Kisakoodarin käsikirja

Antti Laaksonen

6. marraskuuta 2016

Sisältö

Alkusanat **v**

I Perusasiat **1**

1 Johdanto **3**

2 Aikavaativuus **15**

3 Järjestäminen **23**

4 Tietorakenteet **33**

5 Täydellinen haku **43**

6 Ahneet algoritmit **51**

7 Dynaaminen ohjelmointi **57**

8 Tasoitettu analyysi **69**

9 Välikyselyt **75**

10 Bittien käsittely **87**

II Verkkoalgoritmit **95**

11 Verkkojen perusteet **97**

12 Verkon läpikäynti **105**

13 Lyhimmät polut **111**

14 Puiden käsittely **121**

15 Virittävät puut **127**

16 Suunnatut verkot **135**

17 Vahvasti yhtenäisyys **143**

18 Puukyselyt **149**

19 Polut ja kierrokset **155**

20 Virtauslaskenta **161**

III Uusia haasteita 169

21 Lukuteoria **171**

22 Kombinatoriikka **177**

23 Matriisit **183**

24 Todennäköisyys **187**

25 Peliteoria **193**

26 Merkkijonot **199**

27 Neliöjuurialgoritmit **207**

28 Lisää segmenttipuusta **211**

29 Geometria **219**

30 Pyyhkäisyviiva **225**

Osa I

Perusasiat

Osa II

Verkkoalgoritmit

Luku 18

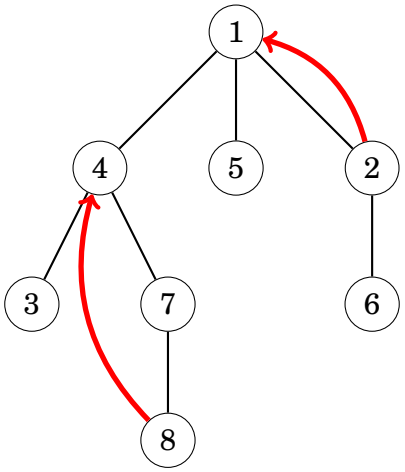
Puukyselyt

Tyypillisiä puukyselyitä ovat puun polkuihin ja alipuihin liittyvät kyselyt. Tämä luku esittelee tekniikoita, joiden avulla on mahdollista toteuttaa puukyselyjä tehokkaasti. Usein esiintyvä idea on muuttaa puu jollakin tavalla taulukoksi, mikä helpottaa puun käsittelyä.

18.1 Tehokas nouseminen

> *Tehtävä:* Annettuna on juurellinen puu, jonka solmut on numeroitu $1 \ldots n$ ja juurisolmu on 1. Tehtäväsi on vastata tehokkaasti kyselyihin muotoa ”mikä solmu on k askelta ylempänä solmua x”.

Merkitään $f(x,k)$ solmua, joka on k askelta ylempänä solmua x. Esimerkiksi seuraavassa puussa $f(2,1) = 1$ ja $f(8,2) = 4$.



Suoraviivainen tapa laskea funktion $f(x,k)$ arvo on kulkea puussa k askelta ylöspäin solmusta x alkaen. Tämän aikavaativuus on kuitenkin $O(n)$, koska on mahdollista, että puussa on ketju, jossa on $O(n)$ solmua.

Kuten luvussa 16.3.1, funktion $f(x,k)$ arvo on mahdollista laskea tehokkaasti ajassa $O(\log k)$ sopivan esikäsittelyn avulla. Ideana on laskea etukäteen kaikki arvot $f(x,k)$, joissa $k = 1,2,4,8,\ldots$ eli 2:n potenssi. Esimerkiksi yllä olevassa puussa muodostuu seuraava taulukko:

x	1	2	3	4	5	6	7	8
$f(x,1)$	0	1	4	1	1	2	4	7
$f(x,2)$	0	0	1	0	0	1	1	4
$f(x,4)$	0	0	0	0	0	0	0	0
$\ldots$								

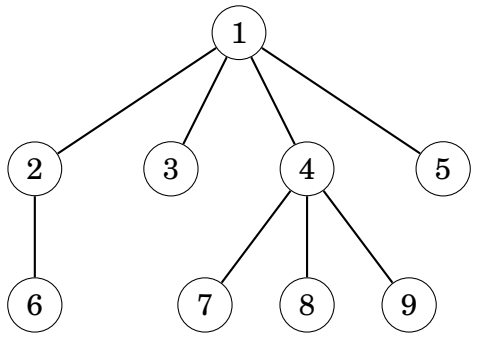
Taulukossa arvo 0 tarkoittaa, että nousemalla k askelta päätyy puun ulkopuolelle juurisolmun yläpuolelle.

Esilaskenta vie aikaa $O(n \log n)$, koska jokaisesta solmusta voi nousta korkeintaan n askelta ylöspäin. Tämän jälkeen minkä tahansa funktion $f(x,k)$ arvon saa laskettua ajassa $O(\log k)$ jakamalla nousun 2:n potenssin osiin.

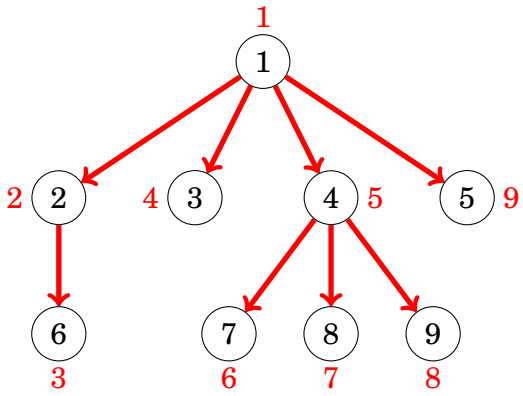
18.2 Solmutaulukko

Solmutaulukko sisältää juurellisen puun solmut siinä järjestyksessä kuin juuresta alkava syvyyshaku vierailee solmuissa.

Esimerkiksi puussa



syvyyshaku etenee



ja solmutaulukoksi tulee:

1	2	3	4	5	6	7	8	9
1	2	6	3	4	7	8	9	5

18.2.1 Alipuiden käsittely

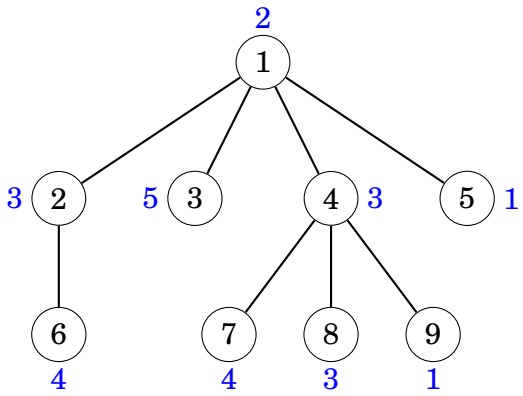
Solmutaulukossa jokaisen alipuun kaikki solmut ovat peräkkäin niin, että ensin on alipuun juurisolmu ja sitten kaikki muut alipuun solmut. Esimerkiksi äskeisessä taulukossa solmun 4 alipuuta vastaa seuraava taulukon osa:

1	2	3	4	5	6	7	8	9
1	2	6	3	4	7	8	9	5

Tämän ansiosta solmutaulukon avulla voi käsitellä tehokkaasti puun alipuihin liittyviä kyselyitä. Voimme ratkaista esimerkiksi seuraavan tehtävän:

> *Tehtävä:* Annettuna on juurellinen puu, jossa on n solmua ja jokaisella solmulla on tietty arvo. Tehtäväsi on käsitellä kyselyt muotoa ”muuta solmun x arvoa” sekä ”laske arvojen summa solmun x alipuussa”.

Tarkastellaan seuraavaa puuta, jossa siniset luvut ovat solmujen arvoja. Esimerkiksi solmun 4 alipuun arvojen summa on $3+4+3+1=11$.



Ideana on luoda solmutaulukko, joka sisältää jokaisesta solmusta kolme tietoa: (1) solmun tunnus, (2) alipuun koko ja (3) solmun arvo. Esimerkiksi yllä olevasta puusta syntyy seuraava taulukko:

1	2	3	4	5	6	7	8	9
1	2	6	3	4	7	8	9	5
9	2	1	1	4	1	1	1	1
2	3	4	5	3	4	3	1	1

Tästä taulukosta alipuun solmujen arvojen summa selviää lukemalla ensin alipuun koko ja sitten sitä vastaavat solmut. Esimerkiksi solmun 4 alipuun arvojen summa selviää näin:

1	2	3	4	5	6	7	8	9
1	2	6	3	4	7	8	9	5
9	2	1	1	4	1	1	1	1
2	3	4	5	3	4	3	1	1

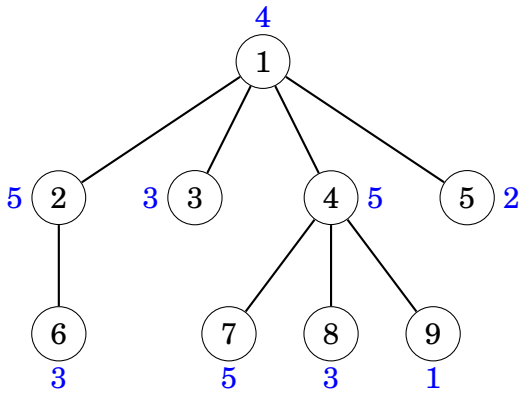
Viimeinen tarvittava askel on tallentaa solmujen arvot binääri-indeksipuuhun tai segmenttipuuhun. Tällöin sekä alipuun arvojen summan laskeminen että solmun arvon muuttaminen onnistuvat ajassa $O(\log n)$, eli pystymme vastaamaan kyselyihin tehokkaasti.

18.2.2 Polkujen käsittely

Solmutaulukon avulla voi myös käsitellä tehokkaasti polkuja, jotka kulkevat juuresta tiettyyn solmuun puussa. Näin on seuraavassa tehtävässä:

> *Tehtävä:* Annettuna on juurellinen puu, jossa on n solmua ja jokaisella solmulla on tietty arvo. Tehtäväsi on käsitellä kyselyt muotoa ”muuta solmun x arvoa” sekä ”laske arvojen summa juuresta solmuun x”.

Esimerkiksi seuraavassa puussa polulla solmusta 1 solmuun 8 arvojen summa on $4+5+3=12$.



Ideana on muodostaa samanlaiset taulukot kuin alipuiden käsittelyssä mutta tallentaa solmujen arvot erikoisella tavalla: kun taulukon kohdassa k olevan solmun arvo on a, kohdan k arvoon lisätään a ja kohdan $k+c$ arvosta vähennetään a, missä c on alipuun koko.

Esimerkiksi yllä olevaa puuta vastaa seuraava taulukko:

1	2	3	4	5	6	7	8	9	10
1	2	6	3	4	7	8	9	5	–
9	2	1	1	4	1	1	1	1	–
4	5	3	−5	2	5	−2	−2	−4	−4

Esimerkiksi solmun 3 arvona on -5, koska se on solmujen 2 ja 6 alipuiden jälkeinen solmu, mistä tulee arvoa $-5-3$, ja sen oma arvo on 3. Yhteensä solmun 3 arvo on siis $-5-3+3=-5$. Huomaa, että taulukossa on ylimääräinen kohta 10, johon on tallennettu vain juuren arvon vastaluku.

Nyt solmujen arvojen summa polulla juuresta alkaen selviää laskemalla kaikkien taulukon arvojen summa taulukon alusta solmuun asti. Esimerkiksi summa solmusta 1 solmuun 8 selviää näin:

1	2	3	4	5	6	7	8	9	10
1	2	6	3	4	7	8	9	5	–
9	2	1	1	4	1	1	1	1	–
4	5	3	−5	2	5	−2	−2	−4	−4

Summaksi tulee $4+5+3-5+2+5-2=12$, mikä vastaa polun summaa $4+5+3=12$. Tämä laskentatapa toimii, koska jokaisen solmun arvo lisätään summaan, kun se tulee vastaan syvyyshaussa, ja vähennetään summasta, kun sen käsittely päättyy.

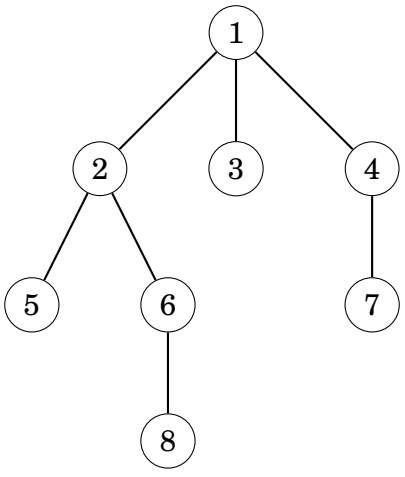
Alipuiden käsittelyä vastaavasti voimme tallentaa arvot binääri-indeksipuuhun tai segmenttipuuhun ja sekä polun summan laskeminen että arvon muuttaminen onnistuvat ajassa $O(\log n)$.

18.3 Alin yhteinen esivanhempi

Solmujen a ja b alin yhteinen esivanhempi (*lowest common ancestor*) on mahdollisimman alhaalla puussa oleva solmu, jonka alipuuhun kuuluvat molemmat solmut a ja b. Luonteva tehtävä on:

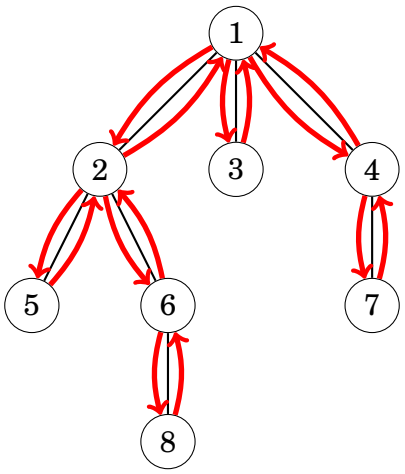
> *Tehtävä:* Annettuna on puu, jossa on n solmua. Tehtäväsi on vastata kyselyihin ”mikä on solmujen a ja b alin yhteinen esivanhempi”.

Esimerkiksi puussa



solmujen 5 ja 8 alin yhteinen esivanhempi on solmu 2 ja solmujen 3 ja 4 alin yhteinen esivanhempi on solmu 1.

Ideana on jälleen järjestää solmut syvyyshaun mukaan:



Erona aiempaan solmu lisätään kuitenkin järjestykseen mukaan *aina*, kun syvyyshaku käy solmussa, eikä vain ensimmäisellä kerralla. Niinpä solmu esiintyy järjestyksessä $x+1$ kertaa, missä x on solmun lasten määrä, ja järjestyksessä on yhteensä $2n-1$ solmua.

Tässä tehtävässä tarvittavat taulukot ovat `node`, joka sisältää solmujen tunnukset, sekä `depth`, jossa on kunkin solmun syvyys puussa. Juuren syvyys on 1, seuraavan tason solmujen syvyys on 2, jne.

Esimerkkipuuta vastaavat taulukot ovat:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
`node`	1	2	5	2	6	8	6	2	1	3	1	4	7	4	1
`depth`	1	2	3	2	3	4	3	2	1	2	1	2	3	2	1

Näiden taulukoiden avulla solmujen a ja b alin yhteinen esivanhempi selviää etsimällä taulukosta `node` kohdat x ja y, joissa `node`$[x] = a$ ja `node`$[y] = b$. Tämän jälkeen taulukon `depth` pienin arvo välillä $x \ldots y$ (tai välillä $y \ldots x$, jos $y < x$) ilmaisee solmun, joka on a:n ja b:n alin yhteinen esivanhempi. Jos on monta vaihtoehtoa valita solmut x ja y, mikä tahansa valinta kelpaa.

Esimerkiksi solmujen 5 ja 8 alin yhteinen esivanhempi löytyy seuraavasti:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
`node`	1	2	5	2	6	8	6	2	1	3	1	4	7	4	1
`depth`	1	2	3	2	3	4	3	2	1	2	1	2	3	2	1
				↑											

Taulukosta `node` selviää, että `node`[3] = 5 ja `node`[6] = 8. Taulukossa `depth` pienin arvo välillä 3…6 on `depth`[4] = 2. Niinpä kohdan 4 solmu eli solmu 2 on solmujen 5 ja 8 alin yhteinen esivanhempi.

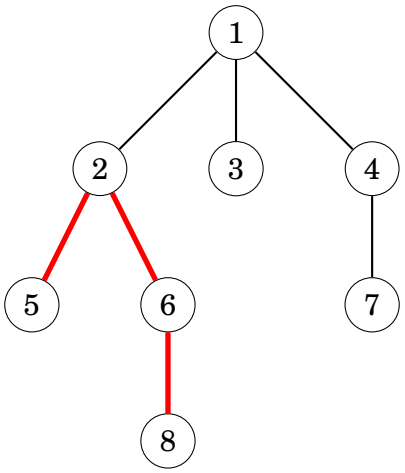
Välin pienin arvo taulukosta `depth` selviää tehokkaasti ajassa $O(\log n)$ segmenttipuun avulla. Koska taulukko ei muutu koskaan, minimin haku on mahdollista toteuttaa myös ajassa $O(1)$, mutta tälle on harvoin tarvetta.

Solmujen etäisyydet

Myös seuraava tehtävä palautuu alimman yhteisen esivanhemman hakuun:

> *Tehtävä:* Annettuna on juurellinen puu, jonka solmut on numeroitu $1 \ldots n$. Tehtäväsi on vastata kyselyihin ”laske solmujen a ja b etäisyys puussa”.

Valitaan ensin mikä tahansa solmu puun juureksi. Tämän jälkeen solmujen a ja b etäisyys on $d(a) + d(b) - 2 \cdot d(c)$, missä c on solmujen alin yhteinen esivanhempi ja $d(s)$ on etäisyys puun juuresta solmuun s. Esimerkiksi puussa



solmujen 5 ja 8 alin yhteinen esivanhempi on 2. Polku solmusta 5 solmuun 8 kulkee ensin ylöspäin solmusta 5 solmuun 2 ja sitten alaspäin solmusta 2 solmuun 8. Solmujen etäisyydet juuresta ovat $d(5) = 3$, $d(8) = 4$ ja $d(2) = 2$, joten solmujen 5 ja 8 etäisyys on $3 + 4 - 2 \cdot 2 = 3$.

Osa III

Uusia haasteita