Fakultet elektrotehnike i računarstva
Zavod za primjenjeno računarstvo

Napredni algoritmi i strukture podataka

1. laboratorijska vježba

<Ime i Prezime, JMBAG >

Zagreb, 05.11.2016.

1. Zadatak

Napisati program koji učitava niz prirodnih brojeva iz ASCII datoteke (po pretpostavci, datoteka nije prazna) i upisuje ih u (inicijalno prazno) AVL 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). Program zatim treba omogućiti dodavanje novih čvorova te nakon svake promjene treba ponovo ispisati stablo.

2. Rješenje zadatka

2.1. Teorijski uvod

AVL stabla razvila su se iz potrebe da stablo bude uravnoteženo lokalno iako ne jamči savršenu uravnoteženost cijelog stabla. AVL stablo binarno je stablo koje zadovoljava AVL definicijsko pravilo- faktor ravnoteže svakog čvora u stablu mora biti 1, 0 ili -1. Faktor ravnoteže računa se po formuli FR(faktor ravnoteže) = visina desnog podstabla - visina lijevog podstabla.

Problem održavanja AVL stabla svodi se na uravnotežavanje stabla nakon dodavanja novih ili brisanja postojećih čvorova. Proučavajući teoriju došlo se do rješenja upravo uz pomoć praćenja faktora ravnoteže pojedinih čvorova.

2.2. Implementacija

Implementacija rješenja problema isprogramirana je u programskom jeziku C++ koristeći Eclipse razvojno okruženje. Program (NASP_LAB1.exe) pokreće se iz naredbene linije(cmd) naredbom „NASP_LAB1.exe <ime_ulazne_datoteke>“.

Program stvara uravnoteženo AVL stablo u kojeg se mogu upisati novi elementi, uz očuvanje ravnoteže stabla. Program se terminira upisom slova 'e' u naredbenu liniju.

2.2.1. Dodavanje novog čvora(add, add_node)

Pri dodavanju novog čvora iznimno je važno pripaziti na to da korijen stabla bude aktualiziran. To omogućava funkcija *add(int).* Samo dodavanje novih elemenata izvršava se isto kao i u klasičnom binarnom stablu- počevši od korijena stabla uspoređuju se vrijednosti postojećih elemenata s vrijednošću novog elementa. Pozivom funkcije *add_node(node*, int),* ukoliko novi element ima manju vrijednost od trenutnog čvora, rekurzivno pozovi funkciju *add_node* koja za *node** argument prima adresu čvorovog lijevog djeteta. Analogno za desnu stranu, ukoliko element ima veću vrijednost od vrijednosti trenutnog čvora.

U slučaju narušavanja faktora ravnoteže nekog od čvorova na višim razinama potrebno je izvršiti odgovarajuću rotaciju kako bi se stanje popravilo.

Pseudokod uravnotežavanja AVL stabla:

```
Izračunaj novi FR roditelja;
Ako je FR roditelja == -2
        Ako je FR čvora == -1
                Rotiraj trenutni čvor udesno;
                Return;
        Inače ako je FR čvora == 1
                R = desno dijete;
                Rotiraj R ulijevo;
                Rotiraj R udesno;
                Return;
Inače ako je FR roditelja == 2
        Ako je FR čvora == 1
                Rotiraj trenutni čvor ulijevo;
                Return;
        Inače ako je FR čvora == -1
                R = lijevo dijete;
                Rotiraj R udesno;
                Rotiraj R ulijevo;
                Return;
Ponovi za roditelja;
```

Iz pseudokoda se mogu prepoznati četiri moguća slučaja neuravnoteženosti stabla. Izvršavanjem jedne rotacije(maksimalno jedne „dvostruke“) stanje AVL stabla će, nakon dodanog novog čvora, ponovno biti ispravno. Nakon dodavanja novog čvora ključno je i osvježiti dubinu svih čvorova.

2.2.2. Ispis stabla(preorder)

Ispis stabla obavlja se na preorder način, tj. ispisivat će se vrijednost roditelja, zatim lijevog djeteta pa desnog djeteta.

Izvorni kod izvedbe ispisa:

```
void AVL_tree::preorder(node* nodee)
{
      if (nodee == NULL)
      {
            return;
      }
      printf("%d ", nodee->value);
      preorder(nodee->left_child);
      preorder(nodee->right_child);
      }
```

2.2.3. Lijeva/desna rotacija(left_rotation, right_rotation)

Sama rotacija implementirana je uz pomoć pomoćnog čvora <left | right>Child, a kao povratnu vrijednost vraća pokazivač na čvor. Za primjer opisat ćemo desnu rotaciju. Pri ulasku u funkciju rotacije kreiramo novi, pomoćni čvor *leftChild* koji pokazuje na lijevo dijete roditeljskog čvora. Nakon toga provjeravamo postoji li *leftChild*. Ako postoji roditeljski čvor pokazivat će na desno dijete *leftChild*-a. U idućem koraku provjeravamo je li *leftChild* == NULL, tj. pokazuje li na išta. Ako **nije** NULL usmjeri njegov desni pokazivač na njegovog bivšeg roditelja(argument rotacijske funkcije), osvježi njegovu dubinu i vrati ga kao *return* rotacijske funkcije. Također, osvježi dubinu argumenta funkcije, bivšeg roditelja *leftChild* čvora.

Ako pak je jednak NULL, osvježi dubinu argumenta i vrati iz funkcije isti argument pomoću kojeg je pozvana.

Postupak je analogan za desnu rotaciju uz zamjene odgovarajućih smjerova pokazivača na djecu.

3. Zaključak

Radom na implementaciji rada AVL stabla potvrdio sam teoretiziranu uspješnost i dinamičnost korištenja ove strukture u svrhu čuvanja cjelobrojnog tipa podatka. Algoritam je brz, relativno jednostavan i precizan.

Daljnja poboljšanja ovog algoritma mogla bi ići u smjeru čuvanja drugih tipova podataka u istoj strukturi u svrhu njihovog bržeg pretraživanja i kvalitetnijeg iskorištavanja memorijskog sustava koji ga koristi.

4. Literatura

[1] N. Hlupić, D. Kalpić, Napredne strukture podataka (2009)