Strukture podataka i algoritmi

Tema 4

Radila: Ema Šahbegović

17.01.2024.g.

Uvod

U mom projektu je bilo potrebno da elementi stabla pored vrijednosti sadrže i prioritete, koji se slučajno generišu pri umetanju u elemenata u stablo. Svaka dva čvora su trebala zadovoljavati osobinu heapa za prioritete.
Operacije koje sam izvršila u ovom projektu su pretraga, umetanje, brisanje, razdvajanje i spajanje.

Struktura projekta:
Projekat se sastoji iz 3 fajla: main.cpp, stablo.cpp i stablo.h

Unutar main.cpp mi se nalazi logika za pozivanje funkcija koje sam implementirala u svom projektu.
U stablo.cpp fajlu mi se nalaze sve funkcije unutar projekta, a u stablo.h je definisana klasa Stablo i svi konstruktori, privatne i javne funkcije unutar te klase.

Funkcija pretraga(int kljuc):

Unutar funkcije pretraga sam proslijeđivala neku vrijednosti i rekurzivnim pozivima sam tražila proslijeđenu vrijednost. Početna funkcija se poziva iz korijena. Ukoliko je proslijeđeni korijen nullptr, vraćamo nullptr čime zaključujemo da tražene vrijednosti nema u stablu.
Ukoliko je korijen jednak traženoj vrijednosti, tu vrijednost vraćamo.
Ukoliko je korijen manji od tražene vrijednosti, rekuzivno pozivamo lijevo dijete našeg korijena te u njoj tražimo traženu vrijednost.
Ukoliko je korijen veći od tražene vrijednosti, rekuzivno pozivamo desno dijete našeg korijena te u njoj tražimo traženu vrijednost.

Funkcija ispisiSve():

Ova funkcija mi je služila kako bih ispisala sve ključeve i prioritete unutar čvoreva u stablu. Prvo počinjemo od korijena. Ukoliko je nullptr to znači da stablo nema članova. Ukoliko nije, ipisujemo dati čvor kao i njegovo lijevo i desno dijete ukoliko postoje.
Rekurzivno se pomijerano ulijevo i udesno dok ne ispišemo sve članove našeg stabla.

Funkcija ispisi():

Ovo je dodatna funkcija koju sam ja dodatno implementirala kako bih lakše pratila da je narušena osobina stabla po ključu. Ona ispisuje sve ključeve od manjeg do najvećeg. Prvo se ide što više u lijevo. Ukoliko je sljedeći pokazivač za dijete nullptr, tada ispisujemo vrijednost na kojoj se nalazimo. Kada ispišemo lijevo dijete, tada prelazimo na ispisivanje desne djece. Funkcija osigurava da se nakon ispisivanja lijevog podstabla ispišu svi čvorovi duž desnog podstabla.

Funkcija rotirajLijevo()

čuva pokazivač na desno dijete a isto tako i na lijevo dijete desnog djeteta.
Postavlja se lijevo dijete desnog djeteta na trenutni čvor.
Zatim se postavlja desno dijete trenutnog čvora na lijevo dijete desnog djeteta.
Na kraju se postavlja ptr na DesnoDijete.

Funkcija rotirajDesno()

čuva pokazivač na lijevo dijete te i na desno dijete lijevog djeteta
Postavlja se desno dijete lijevog djeteta na trenutni čvor
Postavlja se lijevo dijete trenutnog čvora na desno dijete lijevog djeteta
Na kraju se postavlja ptr na LijevoDijete.

Funkcija Insert()

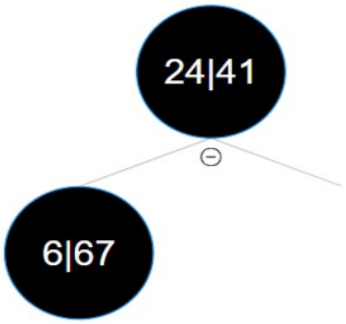
Unutar funkcije Insert proslijeđujemo funkciju InsertPrivate koja će proslijediti i korijen datog stabla. Ukoliko je trenutni pokazivač postavljen na nullptr, potrebno je napraviti novi ključ. U funkciji Razdvajanje će biti potrebno dodatno pojasniti zašto smo prosljeđivali uslov.
Dalje provjeravamo da li je proslijeđena vrijednost manja od trenutne vrijednosti. Ukoliko jeste. Potrebno je postaviti proslijeđenu vrijednosti kao lijevo dijete.
Dalje provjeravamo da li nam je narušena osobina heapa za prioritete. Ukoliko jeste, poziva se funkcija rotirajDesno.

Funkcija rotirajDesno radi dvije stvari:
Ukoliko DesnodijeteLijevogDjeteta je nullptr, ona će postaviti lijevo dijete da postane novi roditelj, a roditelj će postati desno dijete.
Ukoliko DesnoDijeteLijevogDjeteta nije nullptr, tada će se stvari malo drugačije odvijati. Pokazivač Desnog djeteta će sad pokazivati na roditelja, pa to znači da će roditelj postati lijevo dijete, dok će desno dijete postati roditelj.
Sada će bivši roditelj pokazivati na DesnoDijeteLijevogRoditelja, te to znači da će DesnodijeteLijevogaRoditelja ostati upravo desno dijete lijevog roditelja.

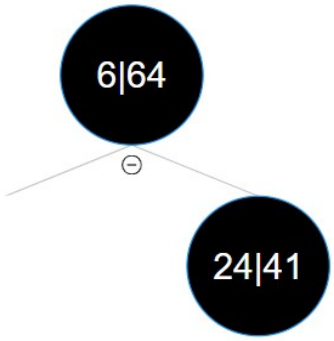
Analogno radi i funkcija rotirajLijevo().

U sljedećem primjeru ću objasniti logiku umetanja Čvorova u stablo:

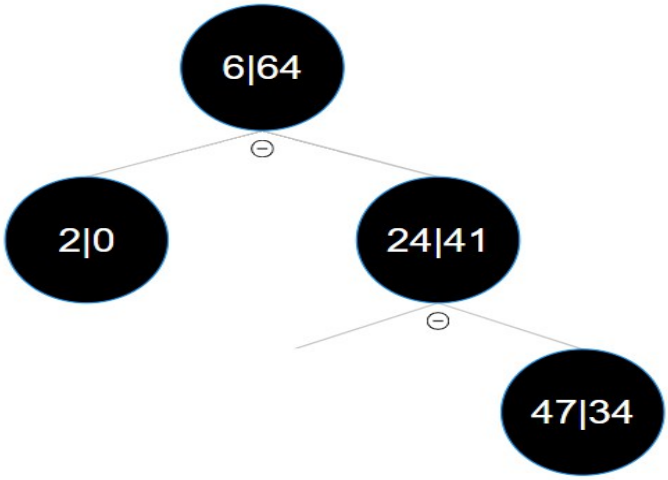
Prvo ubacujemo čvor čiji je ključ jednak 24, a prioritet 41(kraće ću pisati 24,41).On postaje korijen. Zatim unosimo čvor: 6,67. Pošto je 6 manje od 24 čvor 6,67 postaje lijevo dijete.



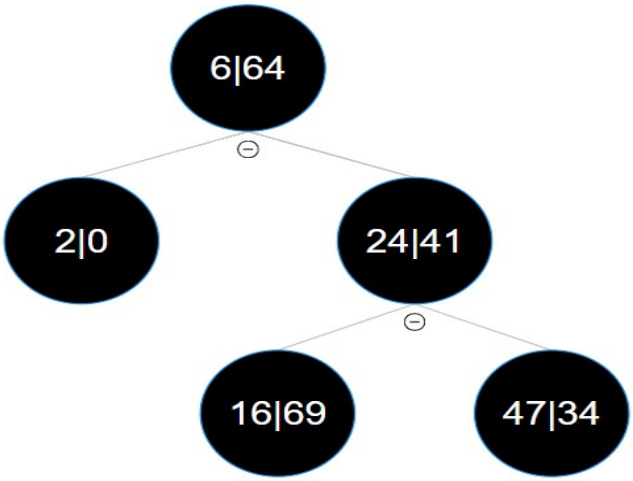
Sada se provjerava uslov da li je lijevo dijete po prioritetu veće od roditelja. Taj uslov je ispunjen pa zato pozivamo funkciju RotirajDesno().
Funkcija rotirajDesno() će postaviti čvor 6,67 na mjesto roditelja, a 24,41 na njegovo desno dijete.



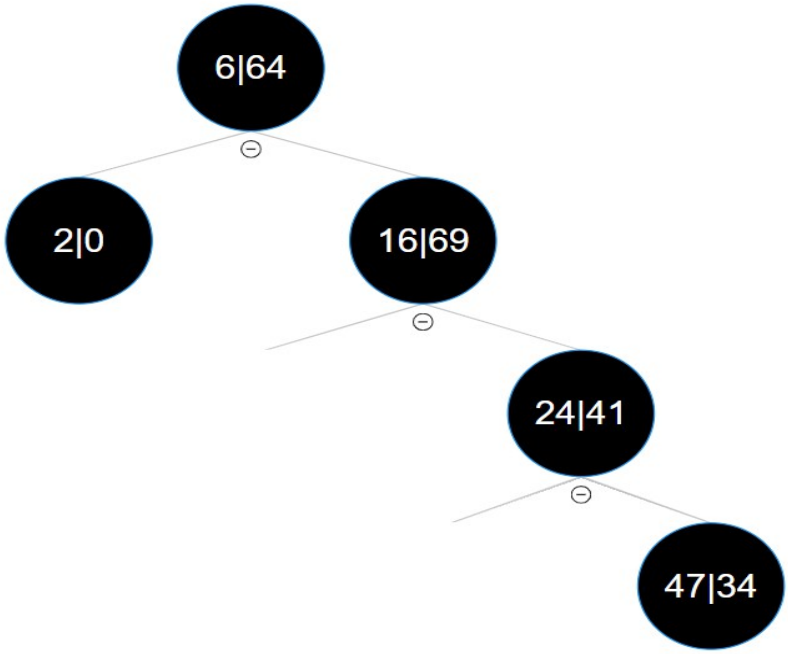
Dalje unosimo čvorove 2,0 i 47,34. One se postavljaju regularno u stablo po ključu.



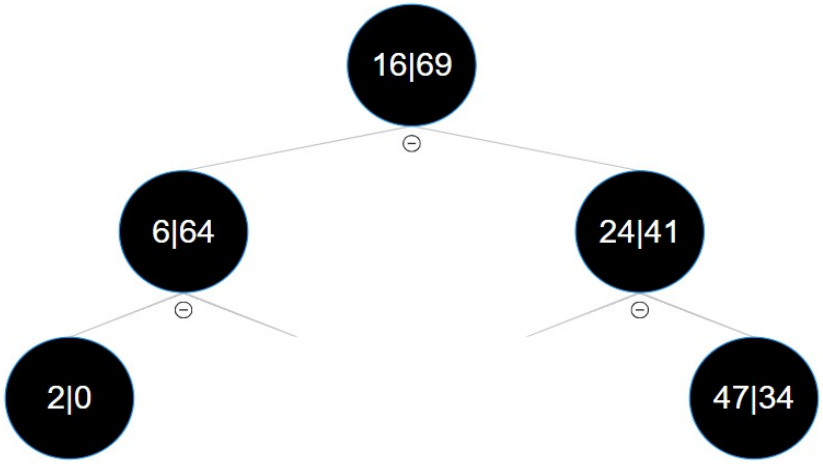
Unosimo u stablo čvor 16,69. On se prvenstveno postavlja kao lijevo dijete čvora 24,41.



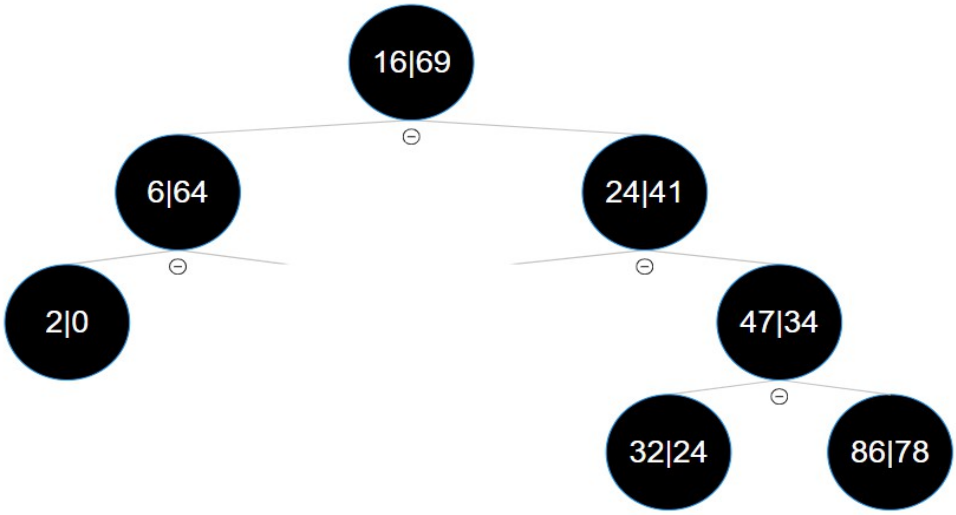
Uviđamo da je prioritet čvora 16,69 veći od 24,41, Oni se zamjenjuju tako što se poziva funkcija RotirajDesno() kojom će lijevo dijete 16,69 postati roditelj, a roditelj 24,41 desno dijete.



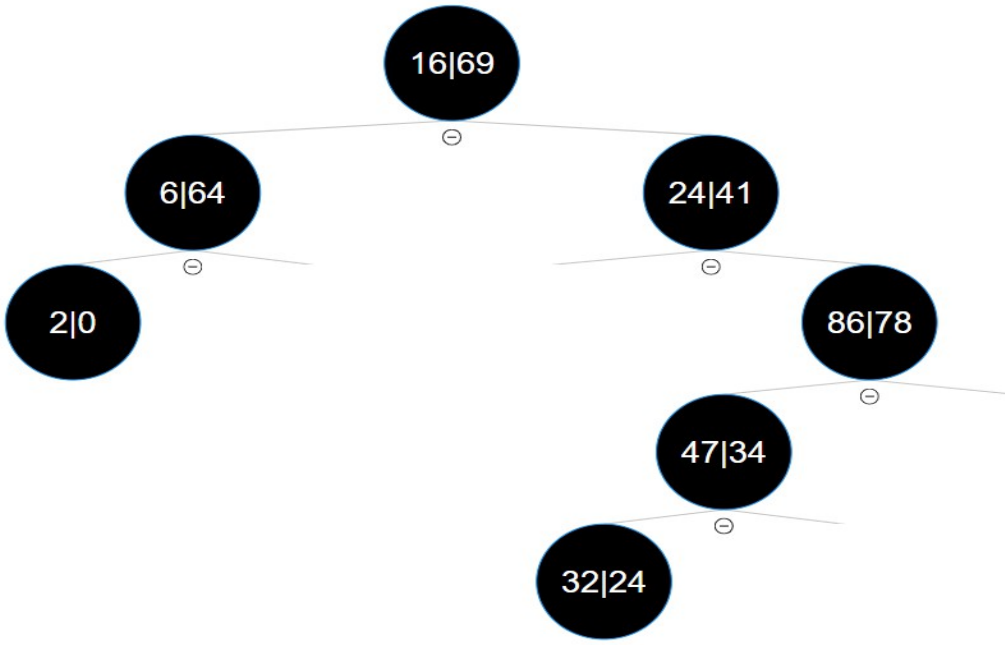
Još uvijek osobina prioriteta nije zadovoljena. 16,69 i 6,64 se rotiraju.



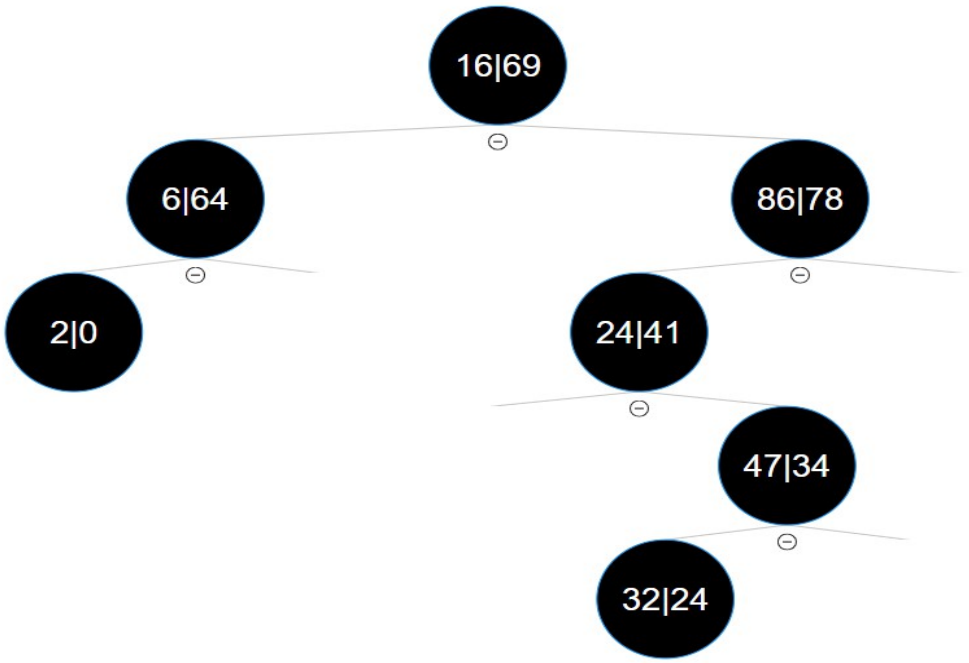
Regularno dodajemo čvor 32,24 a zazim i 86,78



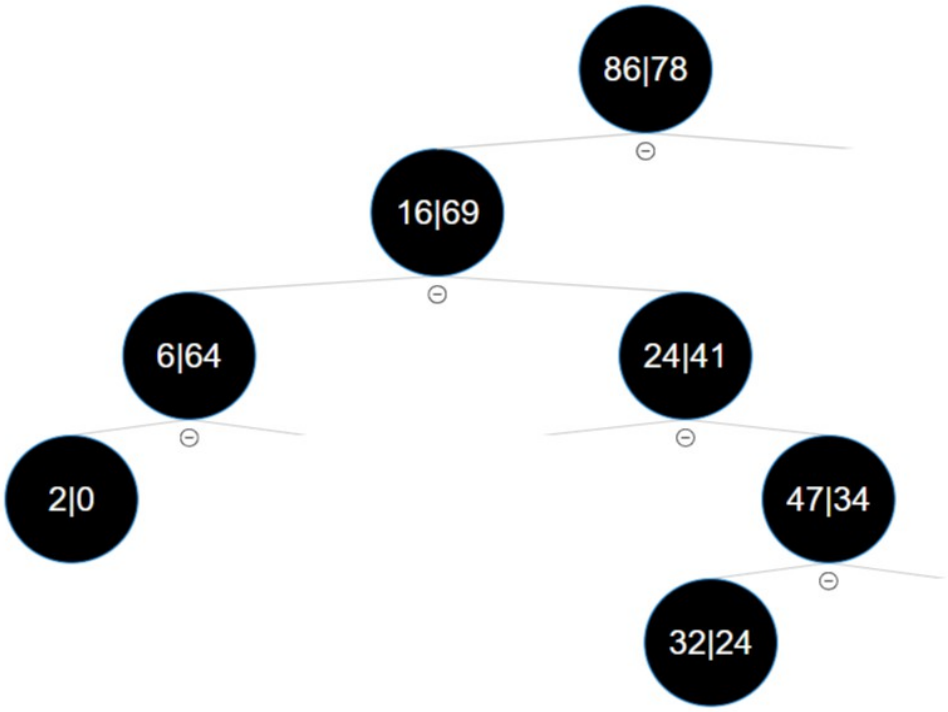
Čvor 87,78 pravi problem jer je njegov prioritet veći od prioriteta njegovog roditelja pa dolazi do rotacije. 86,78 postaje roditelj a roditelj njegovo lijevo dijete.



Još problem postoji. 24,41 i 86,78 se rotiraju. S obzirom da desno dijete ima lijevo dijete, dolazi do drugačije rotacije od prethodnih. Naime, sada će desno dijete postati roditelj, roditelj lijevo dijete, lijevo dijete desnog djeteta ostaje na toj poziciji.



Zadnja promjena se dešava kada se rotiraju 16,69 i 86,78.



Funkcija Erase(int kljuc)

Prvo provjeravam da li je korijen jednak nullptr, jer ako jeste, nemamo šta obrisati. Zatim u sljedećim linijama koda tražimo čvor koji želimo obrisati. Kada ga nađemo, razmatramo sljedeće slučajeve:
-Ako čvor ima desno, lijevo ili nijedno dijete, tada postavljamo pokazivač roditelja na njegovo dijete(ili na nullptr ukoliko nije imao djece)
-Ako čvor ima oba djeteta prvo provjeravamo prioritete lijevog i desnog djeteta. Ukoliko je prioritet lijevog djeteta manji od prioriteta desnog djeteta, rotiramo čvor ulijevo te rekuzivno pozivamo ErasePrivate.
Analogno vrijedi ukoliko je prioritet desnog djeteta manji od prioriteta lijevog djeteta.

Funkcija Razdvajanje(Stablo stablo, int kljuc):

Funkcija poziva da se ključ insertuje u naše stablo i on sa sobom prosljeđuje vrijednost bool. Prosljeđujemo vrijednost bool jer ne želimo da napravimo novi čvor koji će zadavati prioritet na neku random vrijednosti već mi želimo da prioritet bude veći od trenutnog korijena u našem stablu. Ukoliko je vrijednosti prioriteta trenutnog korijena 78, mi ćemo postaviti da je vrijednost prioriteta našeg čvora 79. On se insertuje u stablo. Zahvaljujući funkcijama RotirajLijevo i RotirajDesno mi ćemo rotirati naše čvorove dok ne bude zadovoljena osobina prioriteta za heapove. Kada ona bude zadovoljena, tada će naš insertovani Čvor postati korijen. Na osnovu tog korijena mi možemo razdvojiti stablo na dva podstabla: Lijevom podstablu će korijen biti lijevo dijete stabla, a desnom podstablu će korijen biti desno dijete stabla.

Funkcija Spajanje(Stablo stabloLijevo , Stablo stabloDesno)

Funkcija prvo poziva funkcije nadjiNajveceg i nadjiNajmanjeg koje traže u lijevom stablu najveći ključ a u desnom najmanji ključ. Postavljam ključ na osnovu arimetičke sredine. Dalje pravim stablo te insertam ključ u stablo, on postaje novi korijen. Na dati korijen preko pokazivača povezujem lijevo podstablo s korijenom, i desno podstablo s korijenom.

Unutar petlje, provjerava se da li postoji lijevo dijete i da li je prioritet lijevog djeteta veći od prioriteta korijena. Ako jeste, vrši se rotacija udesno. Slično, provjerava se i za desno dijete, te ako je prioritet desno veći od prioriteta korijena, vrši se rotacija korijena ulijevo. Petlja se prekida kada ni lijevo ni desno dijete nemaju veći prioritet od korijena.