Otsimispuud on ühed paljudest andmestruktuuridest, mis

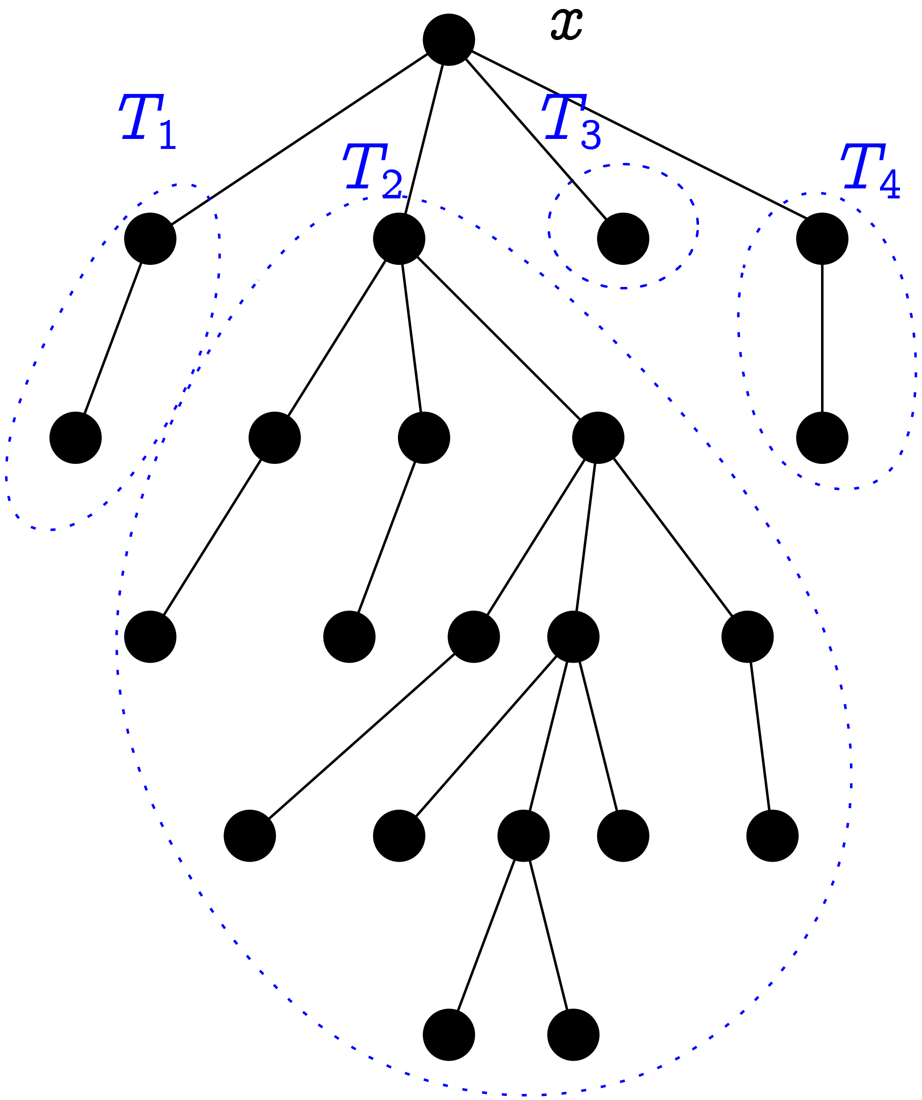
- on mõeldud mingi kirjete hulga hoidmiseks;
  - Kirjed peavad sisaldama võtit, suvalised kaks võtit peavad võrreldavad olema.
- toetab järgmisi operatsioone
  - (otsimispuu loomine,)
  - kirje lisamine puusse,
  - kirje otsimine puust võtme järgi,
  - kirje kustutamine puust.

Otsimispuid on mitut sorti, erinevates puudes on operatsioonide keerukused erinevad.

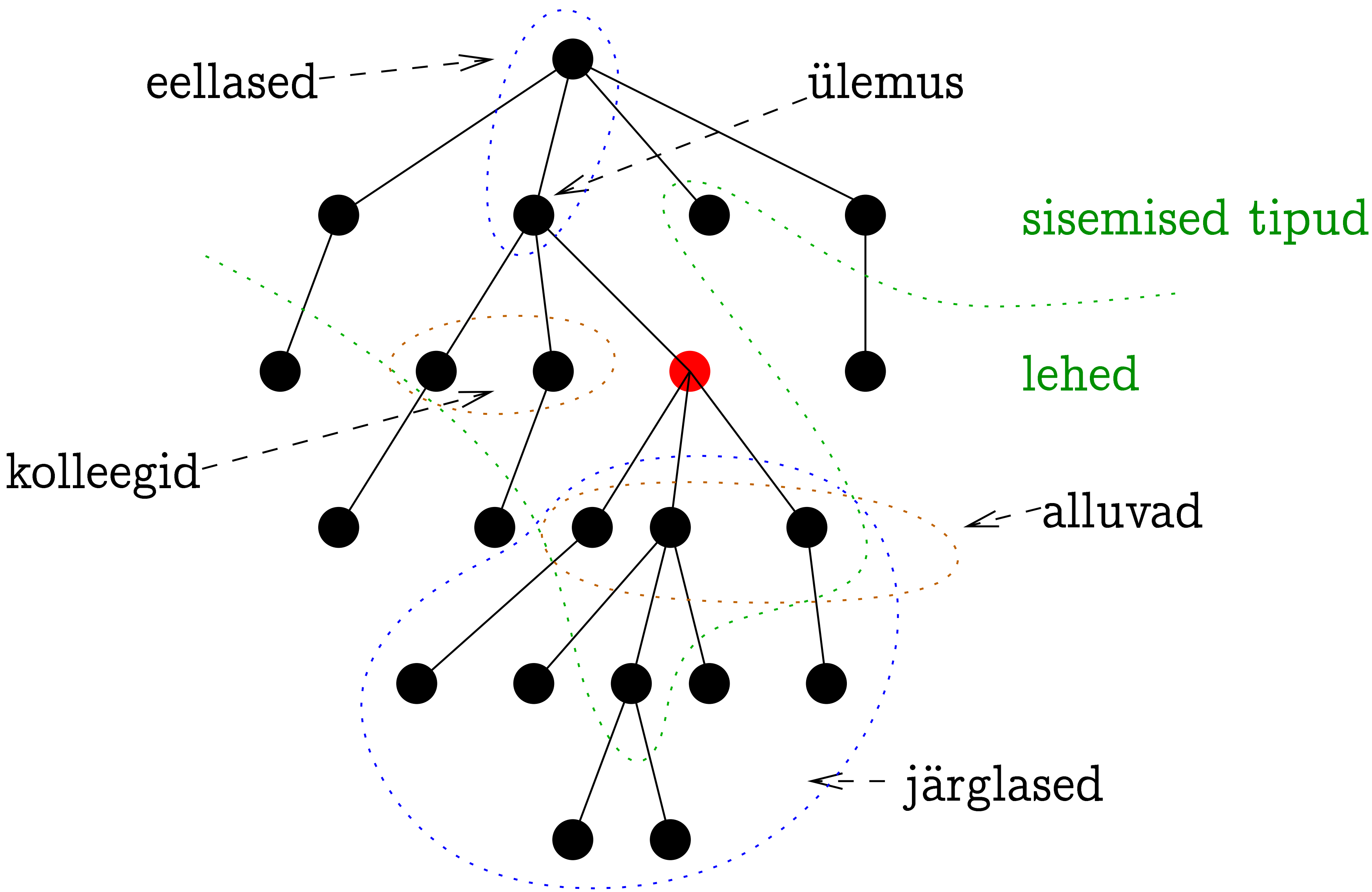
Keerukused pole üldiselt paremad kui $\Theta(\log n)$, kus n on kirjete arv puus.

(Juurega) puu T on üks järgmistest variantidest:

- tühi puu;
- tipp x ja mittetühjad juurega puud $T_1, \ldots, T_m$, kus $m \geqslant 0$.



x
T_1
T_2
T_3
T_4


eellased
ülemus
sisemised tipud
lehed
kolleegid
alluvad
järglased

Kahendpuu T on üks järgmistest variantidest:

- tühi puu;
- üksainus tipp x;
- tipp x ning kahendpuud T_{vasak} ja T_{parem}.

Kahendpuu ei ole samaväärne puuga — kui kahendpuu mingil tipul on ainult üks alluv, siis on ta kas vasak või parem alluv. Puu tipu ainsa alluva korral vasak/parem-erisust ei ole.

Lause. Mittetühjas kahendpuus, kus ühegi tipu aste ei ole 1, on lehti ühe võrra rohkem kui sisemisi tippe.

Tõestus. Induktsiooniga üle kahendpuu struktuuri.

Ühetipulises puus olev ainus tipp on leht.

Kui puus T on juurtipp x ja selle alluvatest koosnevad mittetühjad kahendpuud T_v ja T_p, siis olgu l_v ja l_p lehtede arvud nendes. Puus T on siis lehti $l_v + l_p$ ja sisemisi tippe

$$\begin{aligned} & l_v - 1 + l_p - 1 + && \text{(puudes } T_v \text{ ja } T_p \text{ vastavalt induktsioonile)} \\ & + 1 = && \text{(tipp } x\text{)} \\ & = l_v + l_p - 1 \quad . \end{aligned}$$

Kahendpuud võime arvuti põhimälus kujutada struktuurina, kus igale tipule vastab kirje

Andmed	
.vasak	*.parem*

kus väli *.vasak* viitab vasakule ning *.parem* paremale alluvale.

Teatavad algoritmid võivad nõuda täiendavate väljade olemasolu kas osa „Andmed“ sees või väljaspool seda.

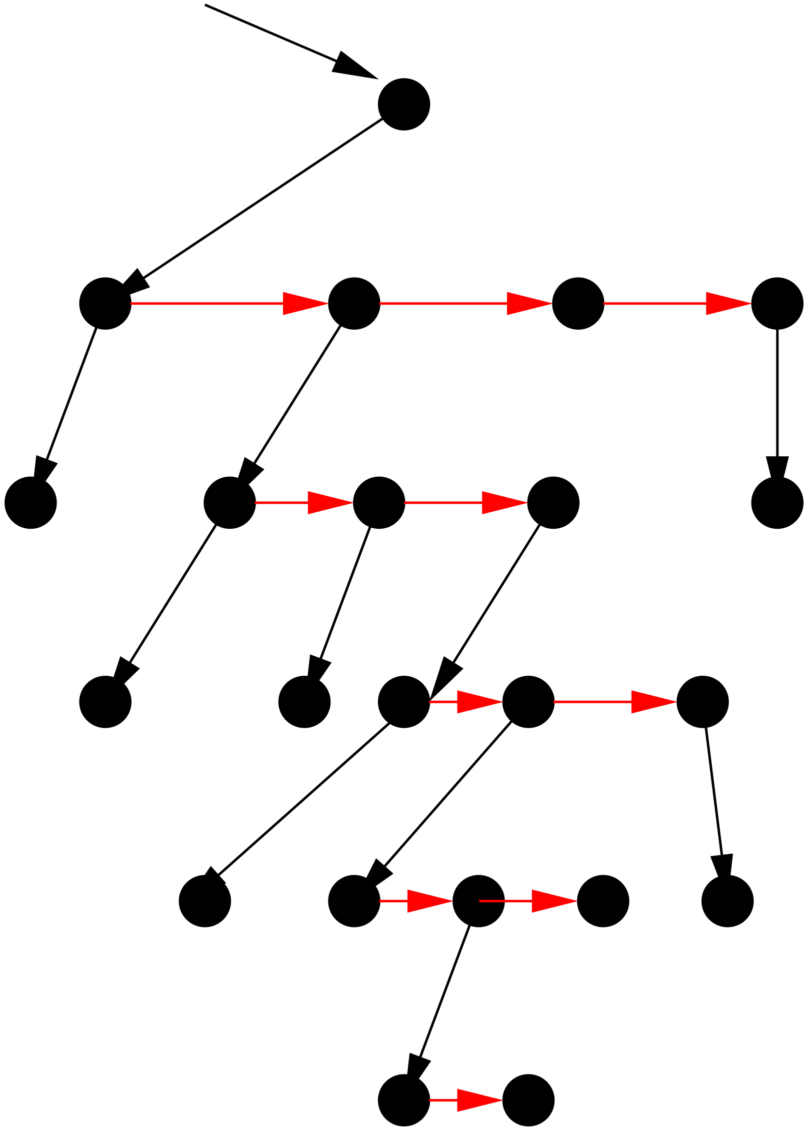
(kahendpuu naturaalesitus)

Puud võime arvuti põhimälus kujutada struktuurina, kus igale tipule vastab kirje

<table>
<tr><td colspan="2">Andmed</td></tr>
<tr><td>.alluv</td><td>.kolleeg</td></tr>
</table>

kus väli *.alluv* viitab vasakpoolseimale alluvale ning *.kolleeg* paremalt järgmisele kolleegile.

(puu naturaalesitus)



viidad: alluv / kolleeg

puuduvad viidad ≡ NIL

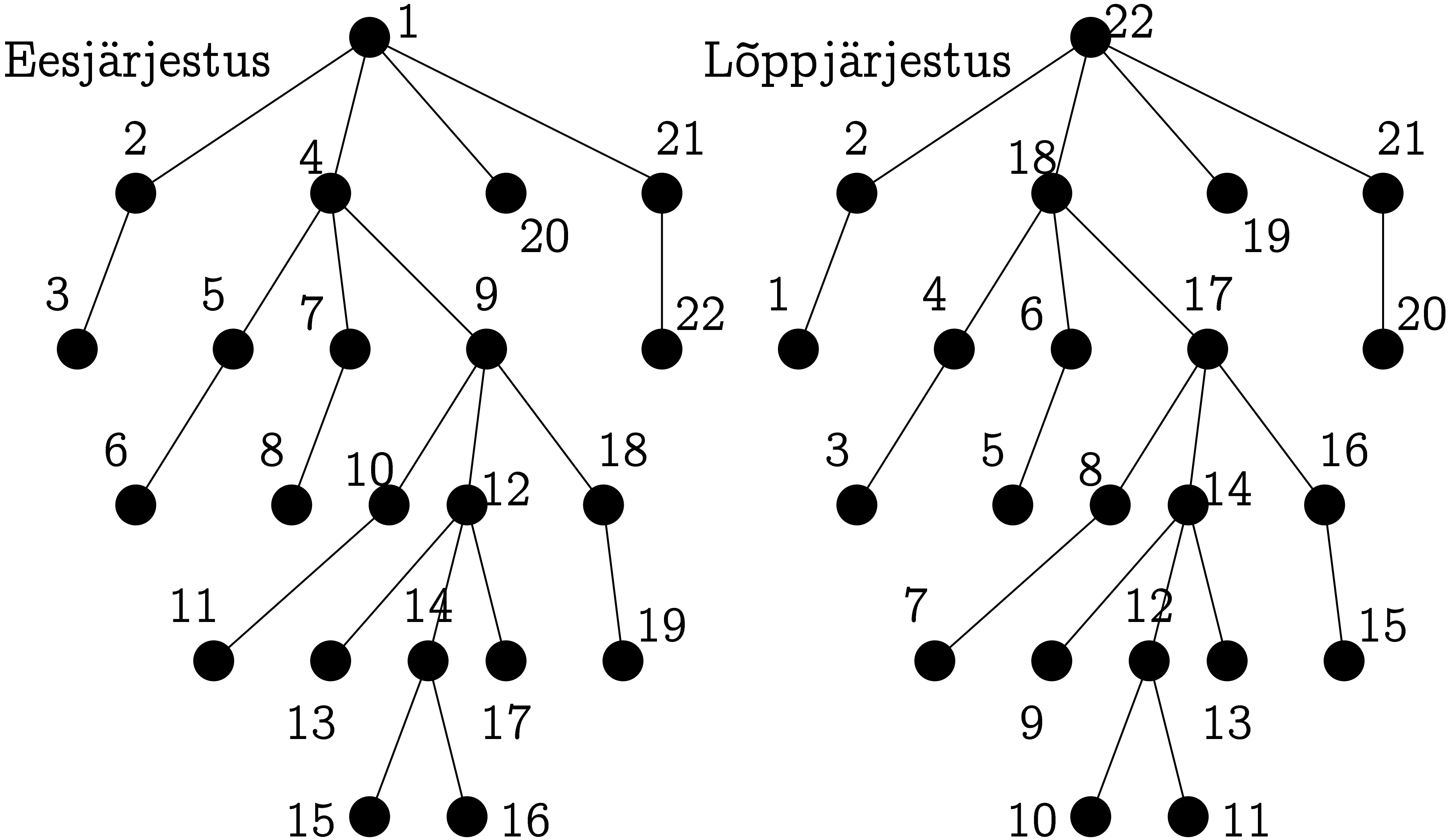
Kui loeme alluv≡vasak ja kolleeg≡parem, siis saame puu naturaalesitusele vastava kahendpuu.

Olgu meil antud mingi protseduur P, mida me puu igal tipul rakendada tahame.

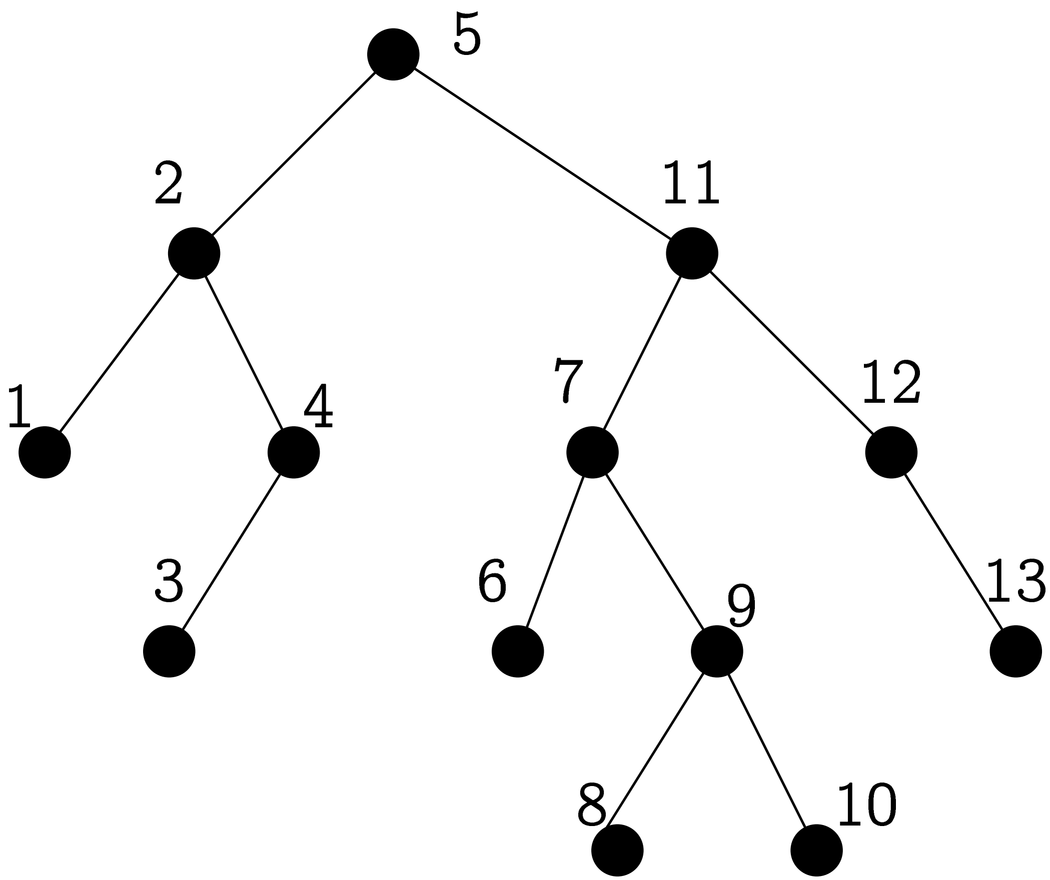
Mis järjekorras rakendada?

Järjestust, kus kõigepealt on puu juur ja seejärel samal viisil järjestatult juure vahetute alampuude tipud, nimetatakse *eesjärjestuseks*.

Järjestust, kus kõigepealt on samal viisil järjestatult juure vahetute alampuude tipud ja lõpuks puu juur, nimetatakse *lõppjärjestuseks*.

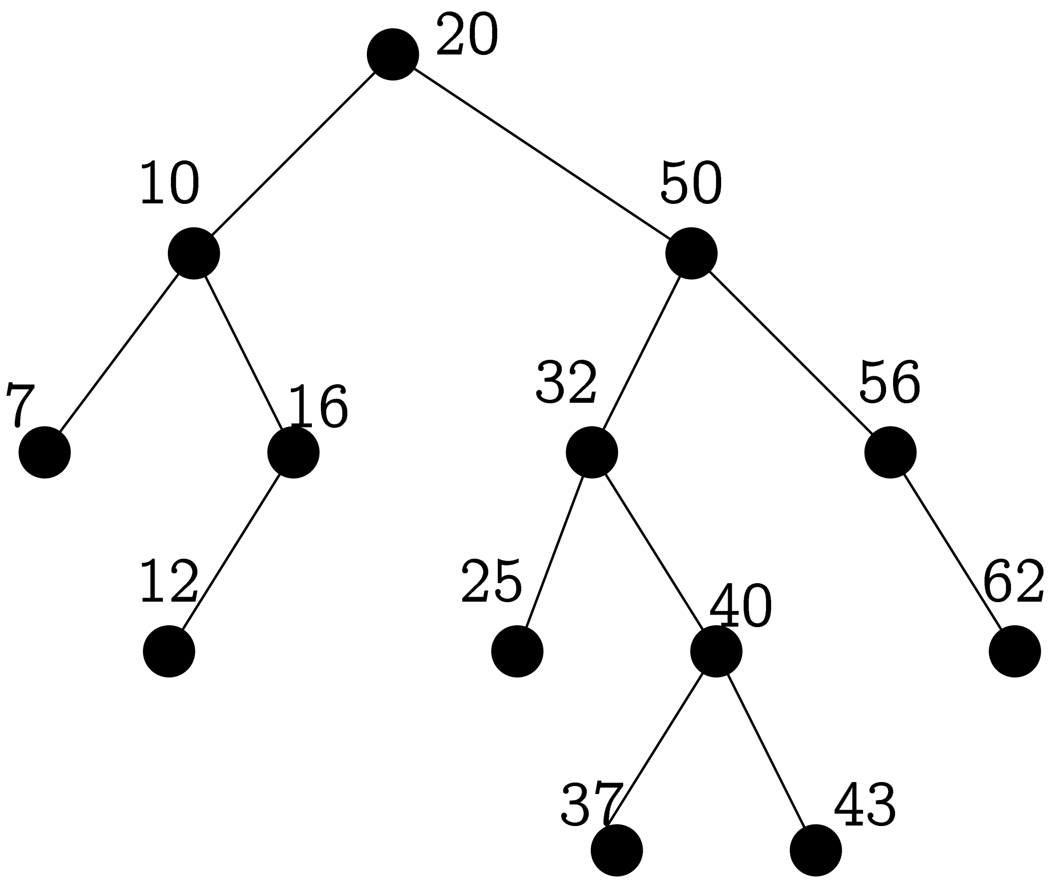

Eesjärjestus
1
2 4 20 21
3 5 7 9 22
6 8 10 12 18
11 13 14 17 19
15 16
Lõppjärjestus
22
2 18 19 21
1 4 6 17 20
3 5 8 14 16
7 9 12 13 15
10 11

Kahendpuu tippude järjestust, kus kõigepealt on samal viisil järjestatult juure vasaku vahetu alampuu tipud, seejärel puu juur ja lõpuks samal viisil järjestatult juure vahetu parema alampuu tipud, nimetatakse *keskjärjestuseks*.


5
2
11
1
4
7
12
3
6
9
13
8
10

Kahendotsimise puu on kahendpuu, kus igal tipul on täiendav väli $.võti$ (kuulub ossa „Andmed“), mille väärtustel on defineeritud täielik järjestus, ning kui puu ei ole tühi, siis

- juure välja $.võti$ väärtus pole väiksem kui tema vasaku alampuu mistahes tipu välja $.võti$ väärtus;
- juure välja $.võti$ väärtus pole suurem kui tema parema alampuu mistahes tipu välja $.võti$ väärtus;
- juure vasak ja parem alampuu on mõlemad kahendotsimise puud.


20
10
50
7
16
32
56
12
25
40
62
37
43

Lause. Võttes kahendotsimise puu tipud keskjärjestuses, on nende väljade *.võti* väärtused mittekahanevas järjekorras.

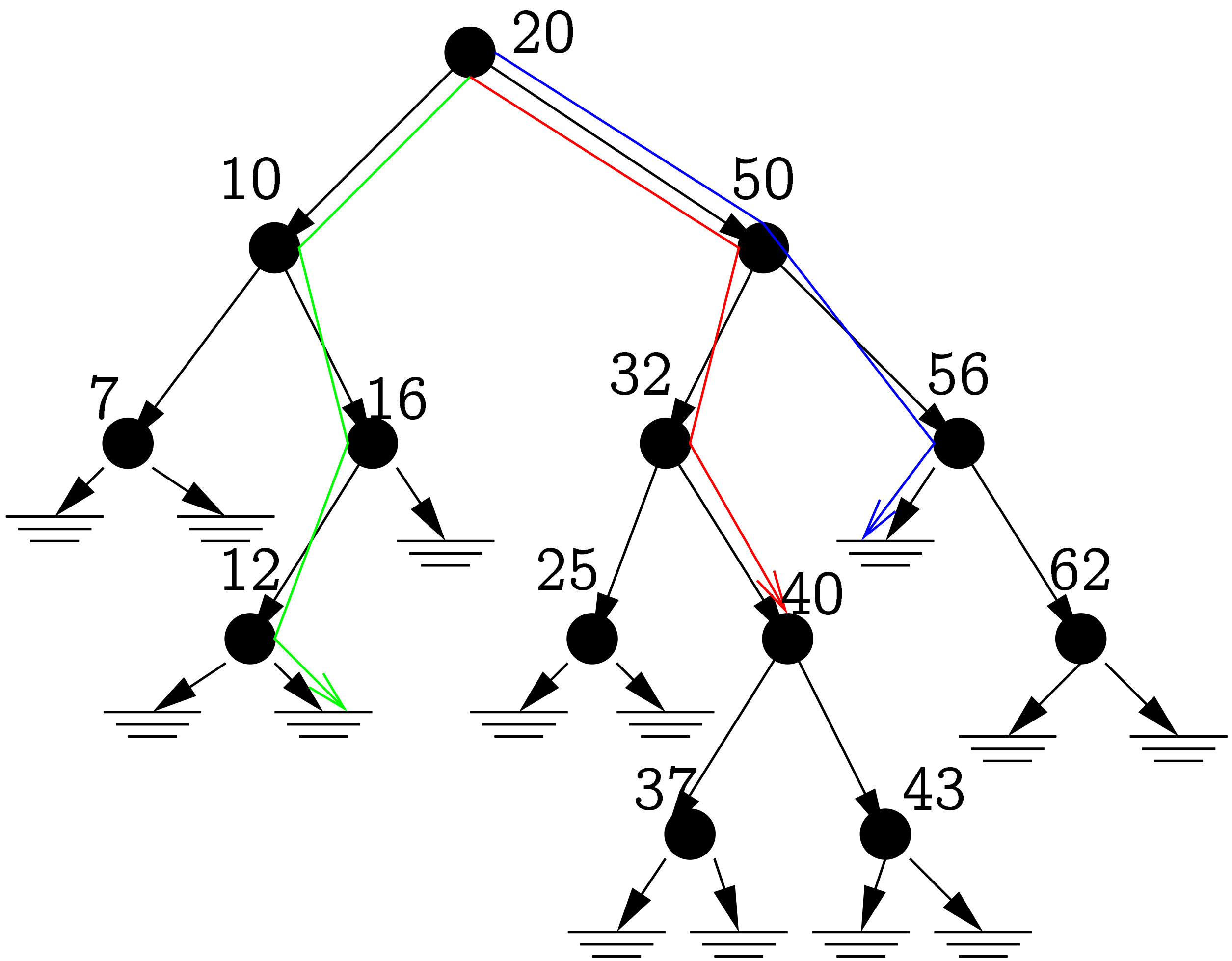
Tõestus. Induktsiooniga üle puu struktuuri. (vt. Kiho konspektist)

Kahendotsimise puust etteantud võtmeväärtusega tipu otsimine. Olgu p viit puu juurtipule ning a otsitav võtmeväärtus. Kui tippu ei leita, siis tagastatakse NIL.

Algoritm: otsi(p, a), kus $otsi(p, x)$ on

```
if p = NIL then
    return NIL
if p.võti = x then
    return p
if x < p.võti then
    return otsi(p.vasak, x)
else                          --- x > p.võti
    return otsi(p.parem, x)
```

Otsimisteed, kui otsitavaks võtmeks on 15, 40 või 52.

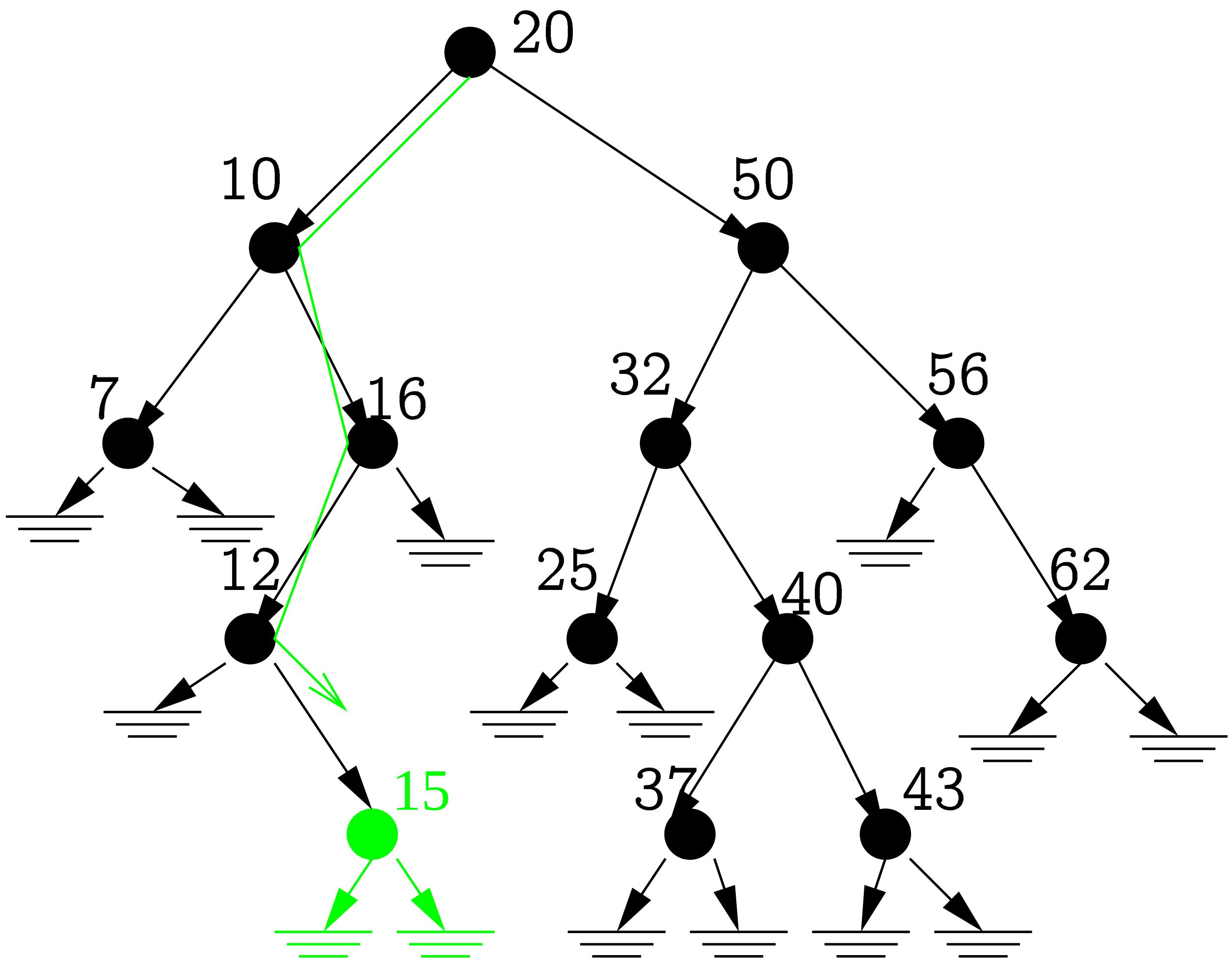


Uue kirje (samade väljadega kui osas „Andmed“) lisamine kahendotsimise puusse. Olgu p viit puu juurtipule ja R lisatav kirje. Tagastatakse lisatud tipp.

Algoritm: lisa(p, R), kus lisa(p, R) on

```
if R.x < p.x then
    if p.vasak = NIL then
        q := new(tipukirje); q.Andmed := R; p.vasak := q
        return q
    else
        return lisa(p.vasak, R)
else
    if p.parem = NIL then
        q := new(tipukirje); q.Andmed := R; p.parem := q
        return q
    else
        return lisa(p.parem, R)
```

Tipu võtmeväärtusega 15 lisamine kahendotsimise puusse.



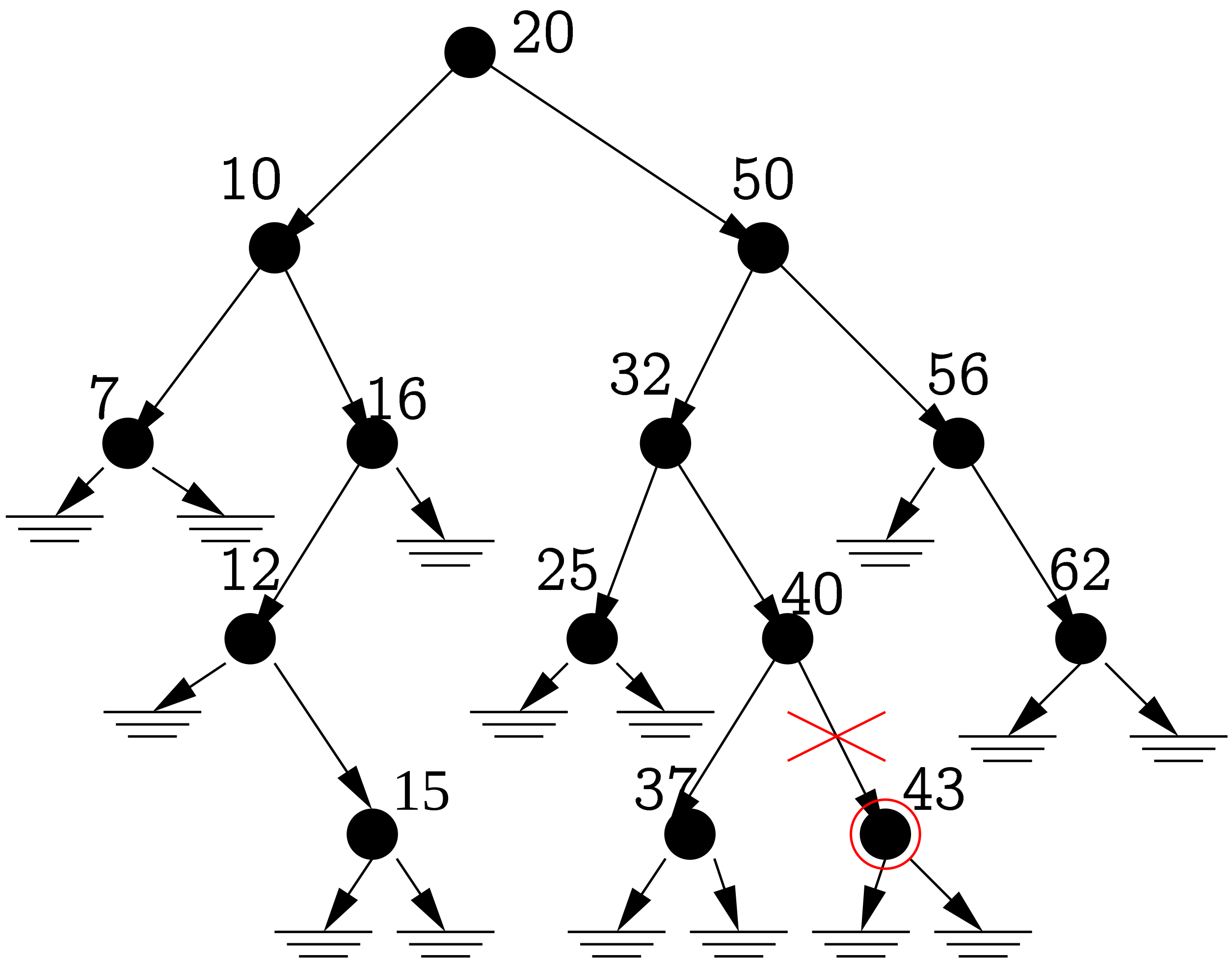
Tipu eemaldamine kahendotsimise puust.

Olgu tippudel väli $.\ddot{u}lem$ (ei kuulu ossa „Andmed“), mis viitab käesoleva tipu ülemusele.

Lisamisalgoritmis tuleb siis uue tipu väli $.\ddot{u}lem$ ka initsialiseerida, s.t. 3. ja 9. rida tuleb täiendada omistamisega $q.\ddot{u}lem := p$.

Olgu p viit vaadeldava kahendpuu juurtipule ja q viit eemaldatavale tipule. Algoritm eemalda(p, q) kustutab puust kirje q.Andmed. Ta tagastab väärtuste paari: viida muudetud puu juurtipule ning viida tegelikult eemaldatud tipule.

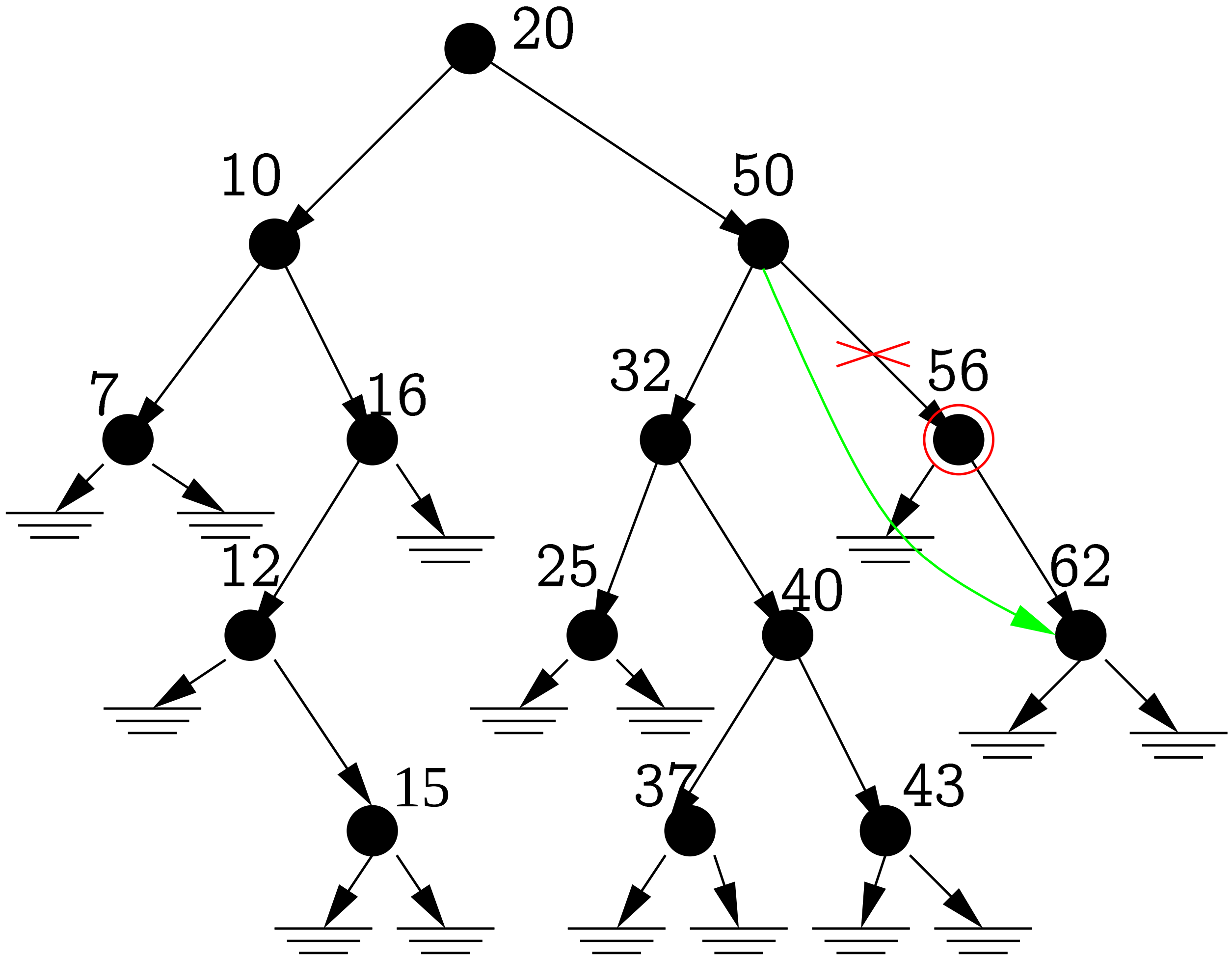
Kui tipul q pole alluvaid, siis lihtsalt kustutame ta.



eemalda(p, q) on

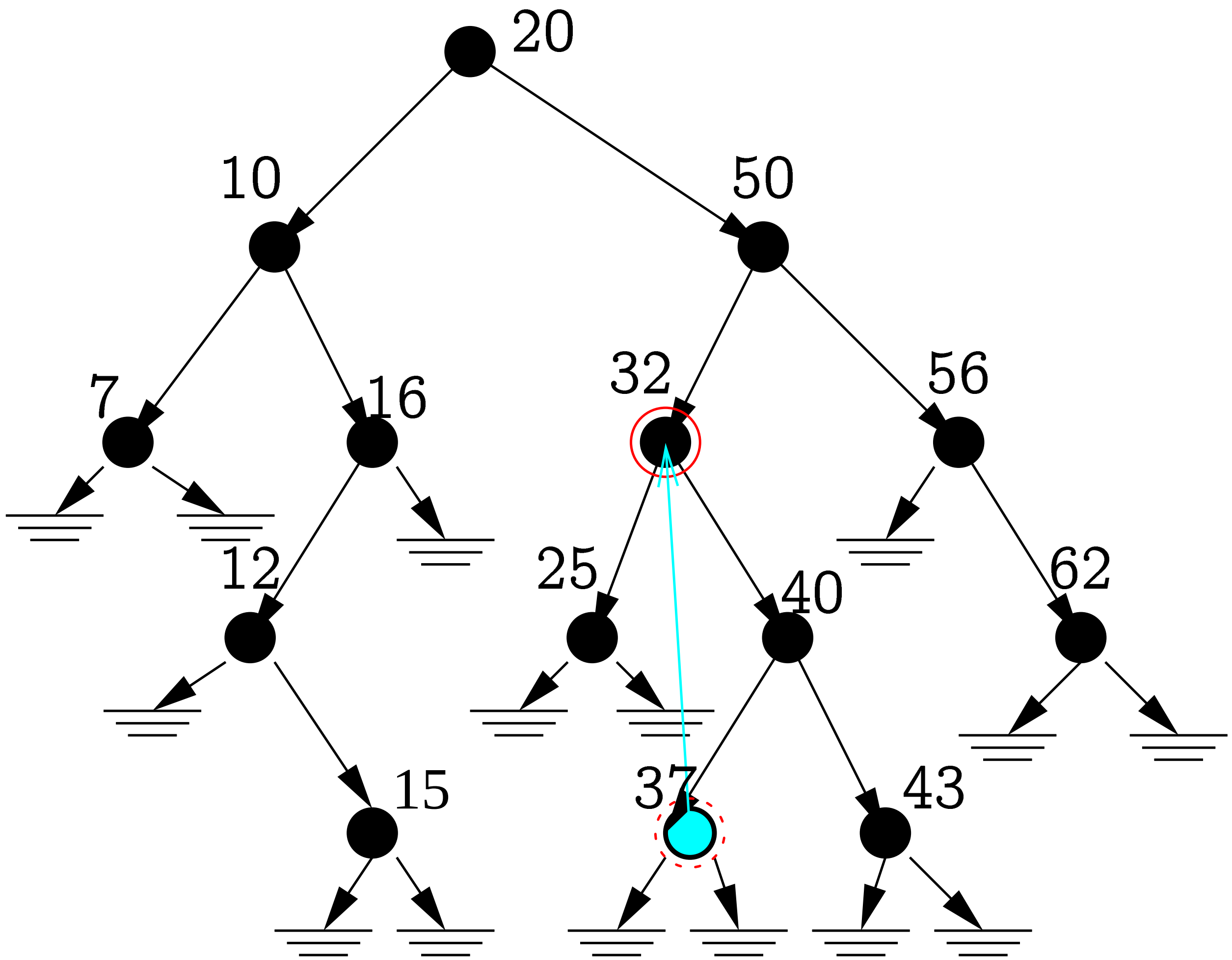
```
if q.vasak = NIL and q.parem = NIL then
    if q.ülem = NIL
        return (NIL, q)
    else if q.ülem.vasak = q
        q.ülem.vasak := NIL
    else
        q.ülem.parem := NIL
    return (p, q)
...
```

Kui tipul q on üks alluv, siis paneme q ülemuse viitama sellele alluvale.



```
else if (q.vasak = NIL and q.parem ≠ NIL)
      or (q.vasak ≠ NIL and q.parem = NIL) then
    r := q.vasak = NIL ? q.parem : q.vasak
    if q.ülem = NIL
        return (r, q)
    else if q.ülem.vasak = q
        q.ülem.vasak := r
    else
        q.ülem.parem := r
    return (p, q)
...
```

Kui tipul q on mõlemad alluvad, siis leiame q-st keskjärjestuses järgmise tipu r, kopeerime tema andmed tippu q ja kustutame hoopis r-i.



Keskjärjestuses järgmine tipp: parempoolse alampuu vasakpoolseim tipp.

```
else
    r := q.parem
    while r.vasak ≠ NIL do
        r := r.vasak
    q.Andmed := r.Andmed
    return eemalda(p, r).
```

Kahendotsimise puu operatsioonide ajaline keerukus on $\Theta(h)$, kus h on puu kõrgus.

Puus kõrgusega h on vähemalt h ja ülimalt $2^h - 1$ tippu.

Seega on kahendpuu operatsioonide ajaline keerukus halvimal juhul $O(n)$, kus n on tippude arv.

Tahaksime keerukust $O(\log n)$. Selleks tuleb puud peale muutmist tasakaalustada.

AVL-puu on kahendotsimise puu, kus iga tipu vasaku ja parema alampuu kõrgus erineb ülimalt ühe võrra.

Lühend „AVL“ tuleb autorite nimedest.

Olgu A_h minimaalne tippude arv AVL-puus kõrgusega h. Siis $A_1 = 1$, $A_2 = 2$ ja $A_h = A_{h-1} + A_{h-2} + 1$.

Fibonacci arvud: $F_0 = 0$, $F_1 = 1$, $F_h = F_{h-1} + F_{h-2}$. Seega $A_h = F_{h+2} - 1$. (tõestus induktsiooniga üle h)

Kuna $F_h = \left[\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^h\right]$, kus $[\cdot]$ tähistab ümardamist täisarvuni, siis $A_h = 2^{\Theta(h)}$ ja AVL-puu kõrgus on alati $\Theta(\log n)$, kus n on tippude arv.

AVL-puus on tippudel täiendav väli *.kõrgus* (ei kuulu ossa „Andmed“), kuhu salvestatakse sellest tipust algava alampuu kõrgus.

Peale tipu lisamist või eemaldamist tuleb nende väljade väärtusi parandada.

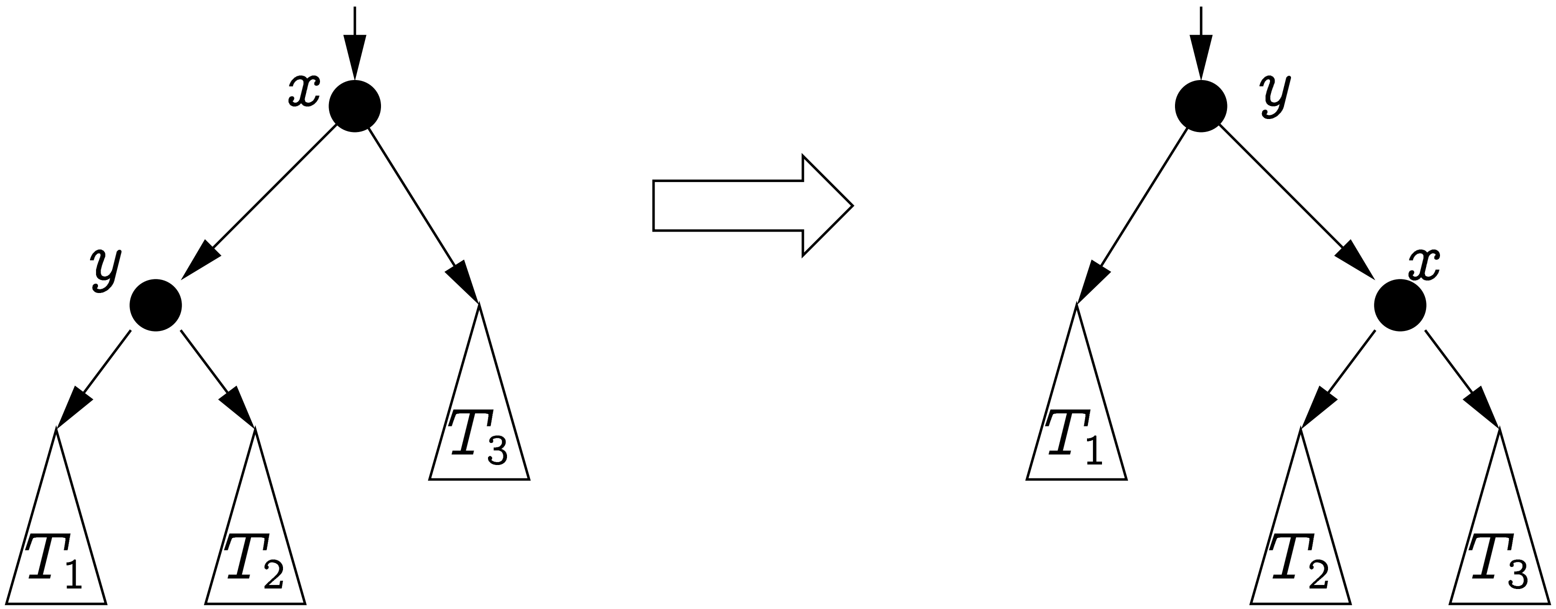
Peale lisamist võivad valed kõrgused olla tippudel, mis asuvad ahelal lisatud tipust juureni.

Peale eemaldamist võivad valed kõrgused olla tippudel, mis asuvad ahelal eemaldatud tipu (endisest) ülemusest juureni.

Kõrguste parandamine toimub samaaegselt puu tasakaalustamisega.

Tasakaalustamisoperatsioonid: pööre_vasakule ja pööre_paremale.

pööre_paremale(p, x), kus p on viit puu juurtipule ja x viit mingile tipule, mille korral $x.vasak \neq$ NIL, muudab alampuud, mille juureks on x, järgmiselt:

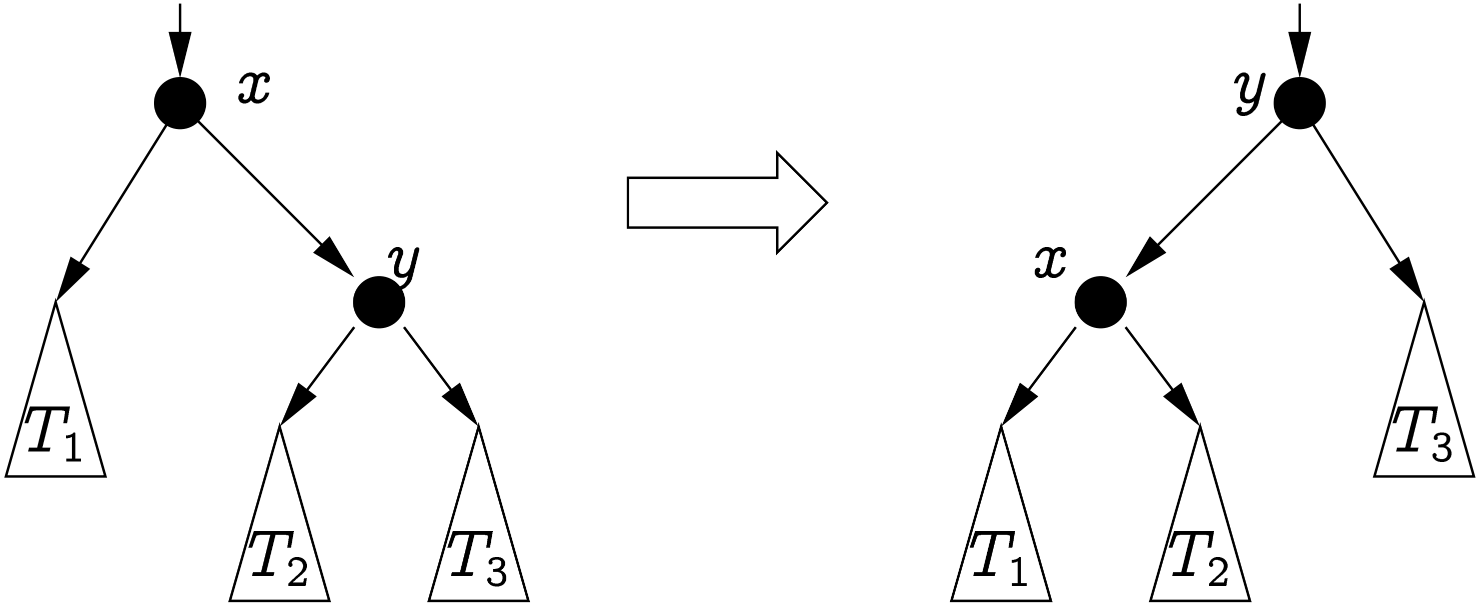


pööre_paremale tagastab muudetud puu juurtipu.

pööre_paremale(p, x) on

$y := x.vasak$; $u := x.\ddot{u}lem$
$T_1 := y.vasak$; $T_2 := y.parem$; $T_3 := x.parem$
$y.vasak := T_1$; $y.parem := x$; $x.vasak := T_2$; $x.parem := T_3$
if $u = \mathrm{NIL}$ **then**
    return y
else
    $\big(u.vasak = x \;?\; u.vasak : u.parem\big) := y$
    return p

pööre_vasakule on vastupidine operatsioon.



pööre_vasakule tagastab muudetud puu juurtipu.

pööre_vasakule(p, x) on

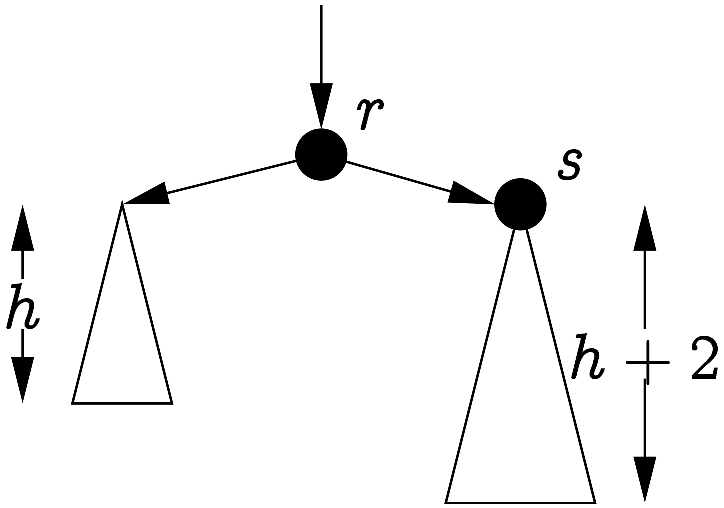
$y := x.parem$; $u := x.ülem$
$T_1 := x.vasak$; $T_2 := y.vasak$; $T_3 := y.parem$
$y.vasak := x$; $y.parem := T_3$; $x.vasak := T_1$; $x.parem := T_2$
if $u = \mathrm{NIL}$ **then**
    return y
else
    $\big(u.vasak = x \;?\; u.vasak : u.parem\big) := y$
    return p

Tipu lisamisel AVL-puusse teeme kõigepealt tavalise lisamise ja hakkame lisatud tipust alates kõrgusi parandama, kuni jõuame tasakaalustamata tipuni. . .

lisaAVL(p, R), mis tagastab paari uuest puu juurest ja lisatud tipust, on

$q := \mathrm{lisa}(p, R)$
$r := q$
while $r \neq \mathrm{NIL}$ **do**
  $h_v := r.vasak = \mathrm{NIL} \ ?\ 0 : r.vasak.k\tilde{o}rgus$
  $h_p := r.parem = \mathrm{NIL} \ ?\ 0 : r.parem.k\tilde{o}rgus$
  if $|h_v - h_p| = 2$ **then break**
  $r.k\tilde{o}rgus := \max(h_v, h_p) + 1$
  $r := r.\ddot{u}lem$
if $r = \mathrm{NIL}$ **then return** (p, q)
  . . .

r viitab tasakaalustamata tipule. Vaatame juhtu, kus tema parempoolne alampuu on kõrgem kui vasakpoolne (teistpidine juht on sümmeetriline).

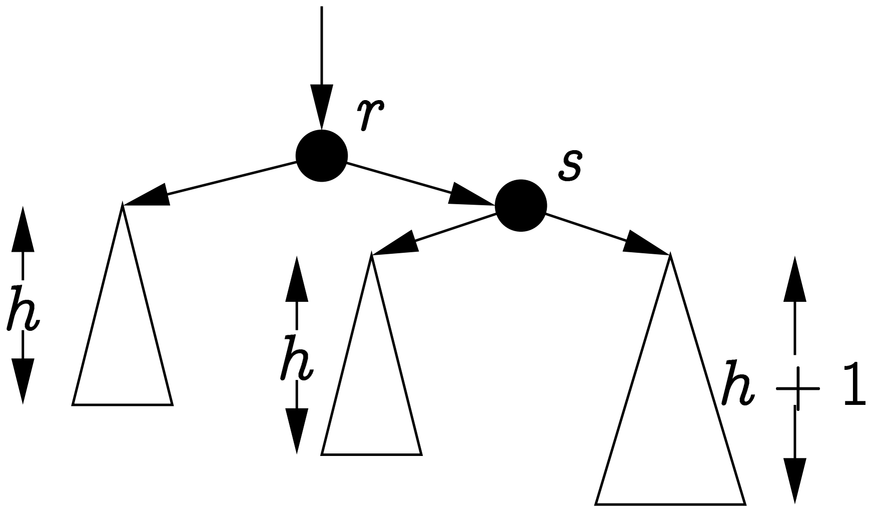


r-i kõrgus: $h + 3$

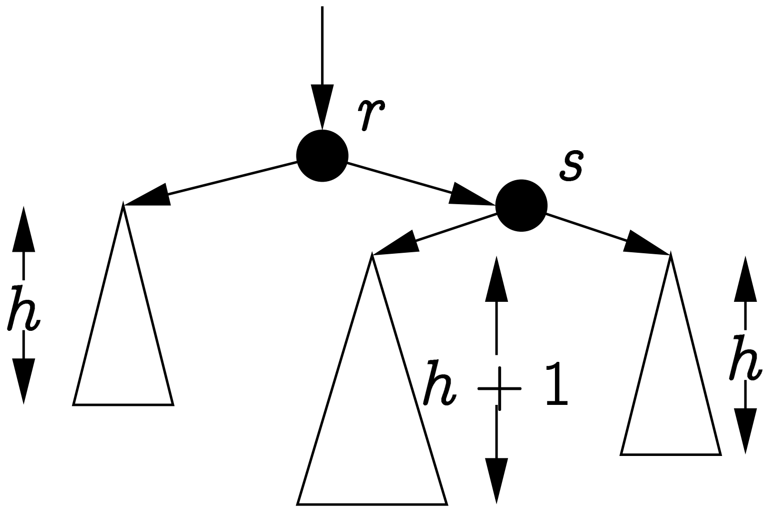
(enne lisamist: $h + 2$)

```
if h_p > h_v then
    s := r.parem
    ...
```

Uus tipp lisati paremasse alampuusse. Kui s-i vasak ja parem alampuu oleks peale lisamist võrdse kõrgusega, siis oleks üks alampuudest juba enne lisamist selle kõrgusega olnud ja s-i kõrgus poleks muutunud. Seega oleks r juba enne lisamist tasakaalustamata olnud. Järelikult on kaks varianti:



1. variant



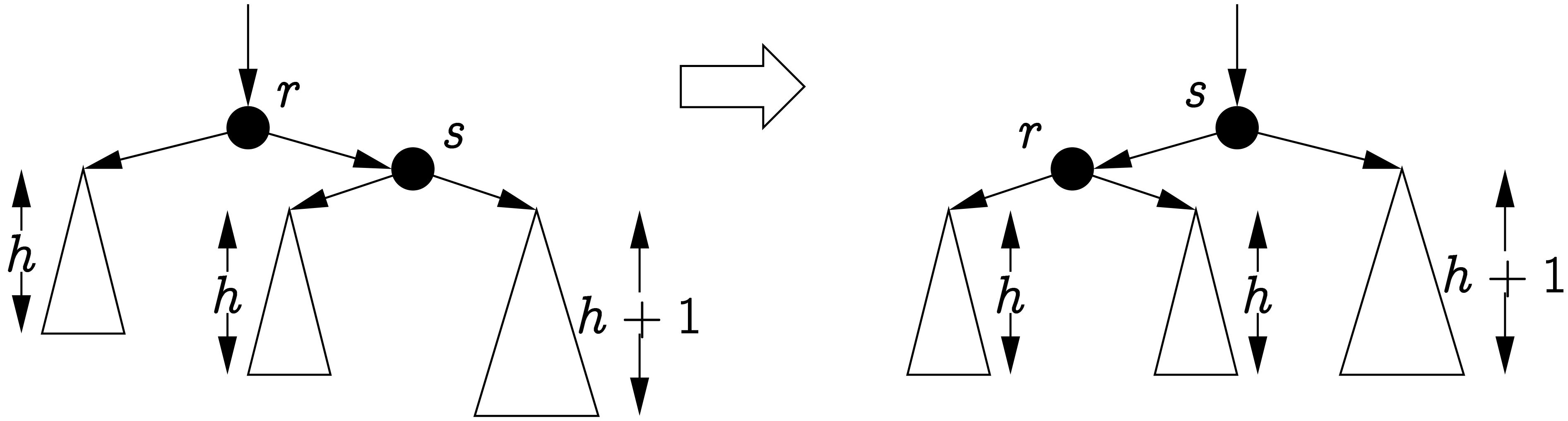
2. variant

$\qquad h_{sv} := s.vasak = \text{NIL}\ ?\ 0 : s.vasak.k\tilde{o}rgus$

$\qquad h_{sp} := s.parem = \text{NIL}\ ?\ 0 : s.parem.k\tilde{o}rgus$

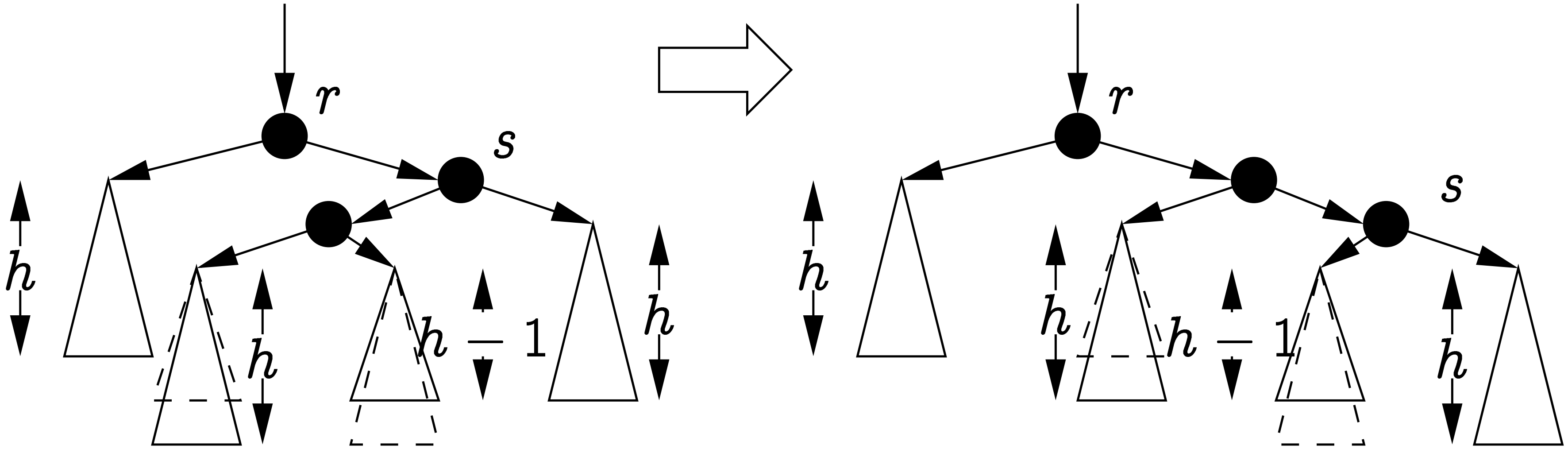
$\qquad \ldots$

1. variandil pöörame r-i vasakule.

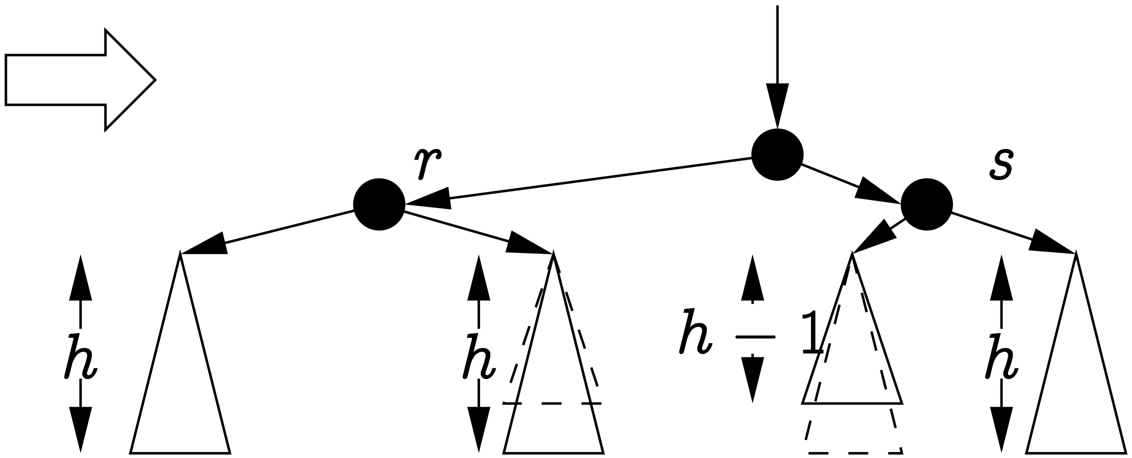


Kogu vaadeldava alampuu kõrgus on nüüd $h + 2$ — sama, mis enne uue tipu lisamist. Kõrgemalasuvate tippude kõrgused seega ei muutunud — puu on tasakaalus.

2. variandil pöörame s-i paremale



saame 1. variandi (s.t. pöörame r-i vasakule).



$\quad\quad$ **if** $h_{sv} > h_{sp}$ **then**

$\quad\quad\quad p :=$ pööra_paremale(p, s); $s.k\tilde{o}rgus := h_v + 1$

$\quad\quad p :=$ pööra_vasakule(p, r)

$\quad\quad r.k\tilde{o}rgus := h_v + 1$; $r.\ddot{u}lem.k\tilde{o}rgus := h_v + 2$

$\quad\quad$ **return** (p, q)

$\quad$ **else** --- $h_v > h_p$

$\quad\quad s := r.vasak$

$\quad\quad h_{sv} := s.vasak = \mathrm{NIL}\ ?\ 0 : s.vasak.k\tilde{o}rgus$

$\quad\quad h_{sp} := s.parem = \mathrm{NIL}\ ?\ 0 : s.parem.k\tilde{o}rgus$

$\quad\quad$ **if** $h_{sv} < h_{sp}$ **then**

$\quad\quad\quad p :=$ pööra_vasakule(p, s); $s.k\tilde{o}rgus := h_p + 1$

$\quad\quad p :=$ pööra_paremale(p, r)

$\quad\quad r.k\tilde{o}rgus := h_p + 1$; $r.\ddot{u}lem.k\tilde{o}rgus := h_p + 2$

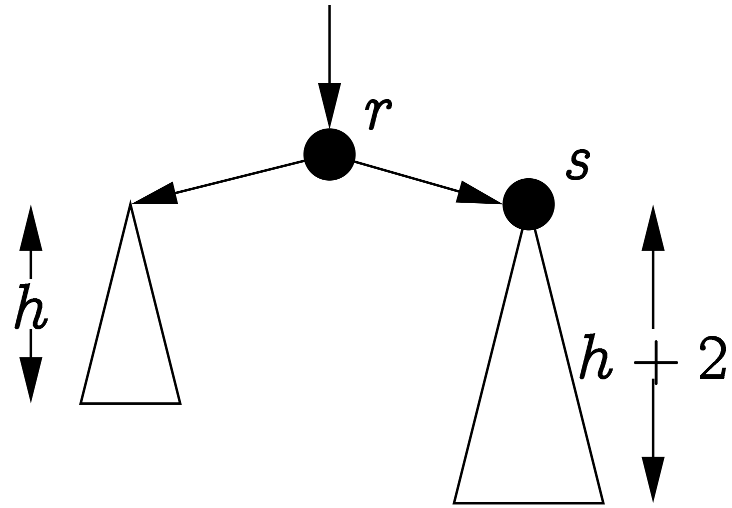
$\quad\quad$ **return** (p, q)

Tipu eemaldamisel AVL-puust teeme kõigepealt tavalise eemaldamise, seejärel teeme tasakaalustamisi teel eemaldatud tipust juureni.

eemaldaAVL(p, q) on

$(p, q) :=$ eemalda(p, q)
$r := q.\ddot{u}lem$
while $r \neq \text{NIL}$ **do**
    $h_v := r.vasak = \text{NIL}\ ?\ 0 : r.vasak.k\tilde{o}rgus$
    $h_p := r.parem = \text{NIL}\ ?\ 0 : r.parem.k\tilde{o}rgus$
    if $|h_v - h_p| = 2$ **then break**
    $r.k\tilde{o}rgus := \max(h_v, h_p) + 1$
    $r := r.\ddot{u}lem$
if $r = \text{NIL}$ **then return** (p, q)
$\ldots$

r viitab tasakaalustamata tipule. Vaatame juhtu, kus tema parempoolne alampuu on kõrgem kui vasakpoolne (teistpidine juht on sümmeetriline).



r-i kõrgus: $h+3$

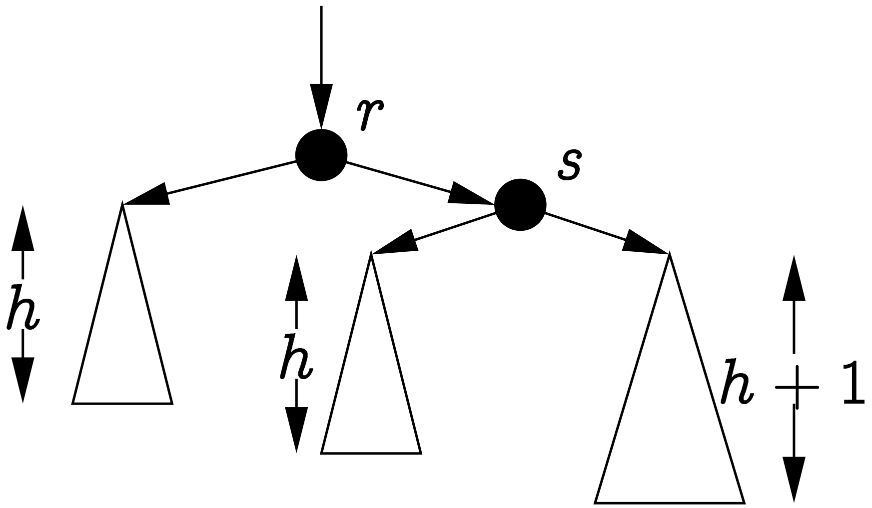
(enne eemaldamist: $h+3$)

if $h_p > h_v$ **then**

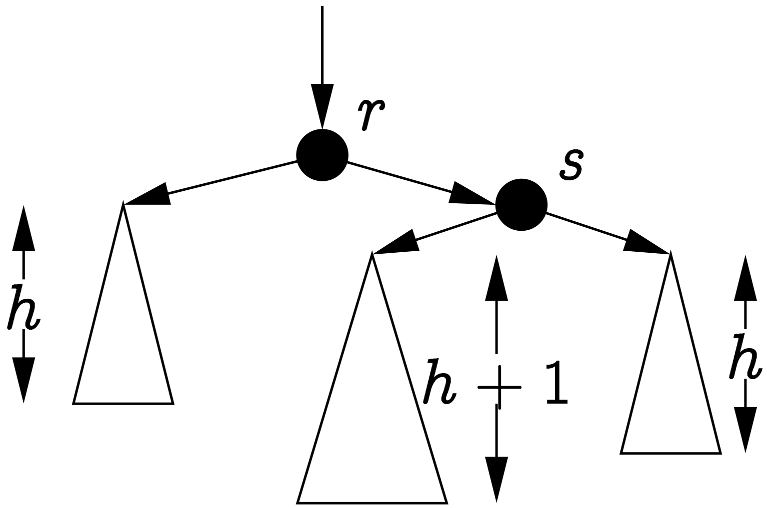
    $s := r.parem$

$\ldots$

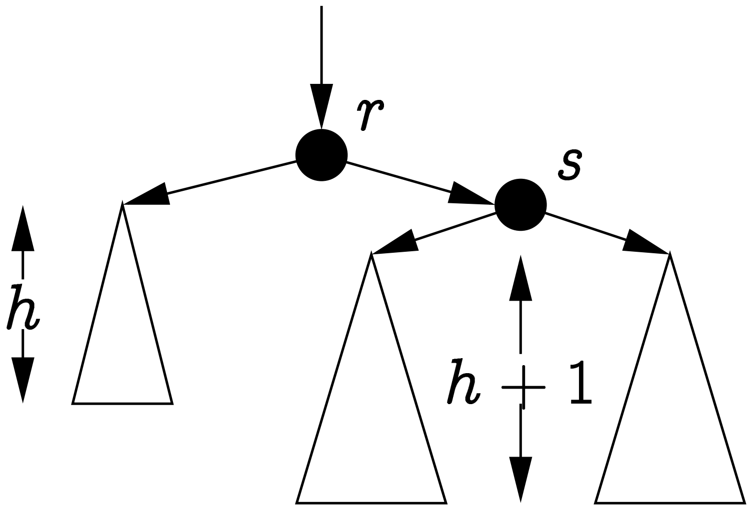
Tipp eemaldati vasakust alampuust. Kolm varianti:



1. variant



2. variant

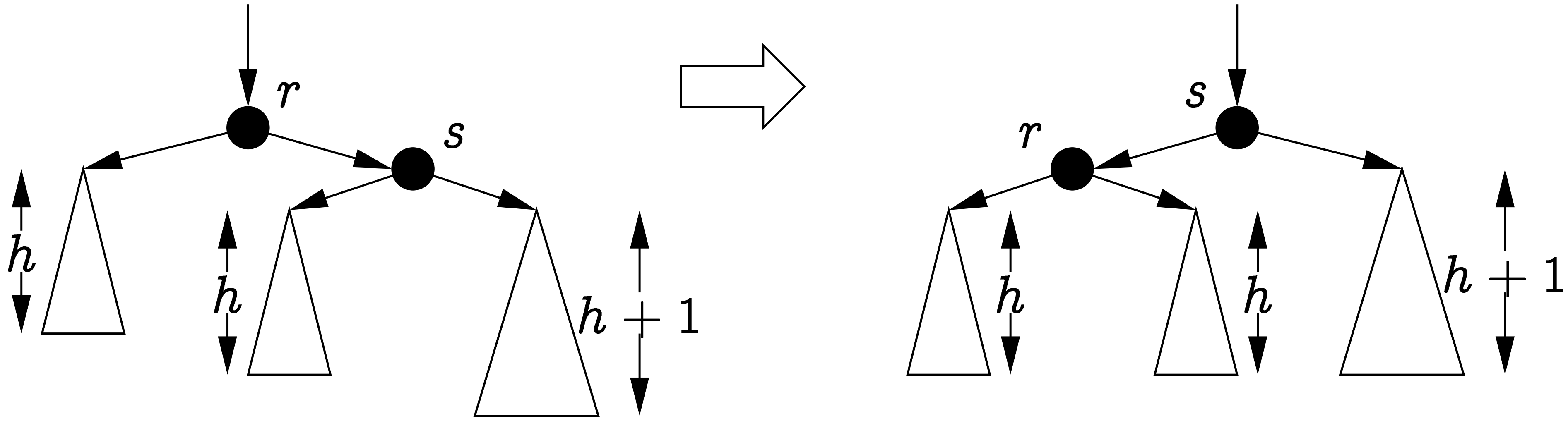


3. variant

$$h_{sv} := s.vasak = \mathrm{NIL}\ ?\ 0 : s.vasak.k\tilde{o}rgus$$
$$h_{sp} := s.parem = \mathrm{NIL}\ ?\ 0 : s.parem.k\tilde{o}rgus$$

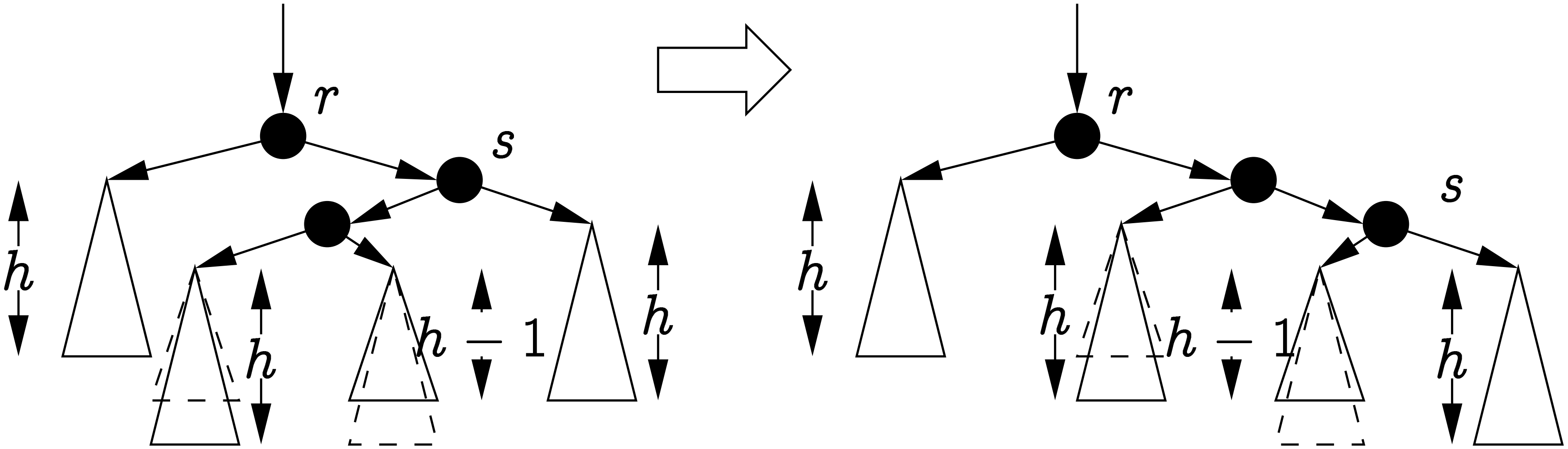
$\ldots$

1. variandil pöörame r-i vasakule.

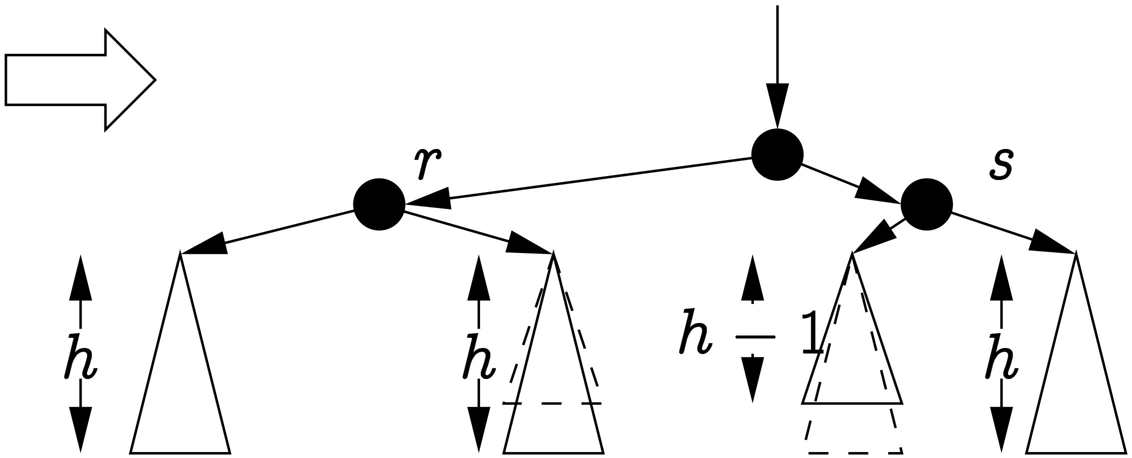


Kogu vaadeldava alampuu kõrgus on nüüd $h+2$ — vähenes ühe võrra. Seega peame me jätkame juure pool asuvate tippude tasakaalustamisi.

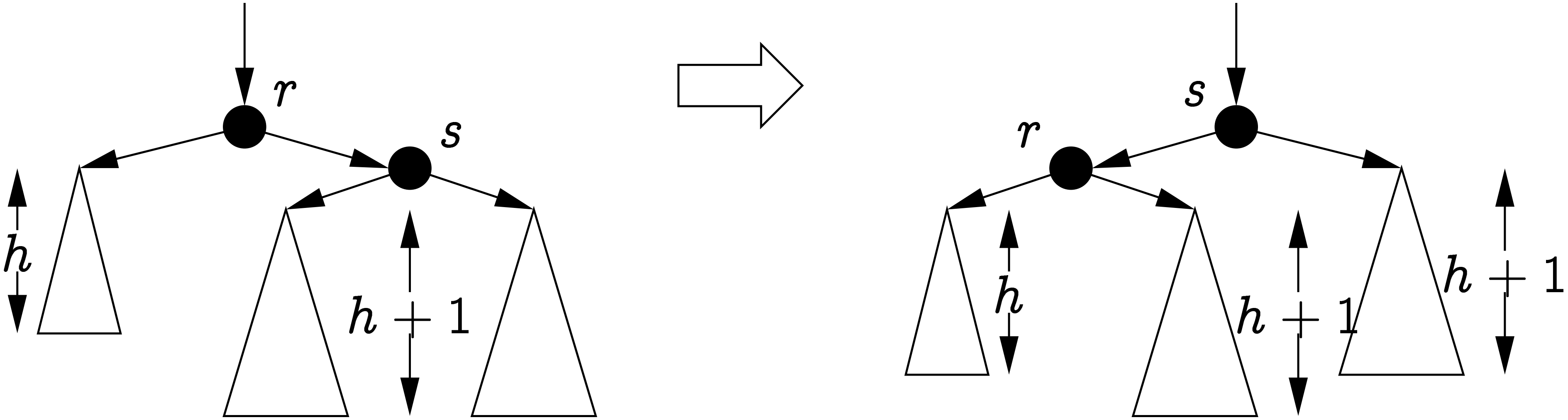
2. variandil pöörame s-i paremale



saame 1. variandi (s.t. pöörame r-i vasakule).



3. variandil pöörame r-i vasakule.



Kogu vaadeldava alampuu kõrgus on nüüd $h + 3$ — sama, mis enne eemaldamist. Seega on nüüd juurepoolsed tipud tasakaalus.

```
14      if h_sv = h_sp then
15          p := pööra_vasakule(p, r)
16          r.kõrgus := h_v + 2; s.kõrgus := h_v + 3
17          return (p, q)
18      if h_sv > h_sp then
19          p := pööra_paremale(p, s); s.kõrgus := h_v + 1
20      p := pööra_vasakule(p, r)
21      r.kõrgus := h_v + 1; r.ülem.kõrgus := h_v + 2
22      r := r.ülem.ülem; goto 3
    ...
```

23 **else** --- $h_v > h_p$

24 $\quad s := r.vasak$

25 $\quad h_{sv} := s.vasak = \mathrm{NIL}\ ?\ 0 : s.vasak.k\tilde{o}rgus$

26 $\quad h_{sp} := s.parem = \mathrm{NIL}\ ?\ 0 : s.parem.k\tilde{o}rgus$

27 $\quad$ **if** $h_{sv} = h_{sp}$ **then**

28 $\qquad p :=$ pööra_paremale(p, r)

29 $\qquad r.k\tilde{o}rgus := h_p + 2;\ s.k\tilde{o}rgus := h_p + 3$

30 $\qquad$ **return** (p, q)

31 $\quad$ **if** $h_{sv} < h_{sp}$ **then**

32 $\qquad p :=$ pööra_vasakule(p, s); $s.k\tilde{o}rgus := h_p + 1$

33 $\quad p :=$ pööra_paremale(p, r)

34 $\quad r.k\tilde{o}rgus := h_p + 1;\ r.\ddot{u}lem.k\tilde{o}rgus := h_p + 2$

35 $\quad r := r.\ddot{u}lem.\ddot{u}lem$; **goto** 3