Univerzitet u Sarajevu
Prirodno-matematički fakultet
Odsjek za matematičke i kompjuterske nauke
Smjer : Kompjuterske nauke

Strukture podataka i algoritmi

Projekat – Tema 4

Studentica : Nejra Ćeranić
Broj indeksa : 5967/M

Predmetni asistent : Admir Beširević
Predmetni profesor : Esmir Pilav
Datum : 17.01.2024.

Uvod

Tema ovog projekta se fokusira na pristup održavanju balansiranosti binarnog stabla pretrage. Postoje mnoge metode koje služe za održavanje balansiranosti binarnog stable pretrage, kao što su na primjer AVL stabla. Jedan od načina je da se simulira slučajni redoslijed umetanja u stablo, kao što mi to radimo u ovom projektu. Ključne operacije, uključujući pretragu, umetanje, brisanje i razdvajanje, trebaju se izvoditi efikasno. Umetanje se vrši na klasičan način, ali dodatno se vrše rotacije kako bi se očuvala osobina heapa. Brisanje uključuje zamjenu čvora sa sljedbenikom ili prethodnikom u sortiranom redoslijedu, dok razdvajanje, posebna operacija, umeće element sa zadatim ključem, kojem se dodjeljuje prioritet veći od prioriteta korijena, pretvarajući ga u novi korijen. Smatra se da će binarno stablo implementirano na ovaj način imati veliku vjerovatnoću da bude balansirano.

Sadržaj

Ovaj projekat se sastoji od sljedećih file-ova :

- *„Stablo.cpp“* – file u kojem su implementirane sve funkcije navedene u file-u *„stablo.h“*
- *„Stablo.h“* – file u kojem je implementirana template klasa Stablo,struktura Cvor, kao i njihovi atributi i deklaracije funkcija koje su implementirane u file-u *„Stablo.cpp“*
- *„main.cpp“* – file u kojem se vrše testiranja implementiranih funkcija

Unutar ovog projekta u file-u „Stablo.h“ je definisana template **klasa Stablo** unutar koje se nalazi sljedeće :

Struktura **struct Cvor –** struktura koja sadrži sve potrebne podatke o jednom čvoru. Njeni atributi su :
Tip element – vrijednost elementa u čvoru, odnosno u našem slučaju ključ za pretragu
int prioritet – random vrijednost koja se dodjeljuje elementu prilikom unosa vrijednosti u stablo, odnosno pozivanjem konstruktora
Cvor *ld, *dd, *rod – tri, redom, pokazivača koji pokazuju na lijevo dijete čvora, desno dijete čvora i na njegovog roditelja

Atributi :
int br_elem – atribut koji nam daje informaciju o broju čvorova unutar stabla
Cvor* korijen – pokazivač na čvor koji je korijen stabla

Konstruktori :
Stablo() – konstruktor bez parametara
Stablo(const Stablo<Tip>& s) - konstruktor kopije
Stablo(Stablo<Tip>&& s) - pomjerajući konstruktor
Stablo(Cvor* korijen) – konstruktor deklarisan u privatnom dijelu klase jer ga koristi samo funkcija članica Separate(). Pošto u toj funkciji stablo dijelimo na dva podstabla – lijevo i desno, uz pomoć ovog konstruktora inicijaliziramo lijevo i desno dijete starog korijena korjenovima novih podstabala, što čini njihove roditelje nullpointerima.

Operatori dodjele :
Stablo<Tip>& operator=(const Stablo<Tip> &rhs) - operator dodjele
Stablo<Tip>& operator=(Stablo<Tip> &&rhs) - pomjerajuci operator dodjele

Destruktor :
~Stablo() – oslobađa zauzetu memoriju uz pomoć funkcije obrisiStablo(Cvor *korijen)

Privatne funkcije :
void obrisiStablo(Cvor* korijen) – rekurzivna funkcija koju koristi destruktor i operator=. Prima pokazivač na korijen stabla koje treba obrisati i prolaskom kroz njega postavlja sve čvorove na nullpointer, te na kraju briše pokazivač na korijen.

Cvor* kopirajStablo(Cvor* korijen, Cvor* rod_korijena, int& broj_elemenata) – rekurzivna funkcija koju koristi konstruktor kopije i operator=. Prima pokazivač na korijen stabla koje se kopira, pokazivač na roditelja korijena što je nullpointer i broj elemenata. Za svaki čvor se formira novi čvor u koji se kopiraju svi elementi starog čvora.

void desnoRotiraj(Cvor* dijete,Cvor* roditelj) – funkcija koju poziva javna funkcija rotiraj(Cvor *cvor). Desna rotacija se poziva ukoliko je čvor koji je potrebno rotirati lijevo dijete roditelja. U ovom slučaju se rotiraju roditelj i dijete, te se adaptiraju potrebni pokazivači poput zamjene roditelja, ispravnog korigovanja djece i ostalih potrebnih atributa.

void lijevoRotiraj(Cvor* dijete,Cvor* roditelj) - funkcija koju poziva javna funkcija rotiraj(Cvor *cvor). Lijeva rotacija se poziva ukoliko je čvor koji je potrebno rotirati desno dijete roditelja. Postupak je isti kao i kod desne rotacije.

Javne funkcije :
bool Find(Tip el) – funkcija koja prima vrijednost elementa koji tražimo i vraća bool da li postoji čvor sa tim elementom u tom stablu ili ne. Prolaskom kroz stablo upoređujemo trenutnu vrijednost, koja je na početku korijen, sa traženom vrijednosti, te se tako krećemo kroz stablo. Ukoliko je tražena vrijednost manja od trenutne, trenutna postaje lijevo dijete, a u supratnom trenutna postaje desno dijete sve dok ne naiđemo na nullpointer. Ukoliko su jednaki vraća true, a ako prođemo kroz čitavo stablo i nismo vratili true, vraćamo false.

void rotiraj(Cvor* cvor) – ova funkcija se poziva prilikom unosa ili brisanja elemenata, odnosno funkcije Insert() ili Delete() i služi da zadrži osobinu prioriteta. Osobina prioriteta nije zadovoljena ukoliko je prioritet roditelja manji od prioriteta djeteta. Koristi funkcije desnoRotiraj(Cvor* dijete,Cvor* roditelj) i lijevoRotiraj (Cvor* dijete,Cvor* roditelj).

void Insert(const Tip& element,const bool& kljuc=false) – funkcija koja nam omogućava unos elemenata u stablo po vrijednosti. Prvi parametar predstavlja element koji unosimo u stablo,a drugi parametar (const bool& kljuc) se pri pozivu stavlja uvijek na defaultnu vrijednost false jer nam je on potreban samo u slučaju kada se Insert() poziva iz funkcije Separate(). Kada vršimo klasični unos (to jeste kada ta vrijednost nije ključ) prvo provjeravamo da li je stablo prazno jer će u tom slučaju ta prva vrijednost biti korijen, te će ujedno i imati najveći

prioritet jer je jedina vrijednost u stablu, tako da neće biti potrebe za ispitivanjima koje se vrše u ostalim slučajevima. U ostalim slučajevima, kada stablo nije prazno, ispitujemo relaciju vrijednosti (atribut element) trenutnog čvora i vrijednosti koju unosimo. Ukoliko je vrijednost koju unosimo veća od trenutne krećemo se u desnom smjeru, a ukoliko je manja krećemo se u lijevom smjeru. Krećemo se sve dok ne dobijemo da je odgovarajuće dijete, gdje možemo smjestiti naš novi čvor sa elementom koji unosimo, nullpointer, te ga tu smještamo. Modifikacije stabla koje vršimo u ovisnosti od prioriteta kako ne bi došlo do slučaja da roditelj ima veći prioritet od djeteta, se rade uz pomoć funkcije rotiraj(Cvor* cvor). Ta funkcija se poziva nakon svakog smještanja novog elementa.

Cvor* nadjiCvor(const Tip& el) – funkcija koja prima vrijednost elementa koji tražimo, a vraća pokazivač na isti. Ovu funkciju pozivamo unutar Delete(). Sadržaj ove funkcije je skoro isti kao i kod funkcije Find(), samo što nam se ovdje vraća pokazivač na traženi čvor, a ne bool, te nismo mogli iskoristiti Find() u ove svrhe.

Cvor* nadjiSljedbenika(Cvor* trenutni) – funkcija koja prima pokazivač na element, a vraća pokazivač na njegovog sljedbenika. Ovu funkciju pozivamo unutar Delete() i to samo u slučaju kada čvor koji želimo izbrisati ima dvoje djece. Samim tim, sljedbenik jednog čvora jeste njegovo desno dijete ili zadnje lijevo dijete njegovog desnog djeteta. Sljedbenik nam je ispočetka desno dijete i onda ukoliko desno dijete ima lijevo dijete, krećemo se while petljom sve dok lijevo dijete ne bude nullpointer, a zadnje koje nije nullpointer jeste upravo sljedbenik.

Cvor* proslijediDijete(Cvor* trenutni) – funkcija koja pronalazi dijete koje ima veći prioritet od roditelja,a ako je ijedan od njih nullpointer, vraća onaj koji nije. Ova funkcija se poziva unutar Delete() u slučaju kada je potrebno opet izvšiti rotaciju jer je prioritet narušen.

void Delete(const Tip& element) – funkcija koja prima vrijednost elementa čiji je čvor potrebno izbrisati. Na početku nađemo čvor koji je potrebno izbrisati uz pomoć funkcije nadjiCvor(const Tip& el). Funkcija Delete() se sastoji iz 3 slučaja :
1. slučaj – čvor koji je potrebno izbrisati nema djece. U tom slučaju jedino postavljamo pokazivač na dijete od roditelja na nullpointer.
2. slučaj – čvor koji je potrebno izbrisati ima jedno dijete. U tom slučaju vršimo zamjenu roditelja i djeteta i korigujemo potrebne atribute.
3. slučaj – čvor koji je potrebno izbrisati ima dvoje djece. U tom slučaju se čvor koji trebamo izbrisati mijenja sa desnim djetetom ili sa zadnjim lijevim djetetom desnog djeteta. Uz pomoć funkcije nadjiSljedbenika(Cvor* trenutni) dolazimo do pravilnog odabira sljedbenika sa kojim ćemo zamijeniti čvor koji je potrebno izbrisati. Ovdje je potrebno izvršiti rotaciju zato što je vrlo moguće da sljedbenik ima manji prioritet od starog djeteta čvora izbrisati, ili da je lijevo dijete desnog djeteta sada u stablu iznad njega, a samim tim što je prije bio ispod, znači da ima manji prioitet.
Na kraju oslobađamo memoriju zauzetu varijablom izbrisati, koja je predstavljala čvor koji je potrebno izbrisati i smanjimo broj elemenata stabla.

Prijateljske funkcije :
friend pair<Stablo<tip>,Stablo<tip>> Separate(Stablo<tip>& s,const tip& k) – funkcija koja prima stablo i vrijednost koja treba da postane korijen stabla,a vraća par podstabala. Ta podstabla se formiraju tako što uz pomoć konstruktora Stablo(Cvor* korijen) lijevo i desno dijete postavimo da budu dva nova korijena, redom, lijevog i desnog podstabla, te na taj razdvajamo naše stablo.

Pored svega implementiranog, imamo i **operator<<** koji služi za ispis stabla. Pored implementacije već navedenog operatora ispisa, koriste se i sljedeće funkcije:

Javna funkcija **void obradiElemente(void (f)(Tip)) const.**
Privatna funkcija **void obradiElementeRekurzivno(Cvor* korijen, void (f)(Tip)) const.**
Funkcija **void ispisi(Tip el).**