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

6. KKP Kahendkuhi. Pakkimine

Mõisteid

Kuhi. Kahendkuhi [Õpik, ptk 3.4]

Eelistusjärjekord (Java: *PriorityQueue**)

Muutuvate eelistustega eelistusjärjekord (Java *PriorityQueue* seda ei toeta)

Huffmani puu

Prefikskood

Teksti pakkimine / lahtipakkimine

---

* Kasutamise näide http://kodu.ut.ee/~kiho/ads/fall18/Abimaterjal/PriorQ.java

Harjutusülesanded

KK_1. Kahendkuhja abioperatsioonid
KK_2. Kahendkuhja kontroll
KK_3. Kahendkuhja käitlemine
KK_4. Pöördkahendkuhja käitlemine
KK_5. Pöördkahendkuhja näiteklass

KK_6. Kahendkuhja klass

KKP_1. Huffman
KKP_2. Prefikskood
KKP_3. Pakkimine ja lahtipakkimine

KK_1. Kahendkuhja abioperatsioonid (vt ka [Õpik, lk 52])
Kirjutada ja testida meetodid järgmiste ülesannete lahendamiseks.

Antud: kahendkuhi *a* listina ja indeks *k* sellel.
Tulemus: *k*-nda tipu vasaku alluva indeks või Λ, kui vasakut alluvat ei ole.

Antud: kahendkuhi *a* listina ja indeks *k* sellel.
Tulemus: *k*-nda tipu parema alluva indeks või Λ, kui paremat alluvat ei ole.

Antud: indeks *k* mingil kahendkuhja kujutaval listil, $k > 0$.
Tulemus: *k-nda* tipu ülemuse indeks.

Antud: list *a* - kompaktse kahendpuu kujutis tasemete kaupa.
Tulemus: *a*-le vastav kahendpuu. (Vt ka ül KP_9)

Antud: kompaktne kahendpuu *kp.*
Tulemus: *kp* tipumärgendite list tasemete kaupa. (Vt ka ül KP_8)

KK_2. Kahendkuhja kontroll
Kirjutada ja testida meetod järgmise ülesande lahendamiseks.
Antud: list *a*.
Tulemus: kontrollitud, kas *a* on kahendkuhja kujutis.

KK_3. Kahendkuhja käitlemine
Kirjutada ja testida meetodid järgmiste ülesannete lahendamiseks.

Antud: kahendkuhi *a* listina ja indeks *i* sellel; kahendkuhi *a* võib olla rikutud seoses *a*[*i*] vähenemisega.
Tulemus: kahendkuhi *a* parandatud. Vt: Õpik, Joonis 3.8]

Antud: kahendkuhi *a* listina ja indeks *i* sellel; kahendkuhi *a* võib olla rikutud seoses *a*[*i*] suurenemisega.
Tulemus: kahendkuhi *a* parandatud. Vt: [Õpik, Joonis 3.9]

Antud: kahendkuhi listina, arv *x* – lisatav.
Tulemus: *x* lisatud antud kahendkuhja. Vt: [Õpik 2003, lk 54]

Antud: kahendkuhi listina.
Tulemus: suurima elemendi väärtus, suurim element on kuhjast eemaldatud. Vt: [Õpik, Joonis 3.10]

Antud: list *a* ja indeks *i* sellel.
Tulemus: tipus *a*[*i*] ja tema järeltulijates kirjed (arvud) ümberpaigutatud nii, et nende jaoks kehtib kuhjaomadus.
Rekursiivne, vt: [Õpik, Joonis 3.11]

KK_4. Pöördkahendkuhja käitlemine
Kirjutada ja testida ülal (KKP_3) loetletud meetodid pöörd-kahendkuhja jaoks .

KK_5. Pöördkahendkuhja näiteklass

Täita lüngad mallis *KahendkuhiDq.java*:

1. Alluvusmeetodid. Kompileerida ja käivitada.
2. Meetod *puuna*. Dekommenteerida vastavad 3 rida testi osas (peameetodis). Kompileerida ja käivitada.
3. Meetodid *viia_alla* ja *võtta_tipuVõti*. Dekommenteerida viimased read testi osas. Kompileerida ja käivitada.

KK_6. Kahendkuhja klass

Lähtudes mallist *KahendkuhiMall.java* koostada kahendkuhja (abstraktne) klass.

KKP_1. Huffman [Õpik, ptk 5.2]
Kirjutada ja testida meetod järgmise ülesande lahendamiseks.
Antud: tekst *s* (sõnena).
Tulemus: Huffmani algoritmiga leitud koodipuu teksti *s* jaoks.
Soovitus. Kuna *s* võib sisaldada suvalisi sümboleid, sh ka erisümboleid, nagu tühemikud ning [ ] ; < jmt,
siis koodipuu lehtippudes tuleks hoida mitte sümbolit vaid selle arvkoodi (sõnena).
Kui *sü* on ühesümboliline sõne (nt ”[”), siis koodipuu lehttipu *t* märgendiks seada sümboli ('[') arvkood sõne kujul (”91”):

```
t.seadaMärgend("" + (int)(sü.charAt(0)));
```

NB! Seni veel ülemuseta tippude frondi pidamiseks kasutada tippude pöörd-kahendkuhja!

Koostada ka eraldi meetod *sagedustabel*:

```
static Hashtable<Character, Double> sagedustabel(String s){
   // Antud: tekst sõnena s
   // Tulemus: luuakse ja tagastatakse sümbolite esinemissageduste tabel,
   //           tabeli element: <sümbol | sagedus>
   int n = s.length();
   Hashtable<Character, Double> tabel = new Hashtable<Character, Double>();
   // loendada sümbolite kordused   (tabeli element: <sümbol | kordusi>) :
//.............................
   // kordused asendada sagedustega (tabeli element: <sümbol | sagedus>) :
//.............................
   return tabel;
}//sagedustabel
```

```
static Kahendpuu huffman(Hashtable<Character, Double> sagedustabel){
   // Antud: sümbolite esinemissageduste tabel
   // Tulemus: konstrueeritakse ja tagastatakse vastav Huffmani koodipuu
   // NB! Fronti Q peetakse tippude pöörd-kahendkuhjana!
   //     Q elemendi (kirje) tüüp on Tipp.
   Kahendkuhi Q = new Kahendkuhi(false) { // front (veel ülemuseta tipud)
       // konkretiseerida abstraktne meetod 'võti' (võtmeks arv tipu väljalt "f" )
      };
   };
   // teha frondi tipud ja vastav isoleeritud tippudega graaf g:
   Graaf g = new Graaf();
   for (char c : sagedustabel.keySet()){
      Tipp t = new Tipp("" + (int)c); // tipu märgendiks sümboli c arvkood
      t.seadaVäli("f", "" + sagedustabel.get(c)); // väljale "f" selle sümboli sagedus
      Q.lisadaKirje(t); // tipp t panna fronti
      g.lisada(t);      // tipp t panna graafi
   }//for
   while(Q.n() > 1){ // Õpik, joonis 5.6
     //.............................
   }//while
   Tipp juur = Q.võttaTipust(); // ainus fronti jäänud tipp
   g.esikohale(juur); // paigutada esikohale (saab kahendpuu juurtipuks)
   return new Kahendpuu(g); // graaf g kahendpuuna
}//huffman
```

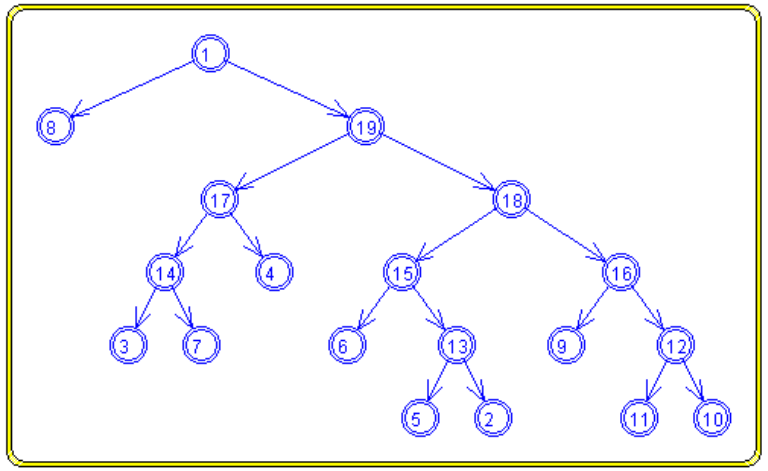
KKP_1

```
»
 –
»»
  import ee.ut.kiho.aa.graaf.*;»
  import java.util.Hashtable;»
»»
  String tekst; // kodeeritav näitetekst;»
  tekst = "ABC.";»
  tekst = "[vanapagana]vanaema.";»
  //tekst = "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
  Aenean luctus nibh a lacus lacinia, id hendrerit orci viverra. Vivamus lectus magna.";»
  //tekst = "Lorem ipsum dolor sit amet, consectetur adipiscing elit,
  sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
  Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
  nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit
  in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
  cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.";»
»»
  System.out.println("Antud tekst: \n" + tekst);»
»»
» TEST sagedustabel:
  Hashtable<Character, Double> sagedused = sagedustabel(tekst);  rida, nt:   <'a' | 0.35>
  System.out.println("\nSagedustabel: \n" + sagedused);»
»»
» TEST huffman:
  Kahendpuu kp = huffman(sagedused);»
  kp.faili("xHuff.txt");»
  Graaf.kuvada("xHuff.txt");»
    »
  Kahendpuu huffman(Hashtable<Character, Double> sagedustabel)
  Hashtable<Character, Double> sagedustabel(String s)
```

```
Antud tekst:
[vanapagana]vanaema.

Sagedustabel:
{[=0.05, p=0.05, n=0.15, m=0.05, g=0.05, e=0.05, a=0.4, v=0.1, ]=0.05, .=0.05}
```

C:_ISIKLIK\ownCloud\AlgJava\xHuff.txt



TABEL
1) f=1.0; -->8 -->19
2) f=0.05;91
3) f=0.05;112
4) f=0.15;110
5) f=0.05;109
6) f=0.05;103
7) f=0.05;101
8) f=0.4;97
9) f=0.1;118
10) f=0.05;93
11) f=0.05;46
12) f=0.1; -->11 -->10
13) f=0.1; -->5 -->2
14) f=0.1; -->3 -->7
15) f=0.15; -->6 -->13
16) f=0.2; -->9 -->12
17) f=0.25; -->14 -->4
18) f=0.35; -->15 -->16
19) f=0.6; -->17 -->18

KKP_2. Prefikskood

Kirjutada ja testida rekursiivne meetod järgmise ülesande lahendamiseks.

Antud: koodipuu *kdp*, selle tipp *t* ja 0-1-rada *kdp* juurtipust tipuni *t* ning prefikskoodide (täiendatav) paisktabel *d*

Tulemus: iga lehttipu *u* korral paisktabelisse *d* lisatud paar < sümbol tipus *u* | leitud 0-1-rada *kdp* juurtipust tipuni *u* >

Vihje. Tipu *u* märgendiks oleva (sõnekujul) arvkoodi teisendamine vastavaks sümboliks:

```
(char)Integer.parseInt(u.nimi())
```

```
static void prefikskood(Kahendpuu kdp, Tipp t, String tee, Hashtable<Character, String> d){
    // Antud: Huffmani koodipuu kdp, selle tipp t ja
    //        tee -- 0-1-rada (sõnena) kdp juurtipust tipuni t ning
    //        d -- täiendatav prefikskoodide tabel (elemendid kujul <sümbol | prefikskood>)
    // Eeldus:  lehttipus on märgendiks üks sümbol (arvkoodina)
    // Tulemus: iga lehttipu u korral täiendatud tabel d
    //          elemendiga    < sümbol tipus u | 0-1-rada kdp juurtipust tipuni u >
    if (t == null) // baasjuht
        return;
    if (kdp.aste(t) == 0){ // kui t on lehttipp puus kdp
        String c = (char)(Integer.parseInt(t.nimi())); // arvkoodist sümboliks
        d.put(c, tee); // element  < c | tee >  ==> tabelisse
    }
    prefikskood(kdp, kdp.vasakAlluv(t), tee+"0", d); // edasi vasaku alluvaga
    //.....................................   // edasi parema alluvaga
}//prefikskood
```

KKP_2

```
»
_
» Ülesanne KKP_2. Prefikskood
»»
  import ee.ut.kiho.aa.graaf.*;»
  import java.util.Hashtable;»
»»
  TEST
    Eeldab, et jooksvas kaustas, failis "xHuff.txt" on Huffmani koodipuu (ülesandest KKP_1)
    Kahendpuu hpuu = Kahendpuu.failist("xHuff.txt");  sisestada
    hpuu.eemaldadaTipuväljad();  sageduse väljad "f"
    hpuu.faili("xKoodipuu.txt");»
    Graaf.kuvada("xKoodipuu.txt", true);»
   »»
    Hashtable<Character, String> kooditabel = new Hashtable<Character, String>();»
    prefikskood(hpuu, hpuu.juur(), "", kooditabel);»
   »»
    println("\tK o o d i t a b e l");»
    println("arvuna\tsümbol | prefikskood");»
    println("\t--------------------");»
    »
   * for(Character c : kooditabel.keySet())»
    »
      println((int)c + "\t  " + c + "     | " + kooditabel.get(c));»

  TEST
»»
 void prefikskood(Kahendpuu kdp, Tipp t, String tee, Hashtable<Character, String> d)
```

arvuna	K o o d i t a b e l sümbol	prefikskood
91	[	11011
112	p	1000
110	n	101
109	m	11010
103	g	1100
101	e	1001
97	a	0
118	v	1110
93	]	11111
46	.	11110

KKP_3. Prefiks-kodeerimine ja dekodeerimine

Kirjutada ja testida meetodid järgmiste ülesannete lahendamiseks.

I.
Antud: tekst *s* (sümbolite sõne).
Tulemus: tagastatakse
nii Huffmani algoritmi abil leitud prefikskoodide tabel teksti *s* jaoks
kui ka selle põhjal prefikskoodides kodeeritud *s* (0-1 sõnena).

II.
Antud: sümbolite prefikskoodide tabel ja selle põhjal prefikskoodides kodeeritud tekst (0-1 sõnena).
Tulemus: tagastatakse dekodeeritud tekst (sümbolite sõne).

KKP_3

```
» Ülesanne KKP_3. Teksti prefiks-kodeerimine ja dekodeerimine
»»
  import ee.ut.kiho.aa.graaf.*;»
  import java.util.Hashtable;»
»»
 String tekst;   kodeeritav näidistekst
 tekst = "[vanapagana]vanaema.";»
 //tekst = "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
 Aenean luctus nibh a lacus lacinia, id hendrerit orci viverra.
 Vivamus lectus magna.";
 System.out.println("\nAntud tekst: \n" + tekst);»
»»
 Object[] kood = kodeerida(tekst);»
 println("\nKodeeritud tekst:\n" + kood[1]);»
»»
 String  s = dekodeerida((Hashtable<Character, String>)kood[0], (String)kood[1]);
 println("\nDekodeeritud tekst:\n" + s);»
»»
```

Object[] kodeerida(String tekst)

String dekodeerida(Hashtable<Character, String> kooditabel, String pakitudTekst)

Kahendpuu huffman(Hashtable<Character, Double> sagedustabel)

Hashtable<Character, Double> sagedustabel(String s)

void prefikskood(Kahendpuu kdp, Tipp t, String tee, Hashtable<Character, String> d)

```
Antud tekst:
[vanapagana]vanaema.

Kodeeritud tekst:
1101111100101010000110001010111111100101010011101001111O

Dekodeeritud tekst:
[vanapagana]vanaema.
```