Fakultet elektrotehnike i računarstva

Zavod za primjenjeno računarstvo

**Napredni algoritmi i strukture podataka**

1. laboratorijska vježba

Valerio Franković 0036471547

Zagreb, 4.11.2016.

# Zadatak

*[Ovdje navesti skupinu zadatka i kopirati tekst odabranog zadatka]*

**Zadaci za 17 bodova**

1. Napisati program koji učitava niz prirodnih brojeva iz ASCII datoteke (po pretpostavci, datoteka nije prazna) i upisuje ih u (inicijalno prazno) AVL stablo istim redoslijedom kao u datoteci. Program može biti konzolni ili s grafičkim sučeljem, po vlastitom izboru. Konzolni program naziv ulazne datoteke treba primiti prilikom pokretanja kao (jedini) argument s komandne linije, a grafički iz odgovarajućeg sučelja po pokretanju programa. Nakon upisa svih podataka, ispisati izgrađeno stablo na standardni izlaz (monitor). Program zatim treba omogućiti dodavanje novih čvorova te nakon svake promjene treba ponovo ispisati stablo.

Isto kao zadatci za 11 bodova, samo treba dodati i brisanje čvora iz stabla.

# Rješenje zadatka

**[*Ovdje ne pisati ništa, početi od potpoglavlja 2.1.*]**

*[****Opće upute:***

*Preporučuje se upotrijebiti* ***font Arial 12 pt crne boje****, osim ako se nešto želi posebno istaknuti kada se preporučuje korištenje podebljanih slova ili neke druge boje.*

*Rješenje zadatka sastoji se od jedne ili više cjelina (funkcionalnosti) koje u zadatku treba ostvariti (ovisno o vrsti i težini zadatka). Svaku funkcionalnost opisati u zasebnom potpoglavlju.*

*Potrebno je* ***ukratko navesti teorijsku podlogu*** *na kojoj se temelji rješenje zadatka te* ***način implementacije*** *rješenja bez ulaženja u detalje implementacije (ne prepisivati tijela funkcija).*

*Tekst rješenja po potrebi* ***strukturirati po listama i natuknicama*** *(npr. za korake nekog algoritma i sl) kako bi dokumentacija bila što preglednija.*

*Ukoliko se ideja rješenja može prikazati dijagramom preporuča se ubaciti i dijagram(e), ali nije neophodno. UML dijagram klasa može biti dobra osnova za objašnjenje implementacije.]*

## Teorijski uvod

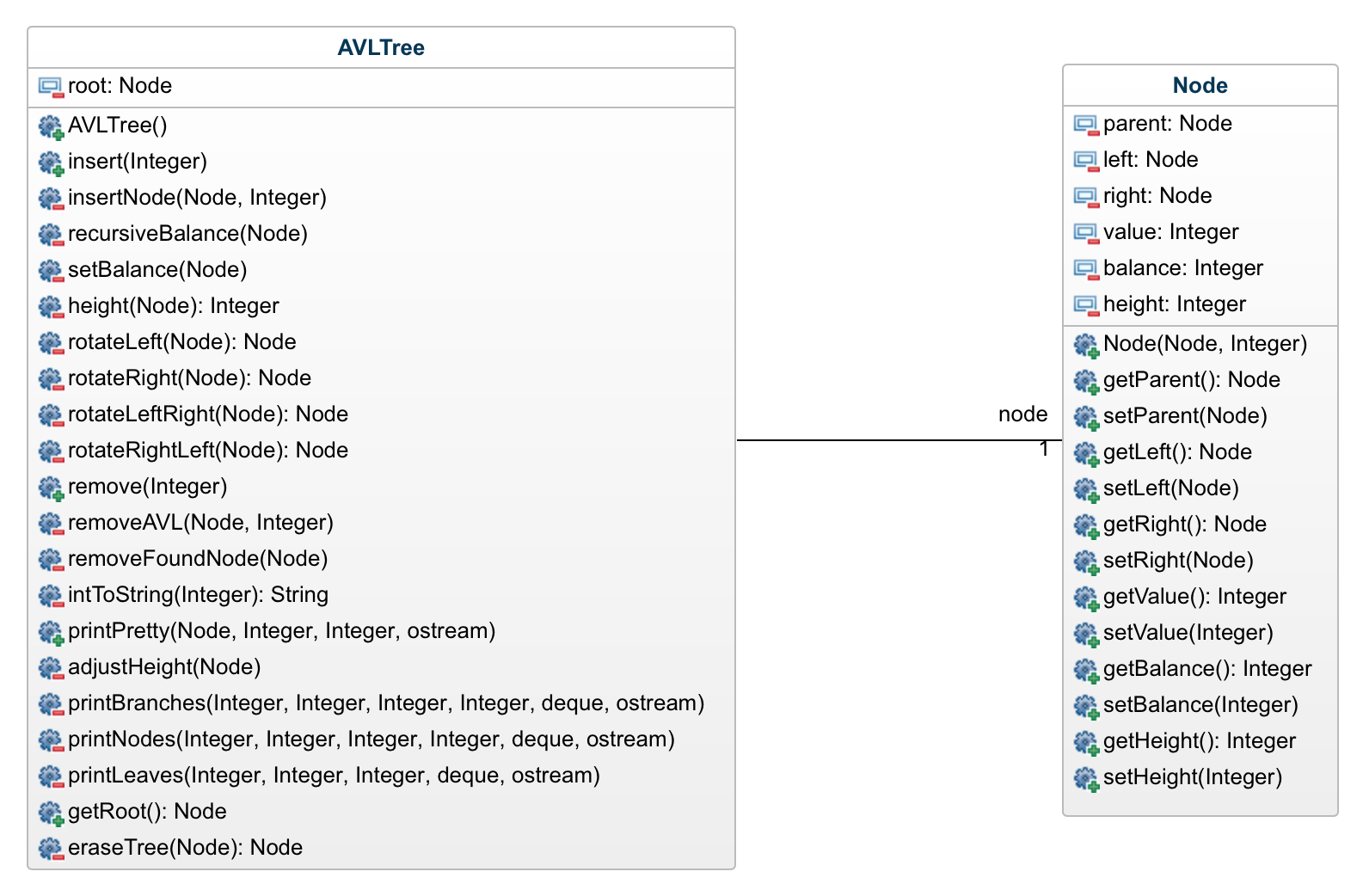
*[U nekoliko rečenica navesti ključne teorijske značajke na kojima se temelji* ***problem i cjelokupno rješenje****. Rješenja pojedinih funkcionalnosti ne pisati ovdje. Ukoliko rješenje implementira neku matematičku zakonitost, istaknite matematičku formulu.]*

AVL stabla su binarna stabla koja se koriste radi bržeg uravnotežavanja stabla. Odnosno, za razliku od algoritama koji stablo uravnotežavaju globalno, AVL to radi lokalno. AVL stabla moraju zadovoljavati AVL definicijsko pravilo prema kojem apsolutna razlika desnog i lijevog podstabla mora biti najviše jedan. U drugim slučajevima, stablo se mora uravnotežiti koristeći lijeve, desne te dvostruke rotacije. Dodavanje novog čvora se izvodi tako da ga se najprije doda kao u binarnom stablu zatim ga se rekurzivno, od roditelja čvora do korijena, uravnoteži, a brisanje se izvodi tako da prije uklonimo čvor (*deletion by copying*) pa ga rekurzivno uravnotežimo. U praksi AVL stablo se koristi u implementaciji *TreeMap*-a u paketu *java.util* za OS Android. Pretraga stabla u najgorem slučaju zahtijeva O(log2 *n*) usporedbi.

## Implementacija

*[Navesti, ukratko, osnovne informacije o implementaciji. (programski jezik, programska podrška/programske knjižnice, ...)]*

AVL stablo je isprogramirano u programskoj jeziku C++ koristeći Eclipse CDT. Korištena su dva razreda – *Node* te *AVLTree* i naravno, metoda *main*. Razred *Node* je programska apstrakcija čvora. Sadrži pokazivače na roditelja, na lijevo i desno podstablo te vrijednosti podatka, visinu i faktor ravnoteže. Razred *AVLTree* je zamišljen kao apstrakcija AVL stabla koji sadrži funkcionalnosti poput dodavanja nove vrijednosti ili brisanje već postojeće vrijednosti, a čuva pokazivač na korijen. Sadrži niz funkcija za ispisivanje, a implementiran je i destruktor koji briše sve čvorove prilikom izlaska iz programa. Programske knjižnice koje su upotrijebljene radi implementacije su <*string*>, <*iomanip*>, <*deque*>, <*fstream*>, <*iostream*>, <*sstream*>, <*cmath*>, <*streambuf*>, <*vector*>, <*unistd.h*> te <*exception*>.



*Slika 2.1 - UML dijagram razreda*

### Dodavanje nove vrijednosti

*[Početi opisom implementacije u kojem povremeno treba napraviti osvrt na teorijska načela koja rješenje implementira, a nisu spomenuta u uvodnom dijelu. Svakako se preporuča uvrštavanje kratkog pseudokoda kojim se implementira rješenje.*

***Napomena:*** *nazive klasa te pripadnih atributa i funkcija istaknuti u tekstu (preporuča se korištenje ugrađenog stila „Coding1“). Pseudokod također formatirati posebnim stilom i uvući u odnosu na ostatak teksta. Primjer je dan u nastavku. ]*

Dodavanje se izvodi tako da se pozove javna metoda *insert* koja kao argument uzima cjelobrojnu vrijednost. Metoda provjerava je li korijen postojan – ukoliko nije, jednostavno napravi novi čvor (razred *Node*) te ga pridijeli korijenu. Ukoliko korijen postoji, poziva se privatna metoda *insertNode* koja za argumente uzima pokazivač na razred *Node* te cjelobrojnu vrijednost. Medota *insertNode* rekurzivno prolazi kroz stablo od korijena prema listovima te dodaje novi čvor kao u slučaju binarnog stabla, samo što nakon dodavanja poziva pomoćnu metodu *recursiveBalance* koja lokalno rekurzivno uravnotežuje stablo. Pseudokod je dan u nastavku:

**AVLTree::insert(value):**

if root

insertNode(root, value)

end if

else

napravi novi čvor i pridijeli ga korijenu

end else

**AVLTree::insertNode(node, value):**

if node->value > value

if !node->left

napravi novi čvor i pridijeli ga lijevom djetetu

rekurzivno izbalansiraj stablo

end if

else

insertNode(node->left, value)

end else

else if node->value < value

if !node->right

napravi novi čvor i pridijeli ga desnom djetetu

rekurzivno izbalansiraj stablo

end if

else

insertNode(node->right, value)

end else

end else if

### Uravnoteživanje stabla

Uravnoteživanje stabla se izvodi tako da se prvo čvoru dodijeli faktor ravnoteže što služi kao odluka treba li se izvesti rotacija. U slučaju da se rotacija izvodi, ona koja mora biti izvedena odluči se prema stablu odlučivanja. Odnosno, ako je faktor ravnoteže roditelja -2, a faktor ravnoteže njegovog lijevog djeteta -1 ili 0, izvodi se desna rotacija. U slučaju da je faktor ravnoteže njegovog lijevog djeteta 1, izvodi se dvostruka rotacija i to lijeva pa desna. Kad je faktor ravnoteže roditelja +2, a faktor ravnoteže njegovog desnog djeteta 0 ili 1, izvodi se lijeva rotacija. No, ako je taj faktor ravnoteže djeteta iznosi -1, opet se izvodi dvostruka rotacija, no ovaj put desna pa lijeva. U slučaju da roditelj ima roditelja, metoda se rekurzivno poziva te joj se za parametar daje pokazivač na roditelja roditelja. Pseudokod metode je dan u nastavku, a vrijedno je spomenuti da je metoda privatna, tako da joj se ne može pristupiti izvan razreda.

**AVLTree::recursiveBalance(node):**

postavi čvoru faktor ravnoteže

if node->balance == -2

if node->left->balance == -1 or node->left->balance == 0

izvedi desnu rotaciju

end if

else

izvedi dvostruku rotaciju (lijevu pa desnu)

end else

end if

else if node->balance == 2

if node->left->balance == 0 or node->left->balance == 1

izvedi lijevu rotaciju

end if

else

izvedi dvostruku rotaciju (desnu pa lijevu)

end else

end else if

if node->parent

recursiveBalance(node->parent)

end if

else

root = node

end else

### Rotacije

U ovom radu implementirane su četiri vrste rotacija – lijeva, desna, dvostruka lijeva desna, dvostruka desna lijeva. Lijeva rotacija se izvodi tako da se roditelja zamijeni s njegovim desnim djetetom pri čemu se pokazivač na desno dijete tog bivšeg roditelja zamijeni pokazivačem na lijevo dijete novopečenog roditelja. Desna rotacija je simetrična lijevoj, odnosno, izvodi se tako da se roditelja zamijeni s njegovim lijevim djetetom pri čemu se pokazivač na lijevo dijete tog bivšeg roditelja zamijeni pokazivačem na desno dijete novopečenog roditelja. Dvostruka lijeva desna rotacija se izvodi tako da se na lijevom djetetu provede lijeva rotacija pa na roditelju desna rotacija, a dvostruka desna lijeva rotacija tako da se na desnom djetetu provede desna rotacija pa na roditelju lijeva rotacija. Pseudokod rotacija je dan u nastavku, a kao i metoda rekurzivnom uravnotežavanja, ona je privatna te joj se ne može pristupiti van razreda.

**AVLTree::rotateLeft(node) Node:**

result = node->right

result->parent = node->parent

node->right = result->left

if (node->right)

node->right->parent = node

end if

result->left = node

node->parent = result

if result->parent

if result->parent->right == node

result->parent->right = result

end if

else if result->parent->left == node

result->parent->left = node

end else if

end if

podesi težine lijevog djeteta node-a, node-a te resulta

podesi faktore ravnoteže node-a i result-a

return result

**AVLTree::rotateRight(node) node:**

result = node->left

result->parent = node->parent

node->left = result->right

if (node->left)

node->left->parent = node

end if

result->right = node

node->parent = result

if result->parent

if result->parent->right == node

result->parent->right = result

end if

else if result->parent->left == node

result->parent->left = node

end else if

end if

podesi težine lijevog djeteta node-a, node-a te resulta

podesi faktore ravnoteže node-a i result-a

return result

**AVLTree::rotateLeftRight(node) node:**

node->left = rotateLeft(node->left)

result = rotateRight(node)

return result

**AVLTree::rotateLeftRight(node) node:**

node->right = rotateRight(node->right)

result = rotateLeft(node)

return result

### Brisanje vrijednosti

Brisanje vrijednosti se provodi tako da se javnoj metodi *remove* kao parametar da cjelobrojna vrijednost koja se želi izbrisati. U toj metodi se pozove pomoćna privatna metoda *removeAVL* koje za parametre uzima korijen te spomenutu vrijednost, a zadatak metode je binarnim pretraživanjem pronaći traženi čvor koji ju sadrži. Taj čvor se dalje prosljeđuje u još jednu pomoćnu privatnu metodu *removeFoundNone* čiji je zadatak da taj čvor i izbriše. Metoda briše metodu tako da uradi *deletion by copying* nakon čega jednostavno ponovno rekurzivno izbalansira stablo. Pseudokod tih metoda je dan u nastavku:

**AVLTree::remove (int):**

removeAVL(node, int)

**AVLTree::renoveAVL(node, int):**

if (!node)

return

end if

else

if node->value > value

removeAVL(node->left, value)

end if

else if node->value < value

removeAVL(node->right, value)

end else if

else

removeFoundNode(node)

end else

end else

**AVLTree::removeFoundNode(node):**

if node->left

temp = node->left

while temp->right

temp = temp->right

end while

if node->left == temp

temp->parent->left = temp->left

end if

else

temp->parent->right = temp->left

end else

node->value = temp->value

rekurzivno uravnoteži roditelja od temp

end if

else if

temp = node->right

while temp->left

temp = temp->left

end while

if node->right == temp

temp->parent->right = temp->right

end if

else

temp->parent->left = temp->right

end else

node->value = temp->value

rekurzivno uravnoteži roditelja od temp

end else if

else

if node == root

root = NULL

end if

else

rebalancing = node->parent

if node->parent->right == node

node->parent->right = NULL

end if

else

node->parent->left = NULL

end else

rekurzivno uravnoteži čvor rebalancing

end else

### Ispis stabla

Algoritam ispisa stabla izračuna visinu stabla, duljinu grane za svaki čvor za svaku razinu, udaljenost između čvora desnog djeteta lijevog susjeda i lijevog čvora desnog susjeda te početan razmak ispisa prvog čvora te razine. Stvori se red te mu se doda korijen. Zatim se za svaku razinu ispišu grane, čvorovi te se izbacuju i izbacuju čvorovi iz reda te se preračunavaju gore spomenute veličine. Na kraju se za listove pozovu metode ispisa grana i listova. Pseudokod je dan u nastavku:

**AVLTree::printPretty(node, level, indentSpace, ostream):**

izračunaj maksimalnu visinu stabla

izračunaj duljinu grane za čvor

izračunaj udaljenost između čvora desnog djeteta lijevog susjeda i lijevog djeteta desnog susjeda

izračunaj početni razmak ispisa prvog čvora

dodaj korijen u red

i = 0

repeat

ispiši grane

izračunaj duljinu grane za čvor

izračunaj udaljenost između čvora desnog djeteta lijevog susjeda i lijevog djeteta desnog susjeda

izračunaj početni razmak ispisa prvog čvora

ispiši čvorove

repeat

izbaci prvog u redu

if prvi u redu postoji

dodaj u red njegovo lijevo i desno dijete

end if

else

dodaj u red pokazivače na NULL

end else

until postoji čvorova u trenutnoj razini

izračunaj broj čvorova u trenutnoj razini

until i != maksimalnoj visini stabla

ispiši grane

ispiši listove

### Naknadno dodavanje nove ili brisanje postojeće vrijednosti

Naknadno dodavanje node ili brisanje postojeće vrijednosti odvija se u *main* funkciji. Korisnik može izabrati između tri opcije. *add* za dodavanje nove vrijednosti, *remove* za brisanje već postojeće vrijednosti, a *exit* za izlaz iz programa. Program se vrti u petlji dokle god korisnik ne odluči izaći iz programa. Pseudokod odsječka funkcije dan je u nastavku.

**Odsječak koda za naknadni upis ili brisanje vrijednosti:**

repeat

ispis uputa na ekranu

dohvati korisnikovu naredbu

if korisnik upisao *add*

ispis uputa na ekranu

dohvati vrijednost

dodaj vrijednost u stablo

end if

else if korisnik upisao *remove*

ispis uputa na ekranu

dohvati vrijednost

izbriši vrijednost iz stabla

end else if

else if korisnik upisa *exit*

izađi iz programa

end else if

else

ispiši da je korisnik upisao krivu naredbu

end else

until

# Zaključak

*[Budite kratki i navedite najvažniji zaključak vašeg rada, opišite svoje viđenje cjelokupne tematike. Uključuje razloge za i protiv, moguća poboljšanja, daljnji rad na temi, i sl. Zaključak mora biti dovoljno kratak i jasan da se daje uvid u mišljenje autora. U zaključku ne upotrebljavajte formule i slike.]*

AVL stabla su zanimljiva u slučaju kad je broj elemenata dovoljno velik (recimo veći od 50), kad je omjer pretrage i unošenja (pretraga/unošenje) dovoljno velik (recimo veći od 5) da bi se opravdalo cijenu ponovnog uravnotežavanja te kad se vrijednosti ne pojavljuju slučajno. Prednosti AVL stabala su da je stablo uvijek uravnoteženo čime se garantira O(log *n*) brzina binarnog pretraživanja dok su nedostatci komplicirane rotacije koje se koriste pri unošenju i brisanju kako bi algoritam očuvao svoju uravnoteženost. Alternativa AVL stablima su crveno-crna stabla.

# Literatura

*[Literatura se navodi redom kojim se citira, a zatim ostala literatura. U tekstu za pozivanje na literature koristite zagradu [1]. Tablica 1 prikazuje osnovna pravila za pisanje pojedinih tipova literature.]*

[1] N. Hlupić, D. Kalpić. 2009. Napredne strukture podataka (*pdf*). Dostupno na: <http://www.fer.unizg.hr/_download/repository/Strukture_podataka.pdf>

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algotihms*. Cambridge, MA: MIT Press, 2009, pp. 333.

[3] A. Drozdek, *Dana Structures and Algorithms in C++*. Thomson Course Technology 2005, pp. 256-261.