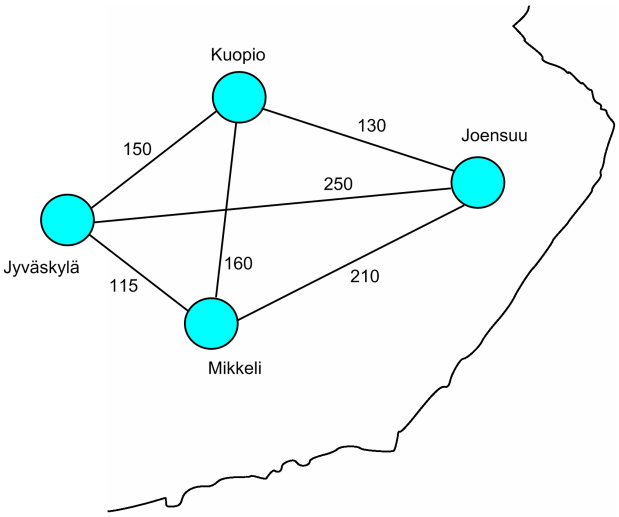
3 Painotetut verkot

3.1 Kauppamatkustajan ongelma

Kauppamatkustajalla on ongelma. Hänen tulee kaupata pölynimureita neljässä eri kaupungissa (Kuopio, Mikkeli ja Jyväskylä) palaten työpäivän jälkeen kotikaupunkiinsa (Joensuu). Kartasta katsomalla hän selvittää paikkojen väliset etäisyydet, mutta mikä on lyhin reitti? (Kuva 51)



Kuva 51: Kauppamatkustaja lähtee Joensuusta, kiertää kaikki kaupungit ja palaa kotikaupunkiinsa. Mikä on lyhin reitti?

Joensuusta kauppamatkustaja voi lähteä kolmeen kaupunkiin, jonka jälkeen vaihtoehtoja on kaksi. Näin ollen reittivaihtoehtoja on

$$3 \cdot 2 \cdot 1 = 6.$$

Yksi tapa ratkaista ongelma on laskea kaikkien reittivaihtoehtojen pituudet ja valita niistä lyhin (Taulukko 2). Taulukosta nähdään, että kauppamatkustajalla on valittavissa kaksi reittiä, jotka molemmat ovat lyhimpiä.

Yleisessä tapauksessa voidaan olettaa, että jokaisesta kaupungista on tie jokaiseen kaupunkiin. Kun kaupunkeja on n kappaletta, niin reittivaihtoehtoja tulee $(n-1)!$ kappaletta. Eli jos kaupunkeja on 9, niin reittivaihtoehtoja on 40 320. Jos kaupunkeja on 10, niin reittivaihtoehtoja on 362 880. Näin ollen reittivaihtoehtojen lukumäärä kasvaa kaupunkien lukumäärän kasvaessa hyvinkin nopeasti.

Tämän tyyppisistä verkoista, jonka kaarilla on jokin pituus tai muu vastaava arvo, kutsutaan painotetuiksi verkoiksi. Kaaren arvoa kutsutaan painoksi.

Määritelmä 3.1.1. *Painotetussa verkossa*, jokaiseen kaareen liittyy jokin luku. Tätä lukua kutsutaan *kaaren painoksi*.

Kauppamatkustajan reitti	Reitin pituus [km]
Joensuu-Kuopio-Mikkeli-Jyväskylä-Joensuu	130+160+115+250=655
Joensuu-Kuopio-Jyväskylä-Mikkeli-Joensuu	**130+150+115+210=605**
Joensuu-Mikkeli-Kuopio-Jyväskylä-Joensuu	210+160+115+250=735
Joensuu-Mikkeli-Jyväskylä-Kuopio-Joensuu	**210+115+150+130=605**
Joensuu-Jyväskylä-Mikkeli-Kuopio-Joensuu	250+115+160+130=655
Joensuu-Jyväskylä-Kuopio-Mikkeli-Joensuu	250+150+160+210=770

Taulukko 2: Taulukossa laskettu kaikki reittivaihtoehdot, joista lyhimpiä on kaksi.

Määritelmä 3.1.2. Painotetussa verkossa *ketjun pituus* on ketjun kaarten painojen summa.

Verkko voidaan esittää kaupunkien välisinä yhteyksinä (Taulukko 3).

	Joensuu	Kuopio	Jyväskylä	Mikkeli
Joensuu	0	130	250	210
Kuopio	130	0	150	160
Jyväskylä	250	150	0	115
Mikkeli	210	160	115	0

Taulukko 3: Taulukoidaan kaupunkien väliset yhteydet, jossa välimatkat kilometreina.

Toisaalta taulukko voidaan esittää myös *paino- eli etäisyysmatriisina*, jossa solmujen välisiä yhteyksiä merkitään niitä vastaavilla painoilla

$$\mathbf{M} = \begin{array}{c} \\ \text{Joensuu} \\ \text{Kuopio} \\ \text{Jyväskylä} \\ \text{Mikkeli} \end{array} \begin{array}{c} \begin{array}{cccc} \text{Joensuu} & \text{Kuopio} & \text{Jyväskylä} & \text{Mikkeli} \end{array} \\ \begin{pmatrix} 0 & 130 & 250 & 210 \\ 130 & 0 & 150 & 160 \\ 250 & 150 & 0 & 115 \\ 210 & 160 & 115 & 0 \end{pmatrix} \end{array}.$$

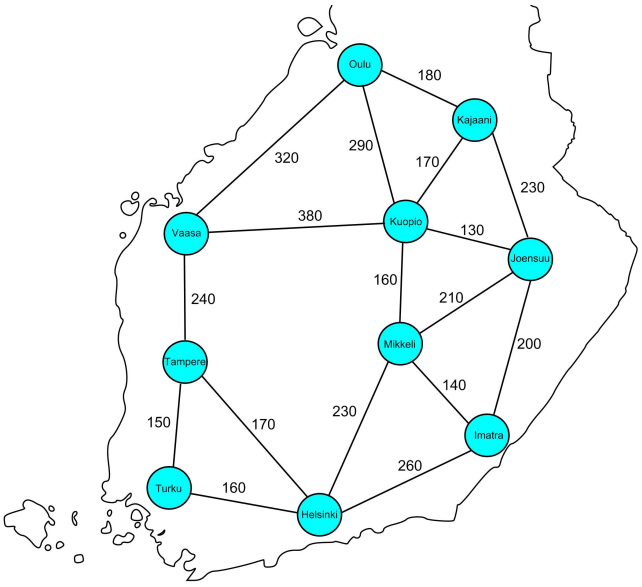
Nykyiset painotettuihin verkkoihin liittyvät ongelmat ja sovellukset ovat lähes poikkeuksetta lyhimmän reitin etsimiseen liittyviä ongelmia. Kauppamatkustajan ongelmassa etsitään **lyhintä suljettua ketjua** n kaupungin kautta.

Kauppamatkustajan ongelman lisäksi usein etsitään **lyhintä avointa ketjua kahden solmun** välille. Esimerkiksi tieliikenneverkosto voidaan kuvata painotettuna verkkona siten, että tiet ovat painotettuja kaaria ja risteykset solmuja. Navigaattori laskee lyhimmän reitin kahden solmun välille, jolloin kyseessä on lyhimmän painotettun ketjun etsiminen, missä painot ovat välimatkoja. Toisaalta navigaattori laskee myös nopeimman reitin, joka tapahtuu yksinkertaisella muunnoksella, missä kaarien painot (matkat) muunnetaan nopeusrajoitusten mukaan ajaksi.

Eräs painotettujen verkkojen ongelmatyyppi on etsiä **lyhin yhtenäinen verkko** n solmun välille. Esimerkiksi 10 kaupungin välille tulee rakentaa sähköverkko siten, että sähköverkko yhdistää kaikki kaupungit toisiinsa. Mutta mikä on pienin määrä kaapelia, jolla sähköverkko voidaan rakentaa?

3.2 Virittävät puut

Suomen tietoverkkoa uusitaan uudella valokaapelitekniikalla. Kuvassa 52 on esitetty kaupunkien välisiä välimatkoja. Koska valokaapeli on kallista, halutaan kaupunkien välille rakentaa mahdollisimman pieni, mutta kuitenkin yhtenäinen verkko. Mikä on minimimäärä valokaapelia, jolla tekniikan uusiminen voidaan toteuttaa?



Kuva 52: Mikä on pienin määrä valokaapelia, jolla kaikki kaupungit saadaan yhdistettyä toisiinsa?

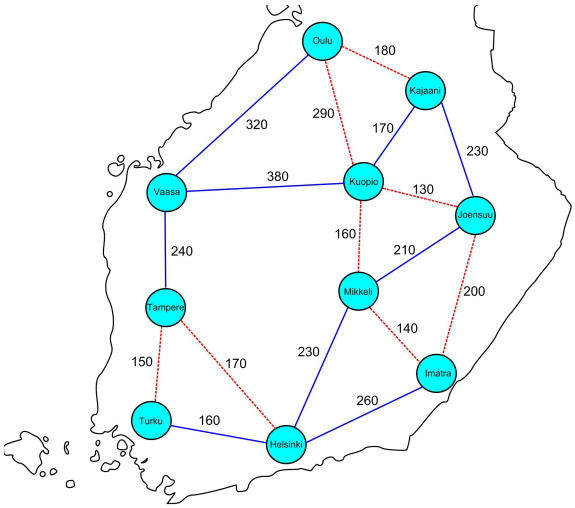
Jos Suomen tietoverkko uusitaan siten, että Kuvan 52 kaikki yhteydet uusitaan, tarvitaan valokaapelia yhteensä 3590 km. Riittää kuitenkin, että kahden kaupungin välillä on vain yksi ketju, joten valokaapelia tarvitaan vähemmän. Jos kahden kaupungin välillä on täsmälleen yksi ketju, niin uusista yhteyksistä muodostuu puu.

Tehtävä voidaan muotoilla matemaattisemmaksi ja täsmällisemmäksi, jos käytetään puun ja kaaren painojen käsitteitä. Tällöin tehtävä kuuluu: Etsi kuvan verkosta sellainen puu, joka sisältää kaikki verkon solmut ja kaarten painojen summa on pienin.

Määritelmä 3.2.1. Puu on verkon *virittävä puu*, jos se sisältää kaikki verkon solmut.

Kuvan 52 verkon virittäviä puita on useita. Esimerkiksi Kuvassa 53 on eräs kyseisen verkon virittävä puu, jonka painojen summa on 2170 km.

Määritelmä 3.2.2. Jos virittävän puun painojen summa on kaikista virittävistä puista pienin, virittävä puu on *minimaalinen virittävä puu*.



Kuva 53: Siniset kaaret muodostavat virittävän puun, jonka painojen summa on 2170 km.

Suurista verkoista minimaalisen virittävän puun etsiminen ilman apuvälineitä on yleensä työlästä. Matematiikassa näitä apuvälineitä kutsutaan algoritmeiksi.

Algoritmi on toimintaohje, joka sisältää tietyn määrän käskyjä, jotka toteuttamalla päästään haluttuun lopputulokseen. Esimerkiksi ruokareseptit ovat algoritmeja. Syömisen algoritmi puolestaan voisi olla seuraava:

1) Laita ruoka suuhun.

2) Pureskele ja niele.

3) Oletko kylläinen?
-Jos vastaat ei, palaa kohtaan 1.
-Jos vastaat kyllä, siirry kohtaan 4.

4) Lopeta ruokailu.

Algoritmissa voidaan siis toistaa joitakin käskyjä useitakin kertoja. Kaikki algoritmit esitetään tässä oppimateriaalissa oranssin laatikon sisällä.

3.3 Minimaalisen virittävän puun algoritmeja

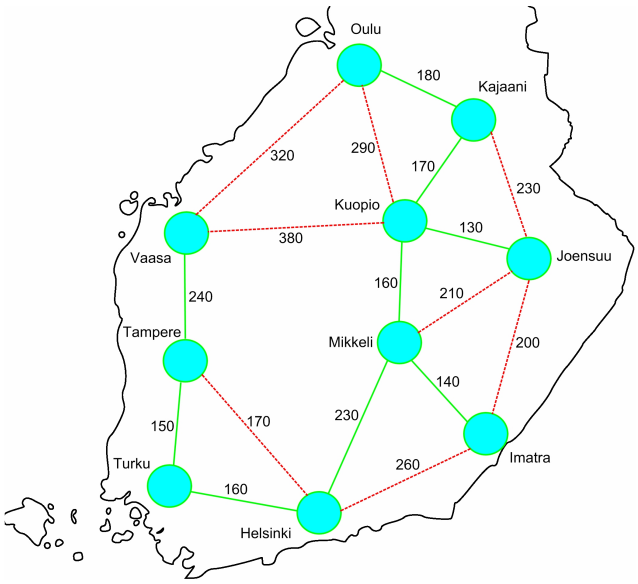
Minimaalisen virittävän puun algoritmeilla voidaan etsiä lyhin yhtenäinen verkko, joka yhdistää kaikki painotetun verkon solmut. Palataan takaisin sivulla 37 olleeseen tehtävään, jossa piti etsiä Kuvan 52 mukaiselle Suomen tietoverkolle minimaalinen virittävä puu.

Tilannetta pohtivat insinöörit keksivät suuressa viisaudessaan, että yhteydet voitaisi uusia lyhimmästä yhteydestä pisinpään, mutta kuitenkin niin, ettei turhia eli moninkertaisia yhteyksiä kaupunkien välille muodostu.

Kartasta katsomalla he selvittävät, että Kuopion ja Joensuun välinen yhteys on lyhin. Siis yhteyksien uusiminen aloitetaan siitä. He merkitsevät tämän yhteyden karttaansa vihreällä. Seuraavaksi lyhin yhteys on Mikkeli-Imatra, joten tämä uusitaan toisena, ja insinöörit merkitsevät tämänkin yhteyden karttaansa vihreällä. Kolmanneksi lyhin yhteys on Tampereen ja Turun välilä, joten sekin merkitään karttaan vihreällä. Seuraavaksi lyhimpiä yhteyksiä on kaksi, nimittäin yhteydet Turku-Helsinki ja Kuopio-Mikkeli. Koska molemmat ovat samanpituisia, insinöörit valitsevat arvalla yhteyden Kuopio-Mikkeli, joka uusitaan ennen yhteyttä Turku-Helsinki. Molemmat kuitenkin uusitaan, joten ne merkitään karttaan vihreällä.

Seuraavaksi lyhimpiä yhteyksiä on jälleen kaksi nimittäin yhteydet Kuopio-Kajaani ja Tampere-Helsinki. Tällä kertaa arpa suosii yhteyttä Tampere-Helsinki, joka uusitaan ennen yhteyttä Kuopio-Kajaani. Juuri kun insinöörit ovat värittämässä tätä yhteyttä, heistä tarkkanäköisin huomauttaa, että Helsinki ja Tampere ovat jo yhteydessä toisiinsa Turun välityksellä. Helsingin ja Tampereen välisen yhteyden uusiminen olisi rahan haaskausta, joten tämä yhteys jätetään uusimatta, ja insinöörit värittävät sen karttaansa punaisella. Sen sijaan tämän yhteyden kanssa samanpituinen yhteys Kuopio-Kajaani uusitaan, ja se väritetään vihreällä.

Insinöörit jatkavat yhteyksien värittämistä tällä tavoin, ja lopulta heillä on edessään Kuvan 54 mukainen kartta, jossa vihreät kaaret kertovat minimaalisen virittävän puun. Tällaista menettelyä sanotaan Kruskalin algoritmiksi.



Kuva 54: Vihreät kaaret muodostavat minimaalisen virittävän puun, jonka painojen summa eli kokonaispaino on 1560 km.

Kruskalin algoritmi

Kruskalin algoritmi tuottaa minimaalisen virittävän puun. Algoritmin idea on seuraava: verkon kaaret väritetään vihreällä pienimmästä painosta suurimpaan siten, että suljettuja ketjua ei muodostu.

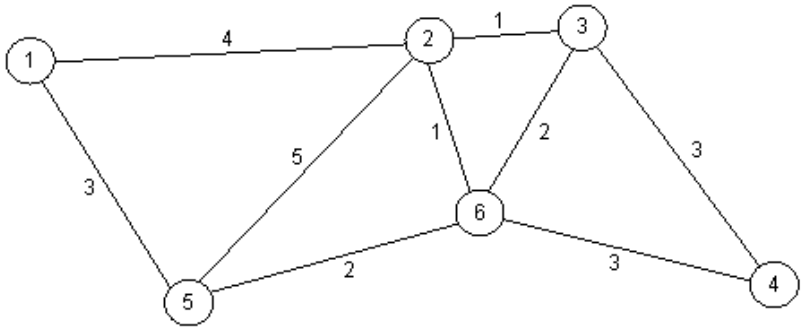
$K1$. Valitse pienimmän painon omaava kaari ja väritä se solmuineen vihreäksi. Mikäli pienimmän painon omaavia kaaria on useita, valitse niistä yksi.

$K2$. Valitse jäljelle jäävästä verkosta jälleen pienimmän painon omaava kaari ja väritä se vihreäksi.

$K3$. Valitse jäljelle jäävästä verkosta pienimmän painon omaava kaari ja väritä se vihreäksi, jos verkkoon ei muodostu suljettua ketjua. Väritä kaari punaiseksi, jos muodostuu suljettu ketju.

$K4$. Toista kohtaa $K3$, kunnes kaikki kaaret on väritetty. Lopuksi korvaa punaisella väritetyt kaaret **mustalla**. Algoritmi on päättynyt.

Etsi Kuvan 55 painotetusta verkosta minimaalinen virittävä puu Kruskalin algoritmin avulla.

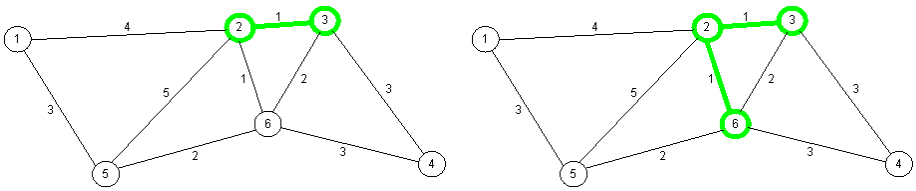


Kuva 55: Etsi painotetusta verkosta minimaalinen virittävä puu.

Kokeillaan Kruskalin algoritmia Kuvan 55 verkolle.

1. Verkon kaarten pienin paino on 1, joka löytyy kaarilta $(2,3)$ ja $(2,6)$. Väritetään niistä toinen, esimerkiksi kaari $(2,3)$, vihreällä. (Kuva 56 vasemmalla)

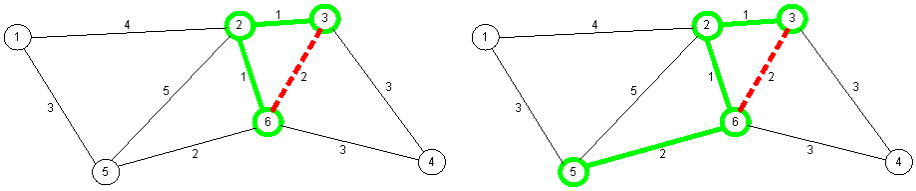
2. Jäljellä olevien kaarien pienin paino on 1, joka on kaarella $(2,6)$. Väritetään kaari (2,6) vihreällä. (Kuva 56 oikealla)



Kuva 56: Kruskalin algoritmissa kaaret väritetään vihreällä pienimmästä painosta suurimpaan.

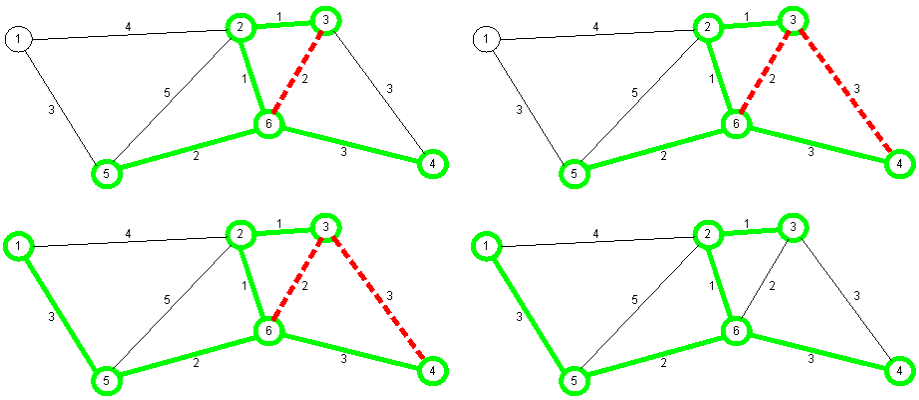
3. Jäljellä olevista kaarista pienin paino 2 löytyy kaarilta $(3, 6)$ ja $(5, 6)$. Väritetään kaari $(3, 6)$ punaiseksi, koska se muodostaa suljetun ketjun vihreällä väritettyjen kaarten kanssa. (Kuva 57 vasemmalla)

4. Jäljellä olevien kaarien pienin paino on 2, joka on kaarella $(5, 6)$. Väritetään kaari $(5, 6)$ vihreällä. (Kuva 57 oikealla)



Kuva 57: Jos pienimmän painon omaavan kaaren värittäminen muodostaa suljetun ketjun vihreällä väritetyn puun kanssa, niin tällöin kaari väritetään punaisella.

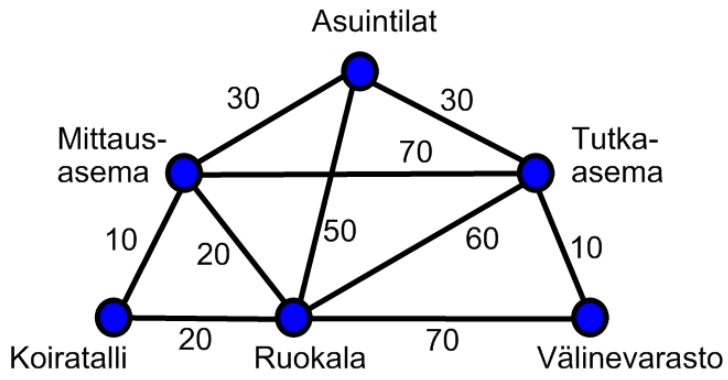
5. Toistetaan algoritmin kohtaa $K3$, kunnes algoritmi pysähtyy. (Kuva 58)



Kuva 58: Minimaalinen virittävä puu muodostuu vihreistä kaarista.

Kruskalin algoritmi pysähtyy, kun kaikki verkon kaaret on väritetty joko vihreällä tai punaisella. Algoritmin viimeisessä vaiheessa $K4$ korvataan punaiset kaaret mustilla kaarilla. Lopputuloksena on vihreällä väritetty verkon minimaalinen virittävä puu. Kuvan 55 verkon minimaalisen virittävän puun painojen summa on 10.

Napajäätiköllä olevalla tutkijalla on ongelma. Yöllä on satanut lunta niin paljon, että kaikki tutkimuskeskuksen tiet ovat kulkukelvottomia. Työpäivä ei pääse käyntiin, ennen kuin hänellä on pääsy jokaiseen tutkimuskeskuksen rakennukseen. Miten tutkijan kannattaa lapioida tiet puhtaaksi, että varsinaisiin töihin päästään mahdollisimman nopeasti? Kuvan 59 painotetussa verkossa painot ovat välimatkoja metreissä.



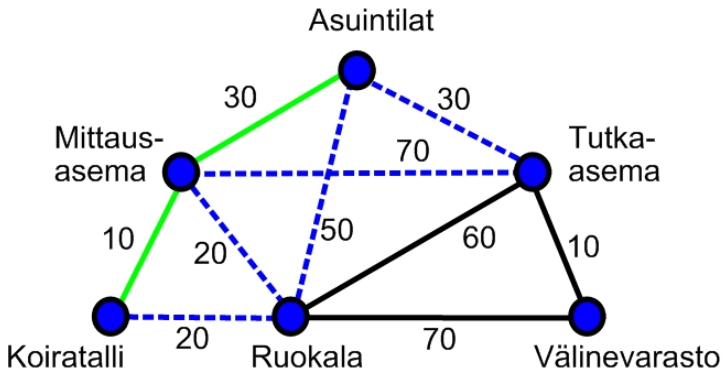
Kuva 59: Mitkä tiet tutkijan kannattaa aurata, jotta lunta täytyisi lapioida mahdollisimman vähän?

Tutkija seisoo asuintilojen edessä ja pähkäilee tilannetta. Hänellä on edessään kolme tietä, joista yksi johtaa mittausasemalle, toinen ruokalaan ja kolmas tutka-asemalle. Mittausasemalle ja tutka-asemalle johtavat tiet ovat lyhyemmät kuin ruokalaan johtava tie, joten tutkija päättää tehdä ensiksi jomman kumman niistä. Hän valitsee mittausasemalle johtavan tien.

Tämän raivattuaan tutkija pohtii seuraavaksi työstettävää tietä. Hän voi lapioida puhtaaksi tien mittausasemalta koiratalliin, ruokalaan tai tutka-asemalle. Toisaalta hän voi palata takaisin äsken raivaamaansa tietä pitkin, ja raivata tien asuintilalta ruokalaan tai tutka-asemalle. Kaikista näistä viidestä vaihtoehdosta mittausaseman ja koiratallin välinen tie on lyhin, joten tutkija valitsee sen (Kuva 60).

Nyt tutkijalla on jälleen vaihtoehtoina viisi tietä, joista mittausaseman ja ruokalan sekä koiratallin ja ruokalan väliset tiet ovat lyhimmät (Kuva 60). Tutkija valitsee koiratallin ja ruokalan välisen tien ja lapioi sen puhtaaksi.

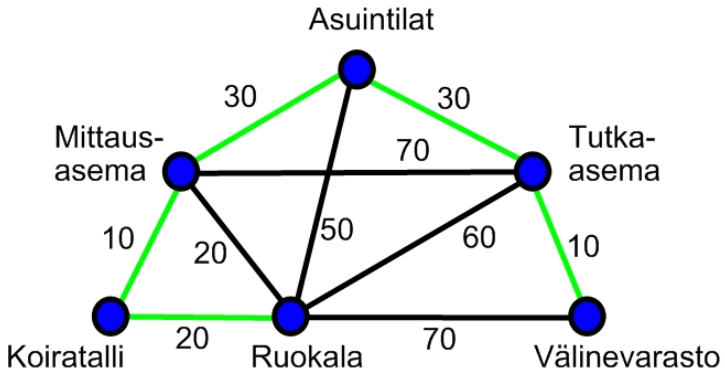
Seuraavaksi edessä olevista vaihtoehdoista lyhin olisi ruokalan ja mittausaseman välinen tie. Tutkijalla kuitenkin on jo auratut tiet mittausasemalle, joten tämän polun raivaaminen olisi turhaa työtä. Niinpä hän katsoo mikä olisi seuraavaksi lyhin vaihtoehto. Se on asuintilan ja tutka-aseman välinen tie, joten



Kuva 60: Kun vihreällä merkityt tiet on aurattu, tutkija voi seuraavaksi aurata jonkin sinisellä merkityistä teistä.

tutkija kävelee puhtaaksi lapioimiaan teitä pitkin takaisin asuintilalle, ja lapioi polun tutka-asemalle puhtaaksi.

Tämän jälkeen jäljellä olevista lumen peittämistä teistä tutka-aseman ja välinevaraston välinen tie on lyhin, joten tutkija lapioi sen puhtaaksi. Nyt hänellä on pääsy kaikkiin tutkimusaseman rakennuksiin, ja varsinainen työpäivä voi alkaa (Kuva 61).



Kuva 61: Tutkija on tehnyt lumitöitä 100 metrin matkalta, kun hänellä on pääsy kaikkiin tutkimuskeskuksen rakennuksiin.

Lumesta puhdistetut tiet muodostavat tutkimusaseman tieverkoston minimaalisen virittävän puun. Tällaista minimaalisen virittävän puun tuottamisen menetelmää kutsutaan Primin algoritmiksi.

Primin algoritmi

Primin algoritmi tuottaa minimaalisen virittävän puun. Algoritmi käynnistyy, kun sille syötetään aloitussolmu, josta puun rakentaminen aloitetaan.

$P1$. Valitse verkosta aloitussolmu ja väritä se vihreäksi. Väritä sinisellä kaikki aloitussolmuun liittyneet kaaret.

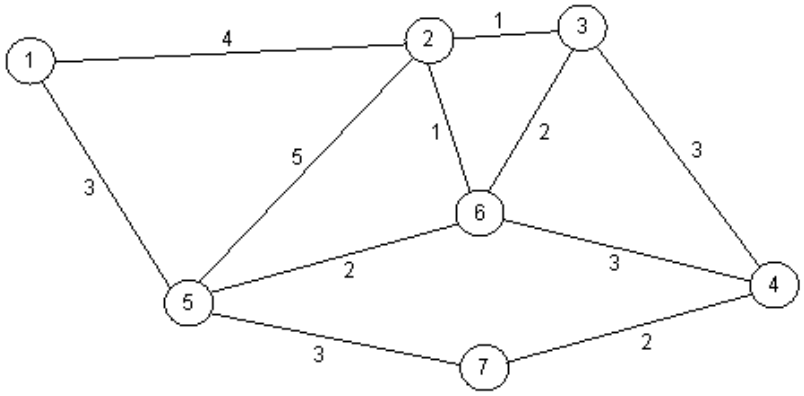
$P2$. Valitse sinisistä kaarista pienin ja väritä kaari k_1 ja kaaren päässä oleva solmu s_1 vihreäksi.

$P3$. Väritä solmuun s_1 kiinnittyneet kaaret sinisellä. Mikäli jokin solmu on päätesolmu kahdelle tai useammalle kaarelle ts. syntyy suljettu ketju, väritä niistä painoltaan suurempi punaisella.

$P4$. Valitse sinisistä kaarista pienin ja väritä kaari ja kaaren päässä oleva solmu vihreäksi.

$P5$. Toista kohtia $P2 - P4$ kunnes et voi jatkaa menettelyä. Lopuksi korvaa punaisella väritetyt kaaret **mustalla**. Algoritmi on päättynyt.

Etsi Kuvan 62 painotetusta verkosta minimaalinen virittävä puu Primin algoritmin avulla.

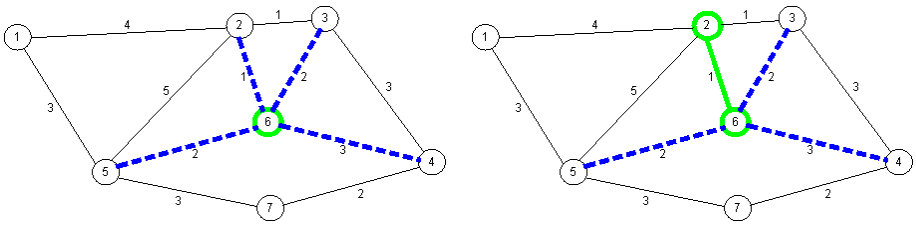


Kuva 62: Muodosta kuvan verkosta Primin algoritmin avulla minimaalinen virittävä puu.

Muodostetaan minimaalinen virittävä puu Kuvan 62 verkolle Primin algoritmin avulla.

1. Valitaan aloitussolmuksi solmu 6. Väritetään kaikki aloitussolmuun liittyneet kaaret $(6,2), (6,3), (6,4)$ ja $(6,5)$ sinisellä. (Kuva 63 vasemmalla)

2. Sinisistä kaarista pienin on kaari $(6,2)$, joten väritetään kaari ja solmu 2 vihreällä. (Kuva 63 oikealla)

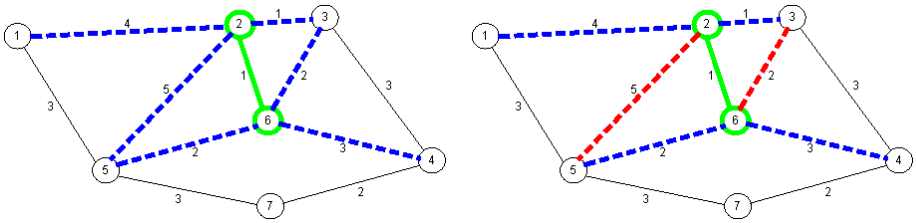


Kuva 63: Väritetään aloitussolmuun liittyneistä kaarista pienin vihreällä.

3. Väritetään kaikki solmuun 2 liittyneet kaaret $(2,1), (2,3)$ ja $(2,5)$ sinisellä. (Kuva 64 vasemmalla)

4. Nyt solmu 3 on päätesolmu kahdelle kaarelle. Väritetään kaari $(3,6)$ punaisella, koska se on suurempi kuin kaari $(2,3)$. (Kuva 64 oikealla)

5. Myös solmu 5 on päätesolmu kahdelle kaarelle, joten väritetään myös kaari $(2,5)$ punaisella, koska se on suurempi kuin kaari $(5,6)$. (Kuva 64 oikealla)

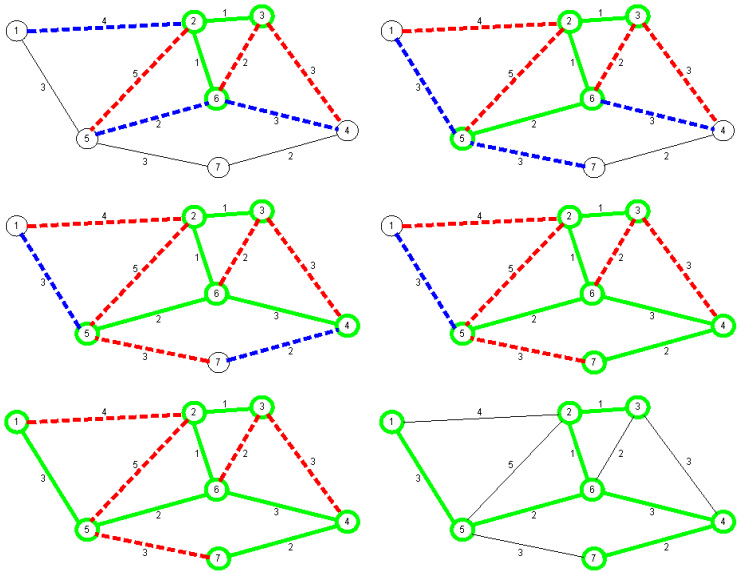


Kuva 64: Mikäli solmu on päätesolmu kahdelle kaarelle väritetään kaarista suurempi punaisella.

6. Sinisistä kaarista pienin on kaari $(2,3)$, joten väritetään kaari ja solmu 3 vihreällä.

7. Jatketaan vastaavalla tavalla, kunnes algoritmi pysähtyy. (Kuva 65)

Lopulta Primin algoritmi tuottaa verkolle minimaalisen virittävän puun, jonka painojen summa on 12. Sekä Primin, että Kruskalin algoritmi tuottavat saman lopputuloksen eli minimaalisen virittävän puun. Tarvitsemmeko kaksi erilaista algoritmia, jotka molemmat tuottavat saman lopputuloksen? Mikä erottaa nämä kaksi algoritmia?



Kuva 65: Primin algoritmi tuottaa minimaalisen virittävän puun.

Verkkoteorian algoritmeja käytetään nykyään pääasiassa tietokoneilla. Silti tietokoneiden laskentakapasiteetti loppuu nopeasti, jos lasketaan esimerkiksi minimaalista virittävää puuta erityisen suurelle verkolle. Tästä johtuen *algoritmien tehokkuutta* mitataan niiden tilan- ja ajankäytöllä. Koska tietokoneiden muistit ovat algoritmien tarvitsemaan tilaan verrattuna suhteellisen suuria, niin algoritmien tehokkuutta mitataankin pääasiassa niiden ajankäytöllä. Toisin sanoen algoritmi on sitä tehokkaampi mitä nopeammin se ratkaisee ongelman.

Primin ja Kruskalin algoritmien ajankäyttö riippuu verkon rakenteesta eli kaarien ja ja solmujen lukumäärästä.

Määritelmä 3.3.1. Verkon $V = (S, K)$ solmujen lukumäärää merkitään $|S|$ ja kaarien lukumäärää $|K|$.

Primin algoritmin ajankäyttö saadaan laskettua kaavalla $|K| + |S| \lg |S|$ ja Kruskalin $|K| \lg |S|$, missä $|S|$ on solmujen lukumäärä ja $|K|$ on kaarten lukumäärä.

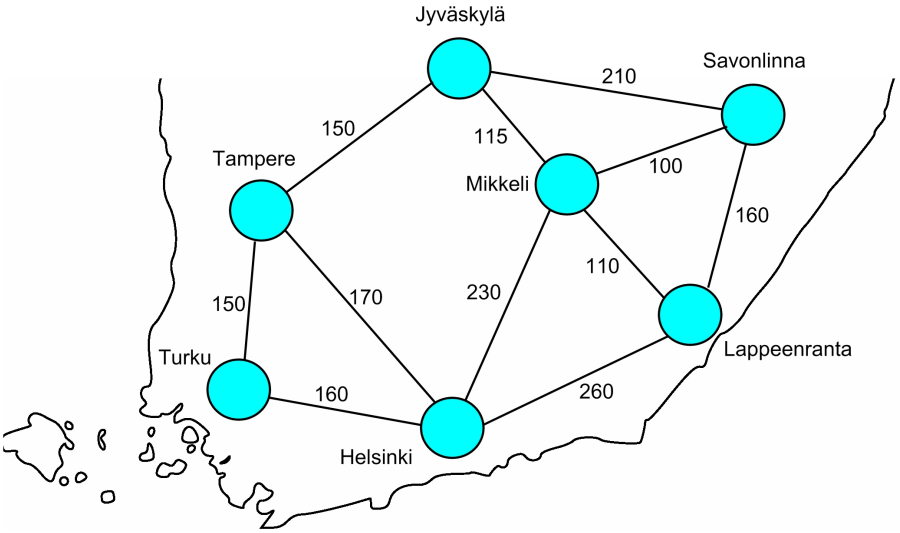
Sivun 37 tehtävässä täytyi ratkaista mikä on pienin määrä valokaapelia, jolla Suomen 10 kaupunkia saadaan yhdistettyä toisiinsa. Kyseisessä painotetussa verkossa on 10 solmua ja 17 kaarta. Tehtävässä täytyi etsiä siis minimaalinen virittävä puu, joten voidaan valita etsitäänkö se Primin vai Kruskalin algoritmilla. Ajankäytön kannalta on järkevää käyttää kyseiselle verkolle Kruskalin algoritmia, koska se suoriutuu ongelmasta nopeammin. (Taulukko 4)

Algoritmi	Tehokkuus	Jos $\lvert S\rvert = 10$ ja $\lvert K\rvert = 17$	Ajankäyttö/yksikköä
Prim	$\lvert K\rvert + \lvert S\rvert \lg \lvert S\rvert$	$17 + 10 \lg 10$	27
Kruskal	$\lvert K\rvert \lg \lvert S\rvert$	$17 \lg 10$	17

Taulukko 4: Prim vs Kruskal.

3.4 Lyhimmän ketjun algoritmi

Ulkomailta Suomeen saapuva posti kulkee Turun sataman kautta, josta posti toimitetaan eteenpäin. Mitkä olisivat lyhimmät reitit kuljettaa posti Turusta Kuvan 66 kaupunkeihin?



Kuva 66: Mitä reittejä pitkin posti kannattaa kuljettaa Turusta eteenpäin?

Kuvan 66 painotetusta verkosta on tarkoitus etsiä sellainen kaikki solmut sisältävä verkko, jossa on Turun ja kaikkien muiden kaupunkien välillä lyhimmät ketjut. Toisin sanoen painotetusta verkosta tulee etsiä virittävä puu, jossa on lyhimmät ketjut Turusta verkon muihin solmuihin. Seuraavaksi tutustutaan Dijkstran algoritmiin, joka tuottaa edellä kuvatun lopputuloksen.

Dijkstran algoritmi

Dijkstran algoritmi tuottaa verkon virittävän puun, jossa on lyhimmät ketjut aloitussolmusta muihin solmuihin.

*D*1. Valitse aloitussolmu s_0, väritä se vihreäksi ja merkitse solmun sisään luku 0.

*D*2. Väritä sinisellä aloitussolmuun s_0 liittyneet kaaret solmuineen. Merkitse sinisten solmujen sisään ketjun pituus aloitussolmusta ko. solmuun.

*D*3. Valitse sinisellä väritetyistä solmuista se, jonka sisällä on pienin numero s_1 (ts. matka aloitussolmusta ko. solmuun). Väritä se kaari ja solmu vihreällä.

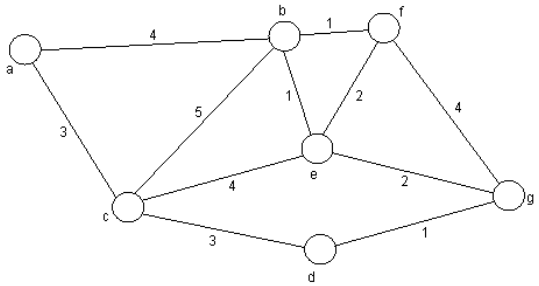
*D*4. Väritä sinisellä solmuun s_1 liittyneet kaaret solmuineen. Merkitse sinisiin solmuihin ketjun pituus aloitussolmusta s_0 solmun s_1 kautta kyseessä olevaan solmuun.

*D*5. Mikäli johonkin solmuun pääsee kahta tai useampaa ketjua pitkin valitse niistä lyhyin ja merkitse sen ketjun pituus ko. solmuun. Väritä punaisella se kaari, jonka kautta ko. solmuun on pidempi matka.

*D*6. Toista kohtia $D3 - D6$, kunnes kaikki solmut kuuluvat virittävään puuhun ja siirry kohtaan *D*7.

*D*7. Lopuksi korvaa punaisella väritetyt kaaret **mustalla**. Algoritmi on päättynyt.

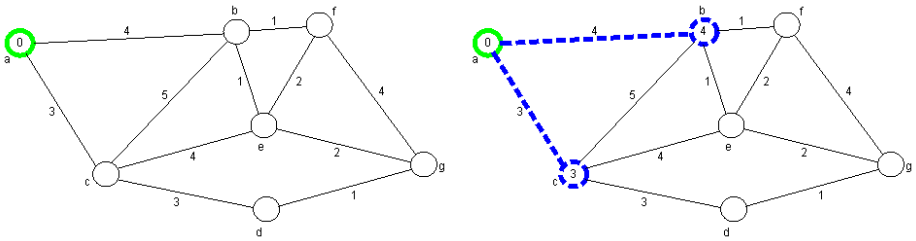
Etsi verkon kaikki lyhimmät ketjut aloitussolmusta a muihin solmuihin Dijkstran algoritmin avulla. (Kuva 67)



Kuva 67: Etsi verkon kaikki lyhimmät ketjut aloitussolmusta a muihin solmuihin.

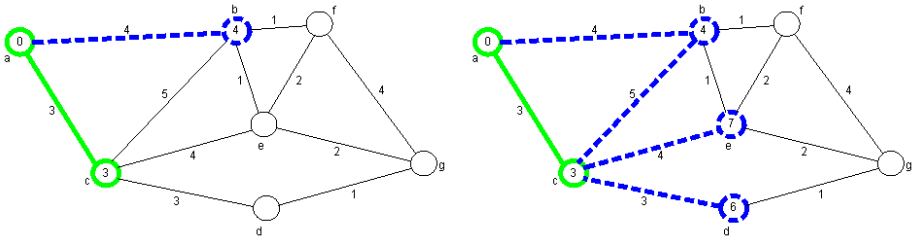
Tutustutaan Dijkstran algoritmiin ja etsitään verkolle kaikki lyhimmät ketjut aloitussolmusta a muihin solmuihin.

1. Väritetään aloitussolmu a vihreällä ja merkitään sen sisään luku 0. (Kuva 68 vasemmalla)
2. Väritetään solmuun a liittyneet kaaret (a, b) ja (a, c) solmuineen sinisellä. Merkitään sinisten solmujen sisään ketjun pituus aloitussolmusta. (Kuva 68 oikealla)



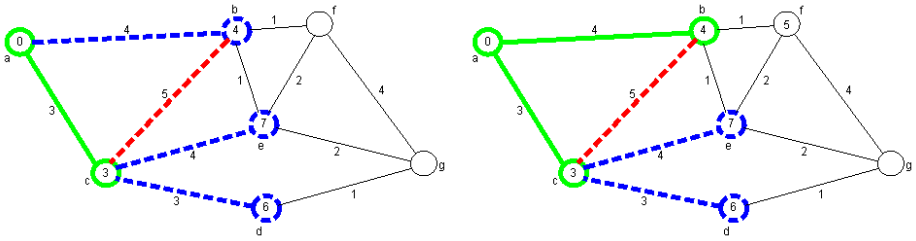
Kuva 68: Valitaan aloitussolmu ja väritetään se vihreällä. Väritetään kaikki aloitussolmuun liittyneet kaaret sinisellä ja merkitään sinisten solmujen sisään ketjun pituus aloitussolmusta.

3. Väritetään sinisistä solmuista solmu c ja kaari (a, c) vihreällä, koska sinisistä solmuista solmun c sisällä on pienin luku. (Kuva 69 vasemmalla)
4. Väritetään solmuun c liittyneet kaaret (c, b), (c, e) ja (c, d) solmuineen sinisellä. Merkitään sinisten solmujen sisään ketjun pituus aloitussolmusta. (Kuva 69 oikealla)



Kuva 69: Väritetään sinisistä solmuista se solmu kaarineen vihreällä, jonka sisällä on pienin luku. Väritetään sinisellä kaikki vihreisiin solmuihin liittyneet kaaret.

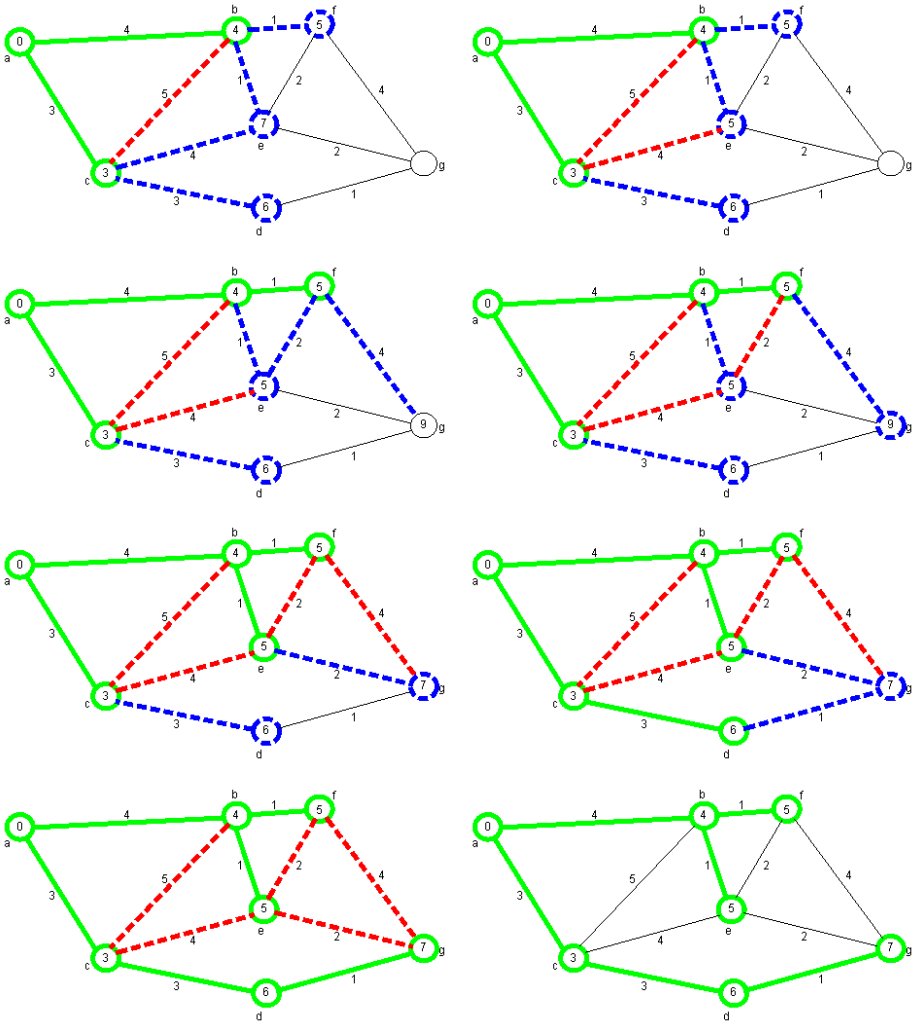
5. Nyt solmu b on päätesolmu kahdelle siniselle kaarelle, joten väritetään niistä suurempi kaari (c, b) punaisella. Merkitään solmun b sisään ketjun pituus sinisiä kaaria pitkin. (Kuva 70 vasemmalla)
6. Väritetään sinistä solmuista solmu b ja kaari (a, b) vihreällä, koska sinisistä solmuista solmun b sisällä on pienin luku. (Kuva 70 vasemmalla)



Kuva 70: Kahdesta reitti vaihtoehdosta pidempi väritetään punaisella.

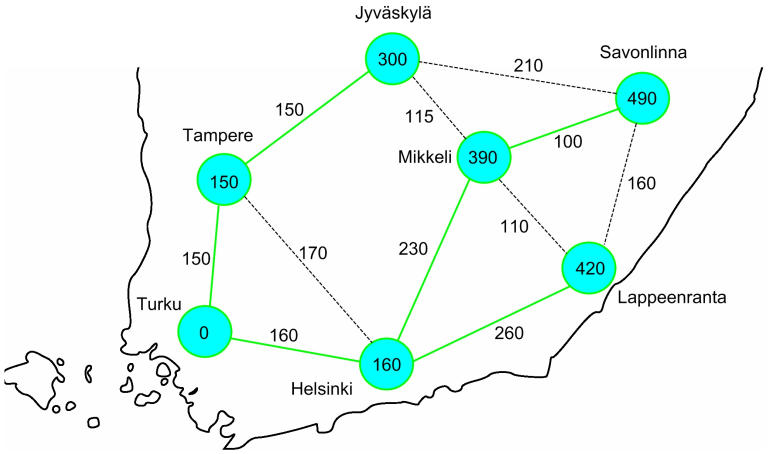
7. Menetelmää jatketaan kunnes algoritmi pysähtyy. (Kuva 71)

Dijikstran algoritmi tuottaa lopulta virittävän puun, jossa on pienimmät etäisyydet aloitussolmusta a kaikkiin muihin verkon solmuihin. Dijkstran algoritmi soveltuu tilanteisiin, jossa etsitään esimerkiksi kahden kaupungin välistä lyhintä etäisyyttä.



Kuva 71: Dijkstran algoritmilla voidaan muodostaa virittävä puu, jossa on lyhimmät ketjut aloitussolmusta muihin solmuihin. Potentiaaliset ehdokkaat tulevaan virittävään puuhun merkitään sinisellä. Kahdesta reittivaihdosta väritetään pidempi reitti punaisella.

Palataan vielä luvun 3.4 alussa olleeseen tehtävään, jossa piti etsiä lyhimmät reitit Turusta Kuvan 66 muihin kaupunkeihin. Dijikstran algoritmi tuottaa lyhimmät ketjut aloitussolmusta kaikkiin muihin solmuihin. Jos Turku on aloitussolmu tällöin saadaan kaikki lyhimmät reitin muihin kaupunkeihin (Kuva 155).



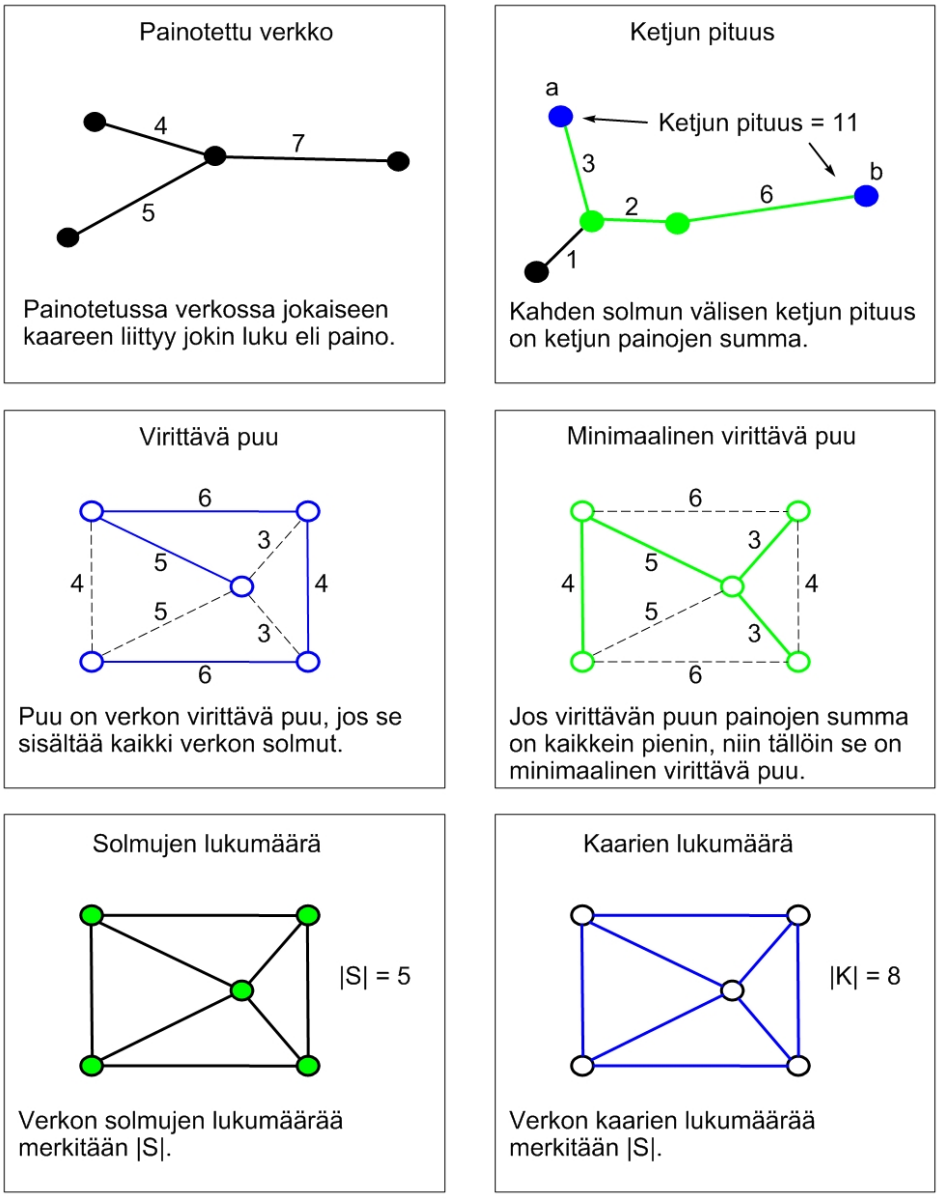
Kuva 72: Dijkstran algoritmi tuottaa lyhimmät etäisyydet Turusta kaikkiin muihin kaupunkeihin.

Kuvan 155 solmujen sisällä oleva luku kuvastaa matkaa Turkusta ko. kaupunkiin.

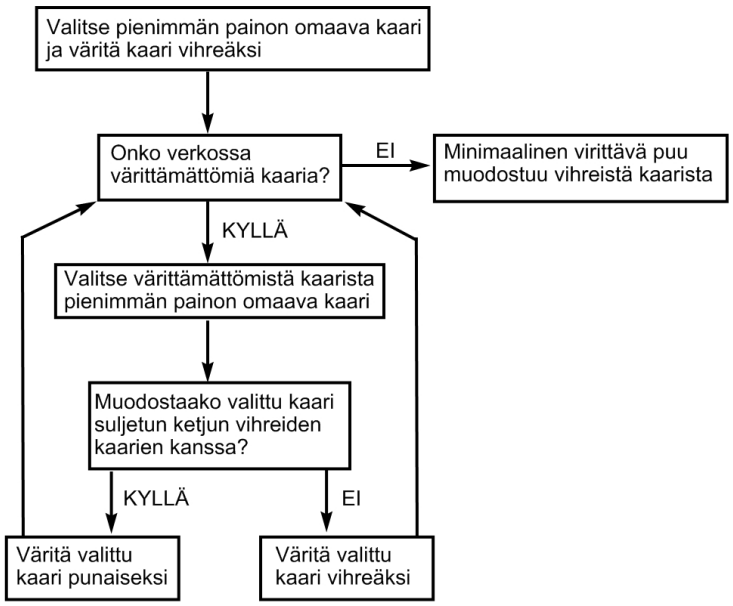
3.5 Käsitteiden kertausta

Tässä kappaleessa on käsitelty painotettuja verkkoja ja muun muassa käsitteitä: kaaren paino, ketjun pituus, virittävä puu, solmujen ja kaarten lukumäärä, kauppamatkustajan ongelma (lyhin suljettu ketju) ja minimalinen virittävä puu (lyhin yhtenäinen verkko). Algoritmeista tutuiksi ovat tulleet Kruskal, Prim ja Dijkstra.

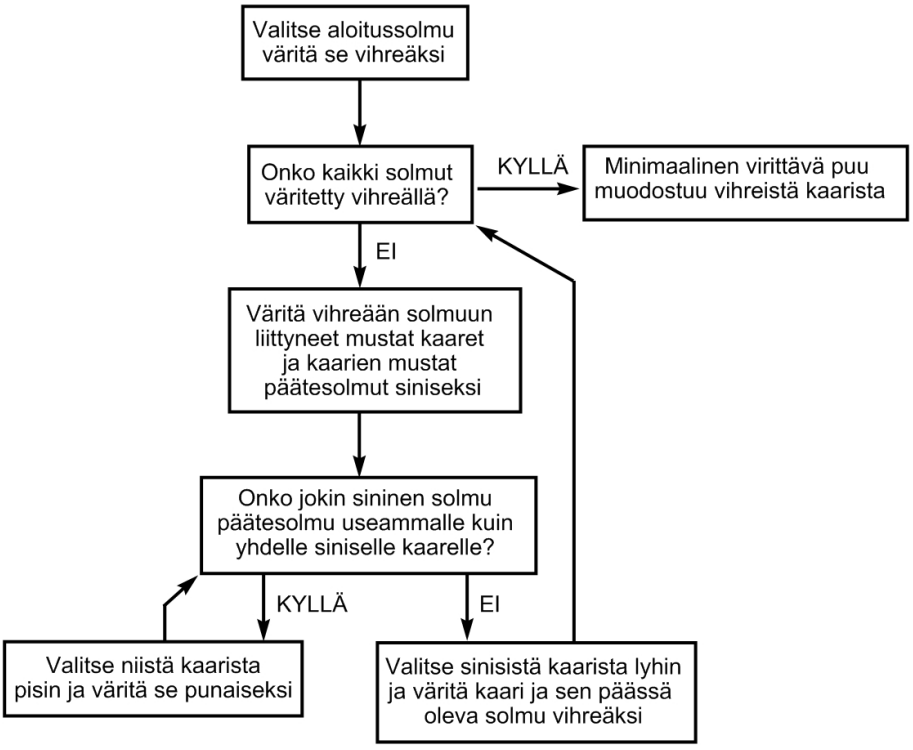
Kertaa aiempia käsitteitä sivuilta 53 - 55.



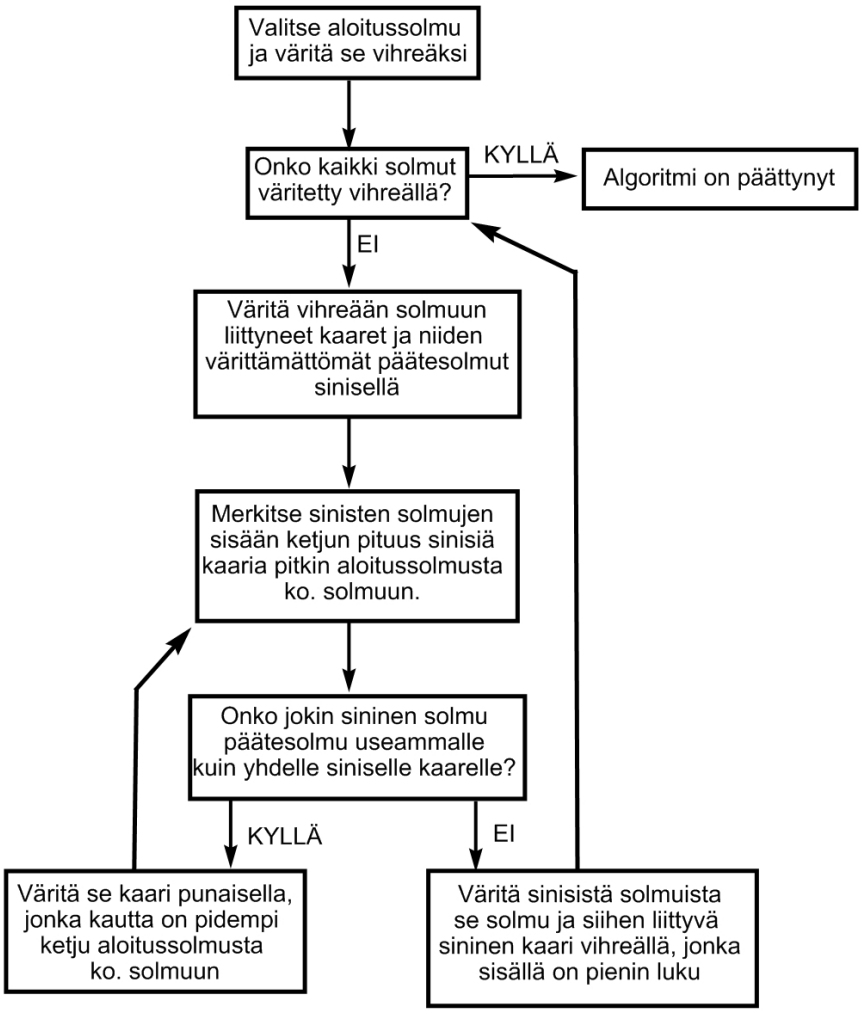
Kuva 73: Kappaleen 3 käsitteiden kertausta.



Kuva 74: Kruskalin algoritmi kaaviona.



Kuva 75: Primin algoritmi kaaviona

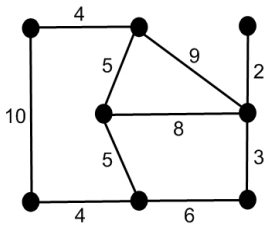


Kuva 76: Dijkstran algoritmi kaaviona

3.6 Tehtäviä

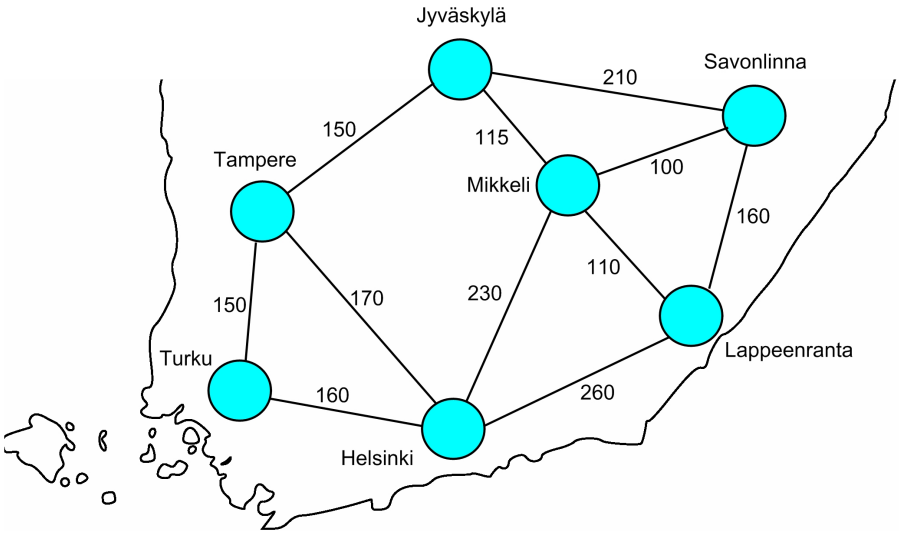
Tehtävä 3.6.1. Etsi Kuvan 77 verkosta virittävä puu. Mikä on virittävistä puista kaikkein pienin?

Tehtävä 3.6.2. Etsi Kuvan 77 verkon minimaalinen virittävä puu sekä Primin, että Kruskalin algoritmilla.



Kuva 77: Mikä on virittävistä puista pienin?

Seuraavat tehtävät liittyvät Kuvan 78 painotettuun verkkoon, jossa painot ovat kaupunkien välisiä etäisyyksiä kilometreissä.



Kuva 78: Verkon painot ovat kaupunkien välisiä etäisyyksiä.

Tehtävä 3.6.3. Perustele millä algoritmilla voidaan laskea nopeiten minimaalinen virittävä puu Kuvan 78 painotetusta verkosta.

a) Prim,

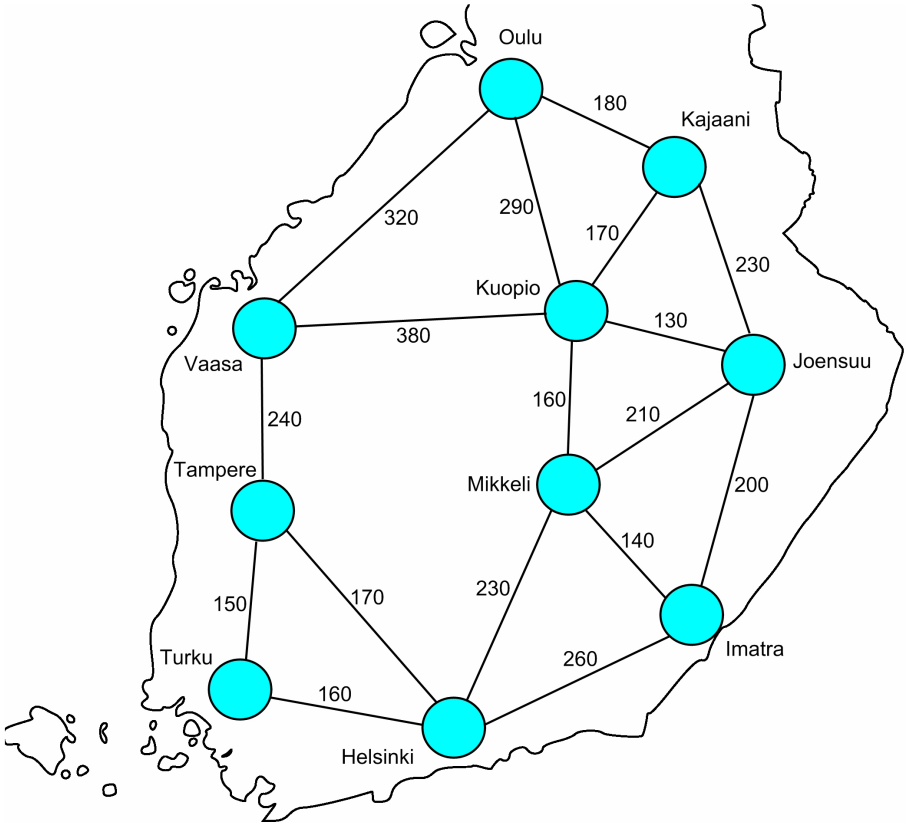
b) Dijkstra vai

c) Kruskal?

Tehtävä 3.6.4. Talvi on yllättänyt taas Etelä-Suomen autoilijat. Kuvan 78 jokaisessa kaupungissa on odottamassa kolme lumiauraa, jotka voidaan tarpeen tullen lähettää puhdistamaan teitä. Mikä on lyhin matka, jonka auraamalla kaikkien kaupunkien välillä on aurattu reitti? Onnistuuko teiden auraaminen, jos kussakin kaupungissa on käytettävissä vain yksi lumiaura?

Tehtävä 3.6.5. Turkulaisella postimiehellä on ongelma. Kaikki muut posteljoonit ovat sairastuneet Turussa riehuvaan hullun lehmän influenssaan. Turkulaisen postimiehen tulee toimittaa postit kaikkiin kaupunkeihin palaten takaisin Turkuun. Mikä mahtaa olla lyhin reitti?

Tehtävä 3.6.6. Olkoon Kuvan 79 verkon painot kaupunkien välisiä etäisyyksiä kilometreinä. Laske lyhimmät reitit Joensuusta kaikkiin muihin kaupunkeihin Dijkstran algoritmin avulla.



Kuva 79: Laske lyhimmät reitit Joensuusta muihin kaupunkeihin.