Programiranje 2

Dinamičke strukture podataka — Stabla

Milena Vujošević Janičić
Jelena Graovac

www.matf.bg.ac.rs/~milena
www.matf.bg.ac.rs/~jgraovac

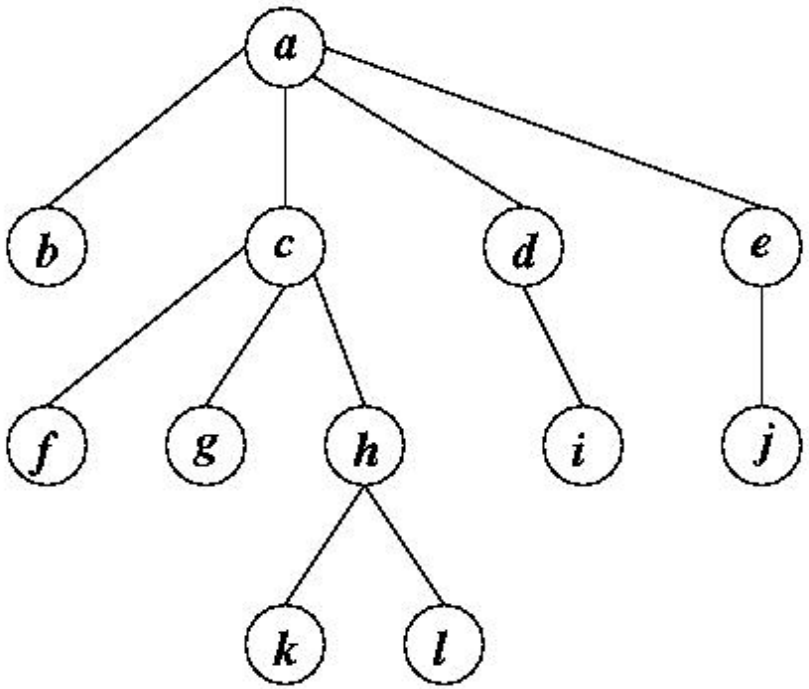
Programiranje 2

Pregled

Pregled

Stabla

- Stablo je struktura koja prirodno opisuje određene vrste hijerarhijskih objekata (npr. porodično stablo, logički izraz, aritmetički izraz, ...).



Stabla

- Stablo se sastoji od čvorova i grana između njih.
- Svaka grana povezuje jedan čvor (koji je u tom kontekstu roditelj) sa njegovim detetom.
- Čvor A je *predak* čvora B ako je A roditelj čvora B ili ako je A predak roditelja čvora B.
- Čvor A je *potomak* čvora B ako je A dete čvora B ili ako je A potomak deteta čvora B.

Stabla

- Čvor koji se zove *koren stabla* nema nijednog roditelja.
- Svaki drugi čvor ima tačno jednog roditelja.
- *List* je čvor koji nema potomaka.
- Čvor koji ima bar jednog potomka naziva se *unutrašnji čvor*.

Stabla

- Koren stabla je predak svim čvorovima stabla (osim sebi). Zbog toga je moguće do bilo kog čvora stabla stići od korena (jedinstveno određenim putem).
- Maksimalni broj dece čvora u stablu naziva se *stepen stabla*.
- Obično je za decu svakog čvora definisan redosled tako da se deca mogu identifikovati svojim rednim brojem.
- Visina stabla je najveći nivo hijerarhije u njemu, tj maksimalno rastojanje od korena do nekog čvora.

Stabla

- Nacrtati stablo za naredni aritmetički izraz:

$$a + b \cdot (c/d - 3)$$

Stabla

- Stablo se može implementirati na razne načine
- Kod eksplicitne implementacije, svaki čvor stabla sadrži pokazivače koji pokazuju gde se u memoriji nalaze njegova deca. Ukoliko je stepen stabla k, onda je potrebno da svaki čvor stabla sadrži k pokazivača
- Stablo stepena većeg od dva može se implementirati i uz pomoć samo dva pokazivača, jedan pokazivač ka detetu, a drugi ka susednom bratu

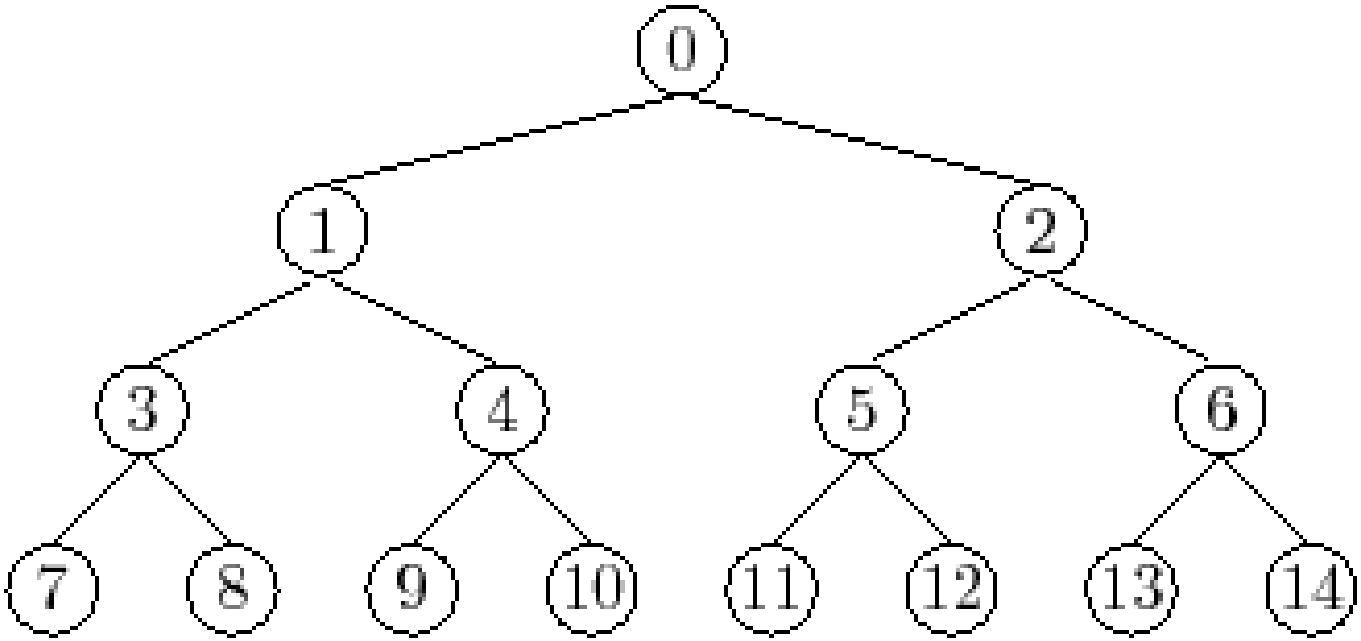
Implicitno predstavljanje stabla

- Implicitno predstavljanje stabla ne koristi pokazivače: svi čvorovi se smeštaju u niz a veze između čvorova određene su njihovom pozicijom u nizu.
- Na primer, ukoliko je stablo stepena dva, koren se smešta u prvi element niza, njegova deca u drugi i treći, u četvrti i peti element se smeštaju deca od čvora koji je smešten u drugi element niza i tako redom, tj ako je čvor smešten u element niza A[i], onda su njegova deca smeštena u A[2i] i u A[2i+1] — ovo se lako može dokazati indukcijom
- Očigledno je da se u ovakvom nizu mora rezervisati prostor i za odsutne čvorove

Implicitno predstavljanje stabla

- Ovakav način predstavljanja stabla je pogodan zbog svoje kompaktnosti
- Ipak, ako je stablo neuravnoteženo ili degenerisano (odnosno neki listovi su mnogo dalje od korena nego drugi), mora se rezervisati i prostor za mnogo nepostojećih čvorova, pa se tada ovakvim predstavljanjem stabla neracionalno koristi memorijski prostor.

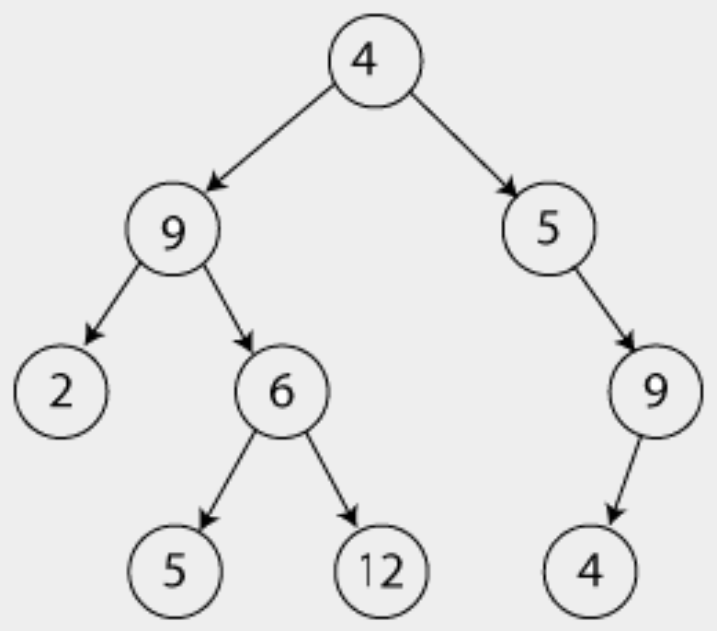
Implicitno predstavljanje stabla



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Binarno stablo

- Binarno stablo je stablo u kojem svaki čvor ima najviše dva deteta.

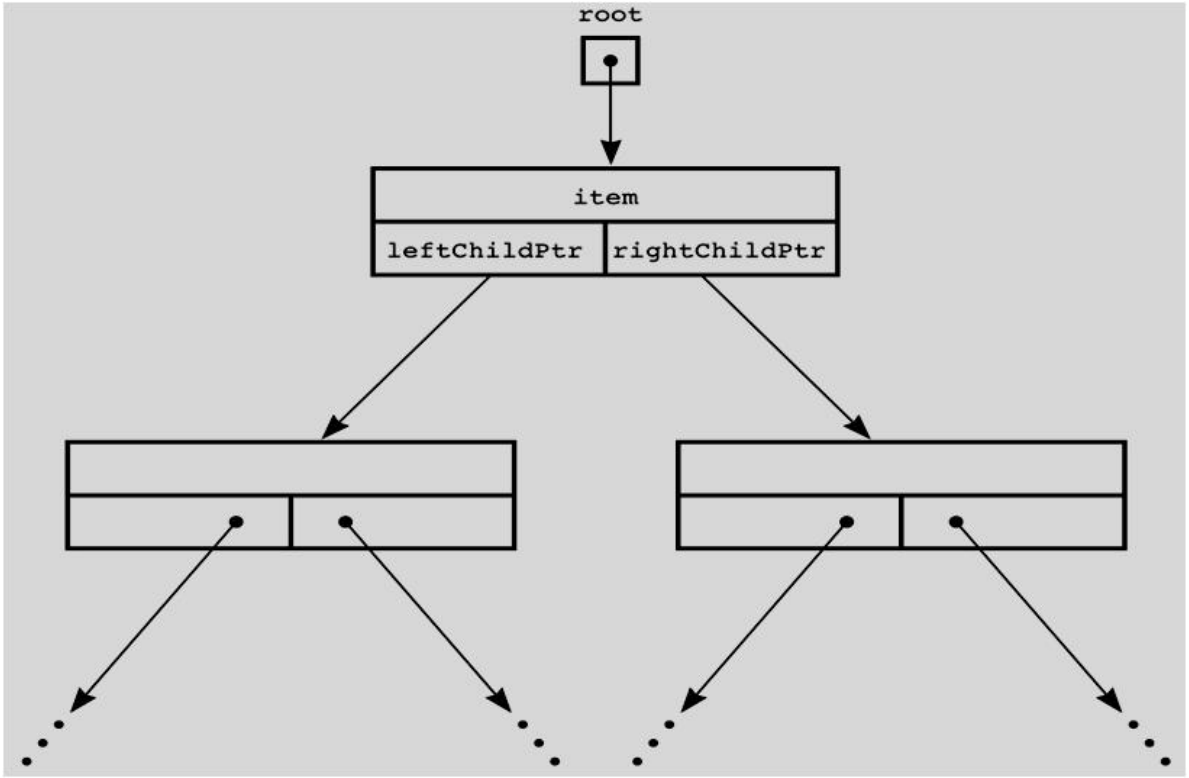


Binarno stablo

Binarna stabla mogu se implementirati korišćenjem dinamičke alokacije i pogodne strukture sa pokazivačima:

```
typedef struct cvor {
  int broj;
  struct cvor *levo;
  struct cvor *desno;
} Cvor;
```

Binarno stablo



Obilazak binarnog stabla

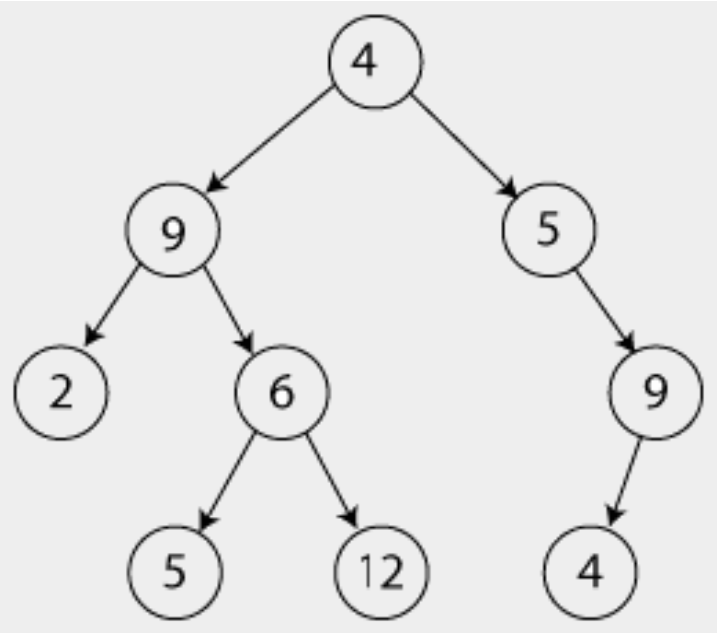
- Obrada redom svih čvorova stabla naziva se *obilazak stabla*.
- Obilazak stabla se može vršiti u širinu i u dubinu

Obilazak binarnog stabla

- Postoji nekoliko vrsta obilazaka u dubinu:
  - infiksni: L-K-D (obilazi (rekurzivno) najpre njegovo levo podstablo, pa sâm taj čvor, pa njegovo desno podstablo) ili D-K-L
  - prefiksni: K-L-D ili K-D-L
  - postfiksni: L-D-K ili D-L-K

Obilazak stabla

Obiđimo ovo stablo infiksno, prefiskno i postfiksno



Obilazak stabla

```
void ispisi_stablo_infiksno_LKD(Cvor * koren)
{
  if(koren != NULL) {
    ispisi_stablo_infiksno_LKD(koren->levi);
    printf("%d ", koren->broj);
    ispisi_stablo_infiksno_LKD(koren->desni);
  }
}
```

Obilazak stabla

```
void ispisi_stablo_prefiksno_KLD(Cvor * koren)
{
  if(koren != NULL) {
    printf("%d ", koren->broj);
    ispisi_stablo_prefiksno_KLD(koren->levi);
    ispisi_stablo_prefiksno_KLD(koren->desni);
  }
}
```

Obilazak stabla

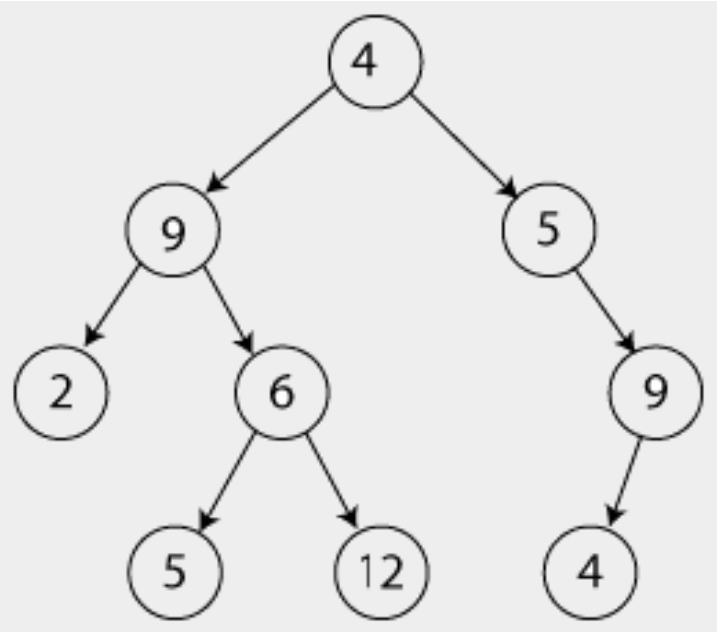
```
void ispisi_stablo_postfiksno_LDK(Cvor * koren)
{
  if(koren != NULL) {
    ispisi_stablo_postfiksno_LDK(koren->levi);
    ispisi_stablo_postfiksno_LDK(koren->desni);
    printf("%d ", koren->broj);
}
}
```

Obilazak binarnog stabla

- Obilazak stabla u širinu odgovara obilasku stabla po nivoima
- Da bi se implementirao obilazak stabla u širinu može se koristiti red
  1. Na početku, u red se stavlja koren
  2. Vrši se obrada čvora koji se nalazi na početku reda a na kraj reda se smeštaju njegova deca
  3. Iz reda se izbacuje čvor sa početka reda
  4. Ukoliko je red prazan, postupak je završen. U suprotnom, ide se nazad na korak 2.

Binarno stablo

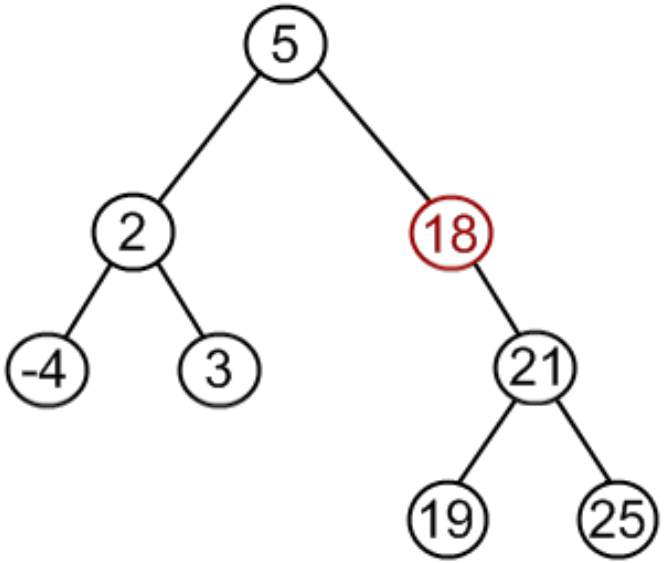
Obilazak stabla u širinu



Uređeno binarno stablo

- Uređena binarna stabla ili binarna pretraživačka stabla
- Za svaki čvor n, vrednosti svih čvorova iz njegovog levog podstabla su manje ili jednake od vrednost u čvoru n, a vrednosti svih čvorova iz njegovog desnog podstabla su veće od vrednosti u čvoru n

Uređeno binarno stablo



Uređeno binarno stablo

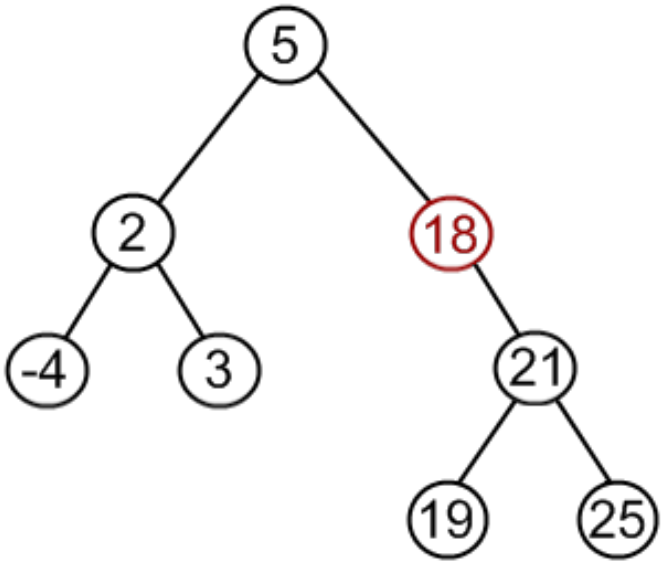
- Uređena binarna stabla omogućavaju efikasnije (logaritamske) obrade podataka u stilu binarne pretrage (jer su podaci ispod svakog čvora podeljeni na one koji su "levo" tj manji i one koji su "desno" tj veći od podatka u posmatranom čvoru).
- Međutim, ta efikasnost se gubi ako je binarno stablo neuravnoteženo (na primer, svaki čvor ima samo desnog potomka) ili skoro neuravnoteženo.

Obilazak uređenog binarnog stabla

- Obilaskom uređenog binarnog stabla infiksnim redosledom L-K-D obrađuju se svi čvorovi stabla u rastućem redosledu
- Obilaskom uređenog binarnog stabla infiksnim redosledom D-K-L obrađuju se svi čvorovi stabla u opadajućem redosledu

Uređeno binarno stablo

Obiđimo ovo stablo u poretku L-K-D i D-K-L



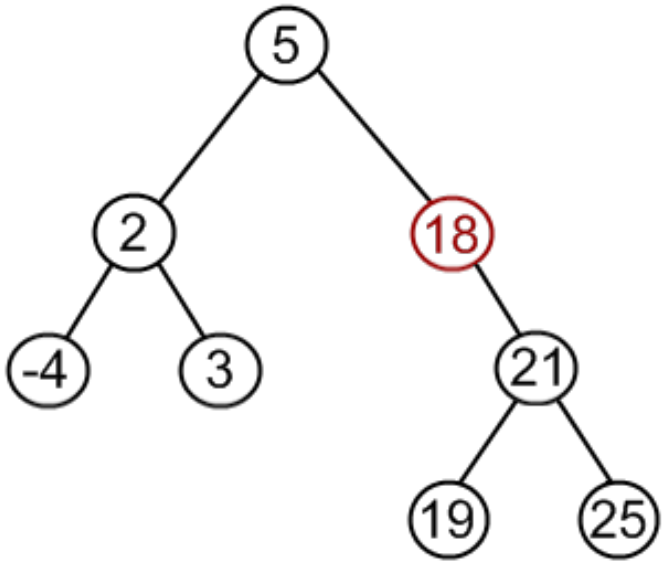
Dodavanje čvora u stablo

Potrebno je niz brojeva smestiti u uređeno binarno stablo.
Precedura je sledeća:

- Kreirati koren stabla i prvi broj iz niza smestiti u koren;
- Za svaku sledeću vrednost v iz niza brojeva:
  - Uporediti vrednost v sa vrednošću broja u korenu; ako je vrednost broja v manja ili jednaka od vrednosti broja u korenu, dodati broj u levo podstablo, inače dodati ga u desno podstablo;
  - Postupak nastaviti sa odgovarajućim podstablom sve dok se ne dođe do lista ili do čvora čije je odgovarajuće podstablo (u koje treba dodati broj) prazno (ima NULL vrednost).
  - Na tom mestu kreirati čvor stabla i broj sa vrednošću v smestiti u taj čvor.

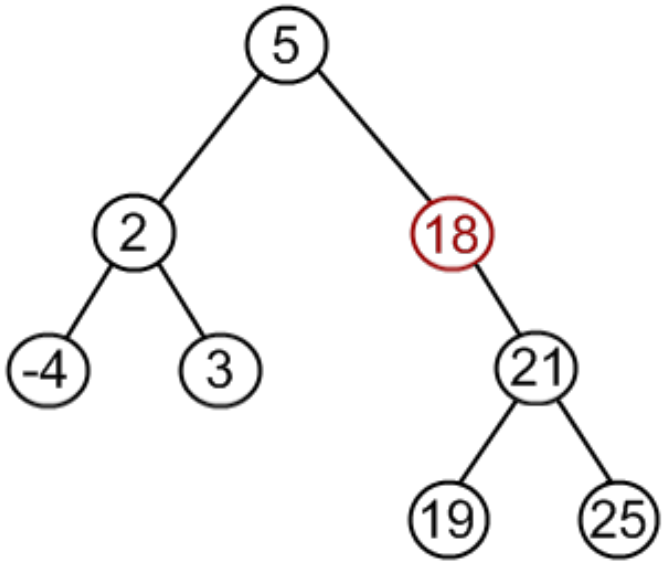
Uređeno binarno stablo

Dodajmo broj 15 u naredno stablo



Uređeno binarno stablo

Dodajmo broj 1 u naredno stablo



Uređeno binarno stablo

- Kreirajmo binarno stablo umetanjem narednih brojeva 7, 2, 5, 9, 8 i 11, tim redom.
- Kreirajmo binarno stablo umetanjem narednih brojeva 1, 2, 3, 4, 5, tim redom.
- Kreirajmo binarno stablo umetanjem narednih brojeva 6, 5, 4, 3, 2, 1 tim redom.

Pronalaženje čvora u stablu

U datom uređenom binarnom stablu, potrebno je pronaći čvor koji sadrži unapred zadatu vrednost v. Algoritam je sledeći:

1. Ukoliko je tekuće stablo prazno, vratiti rezultat da u stablu ne postoji broj sa traženom vrednošću.
2. Početi pretraživanje od korena stabla.
3. Ukoliko je v jednako vrednosti broja u tekućem čvoru stabla, vratiti tekući čvor.
4. Ukoliko je v manje ili jednako od vrednosti broja u tekućem čvoru stabla, pretražiti levo podstablo.
5. Inače, pretražiti desno podstablo.

Pronalaženje čvora u stablu

```
Cvor *pretrazi_stablo(Cvor *koren, int broj)
{
  if (koren == NULL)
    return NULL;
  if (koren->broj == broj)
    return koren;
  if (broj < koren->broj)
    return pretrazi_stablo(koren->levo, broj);
  else
    return pretrazi_stablo(koren->desno, broj);
}
```

Oslobađanje memorije koju zauzima stablo

Oslobađanje memorije koju zauzima stablo vrši se u postfiksnom poretku

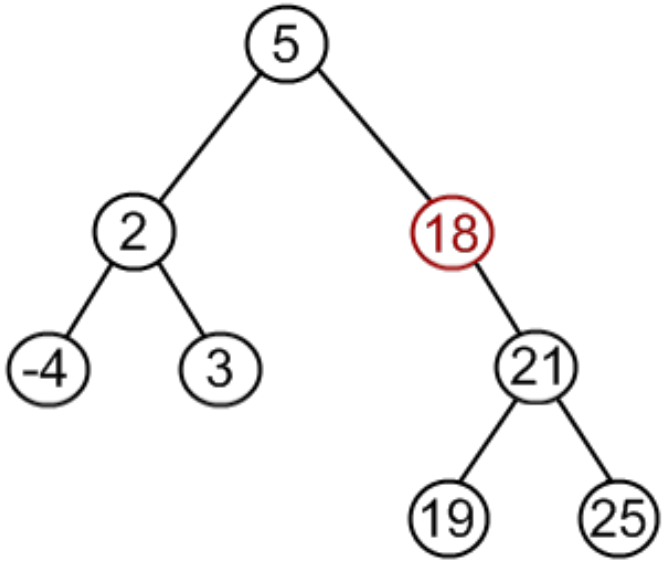
```
void oslobodi_stablo(Cvor** adresa_korena)
{
  if(*adresa_korena == NULL)
    return;

  oslobodi_stablo(&(*adresa_korena)->levo);
  oslobodi_stablo(&(*adresa_korena)->desno);
  free(*adresa_korena);
  *adresa_korena = NULL;
}
```

Brisanje jednog čvora stabla

- Ukoliko želimo da obrišemo jedan čvor iz stabla, razlikujemo tri slučaja:
  - Brisanje lista
  - Brisanje čvora koji nema oba potomka
  - Brisanje čvora koji ima oba potomka

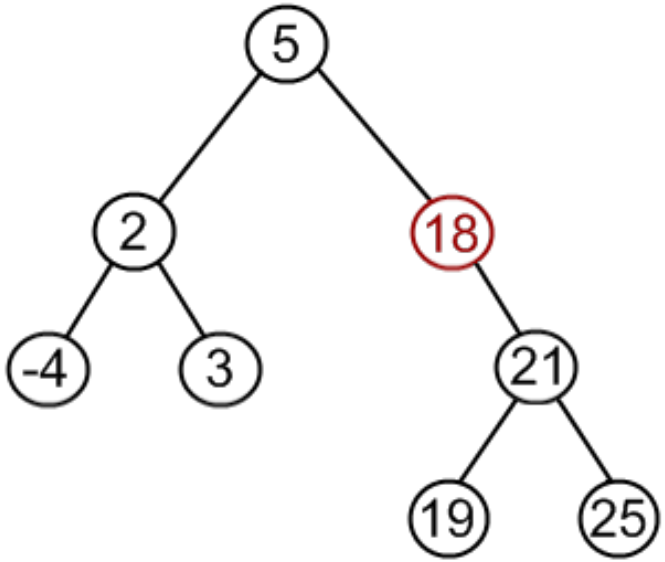
Uređeno binarno stablo

Kako bi obrisali čvor 3?



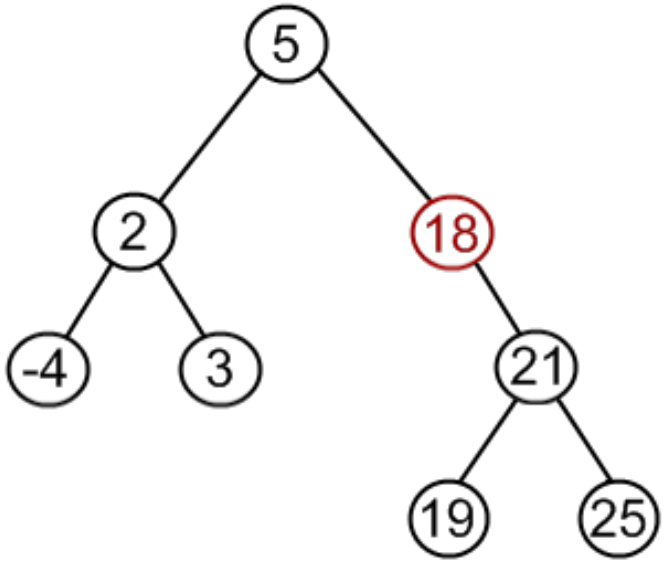
Uređeno binarno stablo

Kako bi obrisali čvor 18?



Uređeno binarno stablo

Kako bi obrisali čvor 5?



Uređeno binarno stablo

- Efikasnost operacija pretrage, umetanja i brisanja zavisi od oblika stabla i položaja čvora na kojem se vrši intervencija
- U najgorem slučaju se pretraga završava u listu stabla — složenost je proporcionalna visini stabla
- Dodavanjem rastućeg niza brojeva u uređeno stablo dobijamo degenerisano stablo koje će imati oblik liste
- Pretraživanje ovakvog stabla je linearne složenosti
- Međutim, ukoliko se stablo dobija umetanjem brojeva u slučajno izabranom poretku, onda je očekivana visina stabla $2 \cdot \ln n$ pa su prethodne operacije efikasne

Uređeno binarno stablo

- Stablo je balansirano (uravnoteženo) ukoliko za svaki čvor važi da je apsolutna vrednost razlike visina levog i desnog podstabla manja ili jednaka od jedan.
- Ukoliko je stablo uravnoteženo operacije za rad sa drvetom su logaritmaske složenosti
- Postoje algoritmi za formiranje balansiranih stabala — dakle nakon umetanja, ukoliko stablo nije uravnoteženo, vrše se dodatne transformacije stabla kako bi ono bilo uravnoteženo

Neki zadaci za rad sa stablima

Napisati sledeće funkcije za rad sa binarnim stablima (ne moraju biti pretraživačka) koja sadrže cele brojeve.

1. Napisati funkciju koja izračunava broj čvorova stabla.
2. Napisati funkciju koja izračunava broj listova stabla.
3. Napisati funkciju koja izračunava sumu čvorova stabla.
4. Napisati funkciju koja izračunava dubinu stabla.
5. Napisati funkciju koja izračunava najveći element u stablu.

Neki zadaci za rad sa stablima

Nivoi drveta su definisani na sledeći način: koren je na nultom nivou, deca od korena su na prvom nivou, njihova deca na drugom nivou i tako redom.

1. Napisati funkciju koja ispisuje sve elemente na n-tom nivou.
2. Napisati funkciju koja izračunava koliko se čvorova nalazi na n-tom nivou.
3. Napisati funkciju koja izračunava maksimalnu vrednost čvorova na n-tom nivou.
4. Napisati funkciju koja izračunava zbir čvorova na n-tom nivou.

Literatura

- Slajdovi su pripremljeni na osnovu sedmog poglavlja knjige Predrag Janičić, Filip Marić: Programiranje 2
- Za pripremu ispita, slajdovi nisu dovoljni, neophodno je koristiti knjigu!