

<http://kodu.ut.ee/~kiho/ads/Praktikum/>

## 5. KOP Kahend(otsimis)puud

# Mõisteid

**Kahendpuu\*** (kujutamine joonisel, graafina; programseid töötlusvahendeid)

**Kahendotsimispuu, selle käitlemine**

**AVL-puu, selle käitlemine**

\*

<http://kodu.ut.ee/~kiho/ads/Praktikum/Arvutipraktikum/--GP2P/Sissejuhatus/Peafail.pdf>

# Harjutusülesanded

## Kahendpuud\*

- KP\_1. Täieliku kahendpuu valmistamine
- KP\_2. Kahendpuu kõrgus
- KP\_3. Tasakaalu kontrollimine
- KP\_4. Kahendpuu ja puu läbimine
- KP\_5. Aritmeetilise avaldise puu
- KP\_6. Tipu järglaste koguarv
- KP\_7. Tippudele kõrguse- ja tasemeväljad
- KP\_8. Tasemete generaator
- KP\_9. Kompaktse kahendpuu valmistamine
- KP\_10. Kahendpuu peegeldus

## Kahendotsimispuud\*\*

- KOP\_1. Kahendotsimispuu valmistamine
- KOP\_2. Kahendotsimispuu kontroll
- KOP\_3. AVL-puu kontroll
- KOP\_4. Otsimine kahendotsimispuus
- KOP\_5. Lisamine kahendotsimispuusse
- KOP\_6. Eemaldamine kahendotsimispuust
- KOP\_7. AVL-puu tasakaalustamine

\*

Vt ka <http://kodu.ut.ee/~kiho/ads/Praktikum/Arvutipraktikum/5-KOP/N%C3%A4ited/Java/KahendpuuMeetodid.java>

\*\*

Abiks: <http://kodu.ut.ee/~kiho/ads/Praktikum/Arvutipraktikum/5-KOP/%C3%9Clesanded/Java/LisamineAVLMall.java>

# Abivahendid kahendpuu loomiseks, kuvamiseks ja töötlemiseks

**Pakett *ee.ut.kiho.aa.graaf*** (kasutamiseks: `import ee.ut.kiho.aa.graaf.*;` )

[http://kodu.ut.ee/~kiho/ads/Praktikum/Arvutipraktikum/--GP2P/Tugi-tarkvara/doc\\_graaf/index.html](http://kodu.ut.ee/~kiho/ads/Praktikum/Arvutipraktikum/--GP2P/Tugi-tarkvara/doc_graaf/index.html)

sisaldab klasside

*Graaf*

*Kaar*

*Kahendpuu*

*Puu*

*Tipp*

kirjeldusi.

Kahendpuud käsitletakse kui graafi/puu erijuhtu. Seetõttu on rakendatavad graafi- ja puutöötlemise meetodid klassidest *Graaf/Puu*. Kahendpuu/puu juureks on graafi esimene tipp (tipp indeksiga 0).

Vasak-parem alluvussuhe on määratud tippude  $x$ -koordinaatidega:

kui tipp  $t$  on tipu  $u$  alluv (st on olemas kaar  $u \rightarrow t$ ), siis

$t$  on tipu  $u$  vasak alluv, kui  $t.x < u.x$ ,

$t$  on tipu  $u$  parem alluv, kui  $t.x > u.x$ .

**$n$ -tipulise juhu-kahendpuu  $kp$  valmistamine:**

```
Kahendpuu kp = new Kahendpuu(Puu.juhupuu(n, 2));
```

Tipumärgendid: A, B, ...

**$n$ -tasemelise juhu-kahendpuu  $kp$  valmistamine:**

```
Kahendpuu kp = Kahendpuu.juhu2puu(n);
```

Tipumärgendid: tühjad

## KP\_1. Täieliku kahendpuu valmistamine [Õpik, lk 28]

Kirjutada ja testida rekursiivne meetod järgmise ülesande lahendamiseks.

Antud: arv  $m$  – loodava kahendpuu tasemete arv.

Tulemus: täielik  $m$ -tasemeline kahendpuu.

Java

Ühetipulise (ainult juur) kahendpuu  $kp$  tegemine:

```
Kahendpuu kp = new Kahendpuu(new Tipp());
```

Kahendpuus  $kp$  alluvateta tipule  $t$  mõlema alluva lisamine:

```
Tipp uus = new Tipp();           -- tühi tipp
```

```
kp.lisadaAlluv(t, uus, true);    -- true: lisamine vasakuks alluvaks
```

```
uus = new Tipp();
```

```
kp.lisadaAlluv(t, uus, false)   -- false: lisamine paremaks alluvaks
```

Ehk lühemalt:

```
kp.lisadaAlluv(t, new Tipp(), true);
```

```
kp.lisadaAlluv(t, new Tipp(), false);
```

Kahendpuu (ka graafi ja puu) kuvamine:

```
kp.faili("fail.txt");           -- kahendpuu  $kp$  faili (tabelkujul)
```

```
Graaf.kuvada("fail.txt", true); -- tippudes märgendid
```

```
Graaf.kuvada("fail.txt");       -- tippudes numbrid
```

## KP\_2. Kahendpuu kõrgus

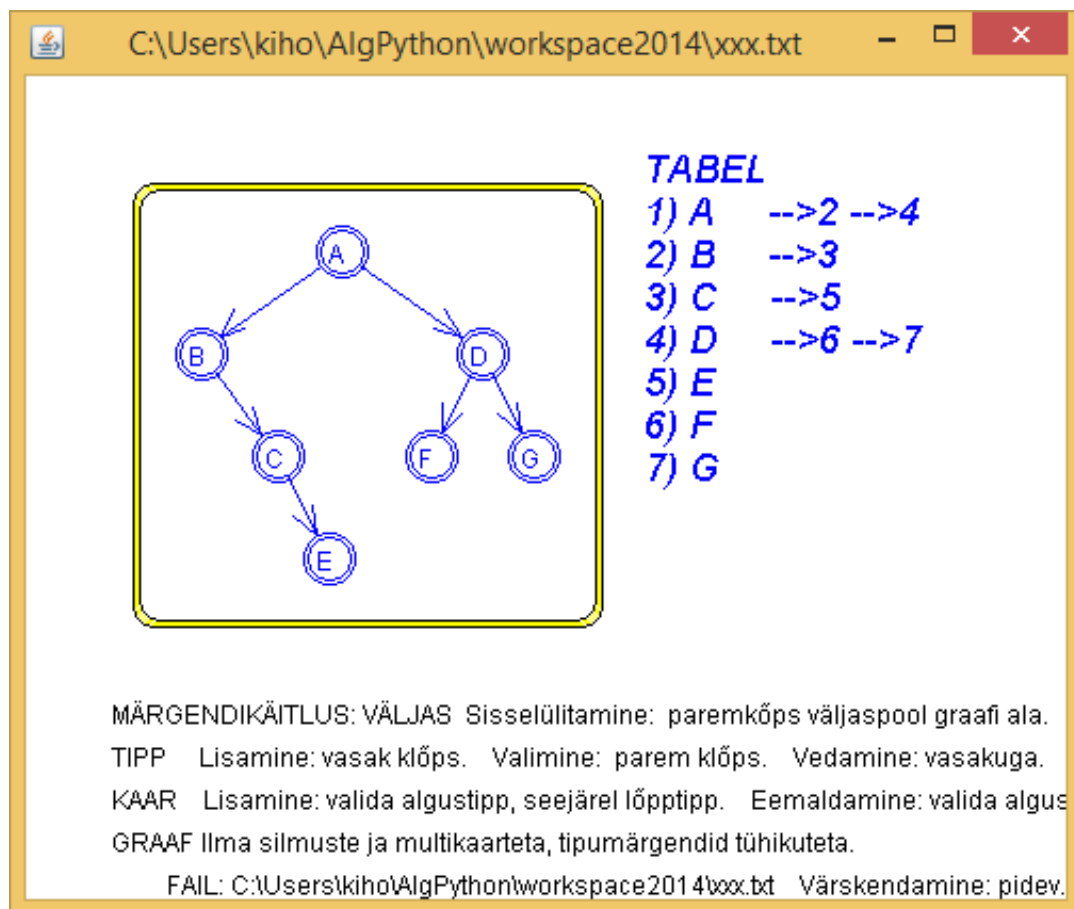
Kirjutada ja testida rekursiivne meetod järgmise ülesande lahendamiseks.

Antud: kahendpuu ja selle tipp.

Tulemus: antud tipust algava (alam)kahendpuu kõrgus.

Kahendpuu (*kp*) isendimeetodid tipu (*t*) alluva leidmiseks: `kp.vasakAlluv(t)` ;  
`kp.paremAlluv(t)` ;

### Testi tulemuse näide



The screenshot shows a text editor window with the following content:

```
C:\Users\kiho\AlgPython\workspace2014\xxx.txt
```

A diagram of a binary tree with root node A. Node A has children B and D. Node B has child C. Node C has child E. Node D has children F and G.

**TABEL**

1) A	-->2 -->4
2) B	-->3
3) C	-->5
4) D	-->6 -->7
5) E	
6) F	
7) G	

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga.  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algus  
GRAAF ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.  
FAIL: C:\Users\kiho\AlgPython\workspace2014\xxx.txt Värskendamine: pidev.

Alampuu  
juur kõrgus

A	4
B	3
C	2
D	2
E	1
F	1
G	1

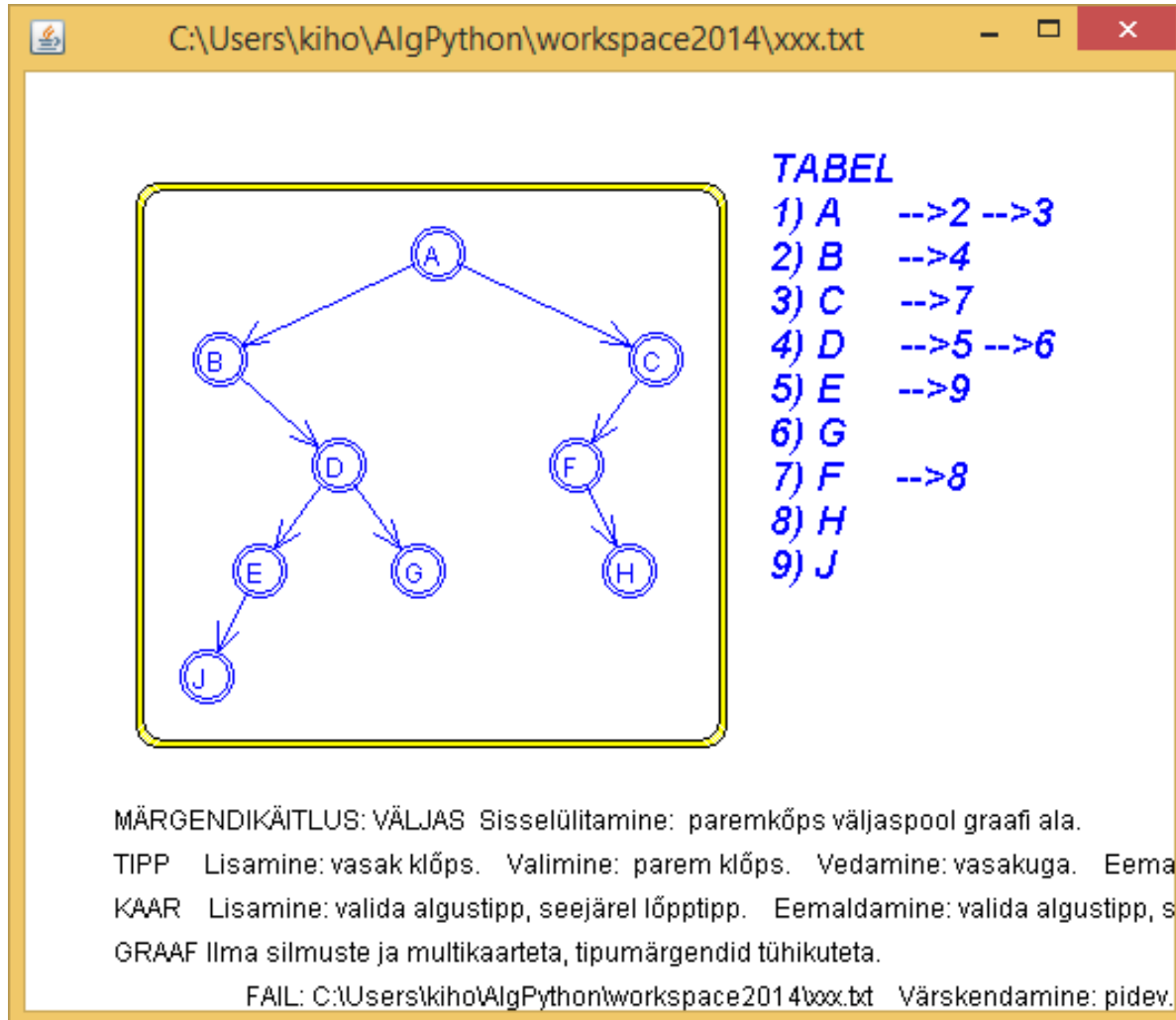
### KP\_3. Tasakaalu kontrollimine

Kirjutada ja testida meetod järgmise ülesande lahendamiseks.

Antud: kahendpuu.

Tulemus: antud kahendpuu nende tippude järjend (listina), kus tasakaal on rikutud, st tipu alluvate kõrgus erineb rohkem kui 1 võrra.

Testi tulemuse näide



The screenshot shows a window titled "C:\Users\kiho\AlgPython\workspace2014\xxx.txt". Inside, a binary tree is drawn with nodes A through J. Node A is the root. Node B is the left child of A, and node C is the right child. Node D is the left child of B, and node F is the left child of C. Node E is the left child of D, and node G is the right child of D. Node J is the left child of E, and node H is the right child of F. The tree is enclosed in a yellow border.

**TABEL**

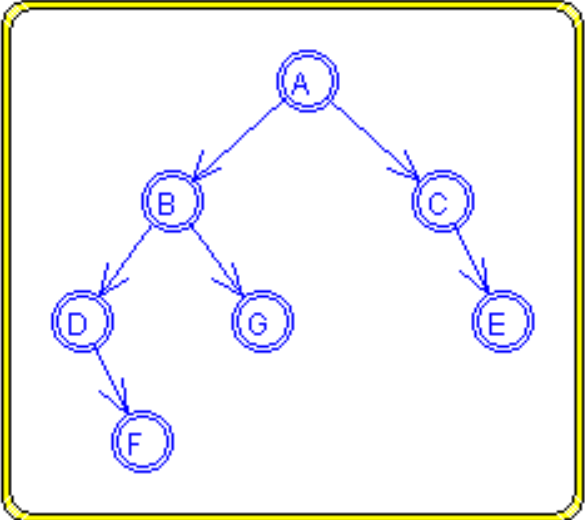
1) A	-->2	-->3
2) B	-->4	
3) C	-->7	
4) D	-->5	-->6
5) E	-->9	
6) G		
7) F	-->8	
8) H		
9) J		

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga. Eema  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustipp, s  
GRAAF ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.  
FAIL: C:\Users\kiho\AlgPython\workspace2014\xxx.txt Värskendamine: pidev.

Tasakaal rikutud  
tipus B  
tipus C

## Testi tulemuse näide

C:\Users\kiho\AlgPython\workspace2014\xxx.txt



```
graph TD; A((A)) --> B((B)); A --> C((C)); B --> D((D)); B --> G((G)); C --> E((E)); D --> F((F));
```

**TABEL**

- 1) A -->2 -->3
- 2) B -->4 -->7
- 3) C -->5
- 4) D -->6
- 5) E
- 6) F
- 7) G

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi a  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasaku  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida a  
GRAAF ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.  
FAIL: C:\Users\kiho\AlgPython\workspace2014\xxx.txt Värskendamine: pidev.

On tasakaalus.



# KP\_4. Kahendpuu ja puu läbimine

## KP\_4-1. Kahendpuu läbimine [Õpik, lk 29]

Kirjutada ja testida rekursiivsed meetodid järgmiste ülesannete lahendamiseks

### KP\_4-1E:

Antud: kahendpuu ja selle tipp  $t$ .

Tulemus: tipust  $t$  algava (alam)kahendpuu tipud eesjärjestuses (listina).

### KP\_4-1K:

Antud: kahendpuu ja selle tipp  $t$ .

Tulemus: tipust  $t$  algava(alam) kahendpuu tipud keskjärjestuses (listina).

### KP\_4-1L:

Antud: kahendpuu ja selle tipp  $t$ .

Tulemus: tipust  $t$  algava (alam)kahendpuu tipud lõppjärjestuses (listina).

### KP\_4-1E lahendus

```
static ArrayList<Tipp> eesjrst(Kahendpuu kp, Tipp t){
//Antud: kahendpuu kp ja selle tipp t (või null, kui tippu ei ole)
//Tulemus:tagastatakse listina kp tipust t algava (alam)kahendpuu tipud eesjärjestuses
    ArrayList<Tipp> tulem = new ArrayList<Tipp>();
    if (t == null) return tulem; // baasjuht
    tulem.add(t);
    tulem.addAll(eesjrst(kp, kp.vasakAlluv(t)));
    tulem.addAll(eesjrst(kp, kp.paremAlluv(t)));
    return tulem;
}
```

## KP\_4-2. Puu läbimine

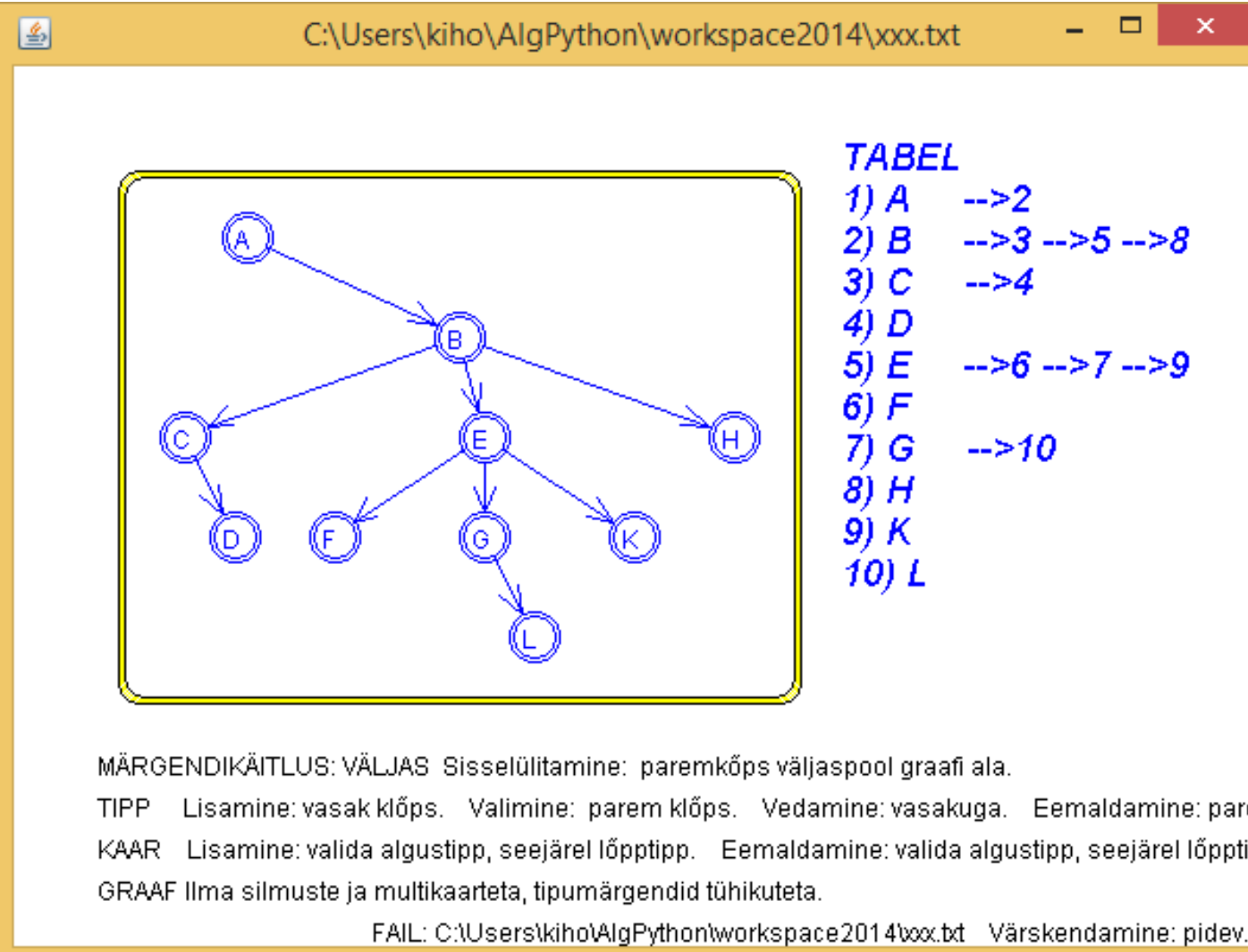
Kirjutada ja testida rekursiivsed funktsioonid järgmiste ülesannete lahendamiseks.

Antud: puu ja selle tipp  $t$ .

Tulemus: tipust  $t$  algava (alam)puu tipud eesjärjestuses (listina).

Antud: puu ja selle tipp  $t$ .

Tulemus: tipust  $t$  algava (alam)puu tipud lõppjärjestuses (listina).



The screenshot shows a window titled "C:\Users\kiho\AlgPython\workspace2014\xxx.txt". Inside, there is a tree diagram with nodes A through L. Node A is the root, with children B and H. Node B has children C, E, and H. Node C has child D. Node E has children F, G, and K. Node G has child L. To the right of the diagram is a table:

TABEL	
1) A	-->2
2) B	-->3 -->5 -->8
3) C	-->4
4) D	
5) E	-->6 -->7 -->9
6) F	
7) G	-->10
8) H	
9) K	
10) L	

Below the diagram and table, there is text in Estonian:

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga. Eemaldamine: parem klõps.  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustipp, seejärel lõpptipp.  
GRAAF Ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.  
FAIL: C:\Users\kiho\AlgPython\workspace2014\xxx.txt Värskendamine: pidev.

Testi tulemuse näide

Puu eesjrst:  
ABCDEFGLKH

Puu lõppjrst:  
DCFLGKEHBA

## **KP\_5. Aritmeetilise avaldise puu**

Koostada programm, mis valmistab (juhusliku) aritmeetilise avaldise kahendpuu ja arvutab vastava avaldise väärtuse. (Vt ka järgmine slaid.)

Eraldi meetoditena kirjeldada järgmised alamülesanded:

### **KP\_5-1. Aritmeetilise avaldise puu valmistamine**

Kirjutada ja testida funktsioon järgmise ülesande lahendamiseks.

Antud: kahendpuu.

Tulemus: antud kahendpuu muudetud 0-2 kahendpuuks (igale ühe alluvaga tipule lisatud teine, puudunud alluv), seejärel rippuvate e. lehttippude märgenditeks seatud juhuarvud, ülejäänutele – binaarsete tehete märgid, juhuvalik (+ - \* /) seast.

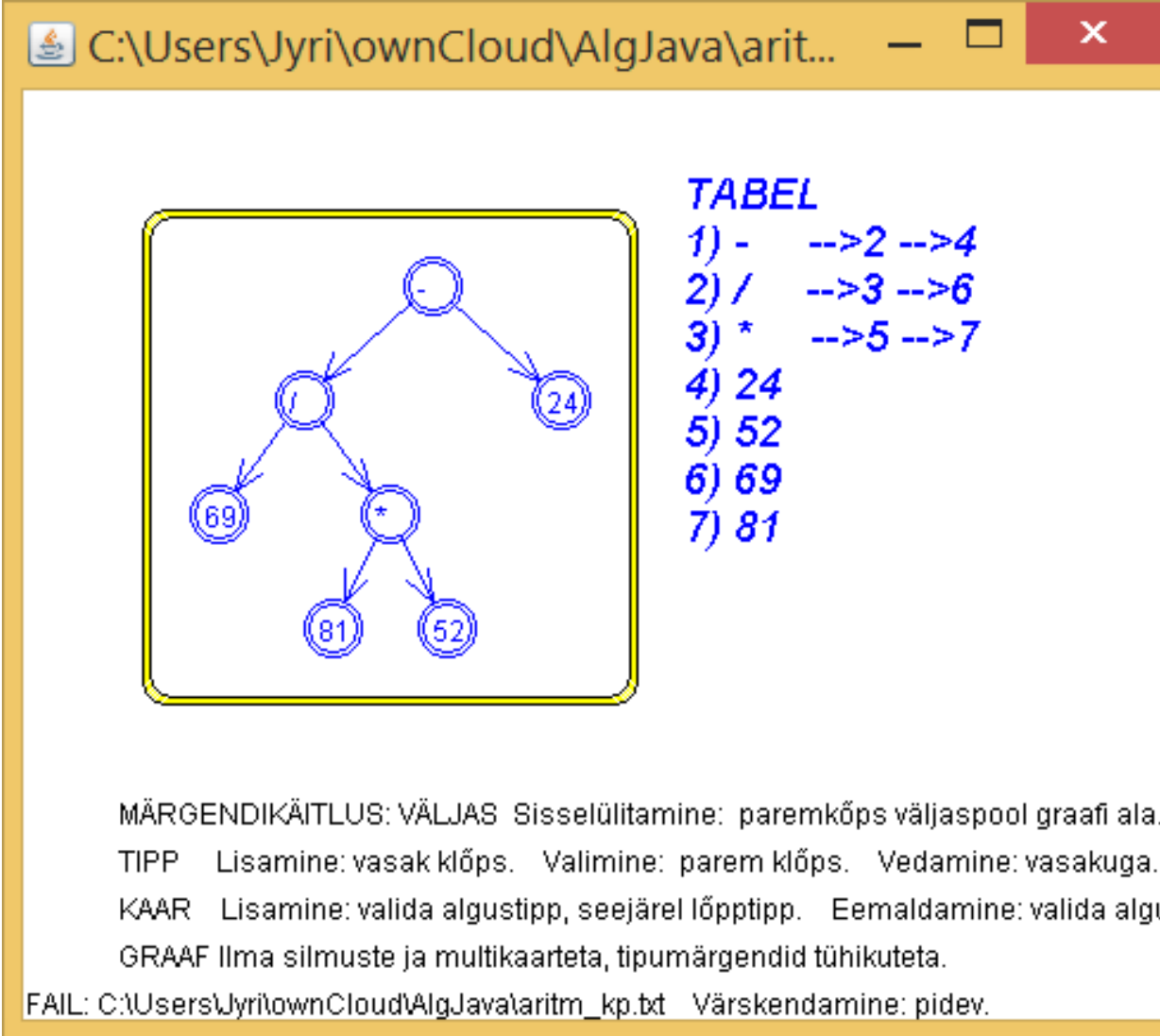
### **KP\_5-2. Aritmeetilise avaldise puust väärtuse arvutamine**

Kirjutada ja testida rekursiivne funktsioon järgmise ülesande lahendamiseks.

Antud: aritmeetiline kahendpuu ja selle tipp  $t$ .

Tulemus: tipust  $t$  algava aritmeetilisest (alam)puust arvutatud väärtus.

# Aritmeetilise avaldise kahendpuu (KP\_5 testi tulemuse) näide.



The screenshot shows a Java application window titled "C:\Users\Jyri\ownCloud\AlgJava\arit...". Inside the window, there is a diagram of an arithmetic expression tree and a table of operations.

The tree diagram shows a root node with a minus sign (-). The left child is a division node (/), and the right child is a leaf node with the value 24. The division node has a left child leaf node with the value 69 and a right child multiplication node (\*). The multiplication node has two leaf children with the values 81 and 52.

Next to the tree is a table titled "TABEL" with the following content:

TABEL		
1) -	-->2	-->4
2) /	-->3	-->6
3) *	-->5	-->7
4)	24	
5)	52	
6)	69	
7)	81	

Below the table, there is a list of instructions in Estonian:

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga.  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algu  
GRAAF ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.  
FAIL: C:\Users\Jyri\ownCloud\AlgJava\aritm\_kp.txt Värskendamine: pidev.

Avaldise väärtus: -23.98.

## KP\_6. Tipu järglaste koguarv

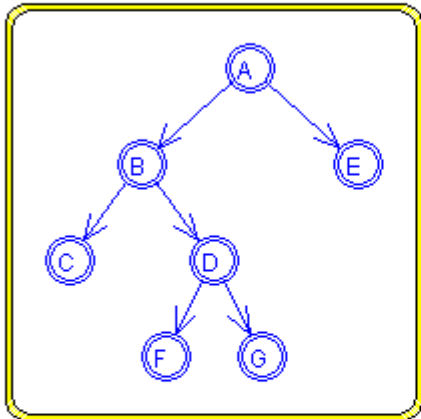
Kirjutada ja testida rekursiivne meetod järgmise ülesande lahendamiseks.

Antud: kahendpuu ja selle tipp.

Tulemus: antud tipu kõigi (vahetute ja kaugemate) järglaste koguarv antud kahendpuus.

Testi tulemuse näide

C:\AlgPyhton\workspace2014\box.txt



```
graph TD; A((A)) --> B((B)); A --> E((E)); B --> C((C)); B --> D((D)); D --> F((F)); D --> G((G));
```

**TABEL**

- 1) A -->2 -->5
- 2) B -->3 -->4
- 3) C
- 4) D -->6 -->7
- 5) E
- 6) F
- 7) G

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga. Eemaldamine  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustipp, seejärel lõpptipp.  
GRAAF ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.

FAIL: C:\AlgPyhton\workspace2014\box.txt Värskendamine: pidev.

Tipp	Järglasi
A	6
B	4
C	0
D	2
E	0
F	0
G	0

## KP\_7. Tippudele kõrguse- ja tasemeväljad

**KP\_7-1.** Kirjutada ja testida rekursiivne meetod järgmise ülesande lahendamiseks.

Antud: kahendpuu  $kp$ , selle tipp  $t$ .

Tulemus: tipust  $t$  algavas  $kp$  alampuus on iga tipu  $u$  väljale "h" omistatud tipust  $u$  algava alampuu kõrgus; tagastatakse tipust  $t$  algava alampuu kõrgus.

**KP\_7-2.** Kirjutada ja testida rekursiivne meetod järgmise ülesande lahendamiseks.

Antud: kahendpuu  $kp$ , selle tipp  $t$  ja tipu  $t$  tasemenumber  $m$ .

Tulemus: tipust  $t$  algavas  $kp$  alampuus on iga tipu väljale "tase" omistatud tasemenumber; antud tipu tasemenumbriks on  $m$ , selle alluvatel  $m+1$  jne.

Tipu ( $t$ ) väljale ("v") arvu ( $n$ ) seadmine:

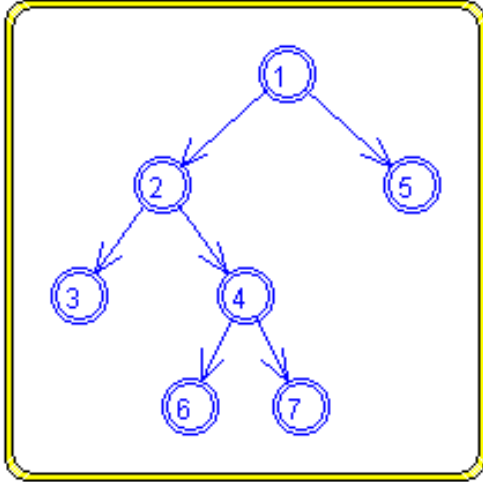
```
t.seadaVäli("v", "" + n);
```

Tipu ( $t$ ) väljalt ("v") arvulise väärtuse lugemine:

```
int n = Integer.parseInt(t.väli("v"));
```

## KP\_7 testi tulemuse näide

C:\Users\Jyri\ownCloud\AlgJava\kpxxx.t... — □ ×



**TABEL**

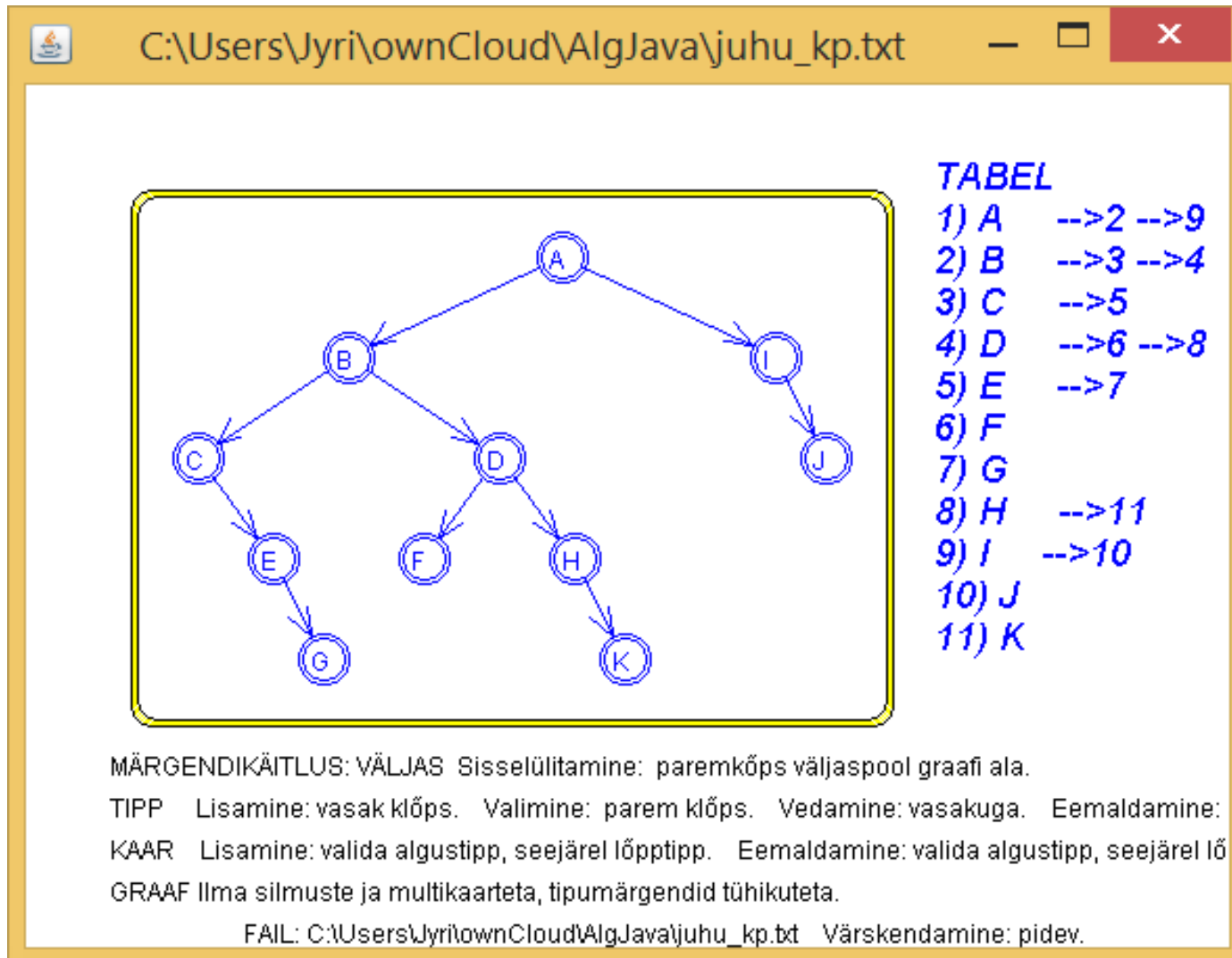
- 1) tase=1;h=4;A-->2 -->5
- 2) tase=2;h=3;B-->3 -->4
- 3) tase=3;h=1;C
- 4) tase=3;h=2;D-->6 -->7
- 5) tase=2;h=1;E
- 6) tase=4;h=1;F
- 7) tase=4;h=1;G

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga. E  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustip  
GRAAF ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.  
FAIL: C:\Users\Jyri\ownCloud\AlgJava\kpxxx.bt Värskendamine: pidev.

## KP\_8. Tasemete generaator

Kirjutada ja testida tasemete generaator genereerimaks järjest antud kahendpoo tasemeid (tippude loeteludena).

Testi tulemuse näide



The screenshot shows a Java application window titled "C:\Users\Jyri\ownCloud\AlgJava\juhu\_kp.txt". The main content area displays a binary tree diagram with nodes labeled A through K. Node A is the root. Node B is the left child of A, and node I is the right child. Node C is the left child of B, and node D is the right child. Node E is the left child of C, and node F is the left child of D. Node G is the right child of E, and node H is the right child of D. Node J is the right child of I, and node K is the right child of H. The tree is drawn with blue lines and circles.

Next to the tree is a table titled "TABEL" with the following content:

TABEL		
1) A	-->2	-->9
2) B	-->3	-->4
3) C	-->5	
4) D	-->6	-->8
5) E	-->7	
6) F		
7) G		
8) H	-->11	
9) I	-->10	
10) J		
11) K		

Below the table is a text box titled "Tipumärgendid tasemeti:" containing the following text:

```
A
B I
C D J
E F H
G K
```

At the bottom of the window, there is a footer with the following text:

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga. Eemaldamine:  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustipp, seejärel lõ  
GRAAF ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.  
FAIL: C:\Users\Jyri\ownCloud\AlgJava\juhu\_kp.txt Värskendamine: pidev.



### KP\_9. Kompaktse kahendpuu valmistamine [Õpik, lk 28]

Kirjutada ja testida meetod järgmise ülesande lahendamiseks.

Antud: arv  $n$  – loodava kahendpuu tippude arv.

Tulemus: kompaktne  $n$ -tipuline kahendpuu.

Meetodi  
üldskeem:

Valmistada (KP\_1) täielik kahendpuu, milles on  $\log_2(n + 1)$  taset.  
Leida (KP\_8) viimase taseme tipud ja lisada neile alluvateks  
veel vajalik arv uusi tippe.

## KP\_10. Kahendpuu peegeldus

Kirjutada ja testida meetod järgmise ülesande lahendamiseks.

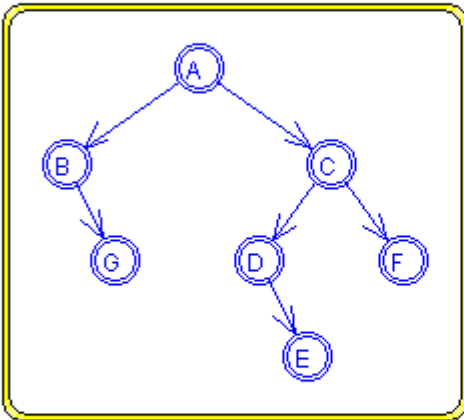
Antud: kahendpuu.

Tulemus: antud kahendpuu peegeldus üle vertikaaltelje.

Idee: tippude x-koordinaadid asendada vastandarvudega.

Testi tulemuse näide

C:\Users\kiho\AlgPython\workspace2014\xxx.txt



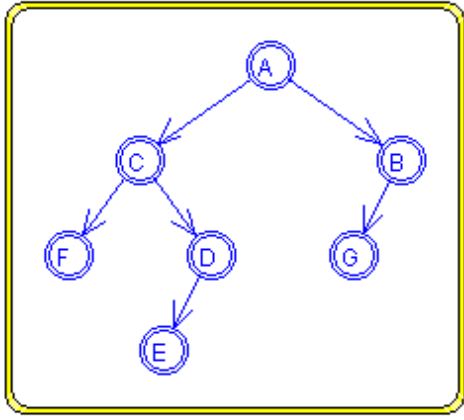
```
graph TD; A((A)) --> B((B)); A --> C((C)); B --> G((G)); C --> D((D)); C --> F((F)); D --> E((E));
```

**TABEL**

- 1) A -->2 -->3
- 2) B -->7
- 3) C -->4 -->6
- 4) D -->5
- 5) E
- 6) F
- 7) G

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasaku  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida  
GRAAF Ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.  
FAIL: C:\Users\kiho\AlgPython\workspace2014\xxx.txt Värskendamine: pidev.

C:\Users\kiho\AlgPython\workspace2014\yyy.txt



```
graph TD; A((A)) --> C((C)); A --> B((B)); C --> F((F)); C --> D((D)); D --> E((E)); B --> G((G));
```

**TABEL**

- 1) A -->2 -->3
- 2) B -->7
- 3) C -->4 -->6
- 4) D -->5
- 5) E
- 6) F
- 7) G

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasaku  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida  
GRAAF Ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.  
FAIL: C:\Users\kiho\AlgPython\workspace2014\yyy.txt Värskendamine: pidev.

# Kahendpuu töötlusprogrammide skeeme

Töötlusprogrammi alguses paiknevat preambulit, näiteks KP\_1 korral

preambul

»

» Algoritmid ja andmestruktuurid

» 2018/2019 sügissemester

» Praktikumitöö: ülesanne KP\_1

» Täieliku kahendpuu valmistamine

» Autor: ...

preambul

ei ole skeemis näidatud.

```
//          --- TÄIELIKU KAHENDPUU VALMISTAMINE ---          Java8 //
/*****
*
*          Algoritmid ja andmestruktuurid
*          2018/2019 sügissemester
*          Praktikumitöö: ülesanne KP_1
*
*
*          Autor: Ülli Õpilane
*
*          --- Lahendamise käsurealt (cmd) ---
*          Eeldus: jooksvas kaustas on failid
*          KP_1.java (see fail) ja ee_ut_kiho_aa.zip
*          Kompileerimine:
*          kui see fail on ANSI kodeeringus, siis
*          Kaustatee>javac -cp .\;ee_ut_kiho_aa.zip KP_1.java
*          kui see fail on utf-8 kodeeringus, siis
*          Kaustatee>javac -encoding utf8 -cp .\;ee_ut_kiho_aa.zip KP_1.java
*          Käivitamine:
*          Kaustatee>java -cp .\;ee_ut_kiho_aa.zip KP_1
*
*****/
```

# KP\_1 skeem

```
import ee.ut.kiho.aa.graaf.*;»
»»
int m = 4; tasemete arv
Kahendpuu kp = tehaTäielik(m);»
kp.faili("fail.txt");»
Graaf.kuvada("fail.txt");»
»»
Kahendpuu tehaTäielik(int m)»
»
» Antud: arv m – loodava kahendpuu tasemete arv
» Tulemus: luuakse ja tagastatakse täielik m-tasemeline kahendpuu
baasjuht
if (m == 1)»
»
return new Kahendpuu(new Tipp()); ühetipuline
baasjuht
Kahendpuu kp = tehaTäielik(m-1);»
»»
int k = (int)Math.pow(2, m-2); tippe kp viimasel tasemel
» kp viimasel tasemel:
int esimeseIndeks = kp.n() - k;»
int viimaseIndeks = kp.n() - 1;»
» igale viimase taseme tipule lisada 2 alluvat:
»
* i = esimeseIndeks .. viimaseIndeks»
»
Tipp t = kp.tipp(i); t on üks tipp viimasel tasemel
kp.lisadaAlluv(t, new Tipp(), true);»
kp.lisadaAlluv(t, new Tipp(), false);»
return kp;»
```

# KP\_2 skeem

```
import ee.ut.kiho.aa.graaf.*;»
```

```
»»
```

```
TEST (peameetod main)
```

```
int h(Kahendpuu kp, Tipp t)»
```

- » Antud: kahendpuu kp ja selle tipp t (või null, kui tippu ei ole)
- » Tulemus: tagastatakse tipust t algava (alam)kahendpuu kõrgus

```
baasjuht
```

```
int vasH = h(kp, kp.vasakAlluv(t)); t vasaku alampuu kõrgus
```

```
int parH = h(kp, kp.paremAlluv(t)); t parema alampuu kõrgus
```

```
tagastada tipu t kõrgus
```

```
TEST (peameetod main)
```

```
»
```

```
Kahendpuu kp = new Kahendpuu(Puu.juhupuu(5, 2));»
```

```
kp.faili("fail.txt"); kahendpuu faili (tabelkujul)
```

```
Graaf.kuvada("fail.txt", true); tippudes märgendid
```

```
»»
```

```
println(" Alampuu");»
```

```
println("juur\tkõrgus");»
```

```
»
```

```
* i = 0 .. kp.n()-1»
```

```
»
```

```
Tipp t = kp.tipp(i); kp tipp indeksiga i
```

```
println(" " + t.märgend() + "\t " + h(kp, t));»
```

```
TEST (peameetod main)
```

# KP\_3 skeem

```
import ee.ut.kiho.aa.graaf.*;»
import java.util.ArrayList;»
```

```
»»
```

```
»» TEST (peameetod main)
```

```
»»
```

```
int h(Kahendpuu kp, Tipp t, ArrayList<Tipp> rikutud)»
```

```
»
```

```
» Antud: kahendpuu kp, selle tipp t (või null) ja formeeritav list 'rikutud'
```

```
» Tulemus: tipp t lisatakse listi 'rikutud', kui selle alluvate kõrgused
```

```
» . erinevad rohkem kui 1 võrra;
```

```
» . tagastatakse tipust t algava (alam)kahendpuu kõrgus
```

```
» baasjuht
```

```
int vasH = h(kp, kp.vasakAlluv(t), rikutud);»
```

```
int parH = h(kp, kp.paremAlluv(t), rikutud);»
```

```
» tipp t lisada/mitte lisada listi 'rikutud'
```

```
return Math.max(vasH, parH) + 1; tagastada tipu t kõrgus
```

```
TEST (peameetod main)
```

```
»
```

```
Kahendpuu kp = Kahendpuu.juhu2puu(5);»
```

```
kp.faili("fail.txt"); kahendpuu faili (tabelkujul)
```

```
Graaf.kuvada("fail.txt", false); tippudes numbrid
```

```
»»
```

```
ArrayList<Tipp> rikutud = new ArrayList<Tipp>();
```

```
h(kp, kp.juur(), rikutud);»
```

```
println("Tasakaal rikutud tippudes" );»
```

```
»»
```

```
* for(Tipp t : rikutud)»
```

```
»»
```

```
println("nr " + (kp.indeks(t) + 1) );»
```

```
println();»
```

```
TEST (peameetod main)
```

# KP\_4-1K skeem

```
import ee.ut.kiho.aa.graaf.*;»
import java.util.ArrayList;»

»»
TEST (peameetod main)
»»

ArrayList<Tipp> keskjrst(Kahendpuu kp, Tipp t)»
Vt ka ül. KP_4-1E
» Antud: kahendpuu kp ja selle tipp t (või null, kui tippu ei ole)
» Tulemus: tagastatakse listina
» .      kp tipust t algava (alam)kahendpuu tipud keskjärjestuses
ArrayList<Tipp> tulem = new ArrayList<Tipp>();»
baasjuht
formeerida tulem
return tulem;»
```

TEST (peameetod main)

```
»
Kahendpuu kp = new Kahendpuu(Puu.juhupuu(8, 2));»
kp.faili("fail.txt");  kahendpuu faili (tabelkujul)
Graaf.kuvada("fail.txt", true); tippudes märgendid
»»
ArrayList<Tipp> list = keskjrst(kp, kp.juur());»
»»
print("Tipud keskjärjestuses: ");»
* for(Tipp t : list)»
»
print(t.märgend() + " ");»
```

TEST (peameetod main)





# KP\_6 skeem

```
»
»
import ee.ut.kiho.aa.graaf.*;»
TEST (peameetod main)
»
Kahendpuu kp = new Kahendpuu(Puu.juhupuu(7, 2));»
println("Tipp \tJärglasi");»
    »
    i = 0 .. kp.n()-1»
    »
    Tipp t = kp.tipp(i); kp tipp ineksiga i
    println(" " + t.märgend() + "\t " + alluvaid(kp, t));»
    »
kp.faili("fail.txt");»
Graaf.kuvada("fail.txt", true); tippudes märgendid
TEST (peameetod main)
»
int alluvaid(Kahendpuu kp, Tipp t)»
»
» Antud: kahendpuu kp ja tipp t selles (või null)
» Tulemus: tagastatakse tipu t alluvate arv kahendpuus kp
» baasjuht
return kp.aste(t) +
alluvaid(kp, kp.vasakAlluv(t)) +
alluvaid(kp, kp.paremAlluv(t));»
```

# KP\_7 skeem

```
import ee.ut.kiho.aa.graaf.*;
```

```
»»
```

```
TEST (peameetod main)
```

```
int seada_kõrguseväljad(Kahendpuu kp, Tipp t)
```

```
void seada_tasemeNr(Kahendpuu kp, Tipp t, int m)
```

```
TEST (peameetod main)
```

```
»
```

```
Kahendpuu kp = new Kahendpuu(Puu.juhupuu(5, 2));»
```

```
kp.faili("fail.txt"); kahendpuu faili (tabelkujul)
```

```
Graaf.kuvada("fail.txt", true); tippudes märgendid
```

```
»»
```

```
seada_kõrguseväljad(kp, kp.juur());»
```

```
kp.faili("failH.txt"); kahendpuu faili (tabelkujul)
```

```
Graaf.kuvada("failH.txt", false); tippudes numbrid
```

```
»»
```

```
seada_tasemeNr(kp, kp.juur(), 1);»
```

```
kp.faili("failHT.txt"); kahendpuu faili (tabelkujul)
```

```
Graaf.kuvada("failHT.txt", false); tippudes numbrid
```

```
TEST (peameetod main)
```

```
int seada_kõrguseväljad(Kahendpuu kp, Tipp t)»
```

```
»
```

```
» Antud: kahendpuu kp ja selle tipp t (või null, kui tippu ei ole)
```

```
» Tulemus: tipust t algavas kp alampuus on iga tipu u väljale "h"
```

```
» . omistatud tipust u algava alampuu kõrgus;
```

```
» . tagastatakse tipust t algava (alam)kahendpuu kõrgus
```

```
baasjuht
```

```
if (t == null)»
```

```
»
```

```
return 0;»
```

```
baasjuht
```

```
int hv = seada_kõrguseväljad(kp, kp.vasakAlluv(t));»
```

```
int hp = seada_kõrguseväljad(kp, kp.paremAlluv(t));»
```

```
int h = 1+Math.max(hv, hp);»
```

```
t.seadaVäli("h", ""+h);»
```

```
return h;»
```

```
void seada_tasemeNr(Kahendpuu kp, Tipp t, int m)»
```

```
»
```

```
» Antud: kahendpuu kp, tipp t selles (või null, kui tippu ei ole)
```

```
» . ja tipu t tasemenumber m
```

```
» Tulemus: tipust t algavas kp alampuus on
```

```
» . iga tipu väljale "tase" omistatud tasemenumber,
```

```
» . antud tipu t "tase" on m, selle alluvatel m+1, ...
```

```
baasjuht
```

```
if (t == null) »
```

```
»
```

```
return; »
```

```
baasjuht
```

```
t.seadaVäli("tase", ""+m);»
```

```
seada_tasemeNr(kp, kp.vasakAlluv(t), m+1);»
```

```
seada_tasemeNr(kp, kp.paremAlluv(t), m+1);»
```

## KOP\_1. Kahendotsimispuu valmistamine

Kirjutada ja testida meetod järgmise ülesande lahendamiseks.

Antud: arv  $n$ ,  $n > 0$ .

Tulemus:  $n$ -tipuline juhu-kahendotsimispuu.

Meetodi võimalik skeem:

- Luua  $n$ -tipuline juhu-kahendpuu  $kp$
- Luua  $n$ -elemendiline juhuarvude sorteeritud järjend  $a$  (näiteks juhuarvudest lõigult [1; 99])
- Omistada arvud järjendist  $a$  kahendpuu  $kp$  tippude märgenditeks tippude keskjärjekorras (vt ül KP\_4-1K)

## **KOP\_2. Kahendotsimispuu kontroll**

Kirjutada ja testida meetod järgmise ülesande lahendamiseks.

Antud: kahendpuu, milles tipumärgenditeks arvud.

Tulemus: kontrollitud, kas antud kahendpuu on kahendotsimispuu.

Vt [Õpik, teoreem lk 31]

### KOP\_3. AVL-puu kontroll

Kirjutada ja testida meetod järgmise ülesande lahendamiseks.

Antud: kahendpuu.

Tulemus: kontrollitud, kas antud kahendpuu on AVL-puu.

#### KOP\_3-1. Rikkekohad

Kirjutada ja testida rekursiivne meetod järgmise ülesande lahendamiseks.

Antud:

- kahendpuu  $kp$ , milles iga tipuga on seotud kõrguseväli  $h$  (vt ül KP\_7)
- $kp$  tipp  $t$
- tipu  $t$  ülemus ( $\Lambda$ , kui  $t$  on  $kp$  juurtipp) /Java:  $\Lambda = \text{null}$  /

Tulemus: tagastatakse

$kp$  tasakaalu rikkekohtade list paaridest  $\langle v, y \rangle$ , kus

$v$  on AVL-rikkega tipp, st  $v$  vasaku ja parema alampuu kõrgused erinevad rohkem kui ühe võrra  
 $y$  on selle tipu ( $v$ ) ülemus (või  $\Lambda$ , kui  $v$  on  $kp$  juurtipp)

Vihje: kahendpuu läbimine toimetada eesjärjestuses.

KOP\_3 skeem:

- Omistada  $kp$  tippudele kõrguseväljad (ül KP\_7-1 )
- Tulemus:  $kp$  on AVL-puu, kui  $kp$  on kahendotsimispuu (ül KOP\_2) ja  $kp$  rikkekohtade list on tühi

## KOP\_3-2. Kõrgused ja AVL-rikkekohad

Kirjutada ja testida rekursiivne meetod järgmise ülesande lahendamiseks.

Antud:

- kahendpuu  $kp$
- $kp$  tipp  $t$
- tipu  $t$  ülemus ( $\Lambda$ , kui  $t$  on  $kp$  juurtipp) / Java:  $\Lambda = \text{null}$  /
- rikkekohtade list (kuhu lisada leitavaid rikkekohti), rikkekohaks on paar  $\langle \text{rikkega tipp, selle ülemus(või } \Lambda) \rangle$

Tulemus: tipust  $t$  algavas  $kp$  alampuu  $alamkp$

- seatud kõikide tippude kõrguseväljad
- list täiendatud selles ( $alamkp$ ) leiduvate rikkekohtadega;
- tagastatakse  $alamkp$  kõrgus

# KOP\_3-2 skeem

## KOP\_3-2 MALL

```
int hJaRikkekohad(Kahendpuu kp, Tipp t, Tipp ülemus, ArrayList<Tipp[]> rikkekohad)»
```

- »
- » Antud: kahendotsimispuu kp,
- » . selle tipp t ja tipu t ülemus (null, kui t on kp juur)
- » . ja rikkekohtade list (kuhu lisada leitavaid rikkekohti <rikkega tipp, selle ülemus>)
- » Tulemus: tipust t algavas kp alampuus (alamkp) seatud kõikide tippude kõrguseväljad
- » . ja rikkekohtade list täiendatud selles alampuus leiduvate rikkekohtadega;
- » . tagastatakse alamkp kõrgus

```
if (t == null) baasjuht
```

```
return ...;»
```

```
Tipp tvas = kp.vasakAlluv(t);»
```

```
Tipp tpar = kp.paremAlluv(t);»
```

- » tipp t on ülemuseks tippudele tvas ja tpar

```
int vas_h = hJaRikkekohad(kp, ..., ..., rikkekohad);»
```

```
int par_h = hJaRikkekohad(kp, ..., ..., ...);»
```

```
if (Math.abs(vas_h - par_h) > 1)»
```

```
...»
```

```
rikkekohad.add(...);»
```

```
int h = ...; tipu t kõrgus
```

```
t.seadaVäli("h", ""+h);»
```

```
return ...;»
```

## KOP\_3-2 MALL

```
-  
preambul
```

```
import ee.ut.kiho.aa.graaf.*;»
```

```
import ee.ut.kiho.aa.util.*; Juhu.järjend(a,b, n)
```

```
import java.util.ArrayList;»
```

```
import java.util.Arrays;»
```

```
»»
```

## KOP\_3-2 TEST

```
»
```

```
Kahendpuu kop = tehaKOPjuhu(9);»
```

```
kop.faili("kopyyy.txt");»
```

```
Graaf.kuvada("kopyyy.txt", true); tippudes märgendid
```

```
»»
```

```
ArrayList<Tipp[]> rikked = new ArrayList<Tipp[]>();»
```

```
hJaRikkekohad(kop, kop.juur(), null, rikked);»
```

```
»»
```

```
println("Rikkekohad:");»
```

```
»
```

```
* for(Tipp[] rike : rikked)»
```

```
»
```

```
Tipp t = rike[0];»
```

```
Tipp ülemus = rike[1];»
```

```
println(t.nimi() + " " + ((ülemus == null)? "null" : ülemus.nimi()));»
```

```
kop.faili("kophhh");»
```

```
Graaf.kuvada("kophhh", true);»
```

## KOP\_3-2 TEST

```
int hJaRikkekohad(Kahendpuu kp, Tipp t, Tipp ülemus, ArrayList<Tipp[]> rikkekohad)
```

```
Kahendpuu tehaKOPjuhu(int n)
```

```
ArrayList<Tipp> keskjrst(Kahendpuu kp, Tipp t)
```

## KOP\_4. Otsimine kahendotsimispuus

Kirjutada ja testida rekursiivne meetod järgmise ülesande lahendamiseks.

Antud:  $kop$  - mittetühi kahendotsimise puu,  $t$  - selle tipp ja arv  $k$ , mille kohta (ja ülemust) otsida alates tipust  $t$ .

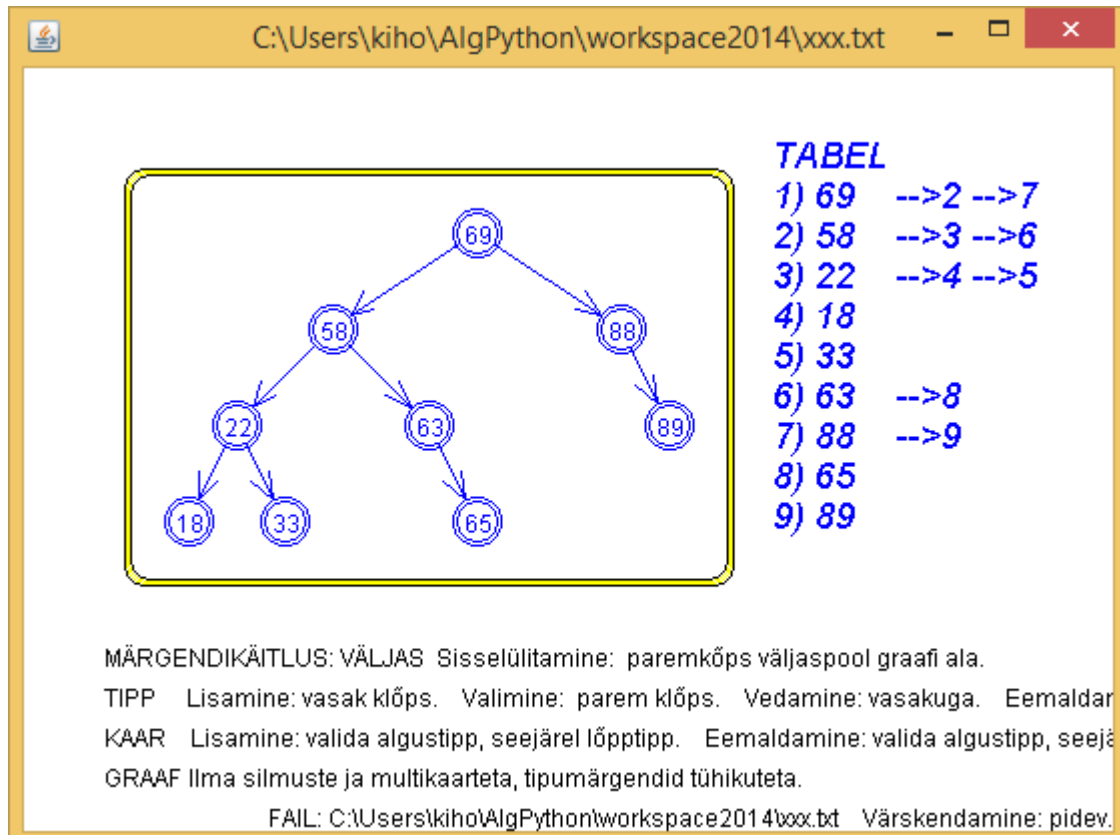
Tulemus: tagastatakse paar  $\langle t1, t2 \rangle$ , kus

$t1$  on tipp, milles arv  $k$  leiti (või  $\Lambda$ , kui arvu ei leitud)

$t2$  on  $t1$  ülemus (või  $\Lambda$ , kui  $t1$  on juurtipp);

juhul kui arvu ei leitud ( $t1 = \Lambda$ ), siis  $t2$  on tipp, millele arv  $k$  sobiks alluvaks.

### Testi tulemuse näide



The screenshot shows a window titled "C:\Users\kiho\AlgPython\workspace2014\xxx.txt". Inside, there is a binary search tree diagram with nodes containing numbers. The root node is 69. Its left child is 58, and its right child is 88. Node 58 has children 22 and 63. Node 22 has children 18 and 33. Node 63 has child 65. Node 88 has child 89. To the right of the tree is a table labeled "TABEL" with 9 rows of test cases. Below the tree and table, there is a block of text providing instructions for using the workspace, including terms like "MÄRGENDIKÄITLUS", "TIPP", "KAAR", and "GRAAF". At the bottom, there is a "FAIL" message.

TABEL		
1)	69	-->2 -->7
2)	58	-->3 -->6
3)	22	-->4 -->5
4)	18	
5)	33	
6)	63	-->8
7)	88	-->9
8)	65	
9)	89	

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga. Eemaldamine: vasak klõps.  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustipp, seejärel lõpptipp.  
GRAAF Ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.  
FAIL: C:\Users\kiho\AlgPython\workspace2014\xxx.txt Värskendamine: pidev.

Otsitav arv: 63

Leitud arv ja selle ülemus: 63 58

Otsitav arv: 75

Otsitavat arvu ei ole, koha ülemus: 88

Otsitav arv: 69

Otsitav arv leitud juurtipus (ülemus: null)



## KOP\_5. Lisamine kahendotsimispuusse

Kirjutada ja testida meetod järgmise ülesande lahendamiseks.

Antud: kahendotsimispuu *kop* (arvud-võtmed unikaalsed) ja lisatav arv *k*.

Tulemus: tipp arvuga *k* lisatud antud kahendotsimispuusse;  
kui arv *k* juba leitud, siis ei tehta midagi.

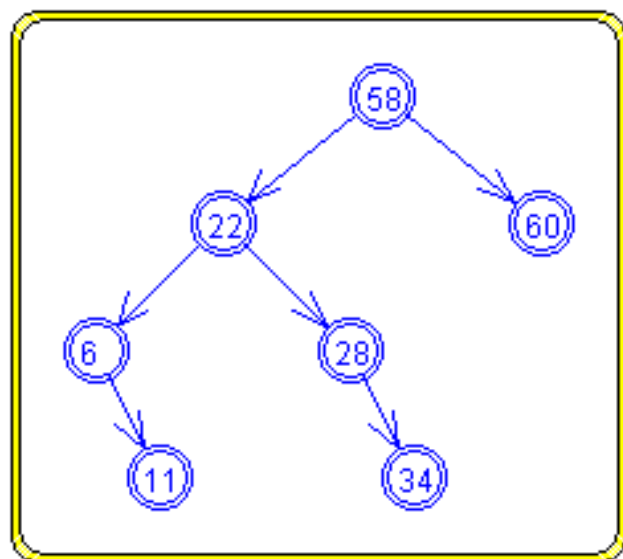
Java

```
Tipp t = new Tipp(k + "[0;0]");  
-- tehtud uus tipp t märgendiga k ja (suva)koordinaatidega  $x = 0; y = 0$ 
```

# Testi tulemuse näide

Lisatav arv: 59  
On KOP = true

C:\Users\kiho\AlgPython\workspace2014\xxx.txt



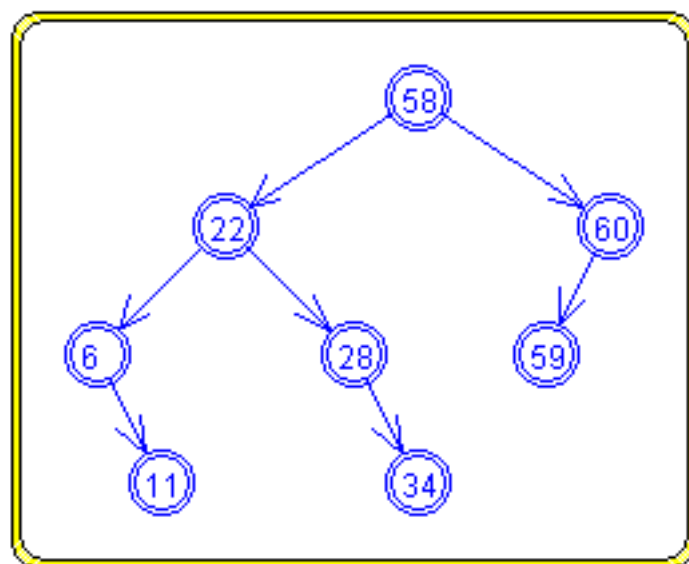
## TABEL

- 1) 58 -->2 -->5
- 2) 22 -->3 -->4
- 3) 6 -->6
- 4) 28 -->7
- 5) 60
- 6) 11
- 7) 34

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremköps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasak klõps.  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustipp.  
GRAAF Ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.

FAIL: C:\Users\kiho\AlgPython\workspace2014\xxx.txt Värskendamine: pidev.

C:\Users\kiho\AlgPython\workspace2014\xxx.txt



## TABEL

- 1) 58 -->2 -->5
- 2) 22 -->3 -->4
- 3) 6 -->6
- 4) 28 -->7
- 5) 60 -->8
- 6) 11
- 7) 34
- 8) 59

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremköps väljaspool graafi ala.

TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasak klõps. Eemaldamine: valida algustipp.  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustipp.  
GRAAF Ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.

FAIL: C:\Users\kiho\AlgPython\workspace2014\xxx.txt Värskendamine: pidev.

## KOP\_6. Eemaldamine kahendotsimispuust

### KOP\_6-1. Tipu eemaldamine [Õpik, lk 31]

Kirjutada ja testida meetod järgmise ülesande lahendamiseks.

Antud:  $kop$  - kahendotsimispuu,  $t$  - selle tipp,  $y$  - tipu  $t$  ülemus ( $\wedge$ , kui  $t$  on  $kop$  juurtipp).

Tulemus: tipp  $t$  eemaldatud kahendotsimispuust  $kop$ .

### KOP\_6-2. Arvu eemaldamine

Kirjutada ja testida funktsioon järgmise ülesande lahendamiseks.

Antud:  $kop$  - kahendotsimispuu,  $k$  – otsitav arv.

Tulemus: tipp arvuga  $k$  (kui leidus) eemaldatud kahendotsimispuust  $kop$ .

## KOP\_7. AVL-puu tasakaalustamise võte

Kirjutada ja testida meetod järgmise ülesande lahendamiseks.

Antud:

- kahendotsimispuu *kop* AVL-rikkega ühes tipus, nagu [Kiho 2003, Joonis 2.4 või selle peegeldus]
- kolm tippu – rikkega tipp *a* (allaviidav), selle alluv *b* (ülesviidav), tipu *a* ülemus *y* ( $\wedge$ , kui *a* on *kop* juurtipp).

Tulemus: antud kahendotsimispuu *kop* tipus *a* sooritatud tasakaalustamise võte.

### Testi tulemuse näide

AVL-rikkega tipu (a) nr: 2  
selle alluva tipu (b) nr: 4  
tipu a ülemuse nr: 1

Antud

Tasakaalustatud

C:\Users\kiho\AlgPython\workspace2014\kop1.txt

**TABEL**

1) 35	-->2 -->8
2) 8	-->3 -->4
3) 5	
4) 20	-->5 -->6
5) 10	
6) 30	-->7
7) 25	
8) 70	-->9 -->15
9) 55	-->10 -->13
10) 45	-->11 -->12
11) 40	
12) 50	
13) 65	-->14
14) 60	
15) 85	-->16 -->18
16) 80	-->17
17) 75	
18) 95	-->19
19) 90	

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga. Eemaldamine: parem klõps valitud tipul (2 x parem klõps)  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustipp, seejärel lõpptipp.  
GRAAF Ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.

FAIL: C:\Users\kiho\AlgPython\workspace2014\kop1.txt Värskendamine: pidev.

C:\Users\kiho\AlgPython\workspace2014\yyy.txt

**TABEL**

1) 35	-->4 -->8
2) 8	-->3 -->5
3) 5	
4) 20	-->2 -->6
5) 10	
6) 30	-->7
7) 25	
8) 70	-->9 -->15
9) 55	-->10 -->13
10) 45	-->11 -->12
11) 40	
12) 50	
13) 65	-->14
14) 60	
15) 85	-->16 -->18
16) 80	-->17
17) 75	
18) 95	-->19
19) 90	

MÄRGENDIKÄITLUS: VÄLJAS Sisselülitamine: paremkõps väljaspool graafi ala.  
TIPP Lisamine: vasak klõps. Valimine: parem klõps. Vedamine: vasakuga. Eemaldamine: parem klõps valitud tipul (2 x parem klõps)  
KAAR Lisamine: valida algustipp, seejärel lõpptipp. Eemaldamine: valida algustipp, seejärel lõpptipp.  
GRAAF Ilma silmuste ja multikaarteta, tipumärgendid tühikuteta.

FAIL: C:\Users\kiho\AlgPython\workspace2014\yyy.txt Värskendamine: pidev.

```
projekt: AlgJavaWorks2015oCDB
Kahendpuu alla_üles(Kahendpuu kop, Tipp v, Tipp w, Tipp vy) sooritada tasakaalustamise võte
spetsifikatsioon
boolean juurtipus = (vy==null);»
Tipp keskmine;»
if (kop.paremAlluv(v) == w) Joonis 2.4
keskmine = kop.vasakAlluv(w); II
else Joonis 2.4 peegel
keskmine = kop.paremAlluv(w);»
» korraldada ümber mõned kaared:
Kaar k;»
if (!juurtipus)
k = kop.kaar(v, w);»
k.seadaLõpptipp(v);»
k.seadaAlgustipp(w);»
if (keskmine != null)»
k = kop.kaar(w, keskmine);»
k.seadaAlgustipp(v);»
» seada "keskmine" v ja w vahele (topeltvõtte puhul vajalik):
nihutadaH(kop, keskmine, (v.x()+w.x())/2 - keskmine.x());»
if (juurtipus)»
» w jääb juureks, tuua esikohale tippude seas:
kop.esikohale(w);»
return kop;»
void nihutadaH(Kahendpuu kp, Tipp t, int xnihe)
```

```
Amadeus - <noname59>*
Fail Toimeti Lisada Minna Teisendus Raam Erisoovid Abi
spetsifikatsioon
»
» Antud: kop - kahendotsimispuu,
» . tipp v - allaviidav,
» . tipp w - ülesviidav,
» . vy - tipu v ülemus või null , kui kui v on kop juurtipp.
» . Joonistel [Kiho 2003, joonis 2.4 või selle peegeldus] ja
» . [Kiho 2003, joonis 2.5 või selle peegeldus]
» . tähistab allaviidavat tippu alla-nooleke ja
» . ülesviidavat tippu üles-nooleke.
» Tulemus: puus kop sooritatud tasakaalustamise võte,
» . vt nt [Kiho 2003, joonis 2.4]
» . tagasatakse (muudetud) kop
spetsifikatsioon
```

```
Amadeus - <noname60>*
Fail Toimeti Lisada Minna Teisendus Raam Erisoovid Abi
void nihutadaH(Kahendpuu kp, Tipp t, int xnihe) abiks
» Antud: kop - kahendpuu, t - selle tipp,
» . xnihe - nihke suurus;
» Tulemus: antud kp tipust t algava alampuu tippude
» . x-koordinaadile liidetud xnihe
if (t != null)»
Tipp vas = kp.vasakAlluv(t);»
Tipp par = kp.paremAlluv(t);»
nihutadaH(kp, vas, xnihe);»
nihutadaH(kp, par, xnihe);»
t.seadaKoordinaadid(t.x()+xnihe, t.y());»
```