Algoritmi i strukture podataka
vežbe 3

Mirko Stojadinović

1. novembar 2013

1 Stabla

Kolika je složenost pretrage i dodavanja novog čvora u BSP?
Uređeno stablo $\Leftrightarrow$ stablo pretrage. Voditi računa o tekstu zadatka jer od njega može bitno zavisiti rešenje. Npr. bitno je da li je stablo binarno ili nije, da li je uređeno ili nije, da li je balansirano ili nije. Kod BSP smatramo da su svi ključevi (brojevi) u stablu *različiti*. Za sve navedene algoritme razmotriti i objasniti koja je složenost.

1. Napisati funkciju u C-u koja proverava da li je stablo koje se prenosi kao argument funkciji uređeno, tj. da li je to stablo ispravno BSP.

```
int uredjeno (cvor *koren) {
  if (! koren) return 1;
  if (koren->levo)
    if (! uredjeno (koren->levo) || max_u (koren->levo) > koren->broj))
      return 0;
  if (koren->desno)
    if (! uredjeno (koren->desno) || min_u (koren->desno) < koren->broj))
      return 0;
  return 1;
}
```

gde je:

```
int min_u (Stablo koren)                /* Najmanji u uredjenom stablu.
  { return koren->levo ? min_u (koren->levo ) : koren->broj; }

int max_u (Stablo koren)                 /* Najveci u uredjenom stablu. */
  { return koren->desno ? max_u (koren->desno) : koren->broj; }
```

Ako se zanemare pozivi fja `max_u` i `min_u` složenost je $O(n)$ (za n čvorova konstantan broj operacija). Ove dve funkcije pozivaju se u svakom čvoru, u

najgorem slučaju svaka od njih se izvršava za $O(h)$ vremena, gde je h visina stabla. Zato je složenost algoritma $O(n \cdot h)$.

Napisati pseudokod algoritma za isti problem složenosti $O(n)$ (uputstvo: spustiti se do listova i kretati se na gore). Olakšica: dovoljno je prekinuti algoritam sa porukom o greški u slučaju utrvrđivanja neuređenosti.

2. Napisati funkciju u C-u koja kao argumente prima BSP i broj i vraća BSP koje se dobija izostavljanjem tog broja iz stabla. Smatrati da se brojevi u stablu ne mogu ponavljati.

```
cvor* izost_u (cvor* koren, int b) {
   if (koren) {
     if (koren->broj > b)
        koren->levo = izost_u (koren->levo, b);
     else if (koren->broj < b)
        koren->desno = izost_u (koren->desno, b);
     else if (koren->levo && !koren->desno){ // samo levi sin
        cvor *tmp = koren->levo;
        free koren;
        koren = tmp;
     }
     else if (koren->desno && !koren->levo){ // samo desni sin
        cvor *tmp = koren->desno;
        free koren;
        koren = tmp;
     }
     else if (koren->levo) { // ima oba sina
        int m = max_u (koren->levo);
        koren->broj = m;
        koren->levo = izost_u (koren->levo, m);
     } else { // cvor je list
        free (koren);
        koren = NULL;
     }
   }
   return koren;
}
```

Složenost navedenog algoritma je $O(h)$. Objasniti zašto.

3. Stablo je balansirano ako *za svaki čvor* važi: broj čvorova u levom podstablu razlikuje se najviše za jedan u odnosu na broj čvorova u desnom podstablu. Napisati funkciju u C-u koja izvršava balansiranje datog BSP.

```
cvor* balans_u (cvor* koren) {
  if (koren) {
    int k = broj_cvorova (koren->levo) - broj_cvorova (koren->desno);
```

```
    for (; k>1; k-=2) {
      koren->desno = dodaj_u (koren->desno, koren->broj);
      koren->broj = max_u (koren->levo);
      koren->levo = izost_u (koren->levo, koren->broj);
    }
    for (; k<-1; k+=2) {
      koren->levo = dodaj_u (koren->levo, koren->broj);
      koren->broj = min_u (koren->desno);
      koren->desno = izost_u (koren->desno, koren->broj);
    }
    koren->levo = balans_u (koren->levo);
    koren->desno = balans_u (koren->desno);
  }
  return koren;
}
int broj_cvorova (cvor *koren)
  { return koren ? 1 + vel (koren->levo) + vel (koren->desno) : 0; }
```

Navedeni algoritam ima složenost $O(n^2 \cdot h)$. Objasniti zašto.
Koja je složenost pretrage kod balansiranih stabala?

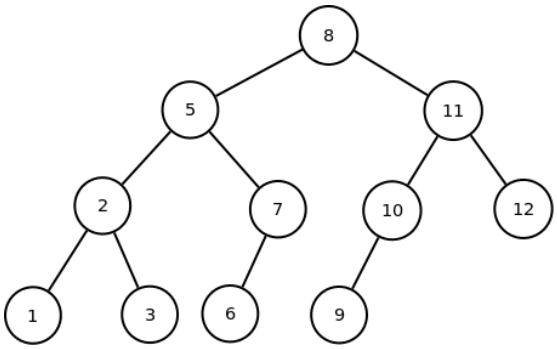
4. Za čvor binarnog stabla kažemo da je *balansiran* ako se broj čvorova u njegovom levom i desnom podstablu razlikuju najviše za 1. Napisati pseudokod algoritma koji u datom binarnom stablu T štampa sve *kritične* čvorove (čvorove koji nisu balansirani čvorovi a čiji su svi potomci balansirani čvorovi). Složenost algoritma treba da je $O(n)$, gde je n broj čvorova u stablu.

Algoritam kriticni(v)
Ulaz v (čvor v, tj. pokazivač na čvor v)
Izlaz vraca -1 ako se negde u stablu v nalazi kritican cvor, inace broj cvorova u stablu v; svi kriticni cvorovi u stablu se ispisuju

```
{
  if (v == NULL)
    return 0;
  l = kritican (v->l);
  d = kritican (v->d);
  if (l == -1 || d == -1) // ako u bilo kom podstablu od v
    return -1;            // ima kriticnih onda i u v ima kriticnih
  else if (abs (l-d) <= 1)  // u ovom stablu nema kriticnih
    return l+d+1;           // pa se vraca broj cvorova
  else {
    stampaj (v); // naisli na kritican cvor => cvor je nebalansiran
    return -1;   // pa medju njegovim precima nema kriticnih cvorova
  }
}
```

5. Elementi skupa A (međusobno različiti) smešteni su u BSP. Konstruisati algoritam koji za dati element a $\in$ A određuje sledeći po veličini element skupa A. Algoritam treba da bude složenosti $O(h)$, gde je h visina stabla.

Rešenje: Sledbenik zadatog čvora je čvor sa najmanjim ključem koji je veći od ključa datog čvora. Nađite sledbenika za 1, 2, 3, 5, 7, 11, 12. Da li je sledbenik potomak u stablu ili predak?



Koji čvor nema sledbenika? Samo najdešnji čvor u stablu (koji nema desnog sina i koji je desni sin svog oca), jer je on čvor sa maksimalnom vrednošću ključa (na slici je to čvor 12).

Karakteristični slučajevi za čvor v čiji ključ ima vrednost elementa a iz skupa S su:

1. v ima desno podstablo => sledbenik se traži kao najmanji u desnom podstablu (ključ najlevljeg čvora u desnom podstablu). Na primer za čvorove 8, 5 ili 11.

2. v nema desnog sina => sledbenik je čvor w koji je najniži predak od v takav da je levi sin od w predak čvora v. Ova operacija se najefikasnije izvodi ako se za svaki čvor čuva i polje otac sa pokazivačem na oca čvora (otac korena ne postoji pa je njegov pokazivač NULL). Tako se ide od čvora v prema korenu sve dok se ne nađe čvor w koji je levi sin svog oca. U slučaju da nema takvog pretka w, onda je a najveći element skupa i nema svog sledbenika.

U svakoj situaciji, sledbenik se nalazi bez poređenja ključeva zahvaljujući principu uređenosti stabla. Sledi algoritam za nalaženje sledbenika za zadatu vrednost ključa čvora ukazanog sa v koji vraća adresu sledbenika ili NULL. Koristi se i poziv algoritma BSP_MIN koji u BSP traži ključ sa minimalnom vrednošću.

Algoritam BSP_sledbenik(v)
Ulaz v (broj v ciji se sledbenik trazi)
Izlaz cvor q koji sadrzi sledbenika broja v ili NULL ako sledbenik ne postoji

```
{
pom = pronadji_cvor_sa_vrednoscu (v);  // obicna pretraga u stablu
if (pom->Desni != NULL)
  return BSP_min (pom->Desni);
else
  {
    q = pom->Otac;

    // dok god je cvor desni sin penjemo se uz stablo
    while (q! = NULL && pom == q->Desni)
    {
      pom = q;
      q = pom->Otac; //pom=koren => q=NULL, situacija kod a=12
    }
    return q;
  }
}
```

Algoritam BSP_min(Koren)
Ulaz Koren (pokazivač na koren BSP)
Izlaz čvor sa minimalnom vrednošću ključa

```
{
  p = Koren;
  while (p->Levi! = NULL)
    p = p->Levi;
  return p;
}
```

Objasniti zašto je složenost algoritma $O(h)$.

6. Konkatenacija je operacija nad dva skupa koja zadovoljavaju uslov da su svi ključevi u jednom skupu manji od svih ključeva u drugom skupu. Rezultat konkatenacije je unija skupova. Konstruisati algoritam za konkatenaciju dva BSP u jedno BSP. Vremenska složenost algoritma mora biti $O(h)$ u najgorem slučaju, gde je h veća od visina dva stabla.

REŠENJE: Neka su zadata dva BSP stabla T, G, tako da su svi ključevi stabla G veći od svih ključeva stabla T. Neka (bez smanjenja opštosti) visina h stabla G nije manja od visine stabla T.

1. Ukoliko je bilo koje od stabala prazno vratiti drugo stablo.

2. Pronaći u stablu G čvor sa najmanjim ključem, neka je to ključ v (npr, sve dok je moguće spuštati se niz stablo polazeći od korena ulevo i to se može obaviti za $O(h)$ vremena u najgorem slučaju)

3. Ukloniti čvor koji sadrži v iz stabla G ($O(h)$ vremena u najgorem slučaju)

4. Formirati za $O(1)$ vremena novo stablo sa korenom koji sadrži ključ v čija podstabla su stabla T (levo) i stablo G bez čvora koji sadrži v (desno). Novoformirano stablo ostaje binarno stablo pretrage, jer je koren sa najmanjim ključem u G, tako da desno podstablo može biti G. S druge strane, po uslovu s početka, svi ključevi stabla G (pa i v) su veći od svih ključeva stabla T, te T može biti levo podstablo.

ZADATAK Napisati pseudokod koji odgovara priloženom objašnjenju!

7. Napisati funkciju u C-u kojom se proverava da li su dva zadata binarna stabla ekvivalentna po strukturi i sadržaju unutar čvorova.

NAPOMENA: Nema pretpostavke da je u pitanju stablo pretrage!!!

Uputstvo: Problem se može rešavati rekurzivnim načinom razmišljanja. Testira se da li su dva zadata binarna stabla neprazna:

1. ako su oba prazna, ekvivalentna su (return 1)

2. ako su oba stabla neprazna, proverava se da li oba korena imaju isti ključ

3. ako ga imaju, rekurzivno se proverava da li su ekvivalentna leva podstabla i da li su ekvivalentna desna podstabla

4. ako su oba uslova ispunjena, onda su stabla ekvialentna; inače, stabla nisu ekvivalentna (return 0)

```
int EKV(cvor *d1, cvor *d2)
{
  if (d1 == NULL && d2 == NULL)
    return 1;    /* ako su oba prazna, ekvivalentna su */
  if (d1 == NULL || d2 == NULL)
    return 0;    /* inace, ako je jedno prazno, nisu ekvivalentna */
  if (d1->broj != d2->broj)
    return 0;    /* razliciti koreni */
  return (EKV(d1->levo,d2->levo) && EKV(d1->desno,d2->desno) );
}
```

Složenost navedenog algoritma je $O(min(n, m))$. Zašto?

8. Napisati funkciju u C-u kojom se proverava da u datom binarnom stablu d1 postoji podstablo koji je u smislu prethodnog zadatka ekvivalentno po strukturi i sadržaju sa zadatim stablom d2. Vratiti ili pokazivač na nađeno podstablo ili NULL u suprotnom.

```
cvor* podstablo (cvor *d1, cvor *d2){
  if (EKV(d1,d2))
    return d1;
  if (d1){    /* provera da li je d1 neprazno */
    cvor *d=podstablo(d1->levo,d2);
    if (d)   /* ako je vracen pokazivac razlicit od NULL */
      return d;
    else
      return podstablo(d1->desno,d2);
  }
  return NULL;
}
```

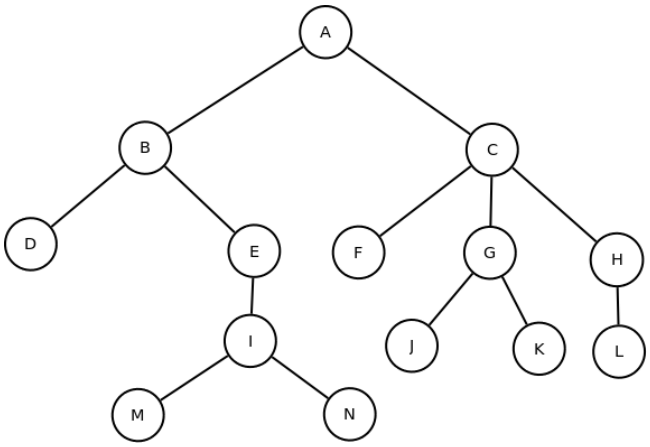
Kolika je složenost navedenog algoritma?

Rešite prethodna 2 zadatka pod pretpostavkom da su u pitanju BSP. Kolika je složenost algoritama u tom slučaju?

9. Da li je operacija brisanja čvora opisana u zadatku 2 komutativna u smislu da brisanje čvora x, te potom y iz BSP daje stablo koje je jednako onom koje se dobija brisanjem čvora y, a zatim čvora x? Obrazložiti odgovor ili dati kontraprimer.

1.1 Obilasci stabala

Koji su sve mogući obilasci datog stabla? (Kada čvor ima više od 2 sina, smatrati da je samo prvi sin levi sin, a svi ostali desni sinovi).

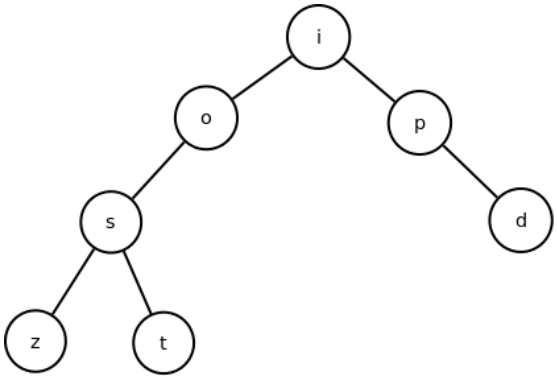


Rezultat preorder (KLD) obilaska datog stabla je A,B,D,E,I,M,N,C,F,G,J,K,H,L.
Rezultat inorder (LKD) obilaska datog stabla je: D,B,M,I,N,E,A,F,C,J,G,K,L,H.
Rezultat postorder (LDK) obilaska datog stabla je: D,M,N,I,E,B,F,J,K,G,L, H,C,A.
Rezultat levelorder (po nivoima) obilaska datog stabla je: A,B,C,D,E,F,G,H,I,J,K,L,M,N.

10. Nacrtati binarno stablo za koje INORDER (LKD) obilazak daje *zstoipd*, a PREORDER (KLD) obilazak *iosztpd*. Ključ čvora je slovo engleske abecede.

SKICA REŠENJA:

1. KLD[0] mora biti koren stabla.
2. Ako je n ukupan broj čvorova i ako je i pozicija korena u niski LKD, onda je LKD$[0..i-1]$ zapis levog podstabla u redosledu LKD i LKD$[i+1..n-1]$ zapis desnog podstabla u redosledu LKD.
3. Ako je i pozicija korena u niski LKD, onda je KLD$[1..i]$ zapis levog poddrveta u redosledu KLD (u levom poddrvetu je i čvorova). Slično, KLD$[i+1..n-1]$ je zapis desnog poddrveta u poretku KLD.
4. Rekurzivno primenimo pravila 1..3 (tj. KLD[1] je koren levog podstabla, KLD$[i+1]$ je koren desnog stabla, $i = 4$ jer LKD[4]=koren='i').



11. U čvoru binarnog stabla zapisano je slovo engleske abecede. Odrediti binarno stablo za koje INORDER (LKD) obilazak daje *dacbfegijhkl*, a PREORDER (KLD) obilazak *gfdcabehijkl*.

12. U čvoru binarnog stabla zapisano je slovo engleske abecede. Ispisati rezultat POSTORDER obilaska, ako je rezultat INORDER (LKD) obilaska *deacbgfhjlkmi*, a PREORDER (KLD) obilaska *hgedcabfijklm*

13. Napisati funkciju u C-u koja za date inorder i preorder zapise (u vidu niski lkd i kld) jednog istog drveta nalazi i ispisuje njegov postorder zapis. Pretpostaviti da stablo ima do 80 čvorova označenih jednim ASCII znakom.

14. Znamo da se iz inorder i preorder obilaska binarnog stabla pretrage može rekonstruisati grafička predstava tog stabla. Da li je to moguće ako imamo zadate preorder i postorder obilaske? Odgovor obrazložiti primerom ili rečima.

REŠENJE: Ne.
Primer:
Preorder: AB
Postorder: BA
Postoje dva stabla koja odgovaraju ovim obilascima: