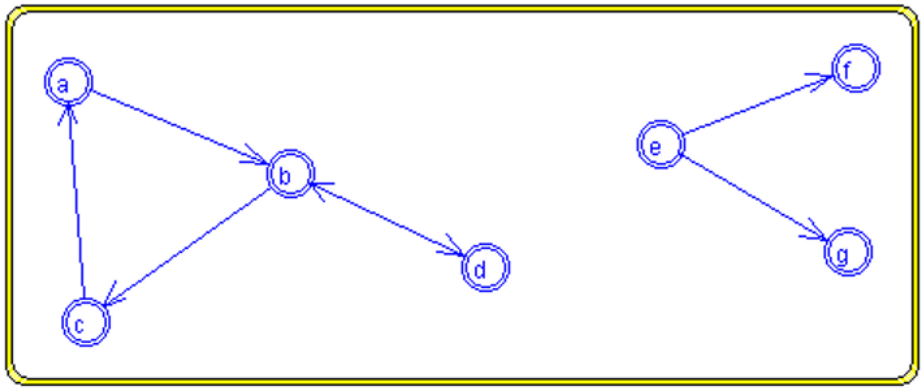
1. Graafid. Puud. Kahendpuud

I Graafi mõiste

Käesolevas piirdume graafi mõnevõrra kitsendatud mõistega (vt ka [1], lk 40).

Ütleme, et on antud **orienteeritud, silmusteta graaf** (edaspidi lihtsalt **graaf**) $G = (V, E)$, kui on antud **tippude** hulk V ja **kaarte** hulk $V = (V \times V) \setminus \{(v,v) | v \in V\}$. Seega kaarteks on järjestatud tipupaarid (v,w), kus $v \in V \wedge w \in V \wedge v \neq w$.

Joonistel kujutame tippe ringikestena, kaari aga nooltena. Tippude paigutus on vabalt valitav. Kaart (v,w) kujutab nool tipu v (kaare algustipu) ringikesest tipu w (kaare lõpptipu) ringikeseni. Joonisel 1 esitatud graafi tippude hulgaks on $\{a,b,c,d,e,f,g\}$ ja kaarte hulgaks $\{(a,b),(b,c),(c,a),(b,d),(d,b),(e,f),(e,g)\}$.



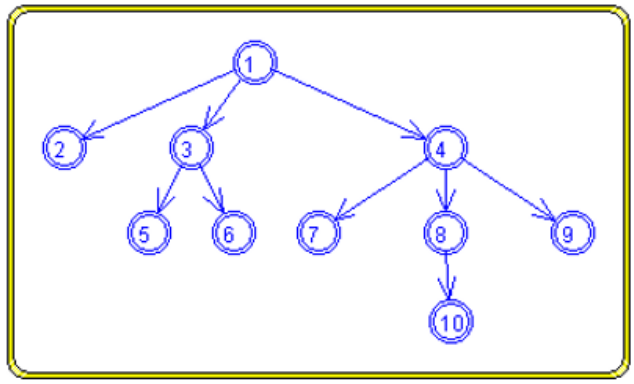
Joonis 1: Seitsmetipuline graaf.

Kaare (v,w) **vastandkaareks** nimetatakse kaart (w,v). Antud näitegraafis leidub üks vastandkaarte paar (b,d) ja (d,b). Vastandkaarte paari kujutame kahesuunalise noolena.

Graafi nimetatakse **sümmeetriliseks**, kui selle igal kaarel leidub vastandkaar, ja **antisümmeetriliseks**, kui kõik kaared on ilma vastandkaarteta. Antud näitegraaf ei ole ei sümmeetriline ega ka antisümmeetriline. Sümmeetriline graaf sobib orienteerimata (suunamata) graafi mudeliks, kus vastandkaarte paar esindab serva vastavas orienteerimata graafis.

II Puu esitus graafina

Puuga seotud tähtsamateks mõisteteks on juur ja alampuu (vt [1], lk 27). Puu loomulikuks graafikujuliseks mudeliks on graaf, milles üks tipp esindab puu juurt ja sellest läheb üks kaar iga selle juure alampuu juurtippu. Selline graaf kujutatakse joonisel nii, et juurtipp paikneb kõigist oma alampuude juurtest ülevalpool.

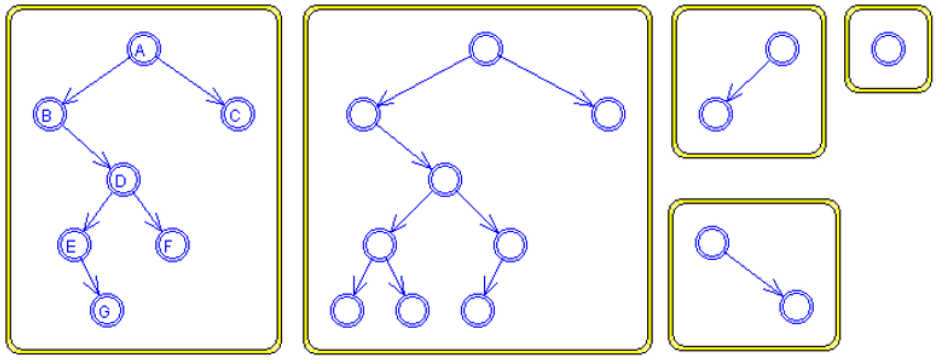


Joonis 2: Puu graafina.

Jooonisel 2 esitatud näitepuus on tippudel 1 ja 4 kolm alampuud, tipul 3 kaks ja tipul 8 – üks alampuu. Ülejäänud tipud $2, 5, 6, 7, 9, 10$ (nn lehttipud) on alampuudeta (alluvateta).

III Kahendpuu esitus graafina

Kahendpuu tipul, mis ei ole lehttipp, on kas vasak alluv või parem alluv või mõlemad (vt ka vt [1], lk 27). Kahendpuu esitatakse joonisel graafina samuti nagu puu, kuid ülemustipu vasak alluv paigutatakse ülemustippu läbivast vertikaalist vasakule ja parem alluv sellest mõttelisest joonest paremale poole.



Joonis 3: Kahendpuude näiteid.

Esimeses kahendpuus joonisel 3 on tipu A vasakuks alluvaks tipp B ja paremaks alluvaks tipp C; tipul on B paremaks alluvaks tipp D, vasak alluv aga puudub.

2. Graafi töötlemise vahendeid keeles Java

Leidub mitmeid graafide käitlemise süsteeme, sealhulgas näiteks küllaltki universaalne ja väga paljude võimalustega raamistik JUNG2. Sellised süsteemid võimaldavad käidelda/visualiseerida suuri graafse struktuuriga andmekogusid.

Käesolevas jaotises tutvustatame lihtsamat, piiratumat õppeotstarbelist paketti *ee.ut.kiho.aa.graaf*, mis on sobivam kasutamiseks just arvutipraktikumis, suhteliselt väikese tippude arvuga graafide korral. Paketis leiduvad vahendid ka puude ja kahendpuude käitlemiseks.

Alljärgnevas mõningaid lisaselgitusi ja näiteprogramme.

I Klass *Tipp*. Tipu info

Graafi tipus hoitakse selle infot sõnekujulise tipukirjena. Tipuga saab seostada (tipukirjes seada) nii tipu nime (identifikaatorit) kui ka nimega lisavälju. Tipukirje lõpus paiknevad ekraanikoordinaadid, millega on määratud selle tipu (ringikese) koht graafi ekraanikuval.

Tipukirje EBNF tähistuses ({X} - X kordub 0 või enam korda, [Y] - Y 0 või 1 kord, st Y võib puududa).

<tipu-kirje> ::= <tipu-märgend><tipu-ekraanikoordinaadid>
<tipu-ekraanikoordinaadid> ::= "[" <x> ";" <y> "]"
<x> ::= **täisarv**
<y> ::= **täisarv**
<tipu-märgend> ::= { <tipu-väli> } [<tipu-nimi>]
<tipu-väli> ::= <välja-nimi> "=" [<välja-väärtus>] ";"
<välja-nimi> ::= **kitsendustega-sõne**
<välja-väärtus> ::= **kitsendustega-sõne**
<tipu-nimi> ::= **kitsendustega-sõne**

kitsendustega-sõne: mittetühi sõne, mis ei sisalda tühikut ega sümboleid ’)’, ’[’, ’]’, >, ’=’, ’;’

täisarv: kümnendsüsteemis, sõnena

Tipu kirje näide:

```
"kaugus=13;B[85;130]"
```

Sellel tipul nimega "B" on väli nimega "kaugus", mille väärtuseks on "13"; tipu ekraanikuva koordinaatideks on $x = 85$ ja $y = 130$.

II Klass *Kaar*. Kaare info

Graafi kaareks on tipupaar – algustipp ja lõpptipp. Kaarega saab seostada lisaks veel sõnekujulise lisainfo – kaare märgendi.

<kaare-märgend> ::= [**kitsendustega-sõne**]

III Graafi tabelesitus

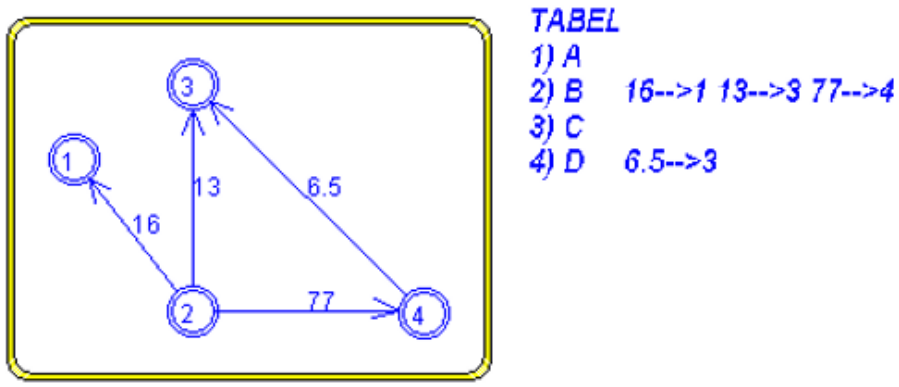
Graafi tabelesitus on sõne, mis kooseb reavahetusmärgiga eraldatud alamsõnedest – tabeli ridadest. Graafi igale tipule vastab parajasti üks rida kujul

$< tipu\ number >$)$< tipu\ kirje > K$

kus K koosneb tühikuga eraldatud kaarekirjeldustest, millest igaüks kirjeldab parajasti ühte sellest tipust väljuvat kaart. Kaarekirjelduseks on sõne kujul

$< kaare\ märgend >$ **-->** $< kaare\ lõpptipu\ number >$

Vt näide joonisel 4.



Joonis 4: Graafi ja selle tabelesituse ekraanikuva.

Graafe säilitatakse (teksti)failides just tabelesituse kujul. Graafi kuvamisel näitatakse joonise kõrval ka selle graafi tabelesitust (ilma tippude ekraanikoordinaatideta).

Joonisel 4 kujutatud graafi tabelesituseks on sõne

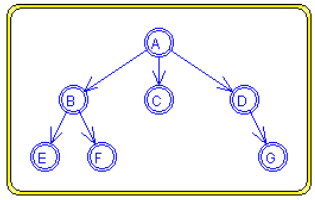
"
1) A[30;66]
2) B[87;139] 16 - -> 1 13 - -> 3 77 - -> 4
3) C[87;30]
4) D[198;140] 6.5 - -> 3
"

IV Klass *Puu*

Klassi *Graaf* alamklass.

Järgmisel leheküljel klassi *Puu* kasutav lihtne näiteprogramm, milles luuakse ja kuvatakse 7-tipuline puu (vt joonis 5).

```
import ee.ut.kiho.aa.graaf.*; // (klassid Puu ja Graaf)

class Puu_Näide00{

   public static void main(String[] args){

      // märgendi kuva stiil tipuringikestes:
      boolean stiil = true; // tipumärgendid
      //stiil = false;      // tipunumbrid

      Puu p = new Puu(new Tipp("A"));
      // p on ühetipuline puu (ainult juurtipp)

      Tipp juur = p.juur(); // ehk  Tipp juur = kp.tipp(0)
      // tipud, mis lisada:
      Tipp b = new Tipp("B");
      Tipp c = new Tipp("C");
      Tipp d = new Tipp("D");
      Tipp e = new Tipp("B");
      Tipp f = new Tipp("C");
      Tipp g = new Tipp("D");

      // kaarte (alluvate) lisamine
         // b juure alluvaks:
      p.lisadaAlluv(juur, b);
         // c juure alluvaks:
      p.lisadaAlluv(juur, c);
         // d juure alluvaks:
      p.lisadaAlluv(juur, d);
         // e tipu b alluvaks:
      p.lisadaAlluv(b, e);
         // b tipu f alluvaks:
      p.lisadaAlluv(b, f);
         // d tipu g alluvaks:
      p.lisadaAlluv(d, g);

      // puu p salvestamine faili ja kuvamine failist:
      p.faili("puu7.txt");
      Graaf.kuvada("puu7.txt", stiil);

      System.out.println("Loodud: seitemetipuline puu,");
      System.out.println("  salvestatud faili puu7.txt");

   }//main
}//class
```



Joonis 5: Väike puu.

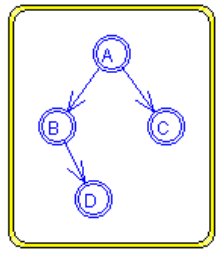
V Klass *Kahendpuu*

Klassi *Puu* alamklass.

Lihtne näiteprogramm, milles luuakse ja kuvatakse 4-tipuline kahendpuu (vt joonis 6):

```
import ee.ut.kiho.aa.graaf.*; // (klassid Kahendpuu ja Graaf)

class Kahendpuu_Näide00{

   public static void main(String[] args){

      // märgendi kuva stiil tipuringikestes:
      boolean stiil = true; // tipumärgendid
      //stiil = false;      //  tipunumbrid

      Kahendpuu kp = new Kahendpuu(new Tipp("A"));
      // kp on ühetipuline kahendpuu (ainult juurtipp)

      Tipp juur = kp.juur(); // ehk  Tipp juur = kp.tipp(0)
      // tipud, mis lisada:
      Tipp b = new Tipp("B");
      Tipp c = new Tipp("C");
      Tipp d = new Tipp("D");

      // kaarte (alluvate) lisamine
         // b juure vasakuks alluvaks:
      kp.lisadaAlluv(juur, b, true);
         // c tipu a paremaks alluvaks:
      kp.lisadaAlluv(juur, c, false);
         // d tipu b paremaks alluvaks:
      kp.lisadaAlluv(b, d, false);
```

```
        // puu kp salvestamine faili ja kuvamine failist:
        kp.faili("kp4.txt");
        Graaf.kuvada("kp4.txt", stiil);

        System.out.println("Loodud: neljatipuline kahendpuu,");
        System.out.println
                          ("        salvestatud faili kp4.txt");

    }//main

}//class
```



Joonis 6: Väike kahendpuu.

VI Klass *Graaf*

Graafi loomiseks on soovitatav kasutada graafi toimetit, vt järgmine jaotis VII. Selle abil saab luua uue graafi (faili) või mingi olemasoleva graafi teisendi.

Graafi **sügavuti läbimisel**, lähtudes mingist tipust "külastatakse" iga (saavutatavat) tippu parajasti üks kord. Vältimaks tippude korduvkülastust, märgistatakse parajasti liikumisteel olevaid tippe.

Üldine graafi sügavuti läbimise algoritmi skeem:

```
läbida_sügavuti(g, t)
--- Antud:  graaf g ja selle tipp t
--- Eeldus: graafis g on märgistatud juba läbitud tipud
--- Tulemus: graafis g märgistatud kõik tipust t saavutatavad,
---          senini veel märgistamata tipud
        panna tipule t märk (märgistada tipp t)
        tipu t iga naabri v korral
                kui tipp v ei ole märgistatud, siis
                        läbida_sügavuti(g, v)
```

Rakendamine.

läbida_sügavuti(g, a) - - - märgistatakse kõik tipust a saavutatavad tipud graafi g sügavuti läbimisel tipust a alustades

VII Graafi toimeti

Töötlusprogrammi töö käigus mingist failist kuvatud graafi saab ekraanil toimetada – muuta tippude asukohti, lisada tippe ja kaari ning muuta tippude ja kaarte märgendeid. Vastavad muudatused tehakse sünkroonselt ka kohe selle graafi failis.

Eraldiseisva programmide *VaataGraafi* ja *VaataGraafiPuuna* otstarbeks on võimaldada graafi (ja selle faili) toimetada väljaspool spetsiifilisi töötlusprogramme. Graafina esitatud puu või kahendpuu toimetamisel tuleb silmas pidada nende joonistele seatud nõudeid: puu tipp paikneb kõigist oma alampuude juurtest ülevalpool ja kahendpuu tipu vasak (parem) alluv seda tippu läbivast vertikaalist vasakul (paremal) pool.

Viited

[1] J. Kiho. *Algoritmid ja andmestruktuurid.* Kolmas, parandatud ja täiendatud trükk. TÜ, 2003, 147 lk. http://dspace.ut.ee/bitstream/handle/10062/16872/9985567676.pdf?sequence=1 (16.04.2017)