskeisessä verkossa voi muodostaa vain yhden polun, koska solmu 4 ei voi esiintyä monella polulla:

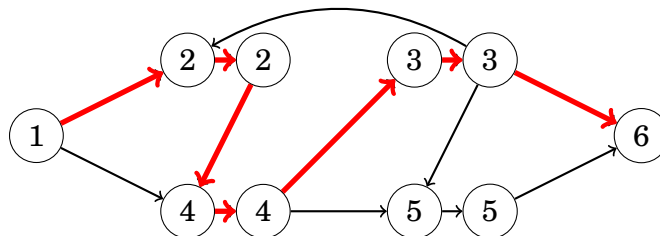


Tavallinen keino rajoittaa solmun kautta kulkevaa virtausta on jakaa solmu tulosolmuksi ja lähtösolmuksi. Kaikki solmuun tulevat kaaret saapuvat tulosolmuun ja kaikki solmusta lähtevät kaaret poistuvat lähtösolmusta. Lisäksi tulosolmusta lähtösolmuun on kaari, jossa on haluttu kapasiteetti.

Tässä tapauksessa verkosta tulee seuraava:



Tämän verkon maksimivirtaus on:



Tämä tarkoittaa, että verkossa on mahdollista muodostaa vain yksi polku alkusolmusta lähtösolmuun, kun sama solmu ei saa esiintyä monessa polussa.

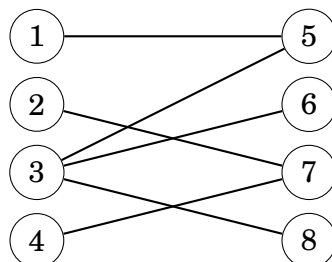
## 20.3 Maksimiparitus

Verkossa oleva **paritus** on kokoelma kaaria, jotka on valittu niin, että jokainen verkon solmu esiintyy enintään yhden paritukseen kuuluvan kaaren päätesolmuna. **Maksimiparitus** on puolestaan paritus, jossa parien määrä on mahdollisimman suuri.

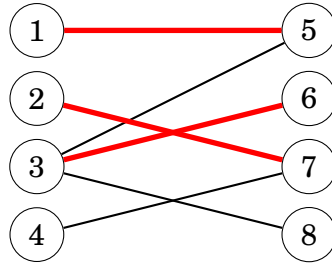
Maksimiparituksen etsimiseen yleisessä verkossa on olemassa polynominen algoritmi, mutta se on hyvin monimutkainen. Tässä luvussa keskitymmekin tilanteeseen, jossa verkko on kaksijakoinen. Tällöin maksimiparituksen pystyy etsimään helposti virtauslaskennan avulla.

### Maksimiparituksen etsiminen

Kaksijakoinen verkko voidaan esittää aina niin, että kukin verkon solmu on vasemmalla tai oikealle puolella ja kaikki verkon kaaret kulkevat puolten välillä. Tarkastellaan esimerkkinä seuraavaa verkkoa:

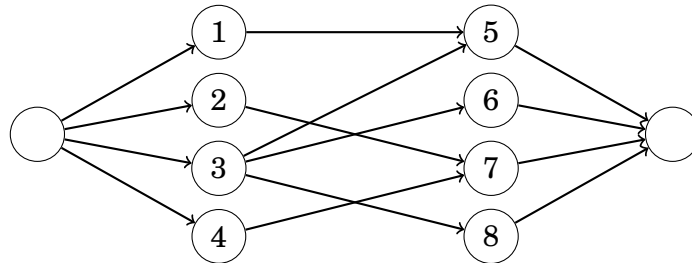


Tässä verkossa maksimiparituksen koko on 3:



Kaksijakoisen verkon maksimiparitus vastaa aina maksimivirtausta verkossa, johon on lisätty alkusolmu ja loppusolmu. Alkusolmusta on kaari jokaiseen vasemman puolen solmuun, ja vastaavasti loppusolmuun on kaari jokaisesta oikean puolen solmusta. Jokaisen kaaren kapasiteettina on 1.

Esimerkissä tuloksena on seuraava verkko:



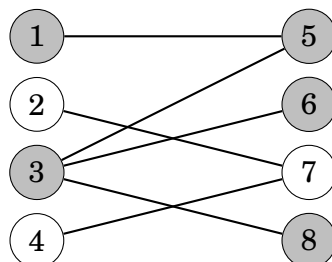
Tämän verkon maksimivirtaus on yhtä suuri kuin alkuperäisen verkon maksimiparitus, koska virtaus muodostuu joukosta polkuja alkusolmusta loppusolmuun ja jokainen polku ottaa mukaan uuden kaaren paritukseen. Tässä tapauksessa maksimivirtaus on 3, joten maksimiparitus on myös 3.

## Hallin lause

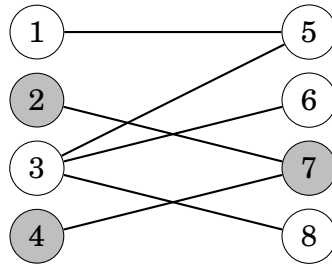
**Hallin lause** antaa ehdon, milloin kaksijakoiseen verkkoon voidaan muodostaa paritus, joka sisältää kaikki toisen puolen solmut. Jos kummallakin puolella on yhtä monta solmua, Hallin lause kertoo, voidaanko muodostaa **täydellinen paritus**, jossa kaikki solmut paritetaan keskenään.

Oletetaan, että haluamme muodostaa parituksen, johon kuuluvat kaikki vasemman puolen solmut. Olkoon  $X$  jokin joukko vasemman puolen solmuja ja joukko  $f(X)$  ne oikean puolen solmut, jotka ovat yhteydessä joukon  $X$  solmuihin. Hallin lauseen mukaan paritus on mahdollinen tarkalleen silloin, kun mille tahansa joukolle  $X$  pätee  $|X| \leq |f(X)|$ .

Tarkastellaan Hallin lauseen merkitystä esimerkkiverkossa. Valitaan ensin  $X = \{1, 3\}$ , jolloin  $f(X) = \{5, 6, 8\}$ :



Tämä täyttää Hallin lauseen ehdon, koska  $|X| = 2$  ja  $|f(X)| = 3$ . Valitaan sitten  $X = \{2, 4\}$ , jolloin  $f(X) = \{7\}$ :



Tässä tapauksessa  $|X| = 2$  ja  $|f(X)| = 1$ , joten Hallin lauseen ehto ei pidä paikkaansa. Tämä tarkoittaa, että ei ole mahdollista muodostaa paritusta, jossa ovat mukana kaikki vasemman puolen solmut (eli täydellistä paritusta). Tämä on odotettu tulos, koska verkon maksimiparitus on 3 eikä 4.

Jos Hallin lauseen ehto ei päde, osajoukko  $X$  kertoo syyn sille, miksi paritusta ei voi muodostaa. Koska  $X$  sisältää enemmän solmuja kuin  $f(X)$ , kaikille  $X$ :n solmuille ei riitä paria oikealta. Esimerkiksi yllä molemmat solmut 2 ja 4 tulisi yhdistää solmuun 7, mutta tämä ei ole mahdollista.

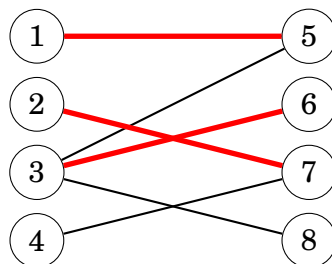
## Königin lause

**Königin lause** tuo yhteyden maksimiparituksen ja solmupeitteen välille. **Solmupeite** on sellainen joukko verkon solmuja, että jokaisen verkon kaaresta ainakin toinen kaaren päätesolmuista kuuluu joukkoon. **Riippumaton joukko** on taas joukko verkon solmuja, jossa minkään solmuparin välillä ei ole kaarta.

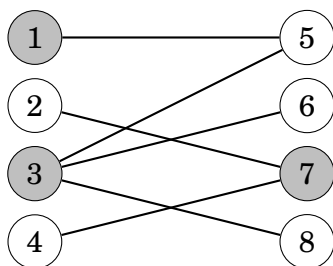
Jokaista verkon solmupeitettä vastaa riippumaton joukko, joka muodostuu niistä solmuista, jotka eivät kuulu solmupeitteeseen. Niinpä jos verkossa on solmupeite, jossa on  $x$  solmua, niin siinä on myös riippumaton joukko, jossa on  $n - x$  solmua. Vastaava riippuvuus pätee myös toiseen suuntaan.

Yleisessä verkossa pienimmän solmupeitteen ja suurimman riippumattoman joukon etsiminen ovat NP-vaikeita ongelmia. Kuitenkin kaksijakoisessa verkossa ongelmat ratkeavat tehokkaasti, koska Königin lauseen nojalla pienin solmupeite on yhtä suurin kuin maksimiparitus.

Esimerkiksi seuraavan verkon maksimiparituksen koko on 3:

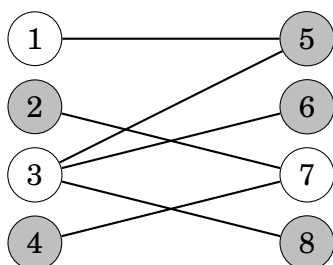


Niinpä myös pienimmän solmupeitteen koko on 3. Solmupeite voidaan muodostaa valitsemalla siihen solmut  $\{1, 3, 7\}$ :



Pienin solmupeite muodostuu aina niin, että jokaisesta maksimiparituksen kaaresta toinen kaaren päätesolmuista kuuluu peitteeseen.

Pienimmästä solmupeitteestä saadaan suurin riippumaton joukko valitsemalla kaikki solmut, jotka eivät kuulu peitteeseen. Esimerkiksi yllä olevassa verkossa suurin riippumaton joukko on seuraava:



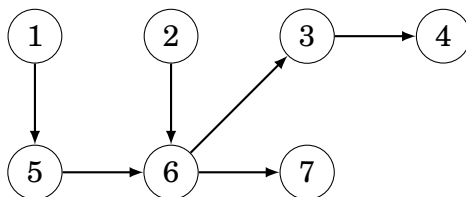
## 20.4 Polkupeitteet

**Polkupeite** on joukko verkon polkuja, joka on muodostettu niin, että jokainen verkon solmu kuuluu ainakin yhteen polkuun. Seuraavaksi näemme, miten virtauslaskennan avulla voi etsiä pienimmän polkupeitteen suunnatussa, syklittömässä verkossa.

Polkupeitteestä on kaksi muunnelmää: **Solmuerillinen peite** on polkupeite, jossa jokainen verkon solmu esiintyy tasan yhdessä polussa. **Yleinen peite** taas on polkupeite, jossa sama solmu voi esiintyä useammassa polussa. Kummassakin tapauksessa pienin polkupeite löytyy samanlaisella idealla.

### Solmuerillinen peite

Tarkastellaan esimerkkinä seuraavaa verkkoa:



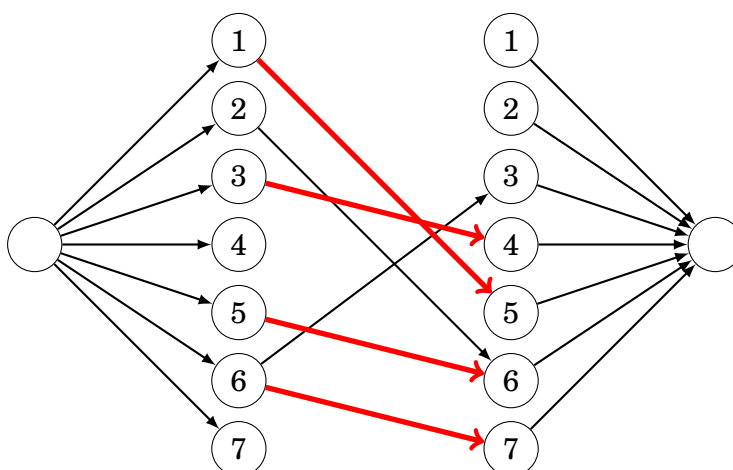
Tässä tapauksessa pienin solmuerillinen polkupeite muodostuu kolmesta polusta. Voimme valita polut esimerkiksi seuraavasti:



Huomaa, että yksi poluista sisältää vain solmun 2, eli on sallittua, että polussa ei ole kaaria.

Polkupeitteen etsiminen voidaan tulkita paritusongelmana verkossa, jossa jokaista alkuperäisen verkon solmua vastaa kaksi solmua: vasen ja oikea solmu. Vasemmasta solmusta oikeaan solmuun on kaari, jos tällainen kaari esiintyy alkuperäisessä verkossa. Ideana on, että paritus määrittää, mitkä solmut ovat yhteydessä toisiinsa poluissa.

Esimerkkiverkossa tilanne on seuraava:

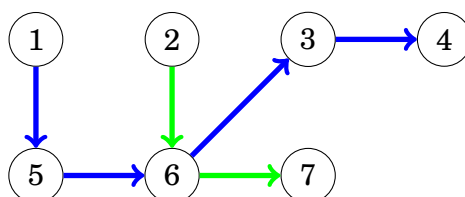


Tässä tapauksessa maksimiparitukseen kuuluu neljä kaarta, jotka vastaavat alkuperäisen verkon kaaria  $1 \rightarrow 5$ ,  $3 \rightarrow 4$ ,  $5 \rightarrow 6$  ja  $6 \rightarrow 7$ . Niinpä pienin solmuerillinen polkupeite syntyy muodostamalla polut kyseisten kaarten avulla.

Pienimmän polkupeitteen koko on  $n - c$ , jossa  $n$  on verkon solmujen määrä ja  $c$  on maksimiparituksen kaarten määrä. Esimerkiksi yllä olevassa verkossa pienimmän polkupeitteen koko on  $7 - 4 = 3$ .

## Yleinen peite

Yleisessä polkupeitteessä sama solmu voi kuulua moneen polkuun, minkä ansiosta tarvittava polkujen määrä saattaa olla pienempi. Esimerkkiverkossa pienin yleinen polkupeite muodostuu kahdesta polusta seuraavasti:



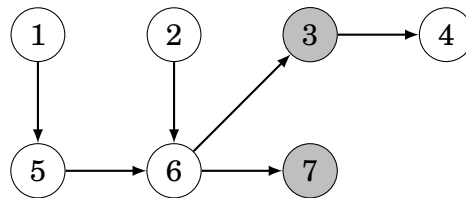
Tässä verkossa yleisessä polkupeitteessä on 2 polkua, kun taas solmuerillisessä polkupeitteessä on 3 polkua. Erona on, että yleisessä polkupeitteessä solmua 6 käytetään kahdessa polussa.

Yleisen polkupeitteen voi löytää lähes samalla tavalla kuin solmuerillisen polkupeitteen. Riittää täydentää maksimiparituksen verkkoa niin, että siinä on kaari  $a \rightarrow b$  aina silloin, kun alkuperäisessä verkossa solmusta  $a$  pääsee solmuun  $b$  (mahdollisesti usean kaaren kautta).

### Dilworthin lause

**Dilworthin lauseen** mukaan suunnatun, syklittömän verkon pienin yleinen polkupeite on yhtä suuri kuin suurin mahdollinen kokoelma solmuja, jossa minäkään kahden solmun välillä ei ole polkua.

Esimerkiksi äskeisessä verkossa pienin yleinen polkupeite sisältää kaksi polkua. Niinpä verkosta voidaan valita enintään kaksi solmua niin, että minäkään solmujen välillä ei ole polkua. Vaihtoehtoja valintaan on monia: voimme valita esimerkiksi solmut 3 ja 7:



Verkossa ei ole polkua solmusta 3 solmuun 7 eikä polkua solmusta 7 solmuun 3, joten valinta on kelvollinen. Toisaalta jos verkosta valitaan mitkä tahansa kolme solmua, jostain solmusta toiseen on polku.



# **Osa III**

## **Lisäaiheita**



# Hakemisto

- #define, 8
- bitset, 39
- complex, 254
- deque, 39
- map, 36
- multiset, 35
- next\_permutation, 45
- priority\_queue, 40
- queue, 40
- random\_shuffle, 37
- reverse, 37
- set, 35
- sort, 27, 37
- stack, 40
- string, 34
- typedef, 8
- unordered\_map, 36
- unordered\_multiset, 35
- unordered\_set, 35
- vector, 33
- 2SAT-ongelma, 148
- aakkosto, 229
- ahne algoritmi, 51
- aikavaativuus, 15
- alijono, 229
- alin yhteinen esivanhempi, 155
- alkuluku, 181
- alkulukupari, 183
- alkuosa, 229
- alkutekijähajotelma, 181
- Andrew'n algoritmi, 267
- aritmeettinen summa, 13
- aste, 101
- Bellman-Fordin algoritmi, 113
- binääri-indeksipuu, 80
- binäärihaku, 29
- binäärikoodi, 56
- binääripuu, 128
- binomijakauma, 215
- binomikerroin, 194
- bittiesitys, 89
- bittijoukko, 39
- Burnsiden lemma, 200
- Catalanin luku, 196
- Cayleyn kaava, 201
- de Bruijnin jono, 165
- determinantti, 205
- Dijkstran algoritmi, 116, 141
- Dilworthin lause, 178
- Diofantoksen yhtälö, 188
- Diracin lause, 164
- dynaaminen ohjelmointi, 59
- dynaaminen segmenttipuu, 247
- editointietäisyys, 68
- Edmonds-Karpin algoritmi, 170
- ehdollinen todennäköisyys, 213
- Eratostheneen seula, 184
- esijärjestys, 128
- etäisyysmitta, 260
- Eukleideen algoritmi, 184, 188
- Eukleideen kaava, 190
- Euklidinen etäisyys, 260
- Eulerin kierros, 160
- Eulerin lause, 186
- Eulerin polku, 159
- Eulerin totienttifunktio, 185
- Fermat'n pieni lause, 186
- Fibonaccin luku, 14, 190, 206
- Floyd-Warshallin algoritmi, 119
- Floydin algoritmi, 144
- Ford-Fulkersonin algoritmi, 168
- funktionaalinen verkko, 142
- geometria, 253
- geometrinen jakauma, 216

geometrinen summa, 13  
 Goldbachin konjektuuri, 183  
 Grundy-luku, 225  
 Grundyn peli, 227  
  
 häviötila, 221  
 hajautus, 231  
 hajautusarvo, 231  
 hakemisto, 36  
 Hallin lause, 174  
 Hamiltonin kierros, 164  
 Hamiltonin polku, 163  
 harmoninen summa, 14, 184  
 harva segmenttipuu, 247  
 Heronin kaava, 253  
 heuristiikka, 166  
 Huffmanin koodaus, 56  
  
 indeksien pakkaus, 87  
 inklusio-eksklusio, 198  
 inversio, 25  
 iteraattori, 37  
  
 jälkijärjestys, 128  
 järjestäminen, 23  
 jakaja, 181  
 jakauma, 215  
 jakso, 229  
 jaollisuus, 181  
 jono, 40  
 joukko, 11, 35  
  
 käänteismatriisi, 206  
 Königin lause, 175  
 kaari, 99  
 kaarilista, 105  
 kaksi osoitinta, 71  
 kaksijakoisuus, 102, 112  
 kaksiulotteinen segmenttipuu, 250  
 keko, 40  
 kekojärjestäminen, 26  
 kierto, 229  
 kiinalainen jäännöslause, 189  
 Kirchhoffin lause, 209  
 kofaktori, 205  
 kokonaisluku, 6  
 kombinatoriikka, 193  
 kompleksiluku, 254  
  
 konvekksi peite, 266  
 koodi, 56  
 koodisana, 56  
 Kosarajun algoritmi, 146  
 Kruskalin algoritmi, 130  
 kuplajärjestäminen, 24  
  
 lähin pienempi edeltäjä, 74  
 lähin pistepari, 265  
 läpimitta, 125  
 Lagrangen lause, 189  
 laiska eteneminen, 244  
 laiska segmenttipuu, 244  
 Las Vegas -algoritmi, 217  
 laskemisjärjestäminen, 27  
 Legendren konjektuuri, 183  
 leikkaus, 168  
 leikkauspiste, 257, 264  
 leksikografinen järjestys, 230  
 leveyshaku, 109  
 liukuluku, 7  
 liukuvan ikkunan minimi, 75  
 logaritmi, 12  
 logiikka, 11  
 lomitussjärjestäminen, 25  
 loppuosa, 229  
 lukuteoria, 181  
 lyhin polku, 113  
  
 makro, 8  
 maksimiparitus, 173  
 maksimivirtaus, 167  
 Manhattan-etäisyys, 260  
 Markovin ketju, 216  
 matriisi, 203  
 matriisipotenssi, 205  
 matriisitulo, 204, 218  
 merkkijono, 34, 229  
 merkkijonohajautus, 231  
 mex-funktio, 225  
 minimileikkaus, 168  
 Mo'n algoritmi, 241  
 modulolaskenta, 6  
 Monte Carlo -algoritmi, 217  
 multinomikerroin, 196  
 muutoshistoria, 248  
  
 naapuri, 101

neliöjuuri, 239  
 neliömatriisi, 203  
 nim-peli, 223  
  
 odotusarvo, 214  
 ohjelmointikieli, 3  
 Oren lause, 164  
 osajono, 229  
 osajoukko, 43  
  
 pakka, 39  
 permutaatio, 45  
 persistentti segmenttipuu, 248  
 peruuttava haku, 46  
 Pickin lause, 260  
 pienin yhteinen moninkerta, 184  
 pikajärjestäminen, 26  
 pino, 40  
 pisin nouseva alijono, 64  
 piste, 254  
 polkupeite, 176  
 polynominen hajautus, 231  
 Prüfer-koodi, 201  
 prefiksi, 229  
 Primin algoritmi, 134  
 prioriteettijono, 40  
 puolivälihaku, 49  
 puu, 123  
 Pythagoraan kolmikko, 190  
 pyyhkäisyviiva, 263  
  
 ratsun kierros, 166  
 rekursioyhtälö, 60, 206  
 repunpakkaus, 66  
 reuna, 230  
 riippumaton joukko, 175  
 riippumattomuus, 214  
 ristitulo, 256  
  
 satunnaisalgoritmi, 217  
 satunnaismuuttuja, 214  
 segmenttipuu, 83, 243  
 seuraajaverkko, 142  
 sisäjärjestys, 128  
 solmu, 99  
 solmupeite, 175  
 SPFA-algoritmi, 116  
 Sprague–Grundyn lause, 224

suffiksi, 229  
 suhteellinen alkuluku, 185  
 suljettu muoto, 193  
 sulkulauseke, 196  
 summataulukko, 78  
 suurin alitaulukko, 20  
 suurin yhteinen tekijä, 184  
 syöte ja tuloste, 4  
 sykli, 111, 137, 143  
 syklin tunnistaminen, 143  
 syntymäpäiväparadoksi, 233  
 syvyyshaku, 107  
  
 täydellinen luku, 182  
 törmäys, 233  
 tasajakauma, 215  
 tasoitettu analyysi, 71  
 tekijä, 181  
 tietorakenne, 33  
 todennäköisyys, 211  
 topologinen järjestys, 137  
 transpoosi, 203  
 trie, 230  
  
 union-find-rakenne, 132  
  
 välikysely, 77  
 väritys, 102, 219  
 vaativuusluokka, 18  
 vahvasti yhtenäisyys, 145  
 vektori, 33, 203, 254  
 verkko, 99  
 vieruslista, 103  
 vierusmatriisi, 104  
 virittävä puu
 

- pienin ja suurin, 129
- yhteismäärä, 209

 virtaus, 167  
 voittotila, 221  
  
 Warnsdorffin sääntö, 166  
 Wilsonin lause, 191  
  
 yhtenäisyys, 100, 111  
 ykkösmatriisi, 204  
  
 Z-algoritmi, 234  
 Z-taulukko, 234  
 Zeckendorfin lause, 190