Tampereen yliopisto

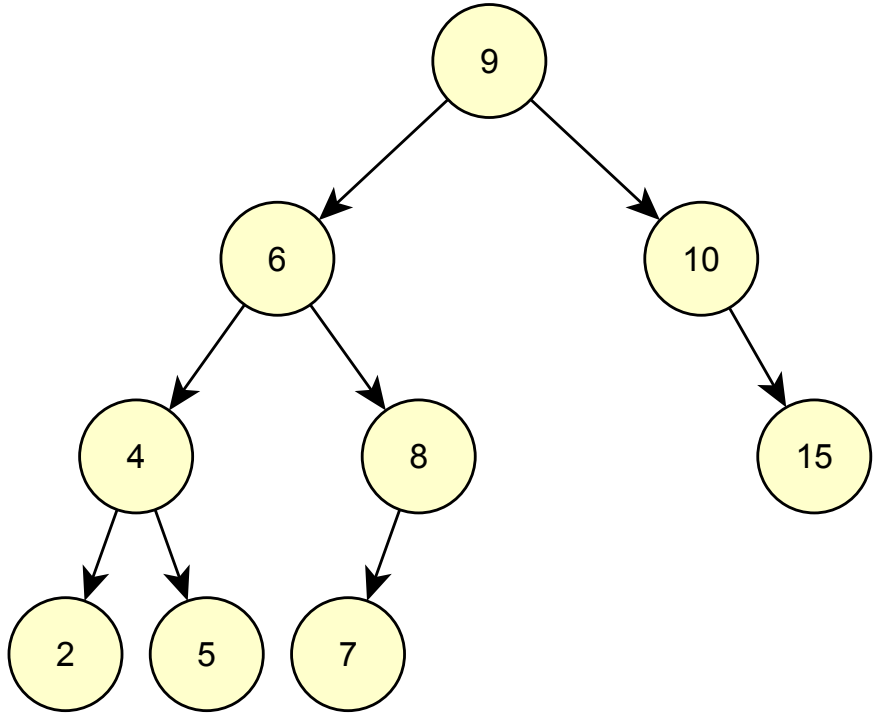
Binäärihakupuut

COMP.CS.300 Tietorakenteet ja algoritmit 1

Matti Rintala (matti.rintala@tuni.fi)

Binäärihakupuu

- Solmulla max. 2 lasta, *vasen* ja *oikea*
- Joka solmun vasemman alipuun alkiot solmua *pienempiä*, oikean *suurempia*
- Ei välttämättä tasapainossa
- Ei siis voi esittää yleensä taulukkona


9
6
10
4
8
15
2
5
7

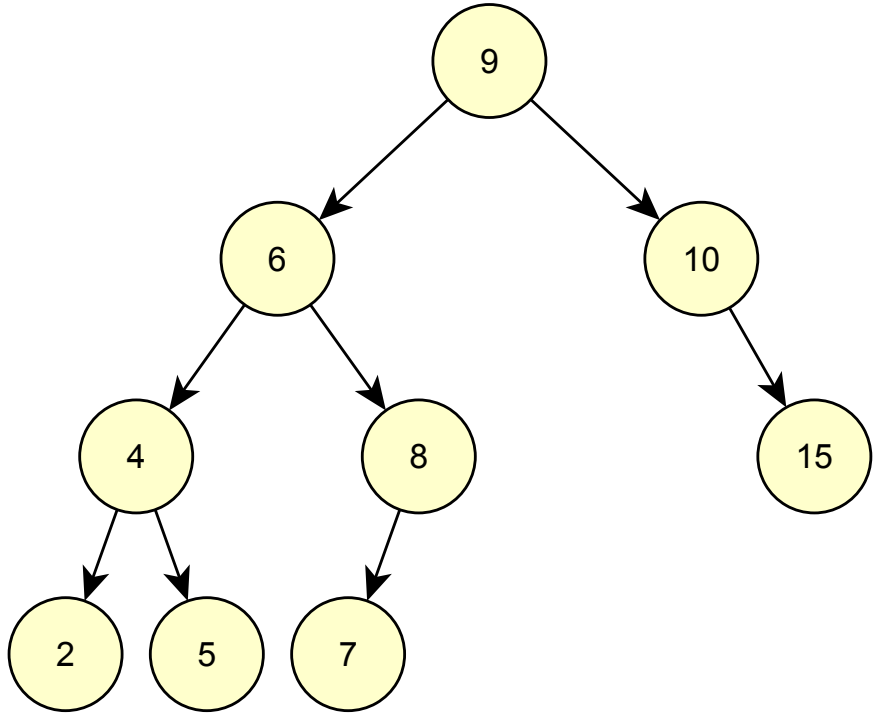
Haku binäärihakupuusta

B-tree-search(*root, value*)

```
1 if root = NIL or root→key = value then      (avain löytyi)
2     return root
3 if value < root→key then                    (jos haetaan pienempää...)
4     return B-tree-search(root→left, value)  (...etsitään vasemmasta alipuusta)
5 else
6     return B-tree-search(root→right, value) (muuten etsitään oikeasta)
```

Haku binäärihakupuusta

```
B-tree-search(root, value)
1 if root = NIL or root→key = value then
2     return root
3 if value < root→key then
4     return B-tree-search(root→left, value)
5 else
6     return B-tree-search(root→right, value)
```


9
6
10
4
8
15
2
5
7

B-tree-search(*root, value*)

```
1 while root ≠ NIL and root→key ≠ value do
2     if value < root→key then
3         root := root→left
4     else
5         root := root→right
6 return root
```

Lisäys bin:hakupuuhun

B-tree-insert(*root, newnode*)

```
1  if root = NIL then
2     ▷ (Puu on tyhjä, lisää newnode juureksi)
3  parent := NIL
4  node := root
5  while node ≠ NIL do                    (mennään alaspäin kunnes puu loppuu)
6     parent := node
7     if newnode→key < node→key then      (valitaan suunta avaimia vertailemalla)
8        node := node→left
9     else
10       node := node→right
11 newnode→parent := parent               (lehtisolmu on uuden solmun vanhempi)
12 if newnode→key < parent→key then       (ja uusi solmu lehtisolmun lapsi...)
13    parent→left := newnode              (...joko vasen...)
14 else
15    parent→right := newnode             (...tai oikea)
16 newnode→left := NIL                    (nollataan uuden solmun lapsiosoittimet)
17 newnode→right := NIL
```

Lisäys bin:hakupuuhun

B-tree-insert(*root, newnode*)

```
1  if root = NIL then
2      ▷ (Puu on tyhjä, lisää newnode juureksi)
3  parent := NIL
4  node := root
5  while node ≠ NIL do
6      parent := node
7      if newnode→key < node→key then
8          node := node→left
9      else
10         node := node→right
11 newnode→parent := parent
12 if newnode→key < parent→key then
13     parent→left := newnode
14 else
15     parent→right := newnode
16 newnode→left := NIL
17 newnode→right := NIL
```

Minimi ja maksimi

B-tree-minimum(*root*)

```
1 while root→left ≠ NIL do
2     root := root→left
3 return root
```

B-tree-maximum(*root*)

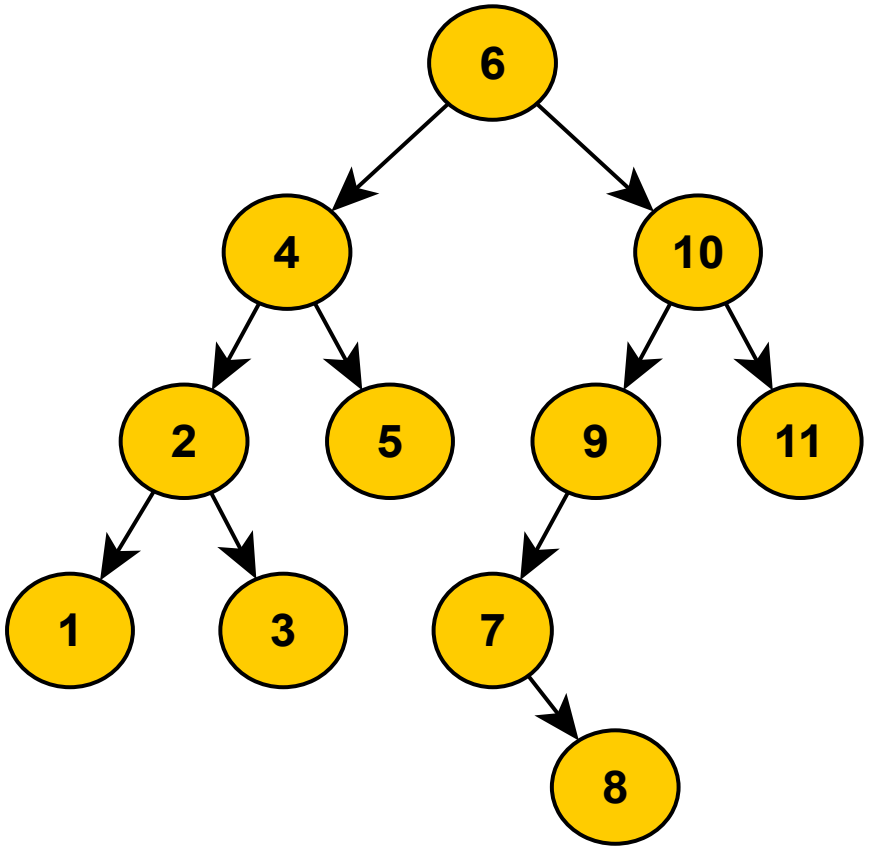
```
1 while root→right ≠ NIL do
2     root := root→right
3 return root
```

Läpikäynti suuruusjärj.

Inorder-tree-walk(*node*)

```
1  if node ≠ NIL then
2      Inorder-tree-walk(node→left-child)
3      ▷ (käsittele alkio node)
4      Inorder-tree-walk(node→right-child)
```

- Binäärihakupuun läpikäynti suuruusjärjestyksessä välijärjestyksellä



Solmua järj. seuraava

B-tree-successor(*node*)

```
1 if node→right ≠ NIL then                          (jos löytyy oikea alipuu)
2     return B-tree-minimum(node→right)             (seuraaja on sen pienin alkio)
3 parent := node→parent                             (muuten lähdetään ylöspäin)
4 while parent ≠ NIL and node ≠ parent→left do      (kunnes tultiin vasemmalta)
5     node := parent
6     parent :=  parent→parent
7 return parent                                     (tulos on löydetty vanhempi)
```

Solmua järj. seuraava

```
B-tree-successor(node)
1 if node→right ≠ NIL then
2     return B-tree-minimum(node→right)
3 parent := node→parent
4 while parent ≠ NIL and node ≠ parent→left do
5     node := parent
6     parent :=  parent→parent
7 return parent
```

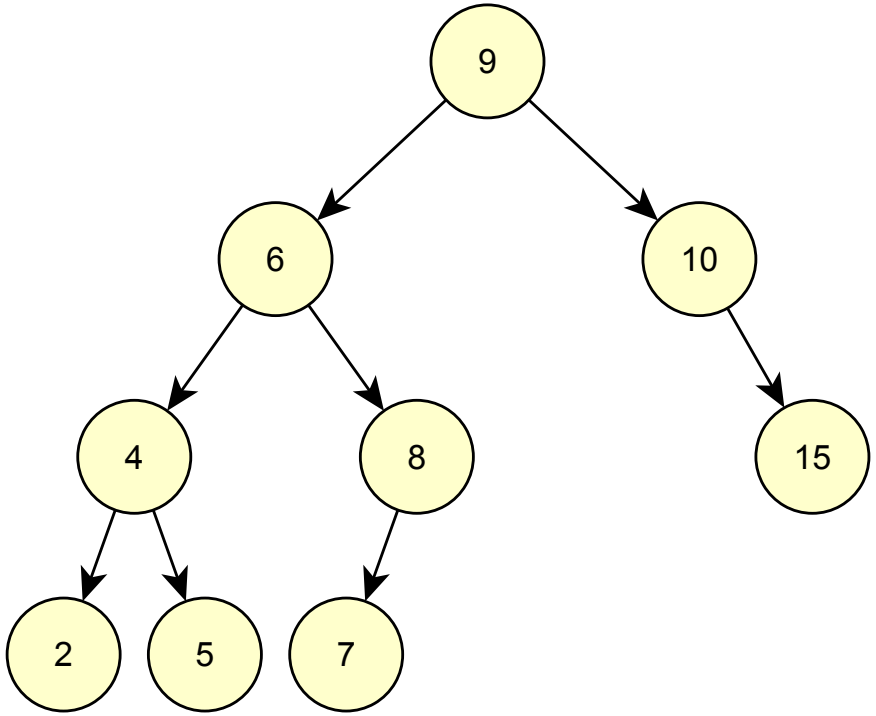
Tampereen yliopisto

Binäärihakupuiden tehokkuus, tasapainotetut bin:hakupuut

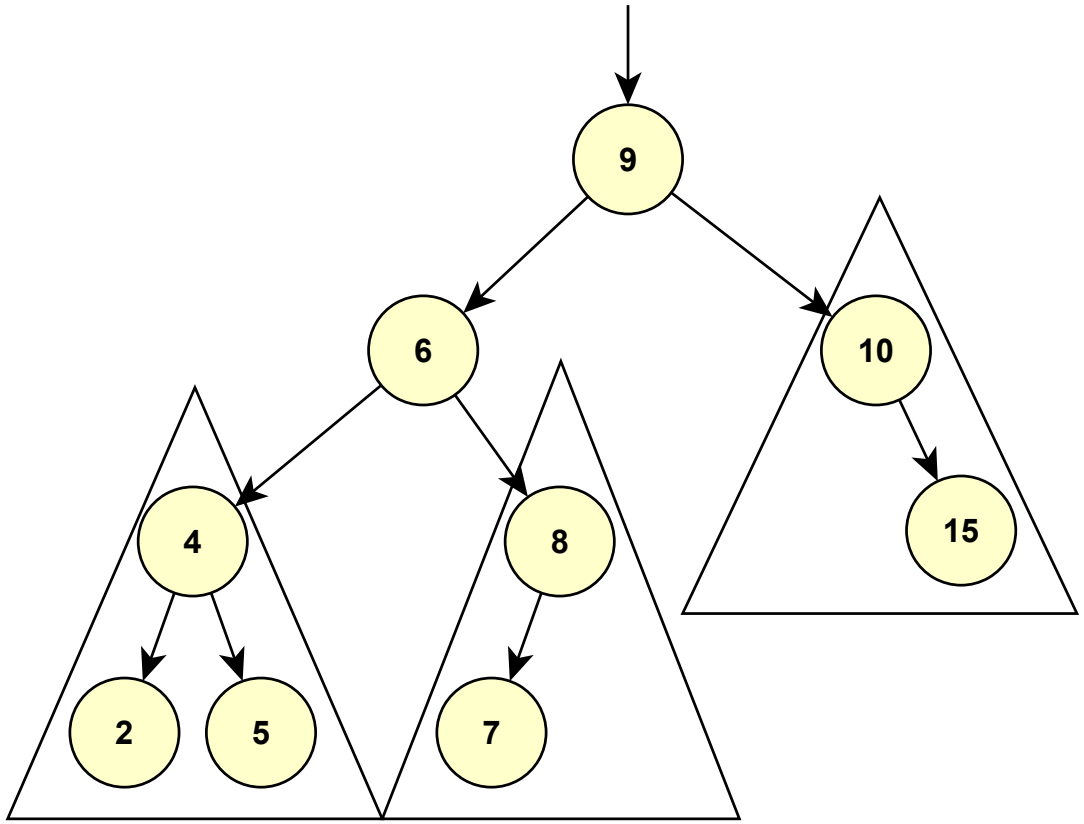
COMP.CS.300 Tietorakenteet ja algoritmit 1

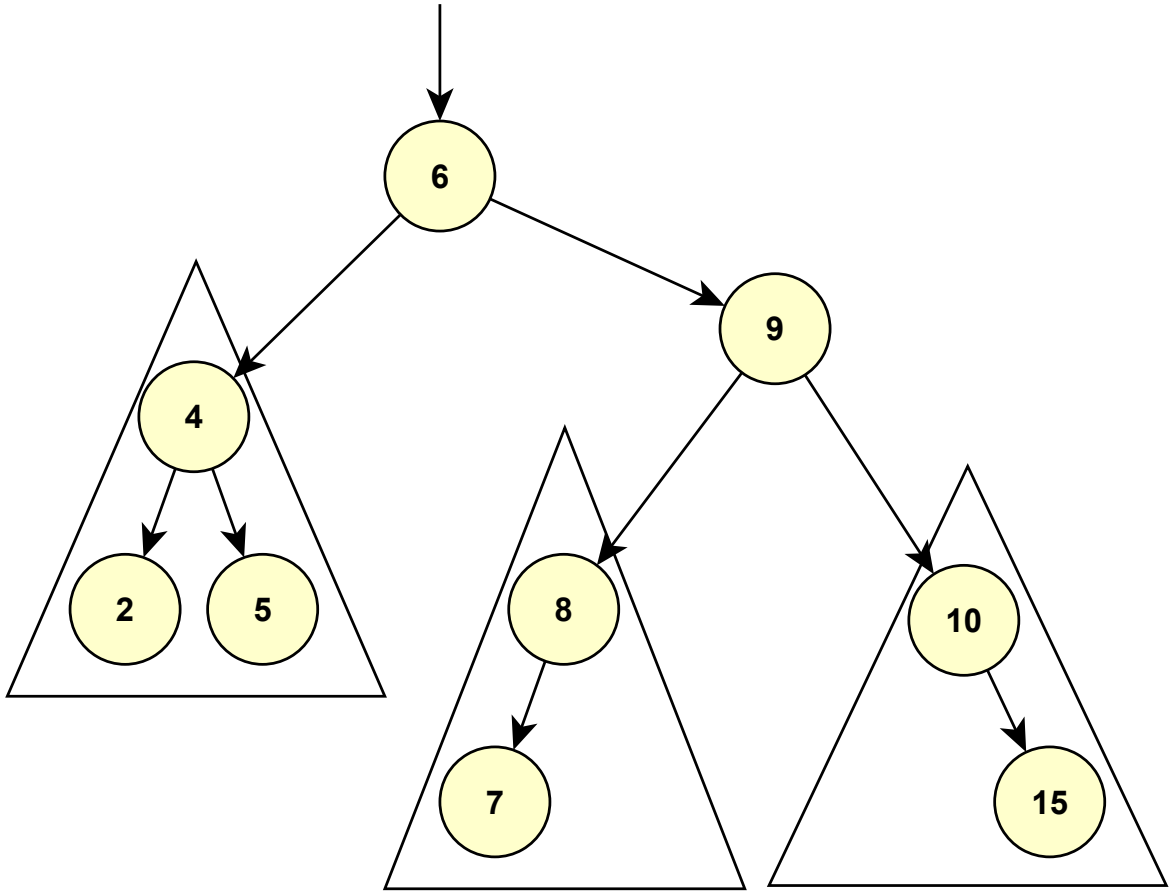
Matti Rintala (matti.rintala@tuni.fi)

Epätasapainon vaikutus

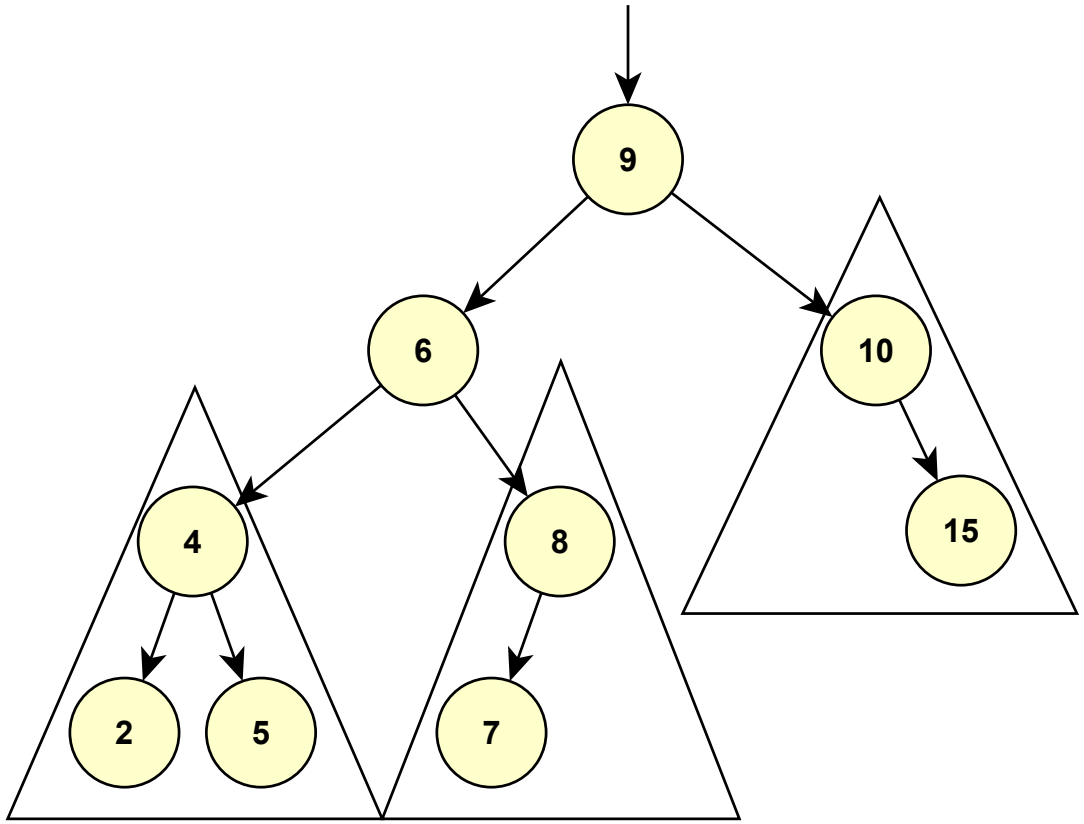


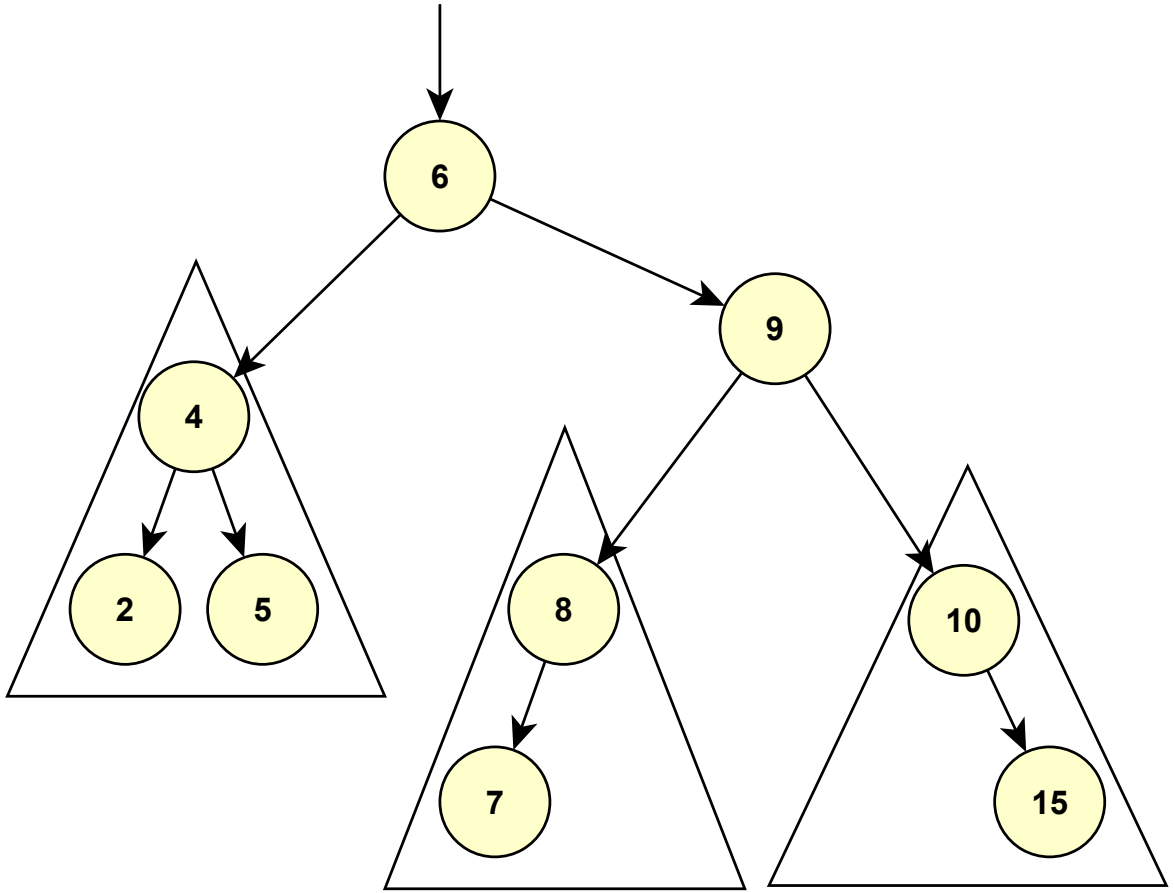
Rotaatiot



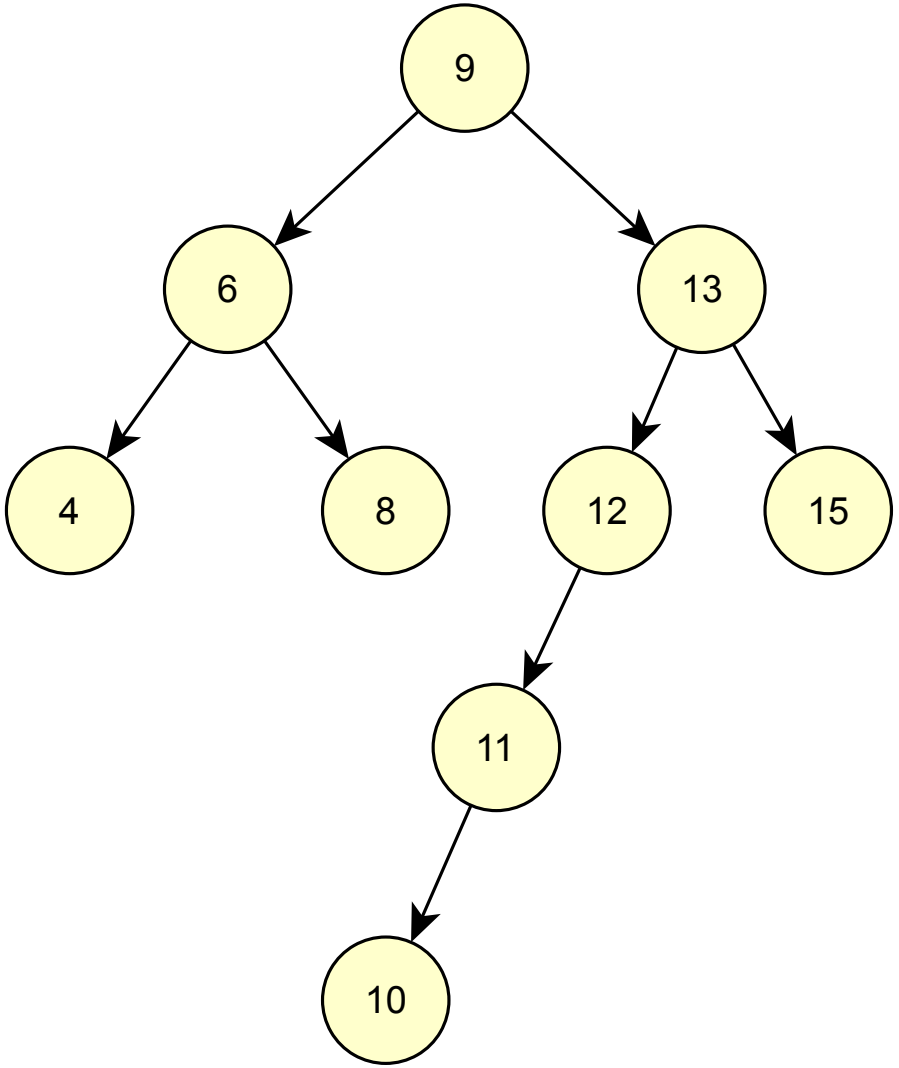

6
4
9
2
5
8
10
7
15

Rotaatiot





"Riittävä" tasapaino



Tampereen yliopisto

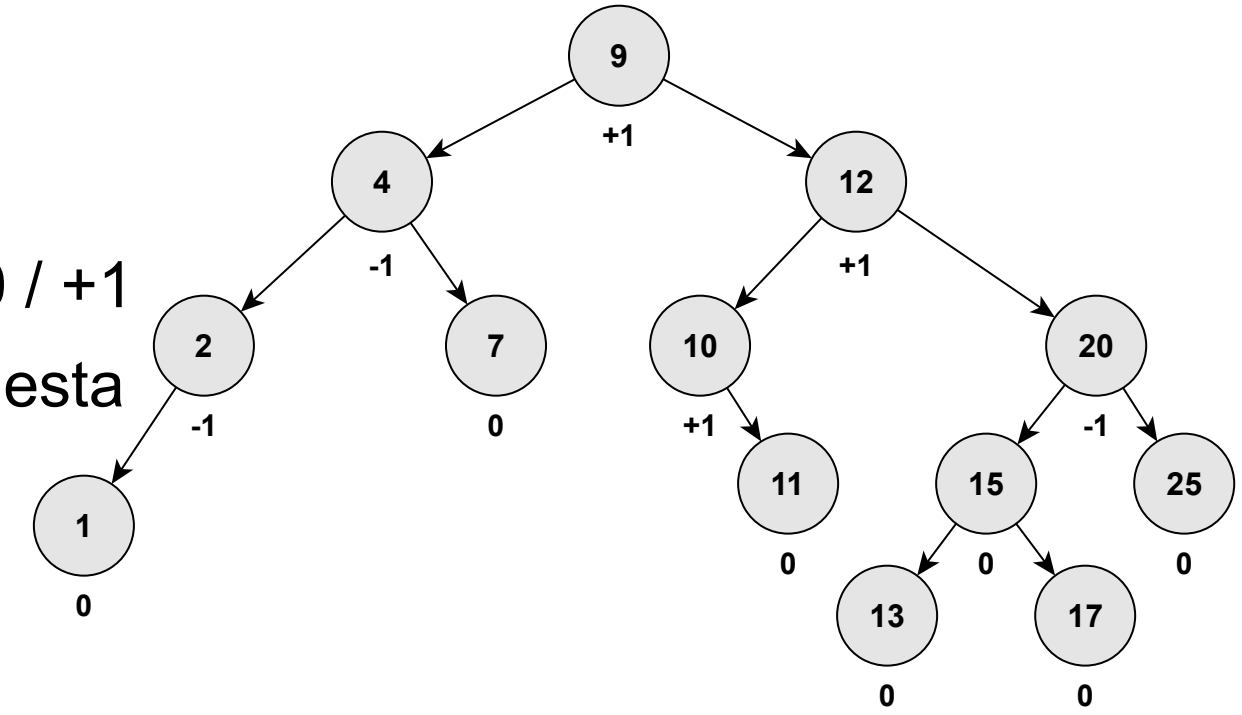
Esimerkkejä binäärihakupuista

COMP.CS.300 Tietorakenteet ja algoritmit 1

Matti Rintala (matti.rintala@tuni.fi)

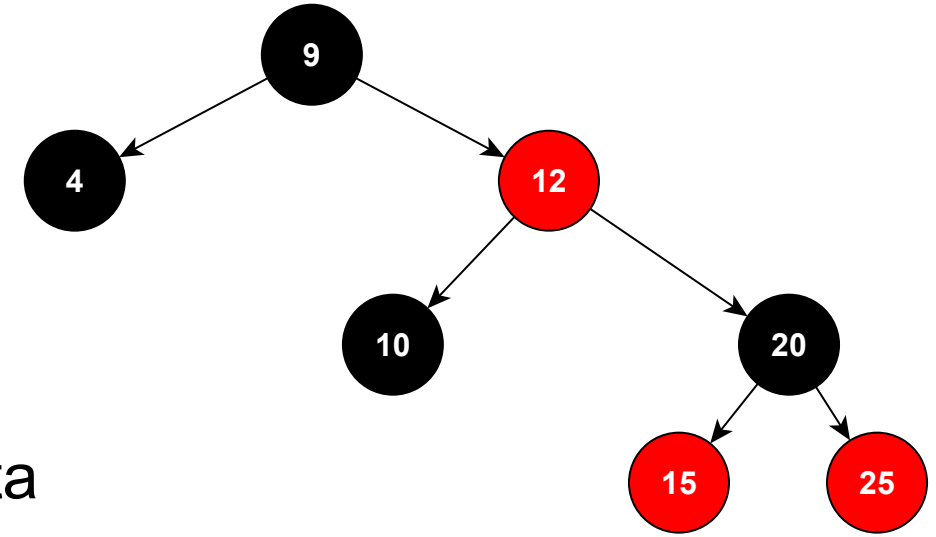
AVL-puut

- Tasapainotettu binäärihakupuu, tasapaino rotaatioilla
- Joka solmun alipuiden korkeudet eroavat korkeintaan yhdellä
- Joka solmussa kirjanpito
  - joko alipuiden korkeuseroista -1 / 0 / +1
  - tai solmusta alkavan puun korkeudesta



Puna-mustat puut

- "Riittävän" tasapainotettu binäärihakupuu, tasapaino rotaatioilla
- Pitkät haarat korkeintaan 2x lyhimmät
- Joka solmussa kirjanpitona "väri" punainen/musta
- Invariantit auttavat tasapainossa:
  - kahta punaista solmua ei peräkkäin
  - mustia solmuja haaroissa aina yhtä monta
  - (juuri aina musta)



Splay-puut

- Eivät tasapainotettuja, mutta muuten mielenkiintoisia
- Lisäyksessä ja haussa lisätty/löydetty solmu rotatoidaan juureksi
- Tuloksena usein haetut/viimeksi lisätyt solmut löytyvät nopeasti läheltä juurta