UNIVERZITET U SARAJEVU
PRIRODNO - MATEMATIČKI FAKULTET
ODSJEK ZA MATEMATIČKE I KOMPJUTERSKE NAUKE

PROJEKAT IZ PREDMETA: “STRUKTURE PODATAKA I ALGORITMI”

Tema 3: Stablo (struktura podataka)

Predmetni profesor: prof. dr. Esmir Pilav
Predmetni asistent: Admir Beširević, MA

Studentica: Berina Spirjan

Sarajevo, januar 2024. godine

UNIVERZITET U SARAJEVU
PRIRODNO - MATEMATIČKI FAKULTET
ODSJEK ZA MATEMATIČKE I KOMPJUTERSKE NAUKE

OSNOVNE INFORMACIJE O PROJEKTU

Zadatak ovog projekta jeste da proširimo klasu **'Stablo'** koju smo obrađivali na vježbama, pri tome ćemo dodati neke nove funkcionalnosti što će nam omogućiti da unaprijedimo navedenu strukturu podataka.

Implementacija **Iteratora** i **Reverse Iteratora** će nam omogućit efikasno kretanje kroz stablo, a čijom implementacijom ćemo imati mogućnost prolaska kroz stablo koja će nalikovati onoj u strukturi podataka set. Pored toga, trebamo implementirati funkcije **Begin()** i **End()** koristeći dva fiktivna čvora, što će doprinijeti jednostavnom pristupu početka i kraja stabla. Ove funkcije će biti od ključnog značaja za navedene iteratore.

Drugi dio obuhvata prijateljsku funkciju koja provjerava da li je jedno stablo podskup nekog drugog. Ova funkcija potrebno je da radi u vremenu **O(m+n)**, gdje su m i n brojevi elemenata ta dva stabla.

Kao posljednju stavku imamo implementaciju indeksiranja, omogućavajući pristup elementima stabla prema veličini indeksa. Funkcija indeksiranja će raditi u vremenu **O(h)**, gdje nam h predstavlja visinu stabla.

Kroz ovaj projekat imat ćemo mogućnost stvaranja optimizovane strukture podataka koje omogućavaju kompleksne operacije nad stablima, istovremeno osiguravajući optimalnu vremensku složenost za različite svrhe upotrebe.

1. ORGANIZACIJA PROJEKTA

U ovom poglavlju naše dokumentacije obradit ćemo organizaciju projekta, raspored po klasama i strukturama, te public i private dijelove klase **Stablo**.

Implementaciju našeg projekta, sprovest ćemo pomoću tri fajla. Tačnije samu organizaciju projekta možemo sprovesti kroz dva foldera **Sources** i **Headers**. U folderu **Headers** kreiramo fajl pod nazivom **stablo.h** u kojem ćemo definisati sve klase i strukture, te prototipe metoda i funkcija. U sklopu foldera **Sources** nalaze se dva fajla main.cpp i **stablo.cpp**. U fajlu **stablo.cpp** sprovodimo programsku implementaciju, tj. tijela funkcija i metoda. A **main.cpp** fajl nam služi za testiranje implementacije programa. Također, dodat ćemo dva dodatna fajla **iterator**.h i **iterator**.cpp u koje ćemo odvojiti deklaraciju i implementaciju **Iterator**-a i **Reverse Iterator**-a, što će učiniti lakše snalaženje u samom projektu, budući da će Iterator i Reverse Iterator biti zasebne klase.

2. IMPLEMENTACIJA

U nastavku ćemo detaljno razmotriti ključne aspekte koji su objedinjeni kroz implementaciju određenih funkcija, iteratora, klasa i sl.

2.1. ITERATOR

Kao što smo već u prethodnom poglavlju spomenuli klasa Iterator je implementirana u fajlu iterator.cpp, dok je njena deklaracija uz neke osnovne konstruktore navedena u fajlu iterator.h. Klasa posjeduje jedan **privatni** član: **Cvor *trenutni** što predstavlja pokazivač na trenutni element u stablu.

Javni dio klase Iterator (public):
U ovom dijelu definišemo dio klase u kojem se nalaze konstruktori, operatori, destruktor, pomoćne funkcije i operatore. Također, u nastavku se nalazi pregled public dijela klase, a nešto kasnije ćemo razmotriti dolje navedeni dio, pojedinačno.

- **Iterator (Cvor *trenutni)**
- **Iterator (const Iterator &it)**
- **Iterator &operator=(const Iterator &it)**
- **Tip &operator***
- **Iterator operator++**
- **Iterator operator++(int)**
- **Iterator operator - -**
- **Iterator operator - - (int)**
- **bool operator ==(const Iterator &cvor)**
- **bool operator !=(const Iterator &cvor)**
- **Tip &operator*()**
- **Tip &operator->()**
- **Iterator najmanjiNaGrani()**
- **Iterator najveciNaGrani()**
- **Iterator pomjeriNaGrani(function<Cvor *(Cvor *)>)**
- **friend class Stablo<Tip>**

2.1.1. Iterator (Cvor *trenutni)

Konstruktor sa jednim parametrom koji kreira objekat klase **Iterator** koji predstavlja iterator na binarno stablo. Parametar **Cvor *trenutni** je čvor u stablu na koji iterator trenutno pokazuje, ako nije naveden tj. proslijeđen postavljamo ga na nullptr.

2.1.2. Iterator(const Iterator &it)

Konstuktor kopije koji kao parametar prihvata **it** - što predstavlja objekat klase **iterator** koji je potrebno kopirati. Ovaj konstuktor kreira novi objekat klase **Iterator** koji je identičan kao objekat koji se prosljeđuje kao argument u konstruktor (**it**).

2.1.3. Iterator &operator=(const Iterator &it)

Operator dodjele omogućava pridruživanje vrijednosti jednog Iteratora drugom Iteratoru. Ako je pozvan na istom objektu this==&it, neće doći do promjene, dok ukoliko nije pozvan nad istim objektom potrebno je da trenutni cvor modifikujemo tako sto mu prekopiramo vrijednost it.trenutni. Operator kao rezultat vraća referencu na trenutni iterator nakon izvršene dodjele.

2.1.4. Tip &operator*()

Ovo je operator dereferenciranja za klasu iterator. On nam omogućava da pristupimo vrijednosti elementa na koju trenutni Iterator pokazuje. Kao rezultat vraća referencu na element trenutnog čvora u stablu.

2.1.5. Iterator operator++()

Operator prefiksne inkrementacije za klasu Iterator (upotreba: ++it), zasniva se na ideji da pomjeramo iterator na sljedeći element u inorder obilasku stabla. Ovo radimo na način da provjeramo da li postoji desno dijete trenutnog čvora, ukoliko da pomjeramo iterator sve dok ne dođemo do najmanjeg elementa u desnom podstablu. Suprotno, ako ne postoji desno dijete, iterator pomjeramo prema roditelju dok ne dođemo do prvog lijevog djeteta. Kada stignemo do korijena znači da će nam iterator pokazivati na nullptr i tu se zaustavljamo. Kao rezultat ovaj operator vraća referencu na modifikovani iterator.

2.1.6. Iterator operator++(int)

Operator postfiksne inkrementacije za klasu Iterator (upotreba: it++) funkcioniše na način da u početku napravimo kopiju trenutnog iteratora, a zatim pomjeramo trenutni (originalni) iterator na sljedeći element u inorder obilasku stabla. Ako postoji desno dijete trenutnog čvora, iterator pomjeramo na najmanji element u desnom podstablu. Ako ne postoji desno dijete, iterator pomjeramo prema roditelju dok ne pronađemo prvo lijevo dijete. U trenutku kada dođemo do korijena, naš iterator će pokazivati na nullptr. Parametar koji prima ovaj operator se naziva dummy parametar, on nam služi samo kako bismo označili postfiksni operator inkrementiranja, ali ga u funkciji ne koristimo. Kao rezultat primjene operatora postfiksne inkrementacije vraćamo kopiju prvobitnog iteratora koja je sada modifikovana.

2.1.7. Iterator operator- -()

Operator prefiksnog dekrementiranja iteratora za klasu Iteratator. Ideja koju koristimo za implementaciju jeste da pomjeramo iterator na prethodni element u inorder poretku stabla. Ako za trenutni čvor postoji lijevo dijete, iterator pomjeramo na najveći element u lijevom podstablu. Suportno, ukoliko ne postoji lijevo dijete, iterator pomjeramo na roditelja sve dok ne dođemo do prvog desnog djeteta. Kada stignemo do korijena, iterator će pokazivati na nullptr. Kao rezultat vraćamo modifikovani iterator.

2.1.8. Iterator operator - - (int)

Operator postfiksnog dekrementiranja iteratora za klasu Iterator. Ovaj operator pravi u početku kopiju trenutnog iteratora, a zatim mijenja trenutni iterator. Ako postoji lijevo dijete trenutnog čvora, iterator pomjeramo na najveći element u lijevom podstablu. Ako ne postoji lijevo dijete, iterator pomjeramo na roditelja dok ne dođemo do prvog desnog djeteta. Ako dođemo do korijena u tom slučaju će iterator pokazivati na nullptr. Iz operatora vraćamo kopiju prvobitnog iteratora koja je sada modifikovana.

2.1.9. bool operator==(const Iterator &cvor) const

Operator jednakosti upoređuje trenutne čvorove dva iteratora. Kao parametar prihvata iterator koji nam sluzi za poređenje. Rezultat koji se vraća primjenom ovog operatora je bool varijabla true ili false u odnosu na to da li je trenutni cvor jednak cvor.trenutni koji je proslijeđen kao parametar unutar operatora.

2.1.10 bool operator !=(const Iterator &cvor) const

Operator nejednakosti upoređuje trenutne čvorove dva iteratora. Kao parametar prihvata iterator koji nam sluzi za poređenje. Rezultat koji se vraća primjenom ovog operatora je bool varijabla true ili false u odnosu na to da li je trenutni cvor različit u odnosuu na cvor.trenutni koji je proslijeđen kao parametar unutar operatora.

2.1.11. Tip &operator*() const

Operator omogućava čitanje i modifikaciju vrijednosti na koju trenutni iterator pokazuje. Vraća referencu na vrijednost elementa trenutnog čvora.

2.1.12. Tip &operator->() const

Operator omogućava pristup članovima objekta (u ovom slučaju, elementu čvora) na koji trenutni iterator pokazuje. Vraća pokazivač na vrijednost elementa trenutnog čvora.

2.1.13. Iterator pomjeriNaGrani(function<Cvor *(Cvor*)>)

Metoda je implementirana kako bismo izbjegli dupliciranje koda. Metoda pomjeriNaGrani kao parametar prima lambda funkciju koja vraća iterator na lijevo ili desno dijete trenutnog čvora, u zavisnosti od potrebe za traženjem najmanjeg ili najvećeg elementa. Iterator se pomjera na najmanji ili najveći element u podstablu tako što se kontinuirano pomjera na lijevo ili desno dijete čvora, dokle god takvo dijete postoji. Kao rezultat poziva metode pomjeriNaGrani dobijamo iterator koji pokazuje na najmanji ili najveći element u podstablu trenutnog čvora.

2.1.14. Iterator najmanjiNaGrani()

Metoda **najmanjiNaGrani** pomjera iterator na najmanji element u podstablu trenutnog čvora. Ova metoda unutar svoje definicije poziva drugu funkciju **pomjeriNaGrani** i proslijeđuje joj lambda funkciju, što kao rezultat vraća iterator na lijevo dijete trenutnog čvora. Iterator se pomjera na najmanji element u podstablu tako što se kontinuirano pomjera na lijevo dijete dok god ono postoji. Kao rezultat metoda vraća iterator na najmanji element u podstablu trenutnog čvora.

2.1.15. Iterator najvećiNaGrani()

Metoda najvećiNaGrani pomjera iterator na najveći element u podstablu trenutnog čvora. Ova metoda unutar svoje definicije poziva drugu funkciju **pomjeriNaGrani** i proslijeđuje joj lambda funkciju, što kao rezultat vraća iterator na desno dijete trenutnog čvora. Iterator se

pomjera na najveći element u podstablu tako što se kontinuirano pomjera na desno dijete dok god ono postoji. Kao rezultat metoda vraća iterator na najveći element u podstablu trenutnog čvora.

2.1.16. friend class Stablo<Tip>

Ova deklaracija omogućava pristup privatnim članovima klase Stablo iz klase Iterator.

2.1. REVERSE ITERATOR

Sve metode, konstruktori, funkcije i metode imaju dosta slične funkcionalnosti kao u klasi Iterator, jedina razlika jeste što idemo od većih prema manjim elementima.

2.3. PROVJERA DA LI JE JEDNO STABLO PODSKUP DRUGOG

U implementaciju klase Stablo dodali smo sljedeće funkcije:

- **Cvor *dajNajveciCvor(Cvor *)**
- **Cvor *dajNajmanjiCvor(Cvor *)**
- **Cvor *Kraj()**
- **Cvor *Korijen()**
- **friend bool provjeriDaLiJePodstablo(Stablo<Tip2> &s1, Stablo<Tip2> &s2)**

2.3.1. Cvor *dajNajveciCvor(Cvor *)

Metoda dajNajveciCvor traži najveći čvor u stablu sa trenutnim čvorom kao početkom. Implementacija metode rekurzivno pretražuje desno podstablo sve dok ne dođe do najvećeg čvora. Prima jedan parametar koji je tipa Čvor i predstavlja pokazivač na trenutni čvor što označava početak pretrage. Kao rezultat ova metode vraća pokazivač na najveći čvor u stablu ili nullptr u slučaju kada je stablo prazno.

2.3.2. Cvor *dajNajmanjiCvor(Cvor *)

Metoda dajNajmanjiCvor traži najmanji čvor u stablu sa trenutnim čvorom kao početkom. Implementacija funkcije rekurzivno pretražuje lijevo podstablo sve dok ne dođe do najmanjeg čvora. Prima jedan parametar koji je tipa Čvor i predstavlja pokazivač na trenutni čvor što označava početak pretrage. Kao rezultat ova metode vraća pokazivač na najmanji čvor u stablu ili nullptr u slučaju kada je stablo prazno.

2.3.3. Cvor *Kraj()

Rezultat poziva ove metode je pokazivač na najveći čvor u stablu ili ukoliko stablo je prazno nullptr. Koristi pomoćnu metodu **dajNajveciCvor** za pronalazak najvećeg čvora.

2.3.4. Cvor *Korijen()

Vraća pokazivač na korijen stabla koji je privatni atribut u klasi Stablo.

2.3.5. friend bool provjeriDaLiJePostablo(Stablo<Tip2>s1, Stablo<Tip2>s2)

Prijateljska funkcija koja vrši provjeru da li je s2 podstabo prvog stabla s1. Vrijeme izvršavanja ove funkcije je O(m + n), gdje m predstavlja broj elemenata stabla s1, a n broj elemenata stabla s2. Funkcija prolazi kroz oba stabla i poredi elemente. Ako su svi elementi stabla s2 sadržani u stablu s1, rezultat koji se vraća je bool vrijednost true. Inače, ukoliko nisu sadržani funkcija vraća false. Logika koju koristimo u ovoj funkciji jeste da vršimo na samom početku provjeru da li je korijen stabla s1 ili s2 jednak nullptr što bi značilo da su stabla prazna i u tom slučaju stablo s2 ne može biti postablo stabla s1 sigurno, pa vraćamo false. U suprotnom kreiramo vektor elementi2 u koji ćemo pohraniti vrijednosti elemenata od čvorova iz stabla s2. Prolazimo kroz drugo stablo s2 i dodajemo njegove elemente. Kada završimo sa dodavanjem elemenata iz stabla s2, potrebno je da čvor koji smo kreirali prethodno kako bismo prošli kroz drugo stablo, sada postavimo na najmanji čvor za prvo stablo. Bitno je da pratimo i poziciju elementa u vektoru drugog stabla. Nakon toga prolazimo kroz oba stabla i vršimo poređenje. Kada smo prošli kroz oba stabla i ukoliko funkcija nije vratila vrijednost false u slučaju kada je element stabla 2 manji od elementa stabla 1, iz funkcije vraćamo true vrijednost jer to bi značilo da stablo s2 jeste podstablo stabla s1.

2.4. INDEKSIRANJE U STABLU

Za implementaciju indeksiranja u klasi Stablo koristimo jednu funkciju i jednu metodu:

- Cvor *dajKtiElement(int pozicija, Cvor *trenutniCvor, int &brojac)
- Tip operator[](int)

2.4.1. Cvor *dajKtiElement(int pozicija, Cvor *trenutniCvor, int &brojac)

Ova metoda traži k-ti najveći element u stablu. Prima tri parametra: pozicija - tražena pozicija elementa, trenutniCvor - trenutni čvor koji posmatramo, i brojac - brojač koji prati

trenutni redni broj najvećeg elementa. Rekurzivno pretražujemo lijevo podstablo sve dok ne pronađemo k-ti najmanji element ili dok ne dođemo do kraja stabla. Ako pronađemo k-ti najmanji element u desnom stablu„ vraćamo odgovarajući čvor, u suprotnom vraćamo nullptr.

2.4.2 Tip operator[](int)

Ova funkcija omogućava pristup elementu na određenoj poziciji u stablu. Prima jedan parametar - **pozicija** - tražena pozicija elementa. Koristi funkciju **dajKtiElement** kako bi pronašla k-ti najveći element u stablu. Ako traženi element ne postoji (tj. ako je k veće od broja elemenata u stablu), bacamo izuzetak **std::out_of_range**. Ova metoda omogućava indeksiranje stabla kao niz, pri čemu je indeksiranje bazirano na redoslijedu elemenata u inorder (uzlaznom) poretku. Vrijeme izvršavanja je **O(h)**, gdje je **h** visina stabla.