

58131 Tietorakenteet (kevät 2007)

1. kurssikoe 26.2., ratkaisuja ja arvosteluperusteita

1. (a) **Väite:** $5 \log_2 n = O(n^3)$

Todistus: Pitää siis löytää sellaiset vakiot a ja n_0 , että $5 \log_2 n \leq an^3$, kun $n \geq n_0$. Tehtävässä annettu arvio $2^n > n$ on yhtäpitävästi $n > \log_2 n$. Valitaan $a = 5$ ja $n_0 = 1$. Kun $n \geq 1$, pätee

$$5 \log_2 n < 5n \leq 5n^3$$

eli haluttu tulos. \square

- (b) **Väite:** ei päde $5 \cdot 2^n = O(n^3)$.

Todistus: Tehdään vasta oletus, että on olemassa sellaiset a ja n_0 , että $5 \cdot 2^n \leq an^3$, kun $n \geq n_0$. Tehtäväpaperissa annetusta epäyhtälöstä saadaan $(2^m)^4 > m^4$ kaikilla $m \in \mathbf{N}$. Koska $(2^m)^4 = 2^{4m}$, pätee siis $2^{4m} > m^4$. Valitsemalla $m = n/4$ saadaan $2^n > (n/4)^4 = n^4/64$. Siis jos $n \geq 64a/5$, niin $5 \cdot 2^n > an^3$. Valitsemalla $n = \max \{ \lceil 64a/5 \rceil, n_0 \}$ saadaan ristiriita oletuksen kanssa. \square

Tehtävän arvosteli Jyrki Kivinen. Arvosteluperusteet olivat seuraavat:

- 1 p, jos oli joko vastannut oikein kumpaankin kyllä/ei-kysymykseen tai muuten osoitti ymmärtävänsä tehtävässä esiintyvien funktioiden suhteelliset kasvunopeudet
 - edellisen lisäksi 2 p, jos on esitetty O -merkinnän määritelmä (tai käytetty sitä todistuksessa)
 - edellisten lisäksi 2 p varsinaisista todistuksista.
2. (a) Yhteen suuntaan linkitetty järjestämätön lista on hyvä valinta. Haun aikavaativuus on joka tapauksessa $O(n)$. Järjestäminen lyhentää hakuajoja jonkin verran, jos avain ei ole listassa, mutta ei muuten (ellei ole syytä olettaa että esim. pieniä avaimia haetaan useammin kuin suurempia). Toisaalta järjestäminen pidentää lisäyksen vakioajasta aikaan $O(n)$, joten se ei tässä ole hyvä. Kahden suuntaan linkitys on tarpeetonta, koska emme tee poistoja. Tunnussolmuilla ei liene suurempaa merkitystä, ellei sitten listoja ole paljon (ja lyhyitä), jolloin niiden lisärasite muodostuu suhteellisen suureksi. Jätetään ne tässä ratkaisussa pois.
- (b) Allaoleva algoritmi siirtää listan S alkiot yksitellen listaan P tai Q avaimen arvon mukaan. Jonkin verran työtä säästettäisiin siirtämällä suoraan kokonaisiasia osalistojaksi, jolloin osa *next*-osoittimista voisi säilyttää vanhan arvonsa. Tällöin koodista tulee kuitenkin monimutkaisempaa ja erilaisia erikoistapauksia pitää ottaa huomioon.

```
SPLIT( $S, k, P, Q$ )
   $P \leftarrow \text{NIL}$ 
   $Q \leftarrow \text{NIL}$ 
   $r \leftarrow \text{head}[S]$ 
  while  $r \neq \text{NIL}$ 
    do  $q \leftarrow \text{next}[r]$ 
    if  $\text{key}[r] \leq k$ 
      then  $\text{next}[r] \leftarrow P$ 
            $P \leftarrow r$ 
    else  $\text{next}[r] \leftarrow Q$ 
           $Q \leftarrow r$ 
     $r \leftarrow q$ 
   $S \leftarrow \text{NIL}$ 
```

- (c) Tasapainoisella hakupuulla ei ole mitään ilmeistä tapaa toteuttaa SPLIT. Niiivi tapa edellyttää puun S avainten viemistä yksi kerrallaan puihin P ja Q , jolloin operaation aikavaativuus on $O(n \log n)$. (Itse asiassa *splay-puut* ovat hakupuuversio, joka toteuttaa mm. SPLIT-Operaation, mutta niiden

tasapainotus on melko erilaista kuin esim. punamustien puiden.) Lisäys veisi ajan $O(\log n)$ listatoteutuksen vakioajan sijaan. Lisäksi tasapainoiset puut ovat hankalia toteuttaa, ja operaatioiden vakiokertoimet ovat suuria. Hyvänä puolena on, että haku toimii ajassa $O(\log n)$ eikä $O(n)$. Hakupuuta ehkä kannattaisi käyttää, jos avaimen etsimistä on paljon suhteessa lisäyksiin ja erittäin paljon suhteessa SPLIT-operaatioihin, ja lisäksi ajansäästö on sen verran tärkeä, että toteuttamisen lisävaiva tulee maksetuksi.

Tehtävän arvosteli Ari Meriläinen. Arvosteluperusteet olivat seuraavat:

- (a) (1 piste) Vastaukseksi hyväksyttiin järjestämätön yhteen suuntaan linkitetyn lista (kuten malliratkaisussa) tai järjestetty vastaava, jos vastauksessa oli todettu sen nopeuttavan SEARCH-operaatiota.
- (b) (5 pistettä) Yhteen pisteeseen riitti jonkinlainen pseudokoodiyritelmä, joka jollain tapaa liittyi listoihin. Kaksi tai kolme pistettä sai, jos ajatus oli suunnilleen oikein, mutta toteutus ei toiminut kaikilta osin tai esitys ei vastannut ajatusta. Neljän pisteen vastaus oli toimiva, mutta ei välttämättä tehokas; täydet viisi pistettä sai ratkaisuista, jotka olivat sekä ideansa että esityksensä puolesta melko virheettömät ja vieläpä suht tehokkaat.
- (c) (1 piste) Pisteeseen sai, olipa mielipide mikä tahansa, jos se oli hyvin perusteltu, eli sisälsi jotain (useampia kuin yhden) seuraavista ajatuksista:
 - puussa haku on nopeampi ($O(\log n)$ vs. listojen $O(n)$)
 - lisäys on puuhun hitaampi, vaatii tasapainotusta (listojen aikavaativuus riippui vastaajan valitsemasta listatyypistä, puulle logaritminen)
 - puussa SPLIT on vaikea toteuttaa (aika moni olisi leikannut puusta jonkun alaoksan joukoksi P ja olisi ollut tähän tyytyväinen) ja hitaampi.

Operaatioiden aikavaativuuksien vertailu oikein tehtynä lisäsi pisteen saamisen todennäköisyyttä, virheellisyyksistä tässä suhteessa sakotettiin.

3. (a) Laskenta tapahtuu kutsumalla $\text{SIZE-COUNT}(\text{root}[T])$, missä SIZE-COUNT on seuraava:

```

SIZE-COUNT(x)
  if x ≠ NIL
  then
    size[x] ← 1
    if left[x] ≠ NIL
    then SIZE-COUNT(left[x])
       size[x] ← size[left[x]] + size[x]
    if right[x] ≠ NIL
    then SIZE-COUNT(right[x])
       size[x] ← size[right[x]] + size[x]

```

- (b) Haluttu luku saadaan kutsulla $\text{SMALLER}(\text{root}[T], k)$, missä SMALLER on seuraava:

```

SMALLER(x, k)
  if x = NIL
  then return 0
  elseif k ≤ key[x]
  then return SMALLER(left[x], k)
  elseif left[x] = NIL
  then return 1 + SMALLER(right[x], k)
  else return size[left[x]] + 1 + SMALLER(right[x], k)

```

- (c) Kun solmu lisätään, kaikissa sen esi-isissä $size$ -arvoa pitää kasvattaa yhdellä. Poistoissa vastaavasti $size$ -arvoja pienennetään. Tämä siis ennen tasapainotusoperaatioita.

Tasapainotuksen yhteydessä (proseduurit INSERT-FIXUP ja DELETE-FIXUP) puun rakenne (mihin siis ei lasketa värejä) muuttuu vain kierroissa. Siis kiertoproseduureihin pitää liittää $size$ -arvojen päivitys aina, kun jonkin solmun lapset muuttuvat.

Tehtävän arvosteli Janne Korhonen. Arvosteluperusteet olivat seuraavat:

- (a) (3 pistettä) Tämä kohta olisi osattu yleisesti aika hyvin. Joillakin ei kuitenkaan ollut rekursio hallussa, mikä yleensä johti toimimattomaan ratkaisuun.

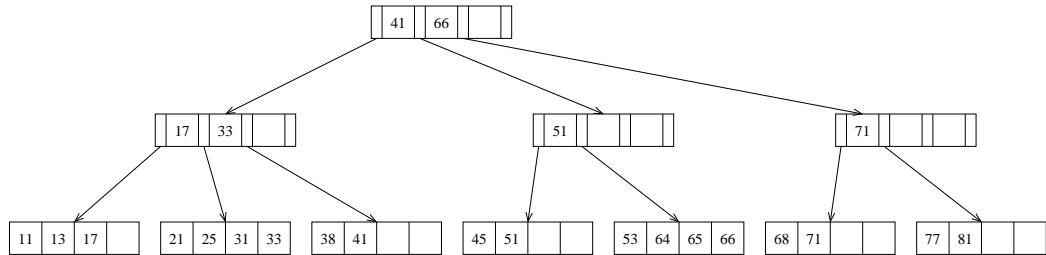
1p, jos vastauksessa on annettu *selkeästi* jokin perusajatus siitä, miten ongelman voisi ratkaista.
 2p, jos on esitetty pääpiirteissään toimiva algoritmi, jossa kuitenkin on joitain pahempia virheitä tai puutteita.
 3p, jos ratkaisussa on korkeintaan pieniä virheitä

- (b) (3 pistettä) Tehtävän toinen kohta oli ylivoimaisesti vaikein. Toimivia algoritmeja oli vastausten joukossa vain kourallinen. Useat olivat yrittäneet ratkaista tehtävän etsimällä solmun, jonka avain on k ja ottamalla sitten joko sen tai jonkin lapsen koon, Tämä ei toimi vaikka puussa sattuisikin olemaan tällainen solmu, mikä ei edes ole taattua. Tehokkuus-vaativuuden tulkittiin tarkoittavan sitä, että algoritmi hyödyntää $size$ -arvoja eikä erikseen laske solmujen määriä.

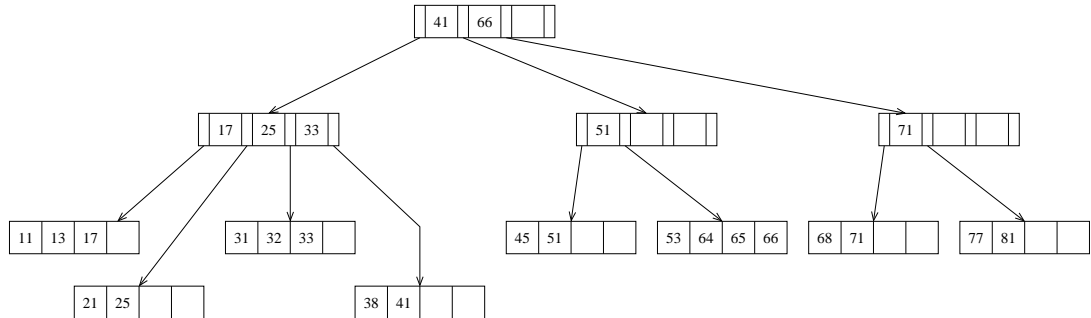
1p, jos on esitetty jokin algoritmi ongelman ratkaisemiseen ja se toimii, tai on esitetty *selkeästi* perusidea tehokkaan algoritmin toiminnasta.
 2p, jos esitetty algoritmi on tehokas. Algoritmissa voi olla jotain isompia virheitä, kunhan perusidea on näkyvissä ja toimiva.
 3p, jos algoritmissa on korkeintaan pieniä huolimattomuus- yms. virheitä.

- (c) (1 piste) Pisteen sai jos oli huomionnut jotenkin sekä varsinaisen lisäyksen tai poiston aiheuttaman muutoksen lapsien määrässä ylempänä puussa sekä kierroissa tapahtuvat muutokset. Perustapaukset olivat siis "päivitetään kaikkia esi-isiä ± 1 ja kierroissa tehdään sopivat muutokset" ja "lasketaan kaikki uudestaan".

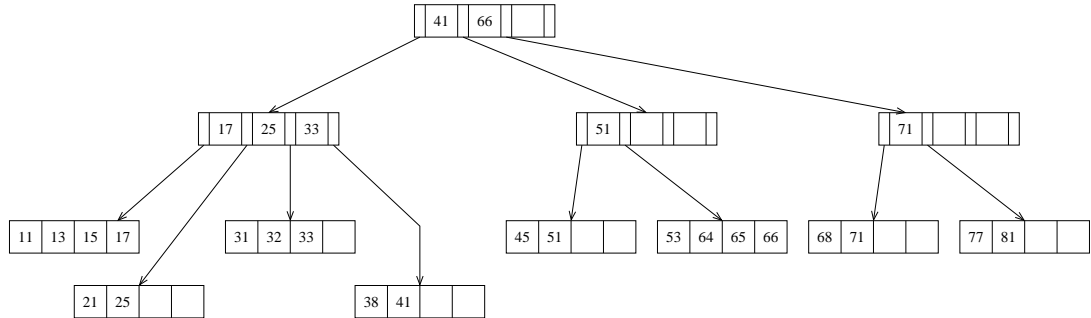
4. (a) Valitaan viitta-arvoiksi kunkin alipuun pienin avain:



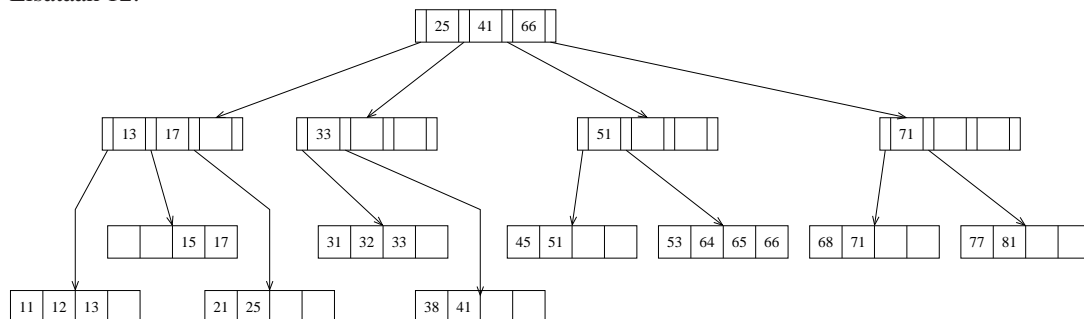
- (b) Lisätään 32:



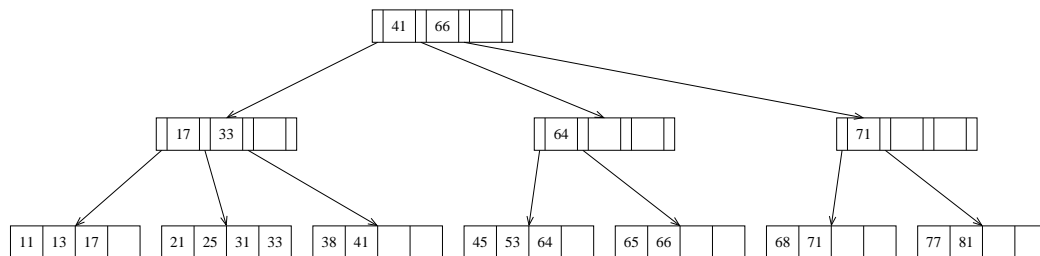
- Lisätään 15:



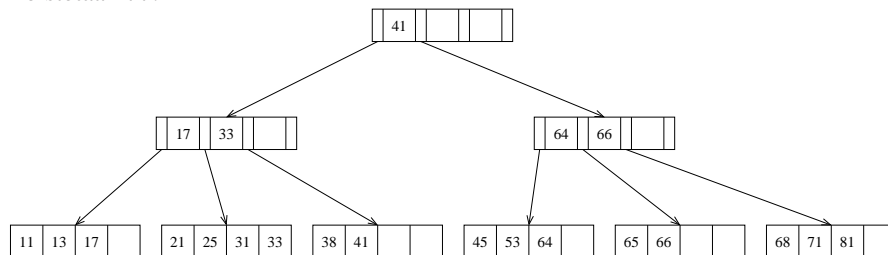
Lisätään 12:



(c) Poistetaan 51:



Poistetaan 77:



Tehtävän arvosteli Jyrki Kivinen. Arvosteluperusteet olivat seuraavat:

Kohdasta (a) on voinut saada yhden pisteen. Tyypillisin virhe oli, että solmuihin oli lisätty ylimääräinen ”suurin” viitta-arvo. Ei liene mitään syytä, miksei sellaisiakin voisi B-puuhun lisätä, mutta tällöin solmun jälkeläisten ja viitta-arvojen lukumäärät eivät täsmää kurssilla esitettyyn, mikä pitäisi algoritmeissa ottaa huomioon.

Kohdista (b) ja (c) on yhteensä voinut saada neljä pistettä. Yhden pisteen ratkaisussa on selvästi yritetty jotain oikeasuuntaista, mutta toteutuksessa ei ole päästy kovin pitkälle. Kahden pisteen ratkaisusta löytyy oikeita elementtejä (solmujen halkomista ja yhdistämistä jne.), mutta selvästi puutteellisia kohtia-kin on enemmän kuin yksi. Kolmen pisteen ratkaisussa voi olla yksi selvästi virheellinen kohta, mutta muuten menetelmiä on sovellettu oikein. Tässä on sallittu melko vapaat tulkinnat B-puun päivytysalgoritmeista niiltä osin, kuin saman asian voi järkevästi tehdä muutenkin kuin luennoilla esitetyn mukaan. Tyypillisiä selviä virheitä oli lisäyksessä avainten siirteleminen monen solmun kesken tavalla, joka ilmeisesti ei olisi tehokkaasti toteutettavissa, ja poistossa solmujen jättäminen yhdistämättä alivuotilanteessa.