Elektrotehnički fakultet Sarajevo
Odsjek za računarstvo i informatiku
Predmet: Algoritmi i strukture podataka

Vježba 8.2 AVL Stablo

Rok za slanje je u informacionom sistemu zamger.

Zadatak 1.

Napravite novu vrstu mape pod nazivom AVLStabloMapa koja implementira AVL balansirano stablo opisano u predavanjima.

Klasu napravite tako što ćete kopirati klasu BinStabloMapa razvijenu u pripremi, a zatim napraviti minimalne prepravke kako bi ova klasa bila AVL stablo. Klasa AVLStabloMapa mora imati sve atribute kao i BinStabloMapa, struktura čvora treba biti ista osim što ćete dodati nove atribut pod nazivom balans i roditelj, sve metode za koje nema potrebe da se mijenjaju trebaju biti iste. Izuzetak od ovoga su samo popravke grešaka (loša implementacija koja za posljedicu ima neoptimalne performanse se također smatra greškom koja se može popravljati), pri čemu svaku popravku greške obavezno treba pratiti odgovarajući komentar.

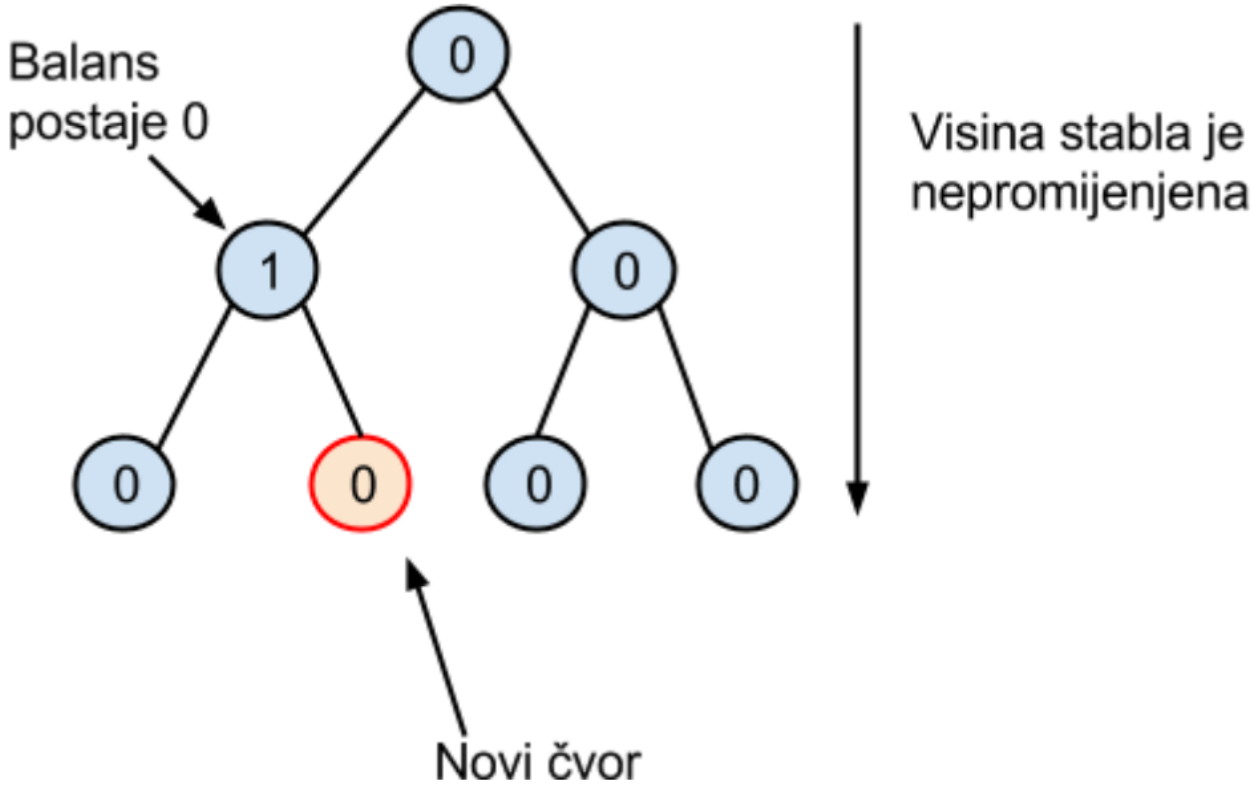
Napravite i main program u kojem demonstrirate razliku u performansama između BinStabloMape i AVLStabloMape.

Dodavanje novog čvora i brisanje čvora trebaju imati složenost O(log n). U predavanjima je data definicija balansa kao razlike u visini lijevog i desnog podstabla. Ako se ova definicija bukvalno prevede u C++ kod pozivajući neku pomoćnu funkciju “visina”, rezultat će biti da dodavanje i brisanje imaju složenost veću od O(log n). U slučaju da se u rekurzivnoj funkciji za ažuriranje balansa napiše nešto poput “balans = visina(c->lijevo) - visina(c->desno)” gdje je visina neka rekurzivna funkcija za izračunavanje visine podstabla nastaje problem jer funkcija visina ima složenost O(log n), pa ako se ona poziva u rekurzivnoj funkciji za ažuriranje balansa koja se izvršava log n puta, ukupna složenost funkcije za ažuriranje balansa bi bila O((log n) * (log n)).
Ovakvo rješenje naravno nije zadovoljavajuće jer se lako može desiti da u testovima sa ubacivanjem slučajno generisanih brojeva AVL stablo pokaže slabije performanse od nebalansiranog stabla. Moguće je implementirati AVL stablo tako da performanse dodavanja i brisanja čvora ostanu O(log n), ali uz konstantni faktor nešto veći nego kod običnog binarnog stabla. Da bi se ovakvo stablo napravilo potrebno je ispuniti određene uslove:

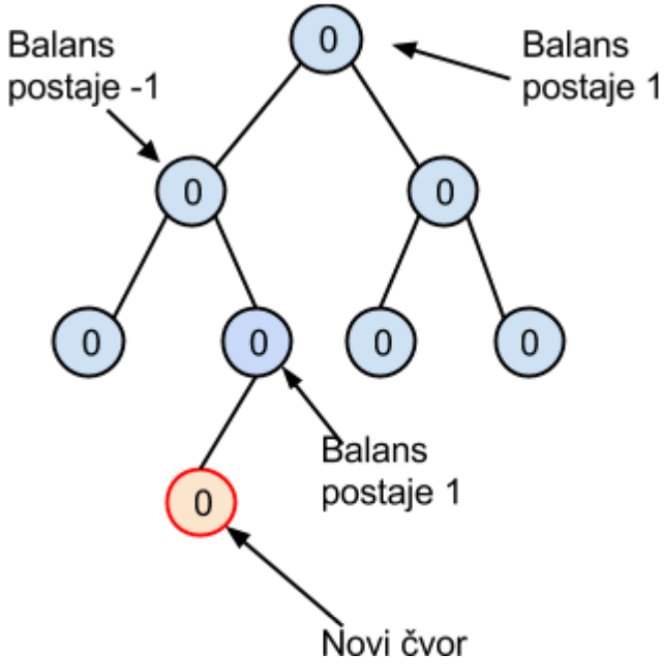
- Struktura Cvor mora sadržavati pokazivač na roditelja.
- Potrebno je u Cvor dodati i novi cjelobrojni atribut balans. Atribut balans se ažurira prilikom dodavanja novog čvora.

Algoritam je sljedeći:

Kada se dodaje novi čvor u stablo njegov balans je naravno 0. Najprije se odredi da li taj novi čvor ima brata/sestru. Brat/sestra (eng. sibling) je čvor sa istim roditeljem. Ako čvor ima brata, balans roditelja čvora koji je bio 1 ili -1 postaje nula. Balans ostalih čvorova u stablu ne mora se ažurirati jer ukupna visina stabla nije izmijenjena (vidjeti sliku - u čvorove su upisani njihovi balansi).



Ako čvor nema brata, poziva se rekurzivna funkcija azurirajBalans koja kao parametar prima dodani čvor:
1. Ako tekući čvor nema roditelja (u pitanju je korijen), rekurzija se prekida.
2. Ako je tekući čvor lijevo dijete svog roditelja, balans roditelja se povećava za jedan, u suprotnom se smanjuje za jedan.
3. Ako je prethodnom operacijom balans roditelja postao manji od -1 ili veći od 1, vrši se odgovarajuća rotacija nad čvorom roditeljem.
4. U suprotnom, funkcija se rekurzivno poziva pri čemu tekući čvor postaje roditelj.
Dodaćemo još dva čvora u prethodno stablo:



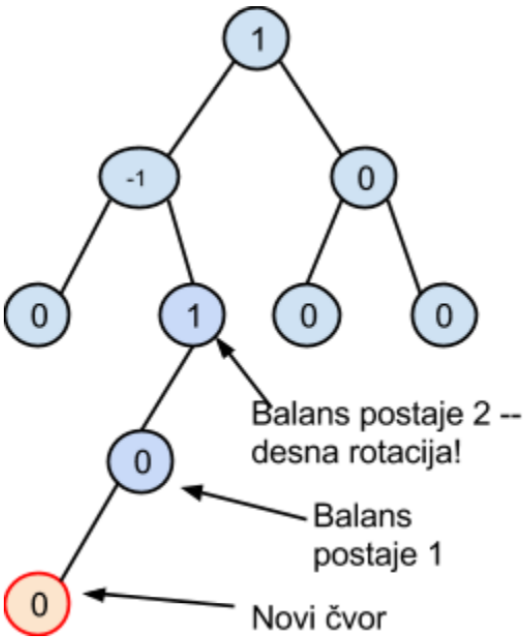


Tabela balansa ključnih čvorova nakon rotacije:

Prije Rotacije			Tip rotacije	Poslije rotacije		
Kriticni	Prvo dijete	Drugo dijete		Kriticni	Prvo dijete	Drugo dijete
2	1		Desno	0	0	
-2	-1		Lijevo	0	0	
2	-1	1	LijevoDesno	-1	0	0
2	-1	0	LijevoDesno	0	0	0
2	-1	-1	LijevoDesno	0	1	0
-2	1	1	DesnoLijevo	0	-1	0
-2	1	0	DesnoLijevo	0	0	0
-2	1	-1	DesnoLijevo	1	0	0