6.2.2 Tasapainotus haarautumisrajoitteilla

- Kalvojen 6.1.6 haarautumisasteen $b > 2$ hakupuu voidaan tasapainottaa solmujen *lapsilukuja* säätelemällä.
- Kaikki lehdet pidetään samalla tasolla.
- Kun lisätään puuhun uusi solmu
  - niin pyritään tekemään lisäys olemassaolevaan (s)isäsolmuun
  - koska silloin puun korkeutta ei tarvitse kasvattaa.
- Jos lisäys on synnyttämässä sisäsolmun, jolla on $b + 1$ lasta
  - niin *halkaistaan* se alipuineen kahtia
  - paikoillaan pysyvään vanhaan puoliskoon
  - uuteen puoliskoon jota lisätään isään.

- Jos (aikaisemmin) täysi juuri halkeaa kahtia
  - niin puu kasvaa korkeutta
  - koska tehdään uusi juuri
  - jolla on (näin aluksi) *vain 2* lasta.
- Tavallisimmat tyypit:

  2-3-4-puu jossa $b = 4$ ja sisäsolmuilla on

  $$2, 3 \text{ tai } 4$$

  alipuuta.

  B-puu jossa $b > 4$ ja sisäsolmuilla on

  $$\left\lceil \frac{b}{2} \right\rceil \ldots b$$

  alipuuta.

  B$^+$-puu eli B-*lehti*hakupuu kalvojen 6.1.5 tapaan.

  B*-puu jonka sisäsolmut pidetäänkin

  $$\frac{2}{3} \text{ eikä } \frac{1}{2}$$

  täynnä.

6.3 Punamusta puu

- Tutustutaan tarkemmin *punamustaan* puuhun (red-black tree):
  - Kalvojen 6 mukainen binäärinen sisähakupuu.
  - Tasapainotus kalvojen 6.2.1 kierroilla.
  - Tasapainotusta varten jokaiseen solmuun s uusi kenttä s.color jolla on 2 arvoa:
    - RED eli punainen
    - BLACK eli musta.
- Etuja kalvojen 6.2.1 AVL-puihin nähden:
  - Tasapainokentäksi riittää *1* bitti.

    Ei enää niin merkittävä etu kuin ennen.
  - Tekee *vakio*määrän kiertoja yhdessä lisäyksessä tai poistossa.

    Siksi soveltuu paremmin muiden tietorakenteiden perustaksi.

6.3.1 Punamustan puun ehdot

- Punamustan puun määritelmässä on 5 osaa:

1. Jokainen solmu on joko punainen tai musta.
2. Koko puun juuri on aina musta.

   (Joissakin muunnelmissa tästä ehdosta luovutaan.)
3. Punamustissa puissa muutetaan hieman yleistä nimityskäytäntöä:
   - Tyhjiä osoittimia NULL kutsutaankin nyt *lehdiksi*.
   - Kaikkia solmuja kutsutaankin vastaavasti *sisäsolmuiksi*.

   Silloin tämä ehto saa muodon:

   Jokainen lehti (eli NULL) on musta.

- Varsinaisessa tasapainoehdossa on 2 osaa:

4. Jos solmu on itse punainen, niin sen molemmat lapset ovat mustia.

   Intuitio: *Polulla ei saa olla kahta punaista solmua peräkkäin!*

5. Jokainen polku tietystä solmusta sen jälkeläisenä olevaan lehteen sisältää saman määrän mustia solmuja.

   Tarkemmin:

   - Jokaiselle solmulle v pätee seuraava:
   - Valitaan mielivaltainen lehti l (= NULL) siitä alipuusta, jonka juuri on v.
   - Kuljetaan vastaava polku
     $$v \to \cdots \to l$$
     puussa alaspäin.
   - Matkalla kohdataan aina sama määrä mustia solmuja, valittiinpa mikä l tahansa.

- Ehdon 5 intuitio: Jokaisen alipuun kaikki *lehdet ovat yhtä kaukana juuresta, jos punaiset solmut unohdetaan* etäisyyttä laskettaessa.

- Eli: Jokaisen solmun *alipuut ovat yhtä korkeat, jos punaiset solmut unohdetaan* korkeutta laskettaessa.

  Solmun v *mustakorkeus* (black-height) määritelläänkin seuraavasti:

  $\mathrm{bh}(v)$ = mustien solmujen lukumäärä (millä tahansa) polulla solmusta v alaspäin lehteen.

  (Itse solmua v ei lasketa mukaan.)

- Ehdot 4 ja 5 yhdessä:

  $$\mathrm{bh}(v) \le \text{solmun } v \text{ korkeus} \le 2 \cdot \mathrm{bh}(v). \quad (16)$$

  Sallii monen muotoiset puut —
  tasapainotusta tarvitaan harvemmin.

  Takaa silti logaritmiset polut.

Lause 6.3.1. *Jos punamustassa puussa on n sisäsolmua (= talletettua avainta), niin sen korkeus $\le 2 \cdot \log(n+1)$.*

Todistus. Yhtälön (16) oikean puolen nojalla

$$\mathrm{bh}(\text{juuri}) \ge \text{korkeus}/2$$

mistä lauseen 6.3.2 nojalla

$$n \ge 2^{\text{korkeus}/2} - 1$$

mistä yksinkertaisin muunnoksin lopputulos

$$2 \cdot \log(n+1) \ge \text{korkeus}.$$

□

Lause 6.3.2. *Punamustan puun solmusta x alkavassa alipuussa on ainakin $2^{\mathrm{bh}(x)} - 1$ sisäsolmua.*

Todistus. Induktiolla yli solmun x *oikean* korkeuden h suhteen:

Jos $h = 0$, niin x on lehti (= NULL) ja sisäsolmuja on väitteen mukaan ainakin
$$2^{\mathrm{bh}(x)} - 1 = 2^0 - 1 = 0$$
kuten pitääkin.

Jos $h > 0$, niin isäsolmulla x on 2 lasta (nyt myös NULL on lapsi).

$$\mathrm{bh}(\text{lapsi}) = \begin{cases} \mathrm{bh}(\text{isä}) & \text{jos lapsi on punainen} \\ \mathrm{bh}(\text{isä}) - 1 & \text{jos lapsi on musta.} \end{cases}$$

Koska lapsen aito korkeus on pienempi kuin isän, induktio-oletus soveltuu solmun x lapsiin.

Solmusta x alkavassa alipuussa on silloin sisäsolmuja ainakin haluttu määrä

$$\begin{aligned} &\text{vasen alipuu} + \text{oikea alipuu} + \text{solmu itse} \\ &\ge (2^{\mathrm{bh}(x)-1} - 1) + (2^{\mathrm{bh}(x)-1} - 1) + 1 \\ &= 2^{\mathrm{bh}(x)} - 1. \end{aligned}$$

□

6.3.2 Avaimen lisäys punamustaan puuhun

- Täydennetään kalvojen 6.1.3 algoritmia palauttamaan kalvojen 6.3.1 punamustat ehdot 1–5 voimaan avaimen lisäyksen jälkeen.
- Käytetään algoritmin iteratiivista versiota.
- Algoritmin *loppuun lisätään tasapainotusvaihe:*
  - Se kulkee lisäyskohdasta takaisin kohti juurta kalvojen 6.1.2 isäosoittimia pitkin.
  - Se tasapainottaa matkalla puuta
    - vaihtamalla solmujen värejä
    - kalvojen 6.2.1.1 kierroilla.
  - Se voidaan lopettaa heti kun ehdot ovat taas kunnossa.

    Jos algoritmi olisi rekursiivinen, niin sen olisi pakko palata koko matka takaisin juureen.

- Muutokset kalvojen 6.1.3 algoritmiin:
  - Rivin 2 uusi *juuri*solmu on s.color := BLACK ehdon 2 nojalla.
  - Rivin 17 uusi *muu* solmu on s.color := RED:
    - Jos BLACK, niin ehto 5 olisi *varmasti* rikkoutunut!
    - Nyt ehto 4 *saattoi* rikkoutua:

      Jos lisätyn solmun isälläkin y.color = RED.
  - Riviksi $22\frac{1}{2}$ lisätään tasapainotusvaiheen kutsu RbInsertFixup(r, t).
    - Rikkovatko lapsi z (= alussa lisätty t) ja isä z.parent (= alussa y) ehtoa 4?
    - Siirretään rikkoumaa puussa ylöspäin, kunnes päädytään
      - juureen
      - kohtaan, josta rikkouma häviää sopivin kierroin.

- Pidetään samalla muut ehdoista 1–5 kuin 4 voimassa:
  - Ehtoja 1 ja 3 ei edes voi rikkoa.
  - Ehto 2 on vaarassa vain jos rikkouma siirtyy aina juureen saakka.
  - Ehto 5 määrää käytettävät kierrot ja värin vaihdot.
  - Ehto 4 pysyy *voimassa muualla* kuin solmun z ja sen isän välillä!
- Jos lapsen z isä z.parent.color = RED, niin isoisä z.parent.parent on
  - olemassa ehdon 2 nojalla
  - BLACK ehdon 4 nojalla.
- Jakaudutaan 3 tapaukseen lapsen z *sedän* y suhteen:

  Isoisän sen haaran joka *ei* ole isä.

procedure RbInsertFixup(
r: **juurisolmuviite**, z: **solmuosoitin**)

```
 1: while z.parent ≠ NULL and
    z.parent.color = RED do
 2:    if z.parent = z.parent.parent.left then
 3:       y := z.parent.parent.right;
 4:       if y ≠ NULL and y.color = RED then
 5:          y.color := BLACK;
 6:          z.parent.color := BLACK;
 7:          z.parent.parent.color := RED;
 8:          z := z.parent.parent
 9:       else
10:          if z = z.parent.right then
11:             z := z.parent;
12:             LeftRotate(r, z) kalvoilta 6.2.1.1
13:          end if;
14:          z.parent.color := BLACK;
15:          z.parent.parent.color := RED
16:          RightRotate(r, z.parent.parent)
             kalvoilta 6.2.1.1
             vasen/oikea-symmetrisesti
17:       end if
18:    else
19:       Kuten rivit 3–17
          vasen/oikea-symmetrisesti
20:    end if
21: end while;
22: r.color := BLACK.
```

1. Myös setä y on (olemassa ja) RED:

   - Ehto 5 säilyy voimassa, kun
     - isoisä
     - isä
     - setä

     vaihtavat värinsä päinvastaisiksi.

     Tämä tehdään riveillä 5–8.

   - Punastunut isoisä on seuraava ehdon 4 mahdollisesti rikkova kohta puussa, eli seuraava mielenkiintoinen z.

     Muita mahdollisia rikkoumakohtia ei synny.

   - Jos punastunut isoisä on

     juurisolmu niin ehto 2 rikkoutuu.

     Mutta samalla rivien 1–21 silmukka purkautuu, ja rivi 22 palauttaa sen jälleen voimaan.

     muu solmu niin ehto 2 säilyy voimassa.

2. Setä y on musta ja z isänsä *oikea* lapsi:

   - Palautuu tapaukseen 3, kun
     - vaihdetaan mielenkiinto z isään
     - kierretään (nyt mielenkiintoista) *isää vasemmalle*.
   - Ehdot 2, 4 ja 5 säilyvät voimassa, koska kierretään punaista oksaa.

   Tämä tehdään riveillä 10–12.

3. Setä y on musta ja z isänsä *vasen* lapsi:

   - Kierretään *isoisää oikealle*.
   - Ehto 5 säilyy voimassa, kun
     - korkeimmalle noussut solmu (eli solmun z entinen isä) on musta
     - sen lapset ovat (eli z itse ja entinen isoisä) ovat punaisia.
   - Ehdon 4 *rikkouma katoaa* solmusta z.

     Uusia ei tule, koska se oli ainoa ja y musta.

     Siis rivien 1–21 silmukka purkautuu.

   Tämä tehdään riveillä 14–16.

- Vertaa kalvoihin 6.2.1:

AVL-kierto	punamusta tap.
2-kertainen oikealle	2
1-kertainen vasemmalle	3

Askelten lukumäärä:

- Algoritmi toistaa rivien 1–21 silmukassa tapausta 1, joka nostaa ylöspäin mielenkiintoista kohtaa z
  2 solmua/kierros.
- Silmukka purkautuu, kun
  - kohdaksi z tulee juuri
  - kohta z katoaa tapauksessa 3.

    Tapaus 2 muunnetaan tapaukseksi 3.
- Siis

$$\begin{aligned} \text{askeleita} &= \mathcal{O}(\text{toistoja}) \\ &= \mathcal{O}(\text{puun korkeus}) \\ &= \mathcal{O}(\log(\text{avainten määrä})) \end{aligned}$$

lauseesta 6.3.1. (17)

Kiertojen lukumäärä:

kiertoja	silmukka purkautui, koska z on
0	juuri
1	tapaus 3
2	tapaus 2