Dvojiška in iskalna drevesa

Programiranje 1

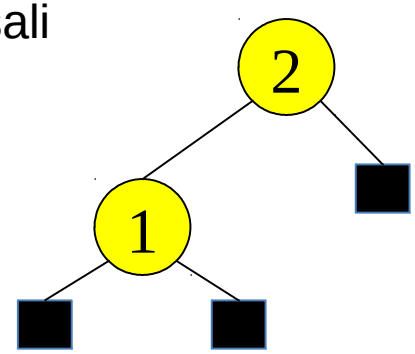
Univerza v Ljubljani, FMF, 2013

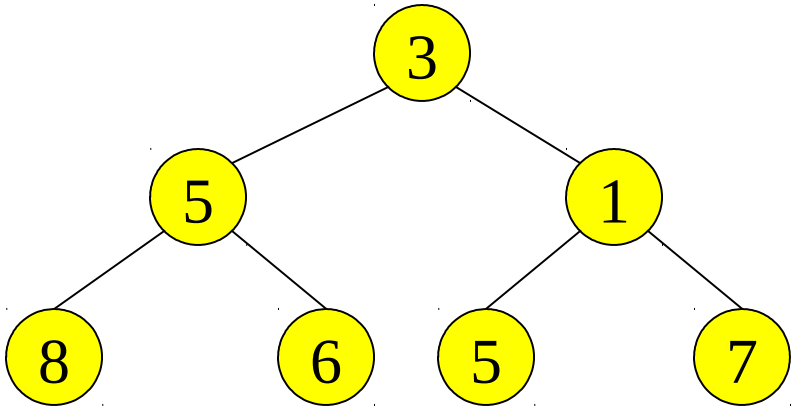
Dvojiško drevo

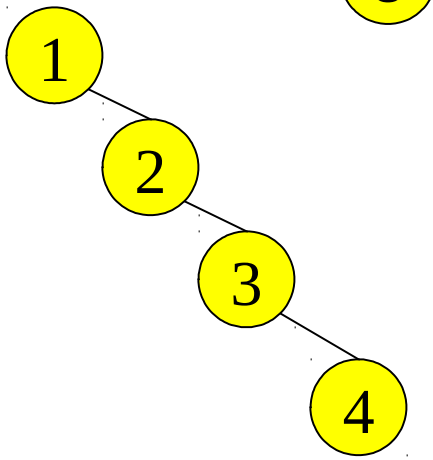
- Sestavljeno iz **vozlišč**, vsako vozlišče ima lahko vsebino, *levega* in/ali *desnega* sina (t.j. določen vrstni red sinov!)
- Dvojiško drevo je:
  - prazno, ali
  - sestavljeno iz vozlišča (*korena*) z neko vsebino, pri čemer sta levi in desni sin spet dvojiški drevesi (poddrevesi).
- Pazi: rekurzivna definicija podatkovne strukture!

Primeri

Opomba: praznih navideznih vozlišč ne bomo risali



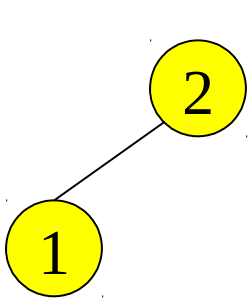


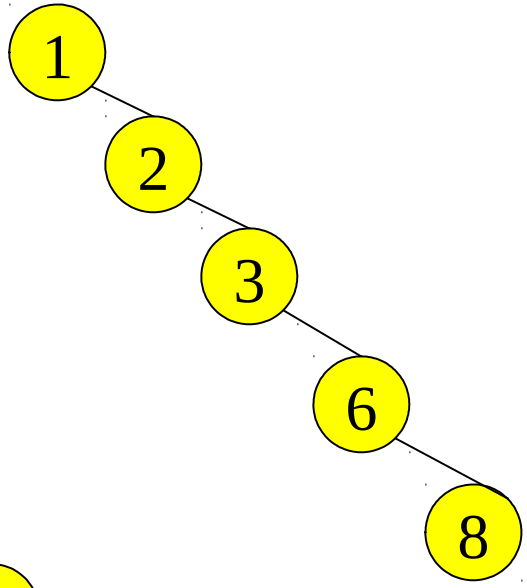


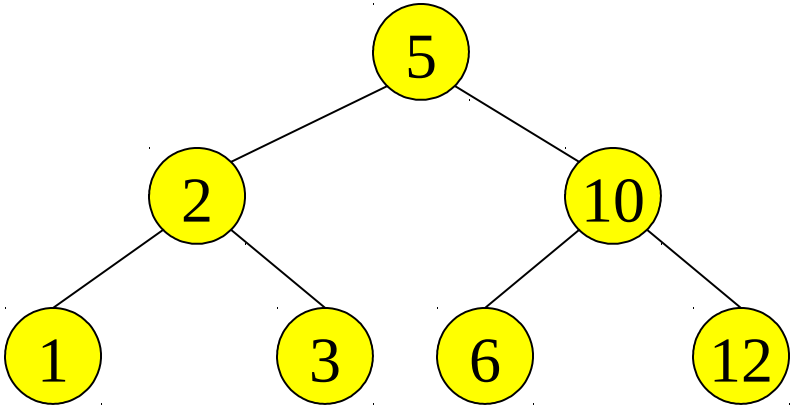
Iskalno dvojiško drevo

- Dvojiško drevo z dodatno strukturo
- Vozlišča vsebujejo podatke iz linearno urejene množice (npr. cela števila)
- Iskalno dvojiško drevo je:
  - prazno, ali
  - sestavljeno iz vozlišča (korena), ki vsebuje nek podatek *x* ter:
  - levo in desno poddrevo sta iskalni dvojiški drevesi,
  - vsi podatki v vozliščih levega poddrevesa so manjši ali enaki *x* in
  - vsi podatki v vozliščih desnega poddrevesa so večji od *x.*

Primeri





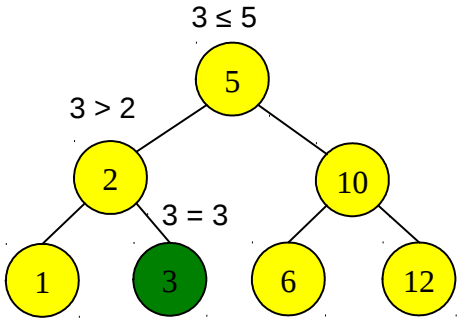


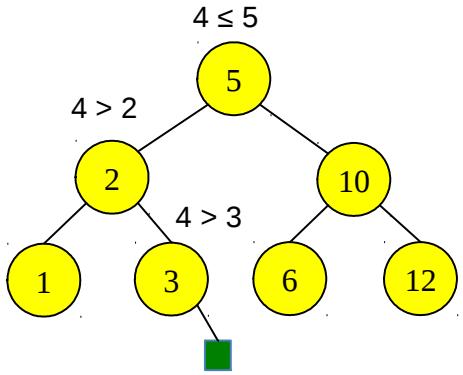
Iskanje elementa x

- Rekurzivno: *išči(koren, x)*:
  - vrne vozlišče (eno izmed) s podatkom *x* v drevesu s korenom *koren*, sicer vrne *None*
  - če je drevo prazno -> vrni *None*
  - sicer, če je *x* v korenu -> vrni koren
  - sicer, če je *x* manjši ali enak od korena -> *išči(koren.levi, x)*
  - sicer -> *išči(koren.desni, x)*
- Premisli iterativno različico z zanko!
- Časovna zahtevnost – O(*globina*)

Primeri

Iščimo: 3, 4





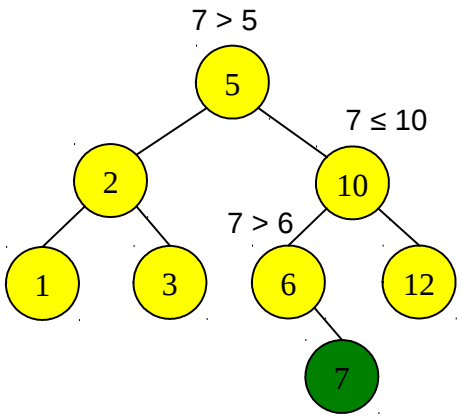
Prazno vozlišče – elementa ni v iskalnem drevesu

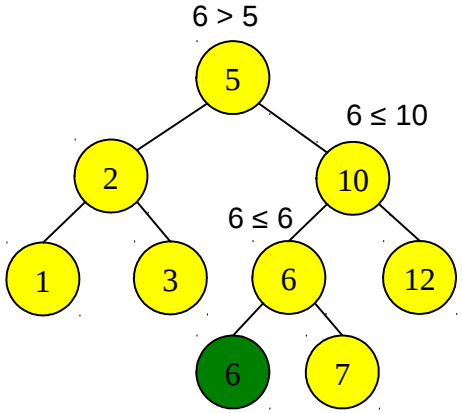
Vstavljanje podatkov

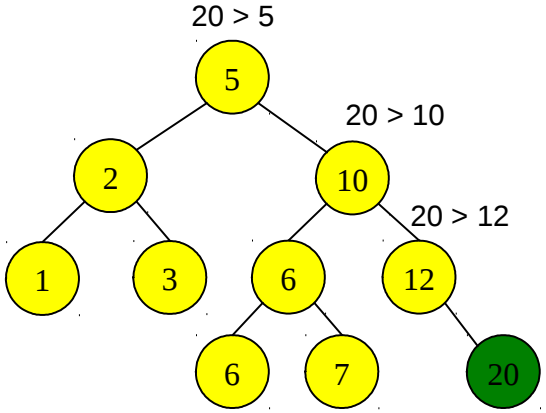
- Rekurzivno
- Podobno kot iskanje – ko naletimo na (navidezno) prazno vozlišče, dodamo novo vozlišče
- Časovna zahtevnost – O(*globina*)

Primer

Vstavimo: 7, 6, 20







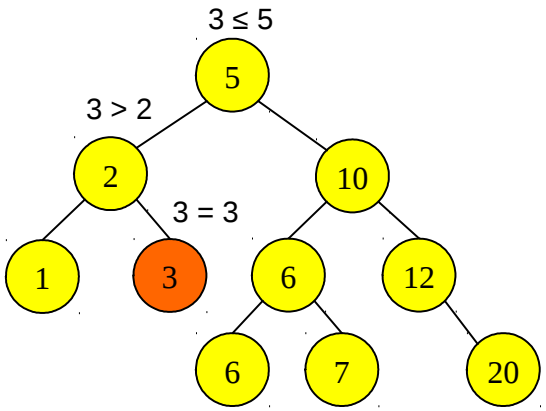
Odstranjevanje podatkov

- Odstranimo eno pojavitev vozlišča s podatkom *x* v drevesu
- Najdemo vozlišče *v* s podatkom *x* (če obstaja, sicer končamo) – algoritem za iskanje
- Če je vozlišče *v* list (ima le prazne sinove), ga odstranimo (nadomestimo s praznim drevesom)
- Če ima vozlišče *v* enega nepraznega sina *s*, potem očeta vozlišča *v* prevežemo na sina *s*.
- Sicer ima vozlišče *v* dva neprazna sinova. Naj bo u najbolj desno vozlišče v levem poddrevesu s korenom *v*. Vozlišče *u* nima desnega sina. Podatek v vozlišču *u* vpišemo v vozlišče *v* in vozlišče *u* zbrišemo.
- Časovna zahtevnost – O(*globina*)

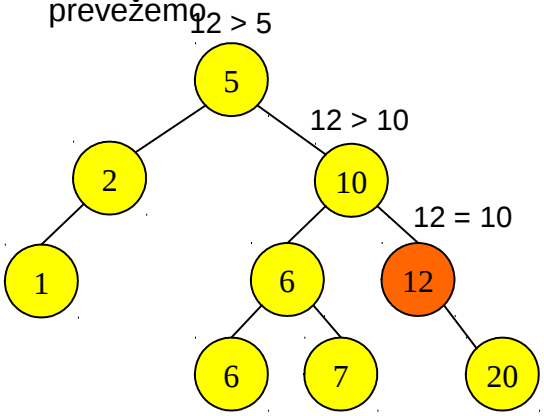
Primer

Brišimo: 3, 12, 10, 5

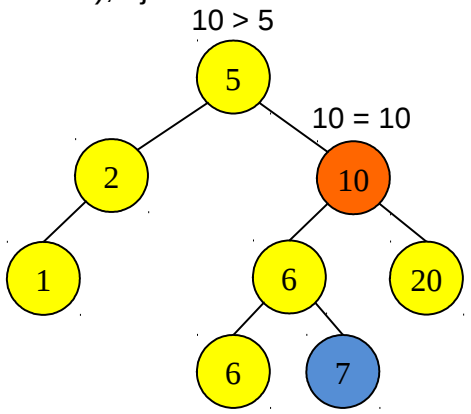
3 je list - brišemo



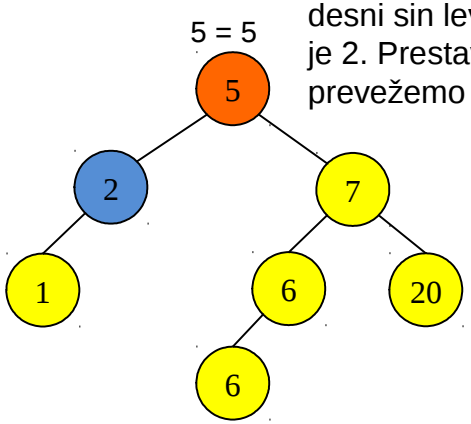
12 ima enega sina - prevežemo

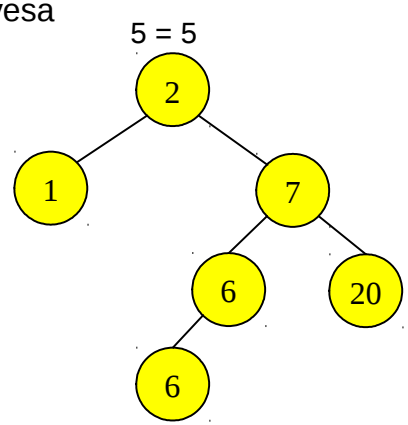


10 ima oba sinova. Najbolj desni sin levega drevesa je 7. Prestavimo 7 v 10 in "prevežemo" list 7 (na virtualno prazno vozlišče), t.j. zbrišemo list.



5 ima oba sinova. Najbolj desni sin levega poddrevesa je 2. Prestavimo 2 v 5 in prevežemo 2.





Kolikšna je lahko globina iskalnih dreves?

- Naj bo *n* število elementov v drevesu
- V najslabšem primeru O(*n*) – npr. vstavljanje elementov po vrsti (nekakšen seznam)
- V najboljšem primeru imamo poravnano drevo, globina O(log *n*)
- Kako ohraniti čimbolj poravnano/uravnoteženo drevo pri operacijah *vstavi* in *odstrani*, ki pa ne smejo postati prezahtevne?

AVL drevo

- Iskalno dvojiško drevo, ki stremi k uravnoteženim levim in desnim poddrevesom
- AVL kratice dveh sovietskih avtorjev: G.M. **A**delson-**V**elskii in E. M. **L**andis
- Vsako vozlišče poleg vsebine vsebuje še faktor ravnotežja:

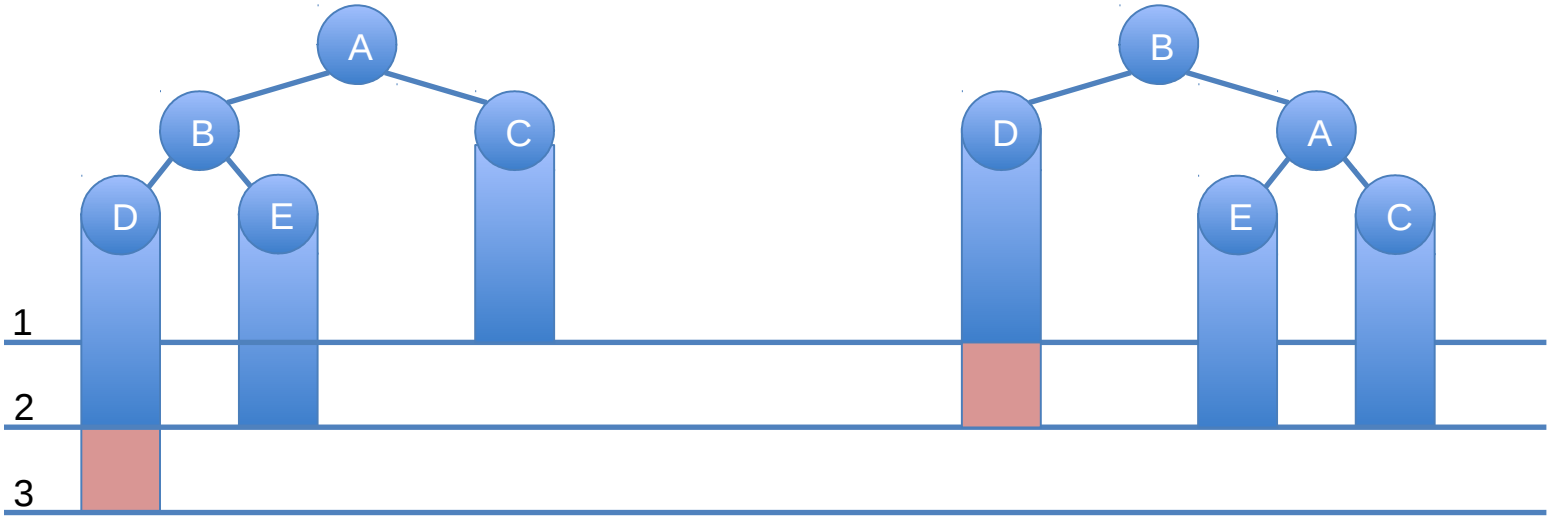
  $$f(v) = globina(l.levi) – globina(l.desni)$$

- Za vsako vozlišče v v AVL drevesu mora veljati $|f(v)| < 2$

AVL - vstavljanje

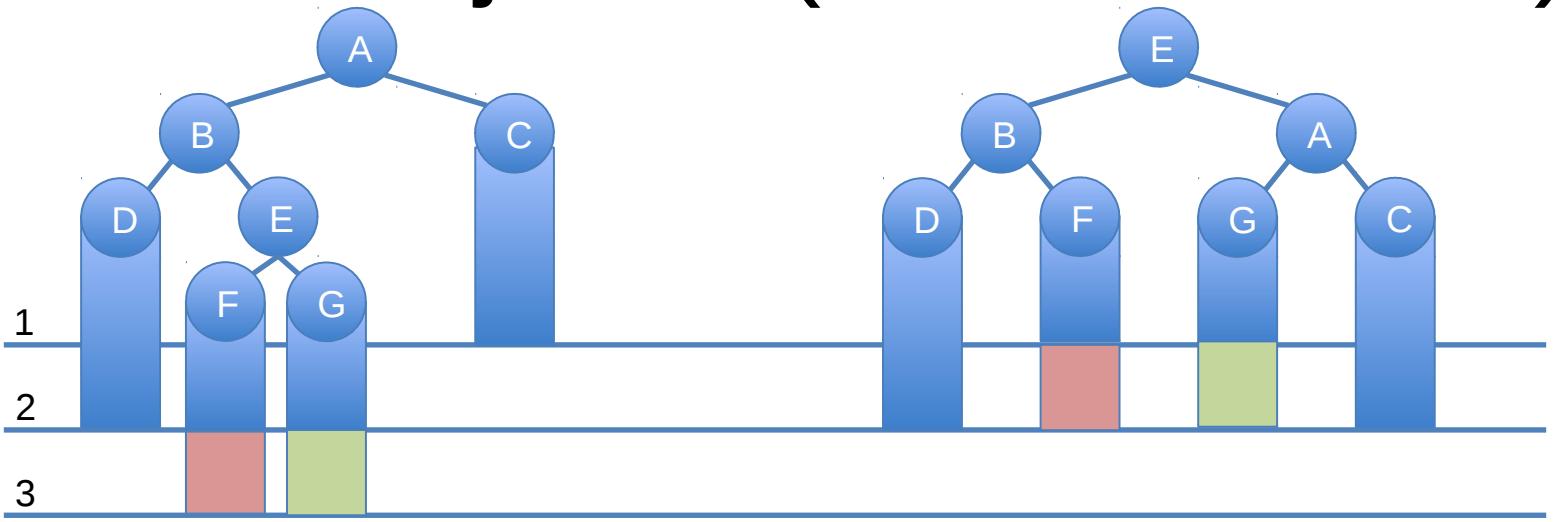
- Izvedemo klasično vstavljanje kot pri iskalnem drevesu
- Izračunamo faktorje ravnotežja, ki se spremenijo po poti od korena do lista, kamor vstavimo
- Ob predpostavki, da gledamo najgloblje poddrevo, kjer se poruši faktor ravnotežja, se zgodijo natanko 4 možnosti
- Z enim popravkom v času O(1) v najglobljem poddrevesu s porušenim faktorjem ravnotežja dosežemo, da drevo z vstavljenim elementom postane AVL drevo
- Kot popravke uporabljamo rotacije LL, LD, DD, DL (L-levo, D-desno)

Rotacija LL (DD simetrično)



- Predpostavka: A je **najgloblje** vozlišče, pri katerem se poruši faktor ravnotežja (postane po abs. vrednosti večji od 1)
- Premislimo: zgoditi se mora natanko takšna situacija, kot je na levem delu slike
  - rdeč kvadratek je dodano vozlišče, ki poveča globino poddrevesa B za 1
  - globini poddreves D in E pred vstavljanjem sta enaki in za ena večji od globine C (sicer bi prišli v protislovje s predpostavko *)
- Vstavljanje je bilo izvedeno v drevo D, ki je levo-levo (LL) glede na A
- Izvedemo prevezavo kot na desnem delu slike (rotacija LL), ki razreši situacijo in novo dobljeno drevo je enako globoko kot prejšnje in je iskalno drevo

Rotacija LD (DL simetrično)



- Predpostavka: A je najgloblje vozlišče, pri katerem se poruši faktor ravnotežja (postane po abs. vrednosti večji od 1)*
- Zgoditi se mora natanko takšna situacija, kot je na levi, pri čemer sta dve izključujoči se možnosti: ali je prišlo do vstavljanja v poddrevo F (rdeč kvadratek) ali v poddrevo G (zelen kvadratek). Dodano vozlišče poveča globino poddrevesa B za 1.
- Globini poddreves F in G pred vstavljanjem sta enaki, najglobji elementi v poddrevesu D pa so na isti globini kot najgloblji elementi v F in G (sicer konflikt s predpostavko)
- Vstavljanje je bilo izvedeno v poddrevo F ali G, v obeh primerih je to levo-desno (LD) glede na A, torej v poddrevo E.
- Izvedemo prevezavo na desni (rotacija LD), ki razreši situacijo in novo dobljeno poddrevo postane enako globoko kot prej.

Analiza

- Iz slik primerov, lahko natančno določimo prevezave in nove faktorje ravnotežja!
- Rotacije se izvedejo v O(1) časa
- Dovolj je ena rotacija (v najglobljem poddrevesu s porušenim faktorjem ravnotežja)
- Vstavljanje se izvede v O(*globina*)
- Odstranjevanje:
  - začetek: enak kot pri iskalnem drevesu;
  - če je *h* globina elementa, ki ga brišemo, potem se da pokazati, da je poterebno izvesti kvečjemo *h* rotacij po poti od korena do odstranjenega elementa
- Torej: preostane nam, da dokažemo, da je globina AVL drevesa z *n* vozlišči O(log n)

Globina AVL drevesa

Nh – minimalno število vozlišč v AVL drevesu globine h

$$Nh \geq Nh\text{-}1 + Nh\text{-}2 + 1 \geq 2Nh\text{-}2 + 1 \geq$$
$$\geq 1 + 2(1 + 2Nh\text{-}4) = 1 + 2 + 22\ N\ h\text{-}4$$
$$> 1 + 2 + 22 + 23N\ h\text{-}2*3 \ ...$$
$$> 1 + 2 + 22 + 23 + ... + 2h/2\ N\ h\text{-}2*h/2$$
$$= 2h/2 - 1$$

Torej:

$$2h/2 - 1 \leq n$$
$$h/2 \leq \log(n + 1)$$
$$h \leq 2 \log(n + 1)$$