Fakultet elektrotehnike i računarstva

Zavod za primjenjeno računarstvo

**Napredni algoritmi i strukture podataka**

1. laboratorijska vježba

Tyrannizer, 10563

Zagreb, 2. 11. 2016.

# Zadatak

Zadatak za 17 bodova

2) Napisati program koji učitava niz prirodnih brojeva iz ASCII datoteke (po pretpostavci, datoteka nije prazna) i upisuje ih u (inicijalno prazno) **crveno-crno (RB) stablo** istim redoslijedom kao u datoteci. Program može biti konzolni ili s grafičkim sučeljem, po vlastitom izboru. Konzolni program naziv ulazne datoteke treba primiti prilikom pokretanja kao (jedini) argument s komandne linije, a grafički iz odgovarajućeg sučelja po pokretanju programa. Nakon upisa svih podataka, ispisati izgrađeno stablo na standardni izlaz (monitor) pri čemu treba nekako naznačiti boje čvorova. Program zatim treba omogućiti dodavanje novih čvorova i brisanje postojećih čvorova te nakon svake promjene treba ponovo ispisati stablo. Napomena: datoteka treba biti tekstualna, a brojevi u datoteci odvojeni po jednim razmakom (space). Kraj datoteke treba biti označen standardnim EOF znakom

# Rješenje zadatka

## Teorijski uvod

Crveno-crno stablo je binarno stablo za pretraživanje kod kojeg se uz podatke dodatno čuva i boja čvora, koja može biti ili crvena ili crna. Crveno-crna stabla su jedna od mnogih schema implementacije „balansiranih“ stabla čime se garantira da se osnovni skup operacija (minimum, maximum, insert, delete, …) izvršava u vremenu u najgorem slučaju, gdje označava duljinu ulaznog niza [1]. Svaki čvor stabla sadrži atribute *boja*, *vrijednost*, *lijevo dijete*, *desno dijete*, *roditelj*.

Crveno-crno stablo je binarno stablo koje zadovoljava sljedeća crveno-crna svojstva [2]:

1. Svaki čvor je ili crven ili crn
2. Korijen stabla je crn
3. Svaki list (null) je crn
4. Ako je čvor crven, oba djeteta tog čvora su crna
5. Svaka staza od nekog čvora do bilo kojeg lista koji je njegov potomak prolazi istim brojem crnih čvorova

## Implementacija

Implementacija crveno-crnog stabla napisana je u jeziku C++ koristeći razvojno okruženje Visual Studio 15, za implementaciju nije korištena nijedna vanjska biblioteka već isključivo standardna biblioteka jezika C++. Implementiran je konzolni program koji se pokreće iz naredbene linije. Program očekuje jedan ulazni parametar koji označava ime ulazne datoteke iz koje čita niz prirodnih brojeva i upisuje ih u stablo, nakon toga moguće je unositi i brisati čvorove upisivanjem naredbi „i 24“ odnosno „d 24“ kako bi se zadalo upisivanje odnosno brisanje čvora s vrijednosti 24. Također unosom naredbe „exit“ završava se izvođenje programa.

### Ispis stabla

Za ispis stabla odabran je jednostavan tekstualni format u obliku „{pozicija čvora} {dubina} {boja čvora}: {vrijednost}“. Pozicija čvora sadrži oznaku \, L, R koje označavaju dali je čvor korijen, lijevo ili desno dijete. Dubina u stablu označava se razmacima, svaki čvor je za jednu razinu više uvučen od svojeg roditelja. Boja čvora sadrži oznaku B, R koja označava da li je čvor crn ili crven. Vrijednost predstavlja vrijednost čvora.

Ispis:

\ B: 18

L B: 5

L R: 2

L B: 1

R B: 3

R R: 4

R R: 16

L B: 15

R B: 17

R B: 24

L B: 22

L R: 21

R R: 23

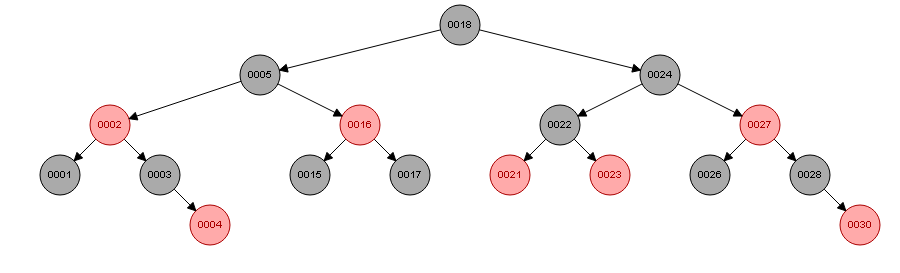
R R: 27

L B: 26

R B: 28

R R: 30

Predstavlja stablo



### Unos novog čvora

Novi čvor se u stablo unosi kao crveni čvor isto kao što se unose i čvorovi u obično binarno stablo za pretraživanje a nakon toga se popravljaju moguća narušena crveno-crna svojstva. Unošenjem čvora u stablo moguće je narušiti svojstva 2, koje zahtjeva da korijen stabla bude crn i svojstvo 4 koje zahtjeva da djeca crvenog čvora budu crna. Pseudokod za popravljanje svojstava dan je u nastavku [1].

fix\_insert\_violations(z)

while parent(z).color == RED

if parent(z) == left\_child(grandparent(z))

y = right\_child(grandparent(z))

if y.color == RED

parent(z).color = BLACK

y.color = BLACK

grandparent(z).color = RED

z = grandparent(z)

else

if z = right\_child(parent(z))

z = parent(z)

left\_rotate(z)

parent(z).color = BLACK

grandparent(z).color = RED

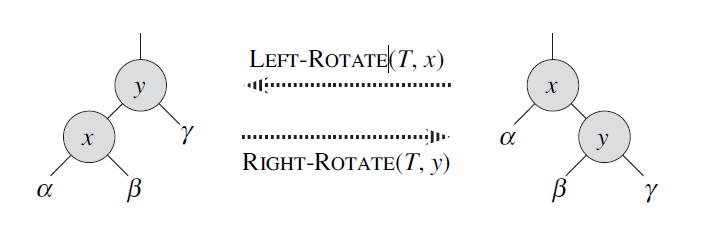
right\_rotate(grandparent(z))

else

// same as if clause with swapped left and right

root.color = BLACK

U pseudokodu se koristi nekoliko pomoćnih funkcija *parent* i *grandparent* koje vračaju roditelja i djeda danog čvora, *left\_child* i *right­\_child* koje vračaju lijevo i desno dijete danog čvora i funkcije *left\_rotate* i *right\_rotate*, koje rotiraju stablo oko danog čvora zadržavajući pritom svojstva binarnog stabla za pretraživanje.



### Brisanje postojećeg čvora

Brisanje čvora iz crveno-crnog stabla provodi se na sličan način kao i brisanje iz običnog binarnog stabla za pretraživanje uz dodatak da se bilježe dodatni podaci kako bi znali da li su narušena crveno-crna svojstva prilikom brisanja. Prilikom brisanja mogu se narušiti svojstva 2, 4 i 5. Pseudokod za brisanje čvora i popravak svojstava dan je u nastavku [1].

delete(z)

y = z

original\_color = y.color

if left\_child(z) == null

x = right\_child(z)

transplant(z, right\_child(z))

else if right\_child(z) == null

x = left\_child(z)

transplant(z, left\_child(z))

else

y = min(right\_child(z))

original\_color = y.color

x = right\_child(y)

if parent(y) == z

parent(x) = y

else

transplant(y, right\_child(y))

right\_child(y) = right\_child(z)

parent(right\_child(y)) = y

transplant(z, y)

left\_child(y) = left\_child(z)

parent(left\_child(y)) = y

y.color = z.color

if original\_color == BLACK

fix\_delete\_violations(x)

fix\_delete\_violations(x)

while x != root and x.color == BLACK

if x == left\_child(parent(x))

w = right\_child(parent(x))

if w.color == RED

w.color = BLACK

parent(x).color = RED

left\_rotate(parent(x))

w = right\_child(parent(x))

if has\_black\_children(w)

w.color = RED

x = parent(x)

else

if right\_child(w).color == BLACK

left\_child(w).color = BLACK

w.color = RED

right\_rotate(w)

w = right\_child(parent(x))

w.color = parent(x).color

parent(x).color = BLACK

right\_child(w).color = BLACK

left\_rotate(parent(x))

x = root

else

// same as if clause with swapped left and right

x.color = BLACK

U pseudokodu se koristi nekoliko pomoćnih funkcija *transplant(u, v)* koja seli podstablo iz čvora *u* sa podstablom u čvoru *v,* i funkcija *has\_black\_children* koja provjerava da li dani čvor ima crnu dijecu.

# Zaključak

Crveno-crno stablo je „jednostavna“ podatkovna struktura koja nam omogućuje održavanje aproksimativno balansiranog stabla uz relativno malen vremenski i prostorni trošak, što je dobra alternativa običnim binarnim stablima za pretraživanje čije performanse mogu degradirati u performanse liste u nekim slučajevima. Moguća poboljšanja trenutne implementacije su interno spremati čvorove u polje kako bi se lokalizirao pristup memoriji.

# Literatura

Tablica . Literatura

|  |  |
| --- | --- |
| **Vrsta** | **Format** |
| Poglavlje knjige | Cormen, et. al.; Red-Black Trees, Introduction to Algorithms, MIT Press, 2009, 308 - 330 |
| Predavanja (slajdovi) | Hlupić, Kalpić: Napredne strukture podataka, 2009. |
| Animacija RB stabla | Red/Black Tree Visualization, <https://www.cs.usfca.edu/~galles/visualization/RedBlack.html>, 2.11.2016. |
|  |  |
|  |  |