Univerzitet u Sarajevu
Prirodno – matematički fakultet
Informacione tehnologije

Naziv projekta: Balansirajuće stablo (Tema 2)

Predmet: Strukture podataka i algoritmi

Predmetni asistent: Beširević Admir

Student: Čajić Dino

Broj indeksa: 122-IT/19

Sadržaj:

1. Uvod

Klasa „stablo“ je generička struktura podataka koja čuva elemente kao binarno stablo pretrage. Odnosno, svi elementi koji se nalaze lijevo od nekog određenog čvora u stablu su manji od njega, a svi elementi desno od njega su veći.

Pretraživanje elemenata se vrši tako što krećemo od vrha stabla (korijena). Traženi element prvobitno poredimo sa korijenom stabla, a zatim u ovisnosti od vrijednosti elementa krećemo se u lijevom ili desnom pravcu od korijena. Ako je npr. tražena vrijednost manja od vrijednosti korijena onda znamo da, ako je elemenat u stablu, sigurno je u lijevom podstablu. Postupak poređenja ponavljamo sa korijenovim djetetom sve dok ne pronađemo traženi elemenat ili čvor koji nema djeteta na koji bismo trebali „preći“.

Umetanje i brisanje elemenata je zasnovano prvenstveno na pretraživanju. Za umetanje elementa radimo u suštini njegovo pretraživanje u stablu samo kada čvor nema dijete u koji bismo trebali „preći“ tu ga umetnemo.

Problem kod ovog umetanja je ako bismo npr. unosili brojeve po veličini, dobili bismo nešto što je po svojim karakteristika slično listama iz c++. Svaki element bi se dodao kao desno dijete i tada bismo imali brzinu pretrage elementa O(n) dok mi koristeći stablo želimo postići vrijeme O(logn).

To je moguće učiniti tako što balansiramo stablo. Stablo održava balansiranost na sljedeći način: kada se na visini barem 3 desi da je broj elemenata u jednom podstablu duplo veći od broja elemenata u drugom podstablu, podstablo tog čvora se mijenja. To se radi tako što se od elemenata podstabla napravi sortirani niz i onda se kreira podstablo koje je idealno balansirano.

Važno za napomenuti je to da se stablo sastoji od čvorova, a čvor je struktura unutar klase stablo. Svaki čvor čuva vrijednost odnosno „element“ tipa Tip jer je stablo generička klasa, pokazivač na čvor „ld“ (tj. lijevo dijete), pokazivač na čvor „dd“ (tj. desno dijete), pokazivač na čvor „rod“ (tj. roditelja), atribut „trenutniNivo“ tipa int koji predstavlja na kojem se nivou u stablu nalazi čvor i atribut „nivoi“ tipa int koji predstavlja najveću

udaljenost (tj. broj nivoa) između trenutno posmatranog čvora i čvorova ispod njega (tj. u njegovom podstablu).

2. Konstruktori

Stablo () - Konstruktor bez parametara koji samo postavlja korijen na nullptr i broj elemenata na 0.

~Stablo () - Destruktor koji briše sve elemente.

3. Funkcije

Cvor* FindCvor(Tip x) – Funkcija koja traži u stablu vrijednost x, te ako je pronađe vraća pokazivač na čvor tog elementa.

bool Find(Tip x) - Funkcija koja traži u stablu vrijednost x, te ako je pronađe vraća true odnosno false ako je ne pronađe.

Tip maximum (Cvor* korijen) – Funkcija koja pronalazi najveći element u stablu krećući se u desno koliko je moguće u odnosu na korijen stabla.

Tip minimum(Cvor* korijen) – Funkcija koja pronalazi najmanji element u stablu krećući se u lijevo koliko je moguće u odnosu na korijen stabla.

Tip maxi() – Funkcija koja samo poziva maximum.

Tip mini() – Funkcija koja samo poziva minimum.

void Insert(Tip x) – Funkcija koja umeće element u stablo, te na kraju poziva funkciju ProvjeriBalansiranost(Cvor* prethodni) da bi provjerio je li potrebno balansirati stablo i balansira ukoliko je potrebo.

void Insert2(Tip x) – Funkcija koja radi na identičan način kao Insert samo bez provjere za balansiranost na samom kraju.

void Obrisi(Tip x) – Funkcija koja briše čvor koji ima vrijednost x ukoliko se takav nalazi u stablu. Kada se pronađe takav čvor, provjerava se da li ima desno dijete. Ukoliko ima, pomjeramo se na taj čvor te onda idemo lijevo kroz stablo koliko god je to moguće. Kada dođemo do kraja, odnosno do tog čvora koji nema lijevo dijete, taj čvor uklanjamo sa te pozicije i postavljamo ga umjesto čvora kojeg smo prvobitno pronašli, odnosno čvor kojeg brišemo. Ovo radimo da bismo zadržali zakonitost koju ima binarno stablo. Ukoliko čvor koji brišemo nema desno dijete onda ga samo izbacimo iz stabla. Na kraju brisanja vršimu provjeru za balansiranost i istu ako je potrebna pomoću funkcije ProvjeriBalansiranost(Cvor* prethodni).

void Obrisi2(Tip x) – Funkcija koja radi isto što i funkcija Obrisi samo bez provjere za balansiranost na samom kraju.

void SmanjiNivoZaJedan (Cvor* c) – Funkcija koja rekurzivno smanjuje „trenutniNivo“ svim čvorovima koji se nalaze ispod čvora „c“ za 1.

void BrojNajduzih(Cvor* c, int k, int& br) – Funkcija koja rekurzivno prolazi sve čvorove ispod čvora „c“ i smiješta u varijablu „br“ broj čvorova koji imaju „trenutniNivo“ jednak broju „k“.

Cvor* idiDoKraja(Cvor* c) – Funkcija koja vraća pokazivač na čvor koji dobijemo krećući se u desno koliko god je moguće u odnosu na čvor „c“.

void DajSveCvorove(Cvor* p, vector<Tip> &v) – Funkcija koja smiješta sve elemente (vrijednosti) čvorova koji se nalaze ispod čvora „p“ (uključujući i njega) u vektor „v“.

Void ObrisiSveCvorove(Cvor* p)– Funkcija koja briše sve čvorove koji se nalaze ispod čvora „p“ (uključujući i njega).

void Balansiraj(vector<Tip> &v, int s, int k)– Funkcija prihvata sortirani vektor „v“ u kojem se nalaze elementi koje treba dodati u stablo u takvom poretku da stablo bude idealno balansirano. To se postiže tako što umetnemo element sa indexom (s+k)/2 gdje „s“ predstavlja početak vektora, a „k“ kraj vektora. Taj postupak ponavljamo za lijevu polovicu vektora, te za desnu dok „polovljenjem“ ne posjetimo i dodamo svaki element u vektoru.

void ProvjeriBalansiranost(Cvor* prethodni)– Funkcija koja provjerava je li stablo balansirano tako što gleda je li atribut „nivoi“ jednog od djece čvora „prethodni“ duplo veće od drugog. Ukoliko jeste, onda pozivamo DajSveCvorove(Cvor* p, vector<Tip> &v) da nam da sve čvorove koje trebamo balansirati, zatim pozovemo ObrisiSveCvorove(Cvor *p) da obrišemo sve te iste čvorove iz stabla. Zatim vektor „v“ koji sadrži sve čvorove za balansiranje sortiramo i pozovemo funkciju Balansiraj(vector<Tip> &v, int s, int k) koja doda čvorove u stablo tako da budu idealno balansirani.

void InOrderRek(Cvor* korijen, void (f)(Tip&)) – Funkcija koja prihvata pokazivač na čvor i pokazivač na funkciju, te posjećuje svaki čvor u stablu u poretku od manjeg ka većem i vrši neku radnju koju smo mi definisali u funkciji „f“ nad tim elementom.

void LevelOrderKorijen(Cvor* korijen,void (f)(Tip&) – Funkcija koja prihvata pokazivač na čvor i pokazivač na funkciju, te vrši šta god da joj definišemo u funkciji „f“ na svaki čvor u stablu ali krećući se kroz stablo nivo po nivo.

void LevelOrderKorijen2(Cvor* korijen) – Funkcija koja ispisuje atribut „nivoi“ za svaki čvor u stablo ali krećući se kroz stablo nivo po nivo.

void InOrder(void (f)(Tip&)) – Funkcija koja poziva void InOrderRek(Cvor* korijen, void (f)(Tip&)).

void LevelOrderKorijen2(Cvor* korijen) – Funkcija koja poziva void LevelOrderKorijen(Cvor* korijen,void (f)(Tip&)).

void LevelOrder(void (f)(Tip&) – Funkcija koja poziva void LevelOrderKorijen2(Cvor* korijen).