

# Mittelineaarsed struktuurid

Puu. Kahendpuu.

Puude läbimine ja realisatsioon.

# Graaf

Ainult lineaarses vormis ei ole võimalik kõiki elulisi andmeobjektide vahelisi seoseid modelleerida.

**Graaf** on üldisem struktuur, mille abil saab modelleerida objektide hulgas paari-kaupa esinevaid suhteid või seoseid.

Erijuhul võib olla ka siin tulemuseks lineaarne struktuur. Üldiselt saame aga üsna suvalise kujuga tippudest ja servadest koosneva võrgu.

Graafist täpsemalt edaspidi.

# Puu

Puu on graafi erivorm. Puus ühendatakse andemobjektid hierarhilisel viisil.

Puu koosneb elementidest, mida nimetatakse **tippudeks** e **sõlmedeks** (*node*), ja seostest tippude vahel, mida nimetatakse **kaarteks** (*edge*). Andmed paigutatakse puu tippudesse.

Elementide paiknemisel puus üksteise suhtes (ehk seostel tippude vahel) on reeglina tähendus.

# Puu definitsioon

Rekursiivne definitsioon *D. Knuth'i* järgi:

**Puu** (*tree*) on lõplik hulk  $T$ , koosnedes ühest või mitmest sõlmest, mis rahuldavad järgmisi tingimusi:

a) eksisteerib üks, teistest erinev sõlm, mis on antud **puu juur** (*root*)

b) teised sõlmed jagunevad  $m$  ( $m \geq 0$ )  
mittelõikuvaks alamhulgaks  $T_1 \dots T_m$  ja iga  
alamhulk on omakorda puu. Hulki  $T_1 \dots T_m$   
nimetatakse antud puu **alampuudeks** (*subtree*)

# Mõisted (1)

**Sõlm** või **tipp** (*node*) – element, millest puu koosneb. Sõlmedes paiknevad andmed. Puu struktuuriga määratakse andmete vahelised seosed.

**Sõlme järk** (*degree*) – sõlme alampuude arv.

**Puu järk** – suurim võimalik sõlme järk (ühe sõlme alampuude arv) antud puus.

Puud, mille kõigil sõlmedel on maksimaalne alampuude arv piiratud arvuga  $n$ , nimetatakse **n-järku puuks**.

Sõlmede paiknemise selgitamiseks kasutatakse sugulussuhetega seotud mõisted (mida ei ole siin mõtet pikemalt kirja panna).

# Mõisted (2)

**Juur** (*root*) – sõlm, millele ei eelne mitte ühtegi sõlme.

**Leht** (*leaf*) – sõlm, mille järk on 0 (talle ei järgne ühtegi sõlme)

Puu sõlmed jagunevad paiknemishierarhia järgi **tasemetesse** (*level*). Juur on tasemel 0, juure lapsed tasemel 1 jne.

**Puu kõrgus** (*height*) määratakse vastavalt tasemete arvule. See on suurim kaugus puus juure ja lehe vahel.

# Mõisted (3)

**Järjestamata puu** (*unordered tree*) on puu, kus ühe sõlme / tipu laste omavaheline järjestus ei ole määratud (ei oma tähendust).

Järjestamata puu on **orienteeritud** (*oriented tree*) – hierarhilised seosed on olemas.

**Järjestatud puus** (*ordered tree*) on ühe tipu laste järjekord mingil viisil määratud, järjekorral on tähendus.

Tavaliselt on puud järjestatud ("puu") ja tema vastandiks on "orienteeritud puu" (järjestamata).

## Mõisted (4)

Kõik antud sõlmest kõrgemal (juure pool) olevad sõlmed on sõlme **eellased** (*ancestor*).

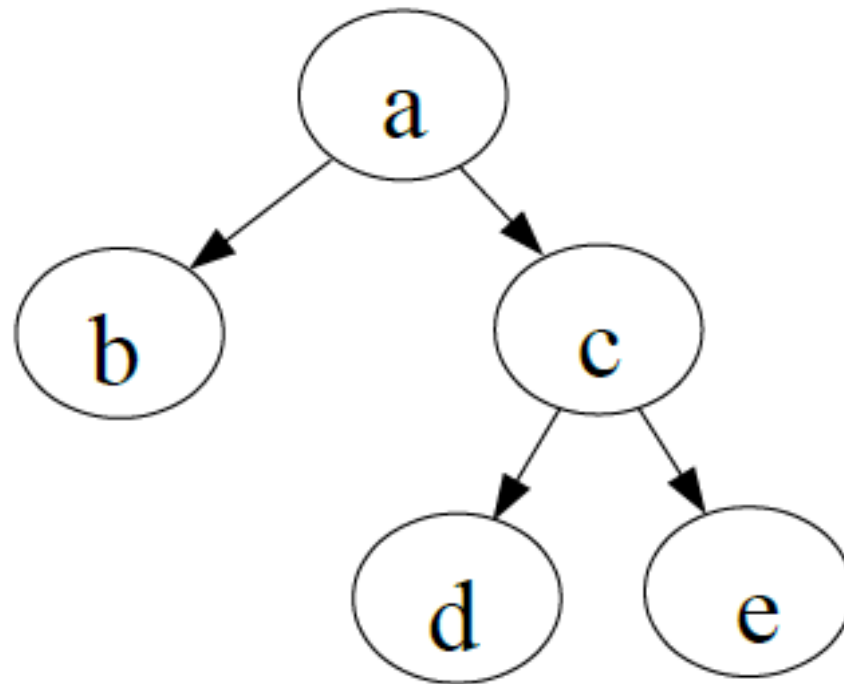
Kõik antud sõlmest allpoololevad sõlmed on sõlme **järglased** (*descendant*).

**Tee** (*path*) on ainus, lühim kaarte järgnevus, mis viib puu juurest leheni. Puu juure ja lehe vahel on ainult üks tee.

Tee võib olla määratud ka suvalise sõlmeni.



# Näide puust



# Puude esitamine

**Joonisena** – lihtsaim mõista. Juur on reeglina ülal.

**Tekstina** (näited eelmise slaidi kohta):

**Sulgavaldis:** puu erinevad tasemed  
"sulustatakse" nt (a(b)(c(d)(e)))

**Dewey kümnendesitus** (*Dewey decimal notation*): süsteem sarnaneb kirjutise peatükkide nummerdusele (kajastub nii omavaheline järjestus kui hierarhia). nt 1 a; 1.1 b; 1.2 c; 1.2.1 d; 1.2.2 e

# Kahendpuu

**Kahendpuu** (*binary tree*) on sõlmede lõplik hulk, mis on tühi või mis koosneb juurest ja kahest mittelõikuvast alampuust, mida nimetatakse antud juure **vasakuks** ja **paremaks alampuuks** (*left and right subtree*).

Kahendpuu igal sõlmel on maksimaalselt kaks alampuud (2-järku puu). On oluline, kas ta on vasak alampuu või parem alampuu.

Eksisteerib **tühja alampuu** mõiste.

Seega kahendpuu ei ole sama kui 2-järku puu.

# Täielik kahendpuu

- Kahendpuu on **täielik** (*perfect / complete*), kui kõik tema lehed paiknevad ühel tasemel ja kõigil ülejäänud tippudel on kaks last.
- Peaagu täielikus kahendpuus võivad lehed puududa vaid viimasel tasemel paremalt poolt.
- Tippude arvudest:
  - tasemel  $n$  on  $2^n$  lehte (juure tase on 0 jne)
  - puus kõrgusega  $h$  on maksimaalselt  $2^{h+1} - 1$  tippu ja  $2^h$  lehte
  - $m$  tipuga puu kõrgus on  $\log_2(m+1) - 1$

# Puude kasutamine

Kui andmete olemus on **hierarhiline**.

Avaldised jt keele osad **süntaksipuuna** (ka HTML DOM).

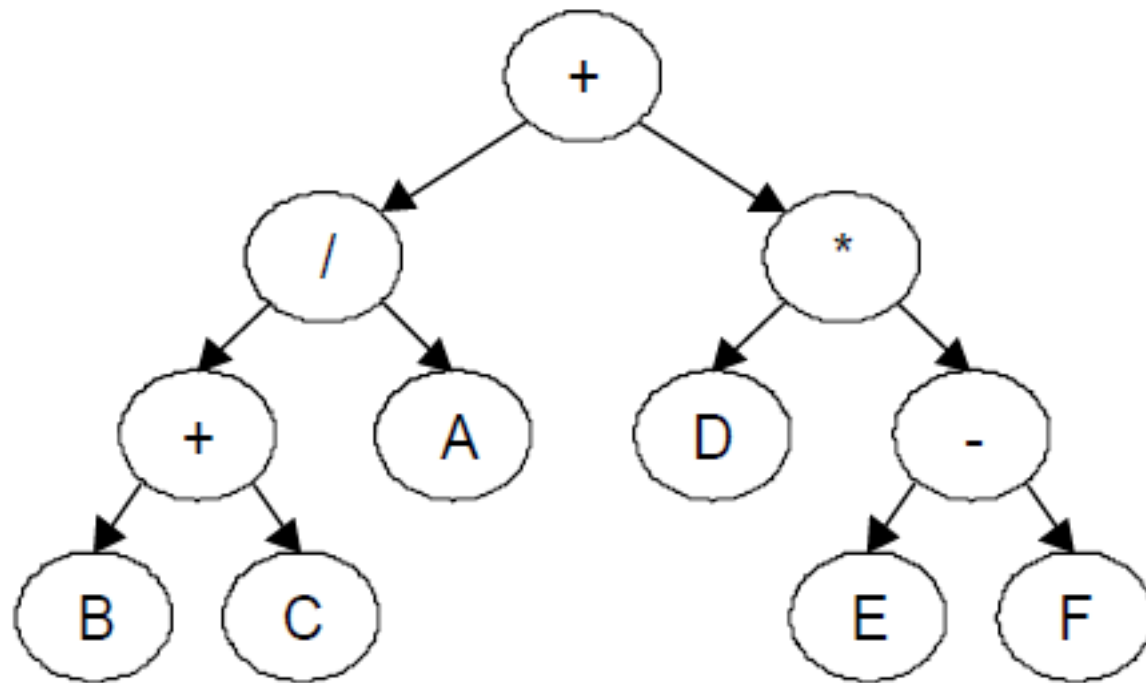
Erinevad **otsimispuud** otsimise kiirendamiseks (kahendotsimispuu, *trie*).

**Kahendkuhi** (*binary heap*) kiireks elementide paigutamiseks ja kättesaamiseks.

Puu sobib aluseks mittelineaarsetele andmestruktuuridele ning mitmed tegevused puus on keerukusega  $O(\log_2 N)$ .

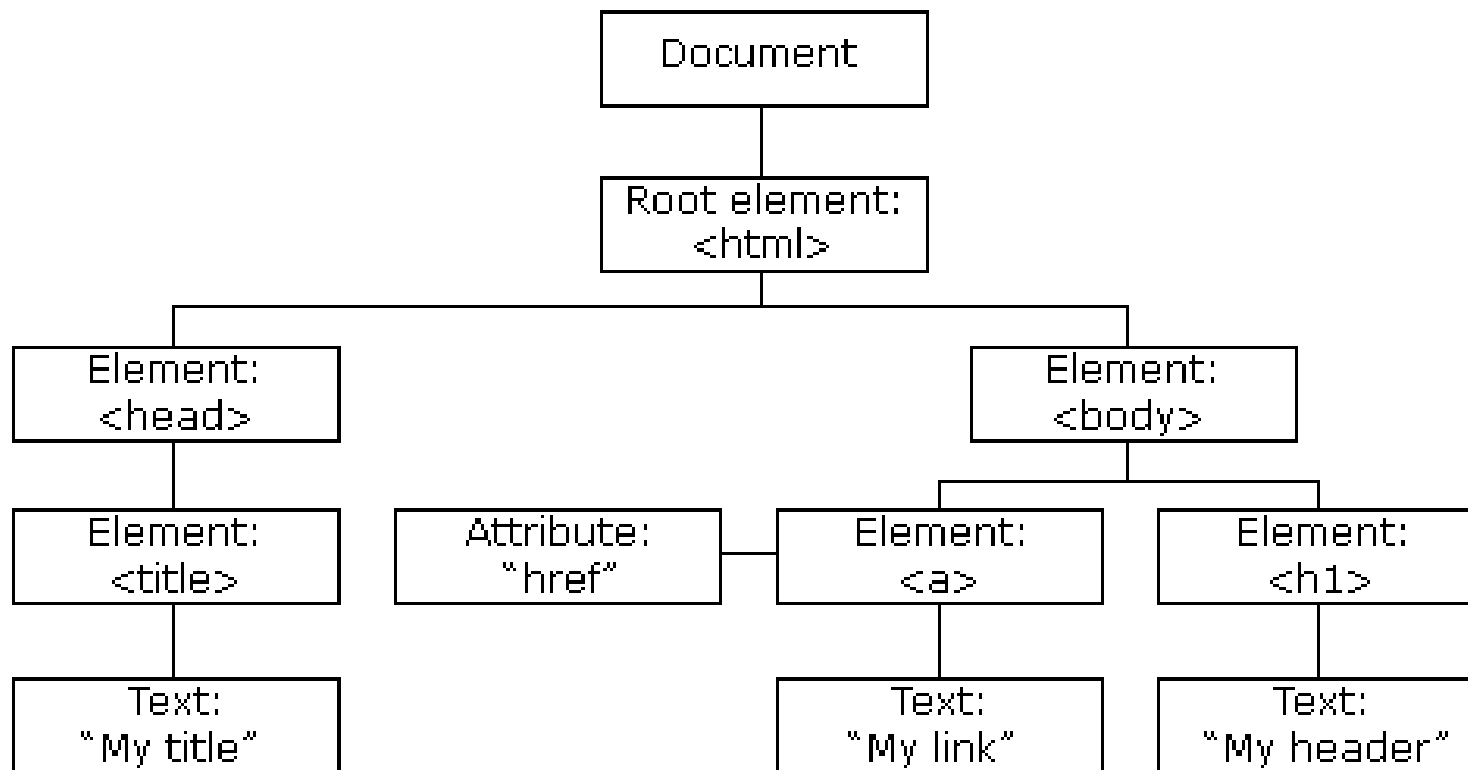
Erinevates puukujulistes struktuurides on elementide paigutamiseks omad reeglid.

# Näidispuu (avaldis)



# HTML-i DOM mudel

[https://www.w3schools.com/whatis/whatis\\_htmlidom.asp](https://www.w3schools.com/whatis/whatis_htmlidom.asp)



# Puu kõigi tippude läbimine

**Süviti otsimine** (*depth-first search*). Tippude läbimise loogika on seotud pinuga.

Pane juur pinusse

Kuni pinu ei ole tühi korda

Võta sõlm pinust (ja trüki välja)

Kui on, lisa pinusse tipu lapsed

**Laiuti otsimine** (*breadth-first search*). Tippude läbimise loogika on seotud järjekorraga.

Pane juur järjekorda.

Kuni järjekord ei ole tühi korda

Võta sõlm järjekorrast (ja trüki välja)

Kui on, lisa järjekorda tipu lapsed



# Kahendpuu läbimine süviti (1)

## Lõppjärjekord (*Postorder e. Endorder*)

1. Kui võimalik, liigu vasakusse alampuusse
2. Kui võimalik, liigu paremasse alampuusse
3. Väljasta (töötle) juur

14. slaidil oleva puu läbimise väljundiks oleks:

B C + A / D E F - \* + .

Selgituseks: "Liigu vasakusse alampuusse" tähendab, et sama 3-sammulist tegevust korratakse antud tipu vasaku lapsega (kui viimane on olemas).

# Kahendpuu läbimine süviti (2)

## Eesjärjekord (*Preorder*)

1. Väljasta (töötle) juur
2. Kui võimalik, liigu vasakusse alampuusse
3. Kui võimalik, liigu paremasse alampuusse

14. slaidil olnud puu läbimine annaks tulemuse:

+ / + B C A \* D – E F

# Kahendpuu läbimine süviti (3)

## **Keskjärjekord** (*Inorder*)

1. Kui võimalik, liigu vasakusse alampuusse
2. Väljasta (töötle) juur
3. Kui võimalik, liigu paremasse alampuusse

Juba tuttav avaldisepuu 14. slaidil annab tulemuse:

$$B + C / A + D * E - F$$

# Puu ja rekursioon

- Puu definitsioon on rekursiivne – puu defineeritakse kasutades mõistet puu.
- Puu on oma olemuselt rekursiivne: puu koosneb juurest ja alampuudest, millest igaüks koosneb juurest ja alampuudest, millest igaüks ...
- Puu tippude süviti läbimisi (kõik kolm varianti) saab realiseerida rekursiivse funktsiooniga.





# Rekursioon

**Rekursiivne funktsioon** on funktsioon, mis iseennast välja kutsub, püüdes selle käigus tavaliselt lahendada ülesande lihtsamat alamjuhtu. (**Rekursioon** vt rekursioon)

Tüüpiline rekursiivne näide faktoriaali leidmisest:

```
int Fact(int n) {  
    int rec;  
    if (n==1)  
        return 1;  
    else {  
        rec = Fact(n-1);  
        return (n*rec);  
    }  
}
```

# Funktsioonide väljakutsed

Funktsiooni väljakutsed pinus	return tagastab 1 või $N \cdot \text{rec}$
Fact(0) $N = 0$	1 (return 1)
Fact(1) $N = 1, \text{rec} = 1$ 	1 (return $N \cdot \text{rec}$ )
Fact(2) $N = 2, \text{rec} = 1$ 	2 (return $N \cdot \text{rec}$ )
Fact(3) $N = 3, \text{rec} = 2$ 	6 (return $N \cdot \text{rec}$ )
main() arv = 3 vastus=6 	

# Veel tegevusi puus

Alati ei ole vaja kõiki sõlmi läbida

- Elementide lisamine (tihti lehtedeks, harvem keskele).
- Elementide kustutamine.
- Paljudel juhtudel on ülesande lahendamiseks vaja läbida üks tee (mis oli tee? - sõlmede jada juurest leheni).
- Kui kaugele jääb leht juurest? - kui puu on täielik, siis  $N$  tipu korral on sammude arv  $\log_2 N$   
Sellest keerukusklass  $O(\log_2 N)$

# Puu realisatsioon

Tavaliselt toimub puu realisatsioon **dünaamiliselt** (analoogiliselt ahelaga).

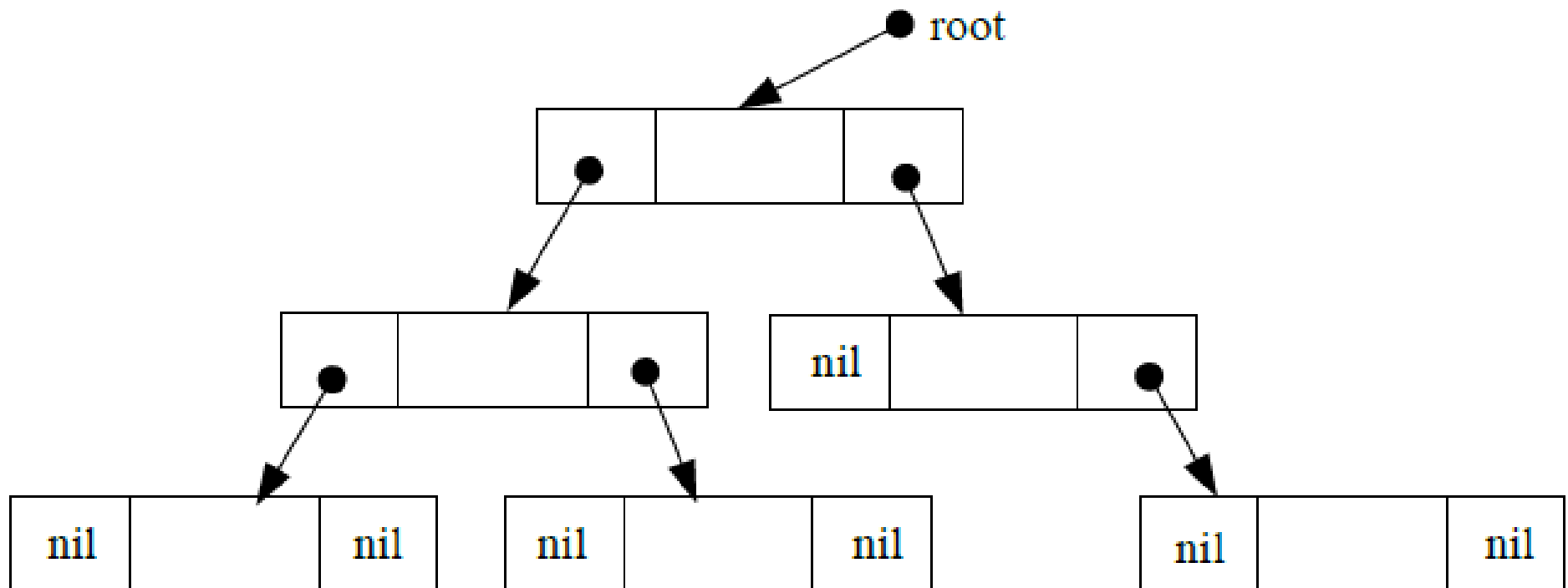
Võimalikud on ka kokkulepped puu realiseerimiseks **massiivina** (nt kahendkuhja puhul).

Koodi on otstarbekam pikast materjalist ja näidetest uurida.

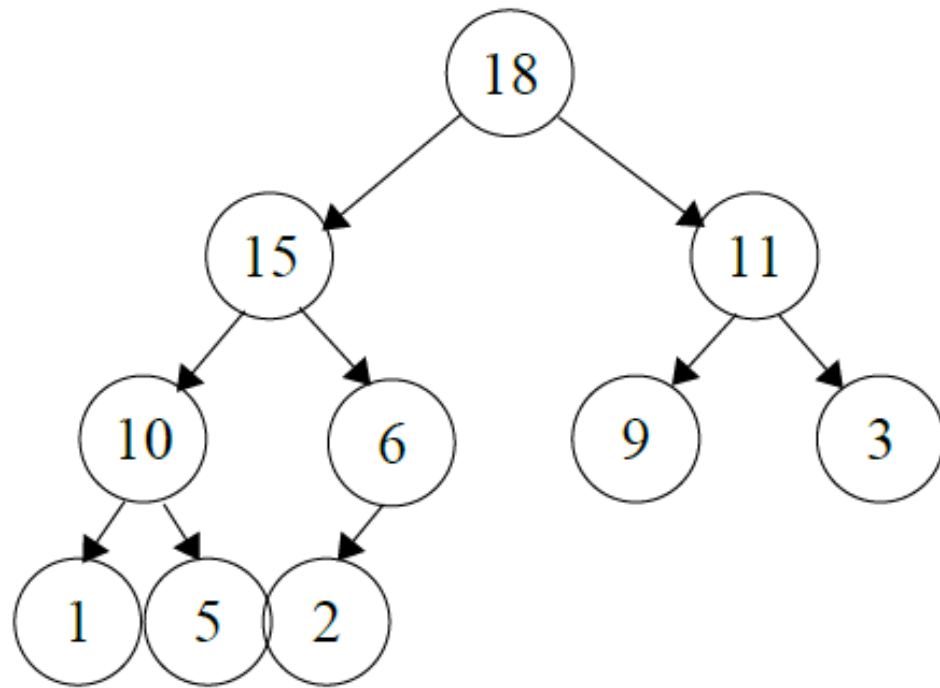
Dünaamilise puu näide on puude materjalis, kahendkuhja on aga kirjeldatud sorteerimise materjalis.



# Dünaamiline realiseatsioon



# Realisatsioon massiiviga (kuhi)



Joonis 1 Kahendkuhi puuna

1	2	3	4	5	6	7	8	9	10
18	15	11	10	6	9	3	1	5	2