

## Korotus-eteen-algoritmi (relabel-to-front)

Tarkennamme geneeristä painamiskorotusalgoritmia kiinnittämällä tarkasti, missä järjestyksessä Push- ja Raise-operaatioita suoritetaan.

Algoritmin peruskomponentiksi tulee proseduur **Discharge**( $u$ ), joka suorittaa operaatioita Push( $u, v$ ) ja Raise( $u$ ), kunnes  $u$  on tasapainossa.

Tietysti on mahdollista, että Discharge( $u$ ) joudutaan myöhemmin uusimaan, jos jokin Push( $v, u$ ) saa sen taas epätasapainoon.

Oleelliset kysymykset ovat

- missä järjestyksessä Discharge-operaation pitää suorittaa ja
- miten yksittäisessä Discharge-operaatiossa valitaan suoritettavat Push- ja Raise-operaatiot.

Kumpaakin kysymystä varten tarvitaan hieman aputietorakenteita.

Oletetaan annetuksi esivuo  $f$  ja laillinen korkeusfunktio  $h$ .

Kaari  $(u, v)$  on **sallittu**, jos operaation  $\text{Push}(u, v)$  esiehdot ovat voimassa, ts.  $r(u, v) > 0$  ja  $h(v) = h(u) - 1$ .

Korkeusehdon takia on ilmeistä, että sallituista kaarista ei muodostu syklejä. Algoritmin perusidea onkin pitää solmut  $u \in V$  järjestetyssä listassa  $L$  siten, että jos  $(v, w)$  on sallittu niin  $v$  on listassa ennen solmua  $w$  (siis topologinen järjestys).

Algoritmin perusrunkona on iteroida seuraavaa:

1. Valitse listasta  $L$  ensimmäinen solmu  $u$ , joka vuotaa yli.
2. Tasapainota  $u$  suorittamalla Discharge.
3. Jos listan  $L$  järjestysehto meni rikki, korjaa.

Viimeisen askelen tehokasta toteuttamista varten tarkastellaan, miten Push- ja Raise-operaatiot voivat vaikuttaa sallittuihin kaariin.

**Propositio A** Oletetaan, että  $u$  vuotaa ylitse ja kaari  $(u, v)$  on sallittu. Nyt  $\text{Push}(u, v)$  ei luo uusia sallittuja kaaria, mutta saattaa tehdä kaaresta  $(u, v)$  ei-sallitun.

**Todistus** Jos painaminen on tukkiva,  $(u, v)$  muuttuu ei-sallituksi.

Koska  $\text{Push}$  ei muuta korkeusfunktioita, kaari voi operaatiossa muuttua sallituksi vain, jos vuo sitä pitkin vähenee.

Ainoa kaari, jota pitkin vuo vähenee, on  $(v, u)$ . Tämä ei kuitenkaan tule sallituksi, koska oletuksen mukaan  $h(u) = h(v) + 1 \neq h(v) - 1$ .  $\square$

**Propositio B** Oletetaan, että  $u$  vuotaa ylitse ja mikään kaari  $(u, v)$  ei ole sallittu. Nyt operaation  $\text{Raise}(u)$  esiehdot ovat voimassa. Operaation  $\text{Raise}(u)$  jälkeen ainakin jokin kaari  $(u, v)$  on sallittu, mutta mikään kaari  $(v, u)$  ei ole.

**Todistus** Algoritmin GeneerinenPainamisKorotus analyysin yhteydessä (sivu 303) todettiin, että jos  $u$  vuotaa ylitse ja operaation  $\text{Push}(u, v)$  esiehdot eivät ole voimassa millään  $v$ , niin operaation  $\text{Raise}(u)$  esiehdot ovat voimassa.

Operaatio  $\text{Raise}(u)$  asettaa  $h(u) := h(v) + 1$  missä  $v$  on eräs solmu, jolla  $r(u, v) > 0$ .  $\text{Raise}$  ei muuta vuota, joten  $r(u, v) > 0$  pysyy voimassa. Siis kaari  $(u, v)$  tulee sallituksi.

Jos toisaalta jollain  $v$  pätee  $r(v, u) > 0$ , on korkeusfunktion määritelmän mukaan  $h(u) \geq h(v) - 1$ . Koska  $\text{Raise}(u)$  kasvattaa arvoa  $h(u)$ , sen jälkeen ei voi päteä  $h(u) = h(v) - 1$ , joten kaari  $(v, u)$  ei voi olla sallittu.  $\square$

Proseduurilla  $\text{Discharge}(u)$  on seuraavat perusominaisuudet:

- vuo  $f$  ja korkeusfunktio  $h$  muuttuvat vain kutsuilla  $\text{Raise}(u)$  ja  $\text{Push}(u, v)$ ,  $v \in V$ ,
- kutsuja  $\text{Raise}(u)$  ja  $\text{Push}(u, v)$  tehdään vain, kun niiden esiehdot ovat voimassa ja
- proseduurin lopuksi  $u$  on tasapainossa.

Palataan hetken kuluttua siihen, miten tällainen  $\text{Discharge}$  toteutetaan.

Tarkastellaan ensin, miten pääalgoritmi toimii kun  $\text{Discharge}$  on annettu.

Algoritmi on seuraava:

**RelabelToFront:**

AlustaEsivuo *% kuten sivulla 302*

$L := V - \{s, t\}$  (mielivaltaisessa järjestyksessä)

$u := head[L]$

**while**  $u \neq Nil$  **do**

$prev-h := h[u]$

    Discharge( $u$ )

**if**  $h[u] \neq prev-h$  **then**

*% Discharge suoritti korotuksen*

        siirrä  $u$  listan  $L$  alkuun

$u := next[u]$

Tässä oletetaan, että  $head[L]$  osoittaa listan  $L$  alkuun ja  $next[u]$  alkiota  $u$  seuraavaan alkioon. Listan loppumerkkinä on Nil.

Seuraavaksi osoitetaan, että kun  $u = Nil$  niin kaikki solmut ovat tasapainossa.

Proseduuria Discharge koskevien oletusten nojalla tästä seuraa, että RelabelToFront on eräs totetus algoritmille GeneerinenPainamisKorotus ja siis erityisesti tuottaa maksimivuon.

Todistus perustuu siihen, että **while**-silmukalla on seuraava **invariantti**:

1. kaikki listan  $L$  solmut ennen solmua  $u$  ovat tasapainossa ja
2. jos kaari  $(v, w)$  on sallittu niin  $v$  on listassa  $L$  ennen solmua  $w$ .

Aluksi  $u$  osoittaa listan alkuun. Korkeusfunktio on vakio 0, joten sallittuja kaaria ei ole. Siis invariantti pätee triviaalisti.

Oletetaan että invariantti pätee ennen kutsua  $\text{Discharge}(u)$ .

Jos  $\text{Discharge}(u)$  ei suorita yhtään  $\text{Raise}(u)$ -operaatiota, uusia kaaria ei tule sallituksi (propositio A). Koska listan järjestyskään ei muutu, invariantin kohta 2 pysyy voimassa.

Koska vuota kasvatetaan vain sallittuja kaaria  $(u, v)$  pitkin ja oletuksen mukaan näillä  $v$  on listassa solmun  $u$  jälkeen, mikään ennen solmua  $u$  oleva solmu ei voi vuotaa yli. Siis invariantin kohta 1 pysyy voimassa.

Jos  $\text{Discharge}(u)$  suorittaa ainakin yhden korotuksen,  $u$  siirtyy listan alkuun, joten invariantin kohta 1 pysyy voimassa. Kohta 2 pysyy voimassa, koska mahdolliset uudet sallitut kaaret alkavat solmusta  $u$  (vrt. propositio B).

Siis invariantti pätee koko suorituksen ajan.

Siis proseduuri RelabelToFront löytää maksimivuon. Jäljellä on proseduurin Discharge toteutus ja aikavaativuusanalyysi.

Proseduuria Discharge varten muodostetaan kullekin solmulle  $u$  naapuruslista  $N[u]$ , jossa on kaikki ne solmut  $v$  joilla  $(u, v) \in E$  tai  $(v, u) \in E$ . Listan  $N[u]$  järjestys on mielivaltainen, mutta kiinteä koko proseduurin RelabelToFront suorituksen ajan.

Lista  $N[u]$  esitetään normaalina linkitettyinä rakenteena. Kun  $p$  on osoitin listan soluun,  $p.vertex$  on listassa sillä kohdassa oleva solmu ja  $p.next$  osoitin listan seuraavaan soluun. Listan lopussa  $p.next = \text{Nil}$ .

Jokaiselle  $u$  pidetään yllä osoitinta  $current[u]$  listaan  $N[u]$ . Aluksi  $current[u] := head[u]$ , missä  $head[u]$  on osoitin listan  $N[u]$  alkuun. Osoittimen  $current(u)$  arvo säilyy proseduurin Discharge( $u$ ) eri kutsukertojen välillä.

Proseduuri on nyt seuraava:

**Discharge( $u$ ) :**

```
    while  $e[u] > 0$  do % vuotaa yli
        if  $current[u] = Nil$  then
(1)       $Raise(u)$ 
           $current[u] := head[u]$ 
        else
           $v := current[u].vertex$ 
          if  $r[u, v] > 0$  and  $h[u] = h[v] + 1$ 
(2)      then  $Push(u, v)$ 
(3)      else  $current[u] := current[u].next$ 
```

Selvästi Push-kutsu tehdään vain, jos sen esiehto on voimassa. Analyysin ei-triviaali osa on osoittaa, että sama pätee Raise-kutsuille; palataan tähän kohta.

Aiemmin todetun mukaan jos  $u$  vuotaa yli, niin joko  $Raise(u)$  tai  $Push(u, v)$  jollekin  $v$  on sallittu. Siis  $Discharge(u)$  johtaa lopulta solmun  $u$  tasapainottamiseen.

**Lemma** Proseduurin  $\text{Discharge}(u)$  aikana kutsu  $\text{Raise}(u)$  tehdään vain, jos sen esiehdot ovat voimassa.

**Todistus** Aiemmin todetun perusteella esiehto on voimassa ainakin, jos mikään kaari  $(u, v)$  ei ole sallittu.

Kun  $\text{current}[u]$  siirtyy solmun  $v$  ohi listassa  $N[u]$ , niin kaari  $(u, v)$  ei ole sallittu. Propositioiden A ja B nojalla  $(u, v)$  voi muuttua sallituksi vain kutsulla  $\text{Raise}(u)$ , joka samalla siirtää osoittimen  $\text{current}[u]$  listan alkuun.

Siis kun  $\text{current}[u]$  saavuttaa listan lopun, kaikki kaaret  $(u, v)$  on kertaalleen todettu ei-sallituiksi, eikä mikään niistä ole sen jälkeen muuttunut sallituksi.  $\square$

Siis proseduurilla  $\text{Discharge}$  on aiemmin oletetut ominaisuudet.

Jäljellä on kokonaisaikavaativuuden analyysi.

**Lause** Algoritmin RelabelToFront kokonaisaikavaativuus on  $O(|V|^3)$ .

**Todistus** Sanotaan kutakin suorituksessa kahden Raise-operaation väliin jäävää osuutta *vaiheeksi*. Aiemmin todetun mukaan Raise-operaatioita on  $O(|V|^2)$ , joka on siis myös yläraja vaiheiden lukumäärälle.

Kunkin vaiheen sisässä osoitin  $u$  liikkuu listaa  $L$  eteenpäin, joten vaiheen aikana voidaan suorittaa vain listan pituuden verran eli  $O(|V|)$  Discharge-operaatiota.

Siis aikavaativuus lukuunotta proseduurin Discharge sisällä kulutettua aikaa on  $O(|V|^3)$ .

Proseduurin Discharge kuluttama aika selvästi määräytyy riveistä (1), (2) ja (3).

Operaation  $\text{Raise}(u)$  aikavaativuus on  $O(|N[u]|)$ . Koska kutsu  $\text{Raise}(u)$  voidaan tehdä kullakin  $u$  korkeintaan  $|V|$  kertaa, ja  $\sum_u |N[u]| = O(|E|)$ , nähdään, että kaikkien Raise-operaatioiden aikavaativuus koko suorituksen ajalta yhteensä on  $O(|V||E|)$ .

Rivi (3) voidaan annetulla  $u$  suorittaa  $|N[u]|$  kertaa suorittamatta kutsua  $\text{Raise}(u)$ . Koska taas kullakin  $u$  kutsu  $\text{Raise}(u)$  voidaan suorittaa  $O(|V|)$  kertaa, riviin (3) kuuluva kokonaisaika on sekin  $O(|V||E|)$ .

Kukin Push-operaatio vie vakioajan, ja ne jaetaan tukkiviin ja ei-tukkiviin. Ei-tukkivan operaation  $\text{Push}(u, v)$  jälkeen solmu  $u$  on tasapainossa ja  $\text{Discharge}(u)$  päättyy. Näitä operaatioita voi siis olla korkeintaan yhtä monta kuin  $\text{Discharge}$ -kutsuja, eli  $O(|V|^3)$ . Tukkivia Push-operaatioita puolestaan on aiemman analyysin mukaan  $O(|V||E|)$ .

Siis kaikkiaan aikavaativuus on  $O(|V|^3 + |V||E|) = O(|V|^3)$ .  $\square$