

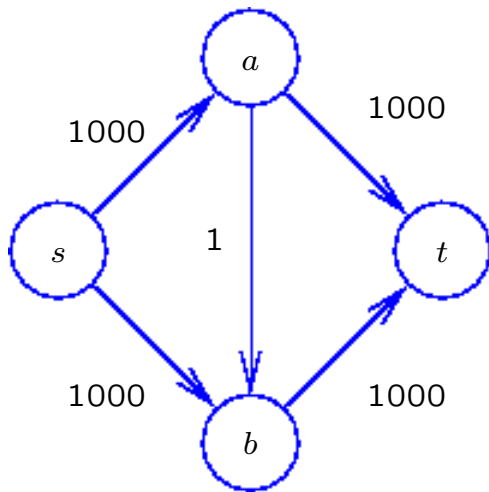
Edellinen tulos johtaa seuraavaan [Fordin-Fulkersonin menetelmään](#) maksimivuon määrittämiseksi:

1. Alusta $f(u, v) = 0$ kaikilla $u, v \in V$.
2. Jos vuolla f ei ole täydennyspolkua, se on maksimivuo; tulosta f .
3. Muuten olkoon p täydennyspolku. Kasvata vuota $f(u, v)$ kaikilla polun p kaarilla (u, v) jäännöskapasiteetin $\text{res}(p)$ verran ja palaa kohtaan 2.

Jos kaarten painot ovat *kokonaislukuja*, menetelmä ilmeisesti tuottaa maksimivuon, jolla on lisäksi se usein toivottava ominaisuus että kaikki arvot $f(u, v)$ ovat kokonaislukuja. Jokainen täydennysaskel kasvattaa vuota ainakin yhdellä, joten iteraatioiden maksimilukumäärä on $|f^*|$ missä f^* on maksimivuo.

Pahimmassa tapauksessa raja $|f^*|$ voidaan myös saavuttaa, jos p valitaan sopimattomasti.

Esimerkki



Jos valitaan aina vuorotellen täydennyspolut (s, a, b, t) ja (s, b, a, t) , vaaditaan 2000 täydennysaskelta.

Jos kaarten painot ovat irrationaalisia (mikä tietysti on sovelluksissa harvinaista), on jopa mahdollista, että menetelmä ei pysähdy, ja voimakkuus $|f|$ lähestyy arvoa joka on aidosti pienempi kuin maksimivuonvoimakkuus.

Kaksi ratkaisua ongelmaan:

1. Valitaan aina täydennyspolku p , jonka jäännöskapasiteetti $\text{res}(p)$ on maksimaalinen.
2. Valitaan aina (kaarten lukumäärällä mitaten) lyhin mahdollinen p .

Ylläolevassa esimerkissä kumpikin heuristiikka antaa maksimivuon kahdella täydennysaskelella.

Heuristiikka 2 on nimeltään **Edmondsin-Karpin algoritmi**.

Tarkastellaan ensin lyhyesti heuristiikkaa 1 (maksimikapasiteettitäydennys).

Voidaan osoittaa, että aina (myös irrationaalisilla kapasiteeteilla) $|f|$ suppenee kohti maksimivuonvoimakkuutta.

Jos painot ovat kokonaislukuja väliltä $[0, C]$, niin täydennysaskelten määrä on $O(|E| \log C)$.

Yksittäinen täydennysaskel (jännöskapasiteetiltaan suurimman täydennyspolun etsiminen) voidaan toteuttaa Dijkstran algoritmia mukaillen ajassa $O(|V| \log |V| + |E|)$.

Kokonaisajaksi tulee siis

$$O(|V||E|(\log |V|)(\log C) + |E|^2 \log C),$$

joka on polynominen verkon esityksen koon $\Theta(m \log C)$ suhteen.

Seuraavaksi näemme, että Edmondsin-Karpin algoritmilla päästään aikavaatimukseen $O(|V||E|^2)$ kapasiteettien suuruuksista riippumatta (kun kuitenkin oletetaan, että niitä voidaan käsitellä vakioajassa).

Olkoon R vuon f jäännösverkko. Solmun v taso $\delta(v)$ on lyhimmän lähteestä s solmuun v verkossa R johtavan polun pituus (eli kaarten lukumäärä).

Tasoverkko L on jäännösverkon R osaverkko, johon on otettu tasan ne verkon R kaaret (u, v) joilla $\delta(v) = \delta(u) + 1$. Siis erityisesti lyhimpiä täydennyspolkuja ovat tasan kaikki polut $s \rightsquigarrow t$ tasoverkossa L . Tasoverkko voidaan konstruoida ajassa $O(|E|)$ leveysuuntaisella haulla.

Lemma Olkoon f vuo ja p täydennyspolku, jota pitkin Edmondsin-Karpin algoritmi täydentää vuon f vuoksi f' . Olkoot jäännösverkko ennen täydennystä R ja täydennyksen jälkeen R' ; vastaavasti tasoverkot L ja L' ja tasofunktiot δ ja δ' . Nyt

1. $\delta'(t) \geq \delta(t)$ ja
2. jos $\delta'(t) = \delta(t)$, niin kaikki polut $s \rightsquigarrow t$ verkossa L' ovat polkuja myös verkossa L .

Todistus Jos kaari (u, v) on verkossa R' mutta ei verkossa R , niin kaari (v, u) on täydennyspolulla p . Siis $\delta(u) = \delta(v) + 1$ eli $\delta(v) = \delta(u) - 1$.

Jos kaari (u, v) on verkossa R , niin $\delta(v) \leq \delta(u) + 1$.
Yhtäsuuruus pätee joss (u, v) on verkossa L .

Joka tapauksessa jos kaari (u, v) on verkossa R' niin $\delta(v) \leq \delta(u) + 1$. Lisäksi yhtäsuuruus voi päteä vain, jos (u, v) on myös verkossa L .

Valitaan nyt mielivaltainen polku q lähteestä s nieluun t verkossa L' . Siis $\delta'(t) = |q|$, missä $|q|$ on polun q pituus.

Koska q on polku verkossa R' , ylläolevasta nähdään $\delta(t) \leq |q|$.

Lisäksi yhtäsuuruus voi päteä vain, jos kaikki polun q kaaret ovat myös verkossa L . \square

Lause Edmondsin-Karpin algoritmi tekee korkeintaan $|E|(|V| - 1)$ täydennysaskelta.

Todistus Tarkastellaan yhtä täydennysaskelta käyttäen edellisen lemmän merkintöjä.

Algoritmin toimintaperiaatteen nojalla valittu täydennyspolku p on polku $s \rightsquigarrow t$ tasoverkossa L mutta ei enää verkossa R' eikä siis etenään uudessa tasoverkossa L' .

Siis täydennyksessä tasoverkosta poistuu ainakin yksi polku $s \rightsquigarrow t$.

Jos $\delta'(t) = \delta(t)$, tasoverkkoon ei tule uusia kaaria, eikä siis uusia polkuja. Siis polkujen lukumäärä pienenee ainakin yhdellä. Tämä voi tapahtua korkeintaan $|E|$ kertaa peräkkäin, ennen kuin polut loppuvat.

Toisaalta tilanne $\delta'(t) \neq \delta(t)$ voi esiintyä vain $|V| - 1$ kertaa, sillä $\delta(t)$ on aluksi vähintään 1, ei koskaan pienene eikä koskaan kasva suuremmaksi kuin $|V| - 1$.
 \square

Korollaaari Edmondsin-Karpin algoritmi löytää maksimivuon ajassa $O(|V||E|^2)$. \square

Esivuot Edmondsin-Karpin algoritmi pitää yllä erästä vuota, kasvattaen sitä pikku hiljaa kunnes se on maksimaalinen.

Tehokkaampia algoritmeja saadaan käyttämällä **esivuota**.

Funktio $f: V \times V \rightarrow \mathbf{R}$ on **esivuo** jos

1. $f(u, v) \leq c(u, v)$ kaikilla $u, v \in V$,
2. $f(u, v) = -f(v, u)$ kaikilla $u, v \in V$ ja
3. $f(V, u) \geq 0$ kaikilla $v \in V - \{s\}$.

Ehto (3) siis korvaa vuolle asetettavan ehdon $f(V, u) = 0$ kaikilla $v \in V - \{s, t\}$. Vuoverkon intuitiivisessa vesijohtotulkinnassa voidaan ajatella, että esivuon tapauksessa jokaisessa solmukohtassa on varaventtiili, josta ylimääräinen vesi voidaan päästää ulos.

Solmuun u tulevaa nettovuota

$$e(u) = f(V, u)$$

sanotaan solmun u **ylivuodoksi**. Jos $u \in V - \{s, t\}$ ja $e(u) > 0$, solmu u **vuotaa yli**. Jos solmu ei vuoda yli, se on **tasapainossa**.

Esivuon voimakkuus, jäännösverkko jne. määritellään kuten vuolla.

Esivuoalgoritmien perusidea on lähteä liikkeelle mahdollisimman voimakkaasta esivuosta ja tasapainottaa solmuja, kunnes saavutetaan vuo.

Jatkossa f , r ja e esittävät vuota, jäännöskapasiteettia ja ylivuotoa, joita algoritmi ylläpitää.

Tasapainottamisen perusoperaatio on vuon **painaminen** ylivuotavasta solmusta u sellaiseen solmuun v , jolla $r[u, v] > 0$. Tällöin asetetaan

$$\begin{aligned}f[u, v] &:= f[u, v] + \Delta \\e[u] &:= e[u] - \Delta \\e[v] &:= e[v] + \Delta\end{aligned}$$

jollain $0 < \Delta < r[u, v]$.

Asetamme painamiselle joitain lisäehtoja **korkeusfunktion** h avulla, jotta jatkossa esitettävä tasapainotusalgoritmi ei juuttuisi painamaan vuota edes takaisin kahden solmun välillä.

Funktio h on laillinen korkeusfunktio, jos $h[s] = |V|$, $h[t] = 0$ ja

$$h[v] \geq h[u] - 1 \quad \text{jos } r[u, v] > 0.$$

Mille tahansa esivuolle ei ole korkeusfunktioita, ja toisaalta laillisia korkeusfunktioita voi olla useampiakin. Jatkossa esitettävän algoritmin esivuolla korkeusfunktio aina on, ja algoritmi pitää yllä erästä korkeusfunktioita h .

Vuon painaminen solmusta u solmuun v on sallittua vain, jos $h[v] = h[u] - 1$.

Painamisoperaatio ehtoineen on nyt seuraava:

Push(u, v):

Oletukset:

$$(a) e[u] > 0, u \notin \{s, t\}$$

$$(b) r[u, v] > 0$$

$$(c) h[v] = h[u] - 1$$

$$\Delta := \min \{ e[u], r[u, v] \}$$

$$f[u, v] := f[u, v] + \Delta$$

$$f[v, u] := f[v, u] - \Delta$$

$$e[u] := e[u] - \Delta$$

$$e[v] := e[v] + \Delta$$

$$r[u, v] := r[u, v] - \Delta$$

$$r[v, u] := r[v, u] + \Delta$$

Jaamme painamisoperaatiot kahteen luokkaan:

Tukkiva: $\Delta = r[u, v]$; painamisen jälkeen $r[u, v] = 0$.

Ei-tukkiva: $\Delta \neq r[u, v]$; siis $\Delta = e[u]$ ja painamisen jälkeen u on tasapainossa.

Helposti nähdään, että jos h on laillinen korkeusfunktio ennen Push-operaatiota, se on laillinen korkeusfunktio myös operaation jälkeen. (Ainoa mahdollinen uusi kaari jäännösverkossa on (v, u) , ja $h[u] = h[v] + 1 > h[v] - 1$.)

Korkeusfunktioita h muutetaan tarpeen vaatiessa **korottamalla** solmu proseduurilla Raise.

Tätä sovelletaan sellaisiin ylivuotaviin solmuihin, joista korkeusrajoituksen takia ylivuotoa ei saada pois painamalla.

Raise(u):

Oletukset:

(a) $e[u] > 0$, $u \notin \{s, t\}$

(b) $h[u] \leq h[v]$ kaikilla v joilla $r[u, v] > 0$

$h[u] := \min \{ h[v] \mid r[u, v] > 0 \} + 1$

Kun u vuotaa yli, jollain v on $r[u, v] > 0$ (koska $f[v, u] > 0$). Siis kutsun Raise(u) minimointi on epätyhjän joukon yli, ja $h[u]$ kasvaa ainakin yhdellä.

Jos $r[u, v] > 0$, niin selvästi korotuksen Raise(u) jälkeen $h[v] \geq h[u] - 1$.

Olkoon toisaalta $r[v, u] > 0$. Jos ennen korotusta h on laillinen korkeusfunktio, niin tällöin $h[u] \geq h[v] - 1$. Koska korotuksessa $h[u]$ kasvaa, tämä ominaisuus pysyy voimassa.

Siis jos h on laillinen korkeusfunktio ennen korotusta, se on sellainen myös korotuksen jälkeen.

Geneerinen painamis-korotus algoritmi

Seuraava proseduuri alustaa suurimman mahdollisen esivuon lähteestä s alkaen, mutta ei yritäkään tasapainottaa solmuja.

Alusta Esivuo:

1. Alusta esivuo kaikilla $(u, v) \in V^2$:

$$f[u, v] := \begin{cases} c(s, v) & \text{jos } u = s \\ -c(u, s) & \text{jos } v = s \\ 0 & \text{muuten.} \end{cases}$$

2. Alusta jäännöskapasiteetit

$$r[u, v] := c(u, v) - f[u, v].$$

3. Alusta ylivuodot kaikilla $v \in V - \{s\}$:

$$e[v] := \begin{cases} c(s, v) & \text{jos } c(s, v) > 0 \\ 0 & \text{muuten.} \end{cases}$$

4. Alusta $e[s] := -\sum_{v \in V} c(s, v)$.

5. Alusta korkeusfunktio

$$h[v] := \begin{cases} |V| & \text{jos } v = s \\ 0 & \text{muuten} \end{cases}$$

Huomaa, että h on laillinen korkeusfunktio, koska $r[s, v] = 0$ kaikilla v , ja siis $h[u] = h[v]$ jos $r[u, v] > 0$.

Varsinainen algoritmi tulee nyt muotoon

GeneerinenPainamisKorotus:

1. *AlustaEsivuo*
2. Toista niin kauan kuin jokin Push- tai Raise-operaatio on mahdollinen:
 - Suorita jokin Push tai Raise.

Aiempien huomioiden perusteella algoritmi säilyttää invariantin, että f on esivuo ja h laillinen korkeusfunktio.

Olkoon u ylivuotava solmu jossain algoritmin iteraatiovaiheessa. Jos $\text{Raise}(u)$ ei ole sallittu, niin $h[v] < h[u]$ ja $r[u, v] > 0$ jollain v . Koska h on laillinen korkeusfunktio, on oltava $h[v] = h[u] - 1$, ja $\text{Push}(u, v)$ on sallittu.

Siis niin kauan kuin ylitsevuotavia solmuja on, algoritmi jatkaa tasapainotusoperaatioita. Kun suoritus päättyy, f on vuo.

Pitää vielä osoittaa, että suorituksen lopuksi vuo f on maksimaalinen.

Lemma Algoritmin *GeneerinenPainamiskorotus* missään iteraatiovaiheessa esivuolla f ei ole täydennyspolkua.

Todistus Olkoon $p = (v_1, \dots, v_k)$ yksinkertainen polku jäännösverkossa, ja $v_k = t$. Koska $r[v_{i-1}, v_i] > 0$, pätee $h[v_{i-1}] \leq h[v_i] + 1$ kaikilla i , joten $h[v_1] \leq h[t] + k - 1 = k - 1 \leq |V| - 1$. Siis $v_1 \neq s$. \square

Korollaari Kun algoritmin *GeneerinenPainamiskorotus* suoritus päättyy, f on maksimivuo. \square

Algoritmin suoritusajan analysoimiseksi osoitamme, että erityyppisiä tasapainotusoperaatioita tehdään korkeintaan seuraavat lukumäärät:

korotukset	$2 V ^2$
tukkivat painamiset	$2 V E $
ei-tukkivat painamiset	$4 V ^2(V + E)$

Kun vielä osoitetaan, että painaminen onnistuu vakioajassa ja korottaminen ajassa $O(|V|)$ per operaatio, saadaan kokonaisaikavaatavuudeksi $O(|V|^2|E|)$.

Seuraava apululos kertoo, että minkä tahansa solmun ylivuoto voidaan jäljittää takaisin lähteeseen s .

Lemma Olkoon u ylivuotava solmu esivuossa f . Nyt jäännösverkossa on yksinkertainen polku $u \rightsquigarrow s$.

Todistus Olkoon $W \subseteq V$ niiden solmujen w joukko, joihin on jäännösverkossa yksinkertainen polku solmusta u .

Jos $w \in W$ ja $v \notin W$, niin jäännösverkossa on yksinkertainen polku $u \rightsquigarrow w$ mutta ei $u \rightsquigarrow v$, joten $r(w, v) = 0$. Siis $f(v, w) \leq 0$.

Laskemalla yhteen saadaan

$$\begin{aligned} e(W) &= f(V, W) \\ &= f(W, W) + f(V - W, W) \\ &= f(V - W, W) \\ &\leq 0. \end{aligned}$$

Koska $e(v) \geq 0$ kaikilla $v \neq s$, on oltava $s \in W$. \square

Lemma Algoritmin *GeneerinenPainamisKorotus* suorituksen ajan kaikilla $u \in V$ pätee $h[u] \leq 2|V| - 1$.

Todistus Aluksi väite selvästi pätee. Osoitetaan, että väite pätee, kun on juuri suoritettu $\text{Raise}(u)$.

Koska solmu u vuotaa ylitse, edellisen lemmän mukaan on olemassa polku (v_1, \dots, v_k) missä $v_1 = u$, $v_k = s$ ja $r[v_{i-1}, v_i] > 0$.

Korkeusfunktion määritelmän nojalla $h[v_i] \geq h[v_{i-1}] - 1$ ja siis $h[s] \geq h[u] - k + 1 \geq h[u] - |V| + 1$.

Koska aina $h[s] = |V|$, väite seuraa. \square

Koska korotettavissa olevia solmuja on $|V| - 2$ kappaletta, ne aloittavat korkeudesta 0 ja kukin Raise kasvattaa solmun korkeutta ainakin yhdellä, saadaan

Korollaari Korotuksia tehdään korkeintaan $(|V| - 2)(2|V| - 1) \leq 2|V|^2$ kappaletta. \square

Lemma Algoritmin *GeneerinenPainamisKorotus* aikana suoritetaan korkeintaan $2|V||E|$ tukkivaa painamista.

Todistus Kiinnitetään solmut u ja v , ja tarkastellaan yhdessä kaikkia tukkivia operaatioita $\text{Push}(u, v)$ ja $\text{Push}(v, u)$.

Oletetaan, että ensimmäinen näistä on $\text{Push}(u, v)$. Tämä tapahtuessa on oletuksen mukaan $h[v] = h[u] - 1$.

Koska kaari (u, v) poistuu jäännösverkosta, seuraavan tähän kaareen kohdistuvan operaation on oltava $\text{Push}(v, u)$. Tämän tapahtuessa on oltava $h[u] = h[v] - 1$, eli korkeus $h[v]$ on kasvanut ainakin 2 yksikköä.

Siis kahden tukkivan operaation $\text{Push}(u, v)$ välissä $h[v]$ kasvaa ainakin kahdella. Edellisen lemmän nojalla tämä voi tapahtua korkeintaan $|V| - 1$ kertaa.

Siis tukkivia Push-operaatioita tulee korkeintaan $|V|$ per suunnattu kaari (u, v) , eli yhteensä korkeintaan $2|V||E|$ kun otetaan huomioon kumpikin suunta. \square

Lemma Algoritmin *GeneerinenPainamisKorotus* aikana suoritetaan korkeintaan $4|V|^2(|V| + |E|)$ ei-tukkivaa painamista.

Todistus Määritellään potentiaali

$$\Phi = \sum_{e[u]>0} h[u].$$

Aluksi $\Phi = 0$, ja aina $\Phi \geq 0$.

Kun tehdään $\text{Raise}(u)$, niin $h[u]$ ja samalla Φ kasvaa korkeintaan $2|V| - 1$.

Kun tehdään tukkiva painaminen $\text{Push}(u, v)$, niin ainoastaan solmusta v voi tulla uusi ylivuotava solmu, ja $h[v] \leq 2|V| - 1$.

Kun tehdään ei-tukkiva painaminen $\text{Push}(u, v)$, niin termi $h[u]$ poistuu potentiaalista. Jos solmu v vuotaa yli, potentiaaliin tulee termi $h[v] = h[u] - 1$. Siis Φ pienenee ainakin yhdellä.

Koska potentiaalin kokonaismuutos on ei-negatiivinen, on siis

ei-tukkivien painamisten lkm.

$$\begin{aligned} &\leq (2|V| - 1) \cdot (\text{korotusten lkm.}) \\ &\quad + (2|V| - 1) \cdot (\text{tukkivien painamisten lkm.}) \end{aligned}$$

joten väite seuraa edellisistä tuloksista. \square

Korollaari Algoritmi *GeneerinenPainamiskorotus* löytää maksimivuon ajassa $O(|V|^2|E|)$.

Todistus Aiemmin esitetyn mukaan kun suoritus päättyy, f on vuo eikä sillä ole täydennyspolkuja. Siis f on maksimivuo.

Algoritmi voidaan toteuttaa siten, että

- $\text{Push}(u, v)$ toimii ajassa $O(1)$,
- $\text{Raise}(u)$ toimii ajassa $O(|V|)$ ja
- jokin suoritettavaksi kelpaava Push tai Raise saadaan valituksi ajassa $O(1)$

(harjoitustehtävä).

Koska Raise -operaatioita on $O(|V||E|)$ ja Push -operaatioita $O(|V|^2|E|)$, väite seuraa. \square

Päätämme verkkoalgoritmien käsittelyn osoittamalla, miten Push - ja Raise -operaatiot tarkemmin organisoimalla maksimivuo löydetään ajassa $O(|V|^3)$.