Matematički fakultet Univerziteta u Beogradu

Mentor:

Prof. dr Miorag Živković

R-stabla i primene

Master rad iz predmeta algoritmi i strukture podataka

Emilija Gregorić 1037/2012

Beograd, 2013

Sadržaj

3

1.Uvod

Comer [1] je 1979 godine uveo termin „sveprisutna B-stabla“. S obzirom na to, da se danas u većini literature koje se bave tematikom baza podataka, može pronaći značajan broj stranica posvećen B-stablima, može se zaključiti da je ovaj izraz koji deluje pretenciozno ipak tačan. Ipak, pored svih svojih pozitivnih strana, zbog kojih su tako široko prihvaćena, pokazalo se da B-stabla imaju i manu. Naime B-stabla su osmišljena tako da prihvataju jednodimenzione podatke kao što su brojevi, karakteri i niske karaktera. Karakteristika svih ovih podataka je da se među njima lako uspostavlja poredak. Međutim, vremenom se pojavila potreba za smeštanje novog tipa - višedimenzionih podataka (objekata sa više atributa) koji nemaju mogućnost linearnog uspostavljanja poretka u klasičnom smislu i zbog toga, su nepodobni za smeštanje u B-stabla.

Kao rezultat ove evolucije podataka pojavio se veliki broj struktura za čuvanje i pristup višedimenzionalnim podacima; pregled ovih struktura može se videti u [2]. S obzirom na široki spektar primene B-stabala i na njene mnogobrojne prednosti, prirodno se pojavila i struktura podataka pogodna za višedimenzione podatke, a zasnovana na B-stablima: R-stabla. U današnje vreme neprestano nadolaze nove potrebe za čuvanjem i obradom višedimenzionih podataka. Primeri sistema koji koriste višedimenzione podatke su: baze podataka koje čuvaju prostorne ili prostorno-vremenske podatke, geografski informacioni sistemi, projektovanje VLSI, slikovne, zvučne i multimedijalne baze podataka. Svi ovi sistemi mogu da koriste R - stabla kao strukturu za smeštanje, indeksiranje i pretragu. Uzevši u obzir zapažanje da je „prostor svuda oko nas“ i široki spektar mogućnosti upotrebe R-stabala u radu [3] autori zaključuju da smo ušli u eru „sveprisutnih R-stabala“ na isti način kao što je pre 25 godina bila era B-stabala.

Gutman (Antoine Guttman) je 1984. godine predstavio prvi put strukturu podataka R-stabla [4]. Njegova osnovna ideja je bila da napravi mehanizam za indeksno pretraživanje baza podataka koji bio mogao da podrži projektovanje pomoću računara i geografske

4

informacione sisteme. R-stabla predložena u njegovom radu su visinski balansirana stabla slična B-stablima. Stablo je potpuno dinamičko, što znači da se ubacivanje i brisanje podataka u stablu može vršiti ravnopravno sa upitima i nije potrebno vršiti periodičnu reorganizaciju stabla, jer se ažuriranje vrši tokom ubacivanja ili brisanja podatka.

Od 1984. godine do danas pojavio se veliki broj različitih varijanti R-stabala [3]. Prema ovom pregledu R-stabla se mogu podeliti u tri grupe. Sve varijante R-stabala se od originalnih suštinski razlikuju po tome kako razdvajaju čvor prilikom upisa novog elementa.

- Dinamičke verzije R-stabala, koje karakteriše pojedinačno ubacivanje i brisanje podataka: Prvobitno R-stablo, R^+-stablo, R^*-stablo, Hilbertovo R-stablo, Kompaktno R-stablo, cR-stablo;
- Statičke verzije R-stabala koja očekuju statične podatke, tj. podatke za koje se ne očekuje često (ili se uopšte ne očekuje) ubacivanje i brisanje podataka nakon inicijalnog popunjavanja stabla. Ovo omogućava preprocesiranje i bolju organizaciju stabla, ali zahteva povremenu reorganizaciju u slučaju ubacivanja ili brisanja podataka: Pakovano R-stablo, Hilbertovo pakovano R-stablo, STR(Sort-Tile-Recursive) R-stablo, Buffer R-stablo;
- Prostornovremenska varijante R-stabla koje su usmerene na prostornovremenske podatke: 3D R-stablo, 2+3 R-stablo, Istorijska R-stabla, R^{to-}stablo, MV3R-stablo, Parcijalno izdržljivo R-stablo, TB-stablo, Vremenski parametrizovano R-stablo.

Svrha ovog rada je da ukratko prikaže strukturu podataka R-stablo i da predstavi implementaciju programa koji upotrebljava R-stablo za čuvanje i pretraživanje podataka vezanih za geografsku lokaciju zgrada od društveno korisnog značaja.

Ovaj rad je podeljen na četiri glave. Najpre se u poglavlju 2 opisuju strukture podataka na čijim osnovama je izgrađeno R-stablo: graf, stablo i na kraju B^+-stablo koje predstavlja tip stabla najsličniji R-stablima. Zatim se u trećem poglavlju prelazi na opis osnovnog podatka R-stabla, strukture R-stabla i algoritama za rad sa R-stablom. Četvrto poglavlje posvećeno je programskoj realizaciji R-stabla i sadrži detaljan opis korišćenja programa, klasa, metoda i

5

funkcija koje se javljaju u programu. Na kraju četvrtog poglavlja je opis rezultata do kojih se došlo testiranjem programa.

6

2. B-stabla

S obzirom na to da R-stabla predstavljaju modifikaciju B-stabala, pre nego što pređemo na detalje vezane za R-stabla, treba se upoznati sa B-stablima. U narednom tekstu sledi opis B-stabala kao najopštije strukture ovog tipa, a zatim i opis B^+-stabala koja predstavljaju najsličniju varijantu B-stabala R-stablima.

2.1. Osnovni pojmovi

Kako je svako stablo u osnovi graf, prvo se navodi definicija grafa, a zatim definisanjem dodatnih karakteristika definiše se stablo, pa tek onda specifična varijanta stabla B-stablo.

2.1.1. Definicija grafa

Neka je X neprazan skup i r binarna relacija, $r \subseteq \{ (u,v) \mid u,v \in X\}$.

Uređeni par $G = (X, r)$ naziva se graf. Elementi skupa X su čvorovi grafa, a elementi skupa r grane grafa.

Za granu $\{u,v\}$, kažemo da izlazi iz u i ulazi u v, tj njen izlazni čvor je u, a njen ulazni čvor je v. Na osnovu toga možemo da definišemo i:

-ulazni red čvora je broj grana kojima je taj čvor ulazni čvor;

-izlazni red čvora je broj grana kojima je taj čvor izlazni čvor.

Put dužine k $(k \geq 1)$ grafa $G = (V,E)$ je niz grana oblika $(v_0,v_1),(v_1,v_2),\ldots,(v_{k-1},k)$.

2.1.2. Definicija stabla

7

U teoriji grafova, stablo je graf u kome su svaka dva čvora povezana tačno jednom stazom. Drugačije rečeno, svaki povezan graf bez ciklusa je stablo. Stablo se naziva korenskim stablom ako se jedan čvor označi kao koren, u kom slučaju grane imaju prirodnu orijentaciju, prema ili od korena.

Za svaka dva čvora u stablu povezana granom važi da čvor iz koga grana izlazi nacivamo otac, a čvor u koga grana ulazi nazivamo sin.

Stablo reda m je stablo u kome je izlazni red svakog čvora manji ili jednak m (svaki čvor ima najviše m sledbenika).

Balansirano stablo je stablo kod koga je put od korena do svakog lista jednake dužine.

2.1.3. Definicija B-stabla

Definicija B-stabla uzeta je sa slajdova [5]:

B-stablo reda m je stablo pretrage reda m takvo da:

– Svi listovi su na najnižem nivou (stablo je balansirano)

– Svi unutrašnji čvorovi (osim možda korena) imaju najmanje $\lceil \frac{m}{2} \rceil$ (nepraznih) sledbenika.

– Koreni čvor, ako nije list, ima najmanje 2 sledbenika, a ako je istovremeno i list onda nema sledbenika (i stablo se sastoji samo od tog jednog čvora)

– Svaki list sadrži najmanje $\lceil \frac{m}{2} \rceil$ - 1 ključeva

2.2. B$^+$-stablo

Sve informacije vezano za B$^+$-stabla, kao i opis i algoritmi preuzeti su sa slajdova [5] i promenjeni su samo u estetskom smislu da bi se uklopili u rad.

8

S obzirom na to da se B-stabla najčešće koriste za pretraživanje indeksa baza podataka, čvorovi stabla predstavljaju blokove fiksne veličine na disku-“stranice“.

B^+-stabla predstavljaju varijantu, a ponekad i sinonim za B-stabla. Podaci mogu da se nađu samo u listovima, dok unutrašnji čvorovi služe samo da usmeravaju pretragu. Listovi su povezani pokazivačima radi efikasne pretrage u poretku atributa indeksiranja kod baza podataka.

Kao i B-stablo:

– B^+ -stablo je balansirana drvolika indeksna struktura

– Može dinamički da se ažurira i sastoji se od stranica

– Podržava operacije pojedinačnog pretraživanja, unošenja i brisanja u $O(\log_p n)$ I/O operacija, pri čemu je n broj slogova u stablu a p-kapacitet čvora izražen brojem slogova

Dodatno B^+-stabla podržavaju operacije intervalnog pretraživanja u $O(\log_p n + t/p)$ I/O operacija, pri čemu je t broj slogova u rezultatu upita.

2.2.1. B-stabla

Za svako B-stablo važi:

– čvorovi stabla su stranice;

– visina balansiranog stabla je $h \geq 0$;

– svaka stranica, osim korena i listova, ima najmanje $d + 1$ neposrednih sledbenika ($d > 0$); koren je ili list ili ima najmanje dva neposredna sledbenika;

– svaka stranica ima najviše $2d + 1$ neposrednih sledbenika.

2.2.2. Struktura čvora

• svaka stranica koja nije list predstavlja niz (p0, (k1, p1), (k2, p2),...,(km, pm)), gde je uređeni par (ki, pi) – indeksni element, koji se sastoji od vrednosti iz domena atributa

9

indeksiranja (ki) i pokazivača na stranicu neposrednog sledbenika (pi), i = 1, 2,...,m. Pokazivač na stranicu –neposrednog sledbenika je i p0;

• svaka stranica koja nije list, osim korena, ima najmanje d indeksnih elemenata, a koren koji nije list – najmanje jedan. Svaka stranica koja nije list ima najviše 2d indeksnih elemenata;

• za svaku vrednost ki iz indeksnog elementa važi sledeće: pokazivač koji joj prethodi (pi-1) pokazuje na podstablo u kome za svaku vrednost kj iz domena atributa indeksiranja važi kj < ki; pokazivač koji za njom sledi (pi) pokazuje na podstablo u kome za svaku vrednost kj iz domena atributa indeksiranja važi kj ≥ ki;

• stranice – listovi sadrže niz elemenata podataka – parova oblika (ki,bi), gde su ki kao i u indeksnom elementu, a bi – pridruženi podaci. List sadrži najmanje d' a najviše 2d' (d' > 0) elemenata podataka;

• elementi stranice (indeksni i elementi podataka) sortirani su u rastućem redosledu vrednosti iz domena atributa indeksiranja;

• sve stranice-listovi uvezane su pokazivačima u rastućem poretku vrednosti atributa indeksiranja

2.2.3. Traženje, upis, brisanje

Tačna pretraga (bez duplikata):

• ulaz: vrednost k iz domena atributa indeksiranja;

• izlaz: skup podataka b pridružen vrednosti k ako par (k, b) postoji u indeksu, inače poruka "u indeksu ne postoji vrednost k";

begin

neka je tekuća stranica – koren B+ -stabla;

while tekuća stranica nije list **do**

begin pretražiti tekuću stranicu;

if za neko i, k < ki, **then**

10

begin

neka je i’ najmanje takvo i;

tekuća stranica neka je stranica na koju pokazuje pokazivač pi’-1

end

else tekuća stranica neka je stranica na koju pokazuje pokazivač pm

end;

pretražiti list stabla;

if pronađen par (k, b) **then** rezultat je b

else poruka “u indeksu ne postoji vrednost k”

end.

Dodavanje:

Unosi se slog sa zadatom vrednošću ključa k. Unosi se uvek u list.

begin

algoritmom tačne pretrage pronaći stranicu-list za unošenje, s;

IF stranica s nije maksimalno popunjena uneti element podataka (k,b);

else begin

stranica s se cepa, alocira se nova stranica-list s’ i s povezuje sa s’;

11

u staroj stranici s ostavlja se d’ elemenata podataka, u novoj d’+1 element podataka;

if stranica-prethodnik nije maksimalno popunjena u stranicu-prethodnik upisuje se indeksni element (k’, p’), gde je k’ - najmanja vrednost na novoj stranici s’, a p’- pokazivač na novu stranicu s’– “copy up”

else begin

stranica-prethodnik s se cepa, alocira se nova stranica s’ sortira se niz od 2d+1 indeksnih elemenata

u stranici s ostaje pokazivač p0 i d manjih indeksnih elemenata u stranicu s’ upisuje se d+1-vi pokazivač pd+1 i d indeksnih elemenata – od d+2-og do 2d+1-og u prethodnika stranice s unosi se indeksni element (kd+1, p’)

gde je p’ pokazivač na stranicu s’ – “push up” (ponavlja se po potrebi)

end

end

end.

Brisanje:

• Briše se slog sa zadatom vrednošću ključa

• Brisanje elementa podataka iz lista

• Pretraga: pronalaženje lista i brisanje elementa podataka (nema duplikata ključa!!!)

12

• Ako je list nedovoljno popunjen:

– redistribucija ključeva sa susednim bratom uz ažuriranje indeksnog elementa u neposrednom prethodniku, zamenom vrednosti u indeksnom elementu koji pokazuje na desni čvor najmanjom vrednošću u tom čvoru

– ili spajanje stranica – braće, uz ažuriranje neposrednog prethodnika brisanjem indeksnog elementa za drugi čvor (operacija inverzna operaciji “copy up”)

• Ako je unutrašnji čvor nedovoljno popunjen:

– redistribucija ključeva kao za list

– ili spajanje suseda na ne-list nivou: spuštanje elementa iz čvoraprethodnika (“drag down” – operacija inverzna operaciji “push up”)

– moguće smanjenje visine B+-stabla

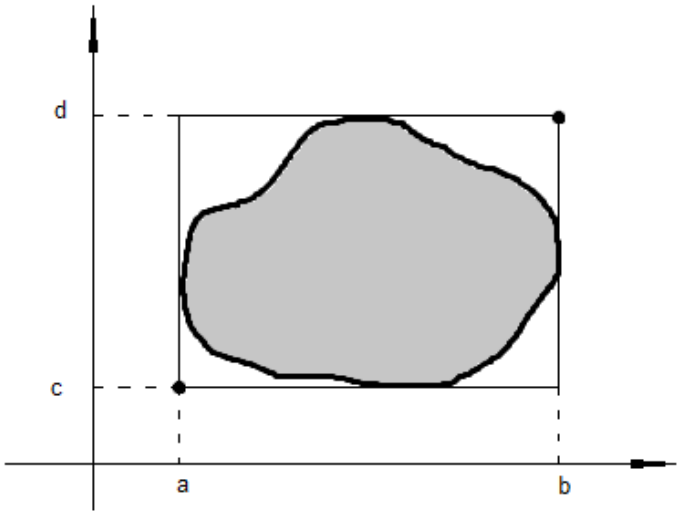
3. R-stabla

13

U ovom poglavlju se najpre definiše izraz „minimalni obuhvatajući pravougaonik“ koji predstavlja osnovni podatak u R-stablu i čije razumevanje je neophodno za dalji rad. Uz definiciju ide i detaljan opis osnovnih operacija za rad sa njim. Dalje slede algoritmi za rad sa stablom i to u redosledu: algoritmi unošenja, gde se razmatraju tri varijante različite složenosti, algoritam brisanja i algoritam pretraživanja stabla.

3.1. Minimalni obuhvatajući pravougaonik

U R-stablu se objekti čuvaju „spakovani“ u svoj minimalni obuhvatajući pravougaonik (MBR, skraćeno od engleskog minimal bounding rectangle). MBR objekta σ, nenula veličine, je pravougaonik, definisan intervalom $[(a,c),(b,d)]$, takav da predstavlja najmanji pravougaonik koji obuhvata prostor koji objekat σ zauzima, videti [6]. Na slici 1 prikazan je



primer MBR $[(a,c),(b,d)]$ koji obuhvata skup tačaka koji je obojen sivom bojom.

Termin pravougaonik ostao je u upotrebi od kada je Guttman prvi put uveo R-stabla koja su obrađivala samo dvodimenzione prostore. Isti termin koristi se i u slučaju prostora proizvoljne dimenzije. Zarad jednostavnosti, u ovom radu nastavićemo da koristimo termin pravougaonik. MBR predstavlja osnovni podatak koji se smešta u R-stablo. Iako ni nad

14

pravougaonicima ne postoji jasno definisan poredak kao ni nad objektima, on se može veštački uvesti, kao na primer poredak površina pravougaonika ili njegove sadržine ili Hilbertove koordinate njegovog centra u zavisnosti od primene.

3.2. Operacije sa minimalnim obuhvatajućim pravougaonicima

Tokom rada sa R-stablom nailazi se na potrebu da se izvršavaju neke osnovne geometrijske operacije nad MBR. Da ne bi bilo zabune oko toga na šta se odnosi koja operacija i kako je treba izvršiti u ovom poglavlju biće opisane četiri osnovne operacije.

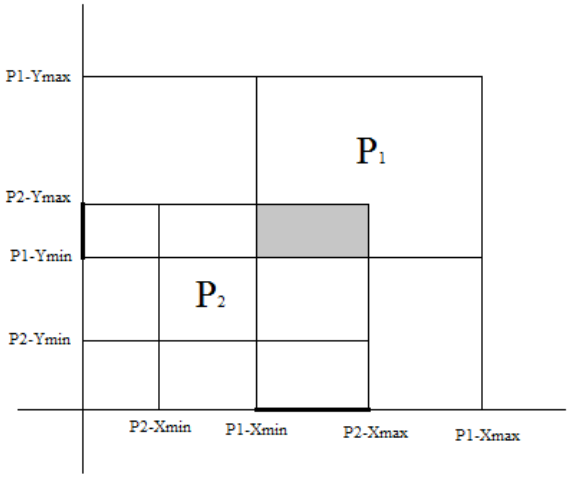
3.2.1. Površina MBR

Slika **: Izgled MBR**

Pod površinom ili veličinom MBR računa se po formuli $P=(x_{max}-x_{min})*(y_{max}-y_{min})$, gde je x_{max} najveća, a x_{min} najmanja vrednost projekcije pravougaonika na x osu, y_{max} najveća vrednost projekcije MBR na y osu, a y_{min} najmanja vrednost projekcije MBR na y osu.

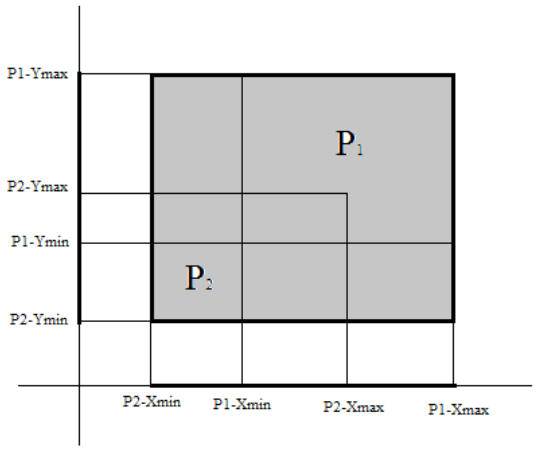
3.2.2. Presek dva MBR

Dva MBR se seku ako sadrže makar jednu zajedničku tačku. Za dva pravougaonika P_1 i P_2 smatramo da imaju zajedničku tačku ako se projekcija P_1 na x osu, preklapa u barem jednoj tački, sa projekcijom P_2 na x osu i ako se u isto vreme projekcija P_1 na y osu preklapa, u barem jednoj tački, sa projekcijom P_2 na y osu. Primer preseka (slika 2), presek MBR P_1 i P_2 je osenčen, a projekcije preseka na x i y osu, su prikazane zadebljanom linijom.



3.2.3. Proširivanje MBR

MBR se može proširiti tako da obuhvati i drugi MBR. Proširivanje pravougaonika P_1 tako da obuhvata pravougaonik P_2 podrazumeva njegovo minimalno povećanje tako da sve



tačke P_2 postanu podskup P_1. Drugim rečima unija projekcija P_1 i P_2 na x osu postaje projekcija na x osu proširenog P_1, a unija projekcija P_1 i P_2 na y osu postaje projekcija proširenog P_1 na y osu. Na primeru (Slika 3), rezultat proširenja prikazan je sivom bojom, a projekcije proširenog pravougaonika na x i y osu, prikazane su zadebljanom linijom.

3.2.4. Upoređivanje proširenja dva MBR

Odnos proširenja dva MBR je zapravo odnos njihovih veličina nakon proširenja. Odnosno, MBR P_1 zahteva veće proširenje da bi primio MBR Q, od MBR P_2, ako je površina P_1 nakon proširenja sa Q veća od površine P_2 nakon proširenja sa Q. U slučaju da se ovako računa proširenje može doći do greške ukoliko je jedan od MBR drastično veći od drugog, jer će samim tim i njegovo proširenje biti veće. Da bi se ovo izbeglo od površine proširenog MBR P_1 oduzima se površina samog MBR P_1, a od površine proširenog MBR P_2 oduzima se površina P_2. Tako dobijene vrednosti se upoređuju.

16

3.3. Struktura R-stabla

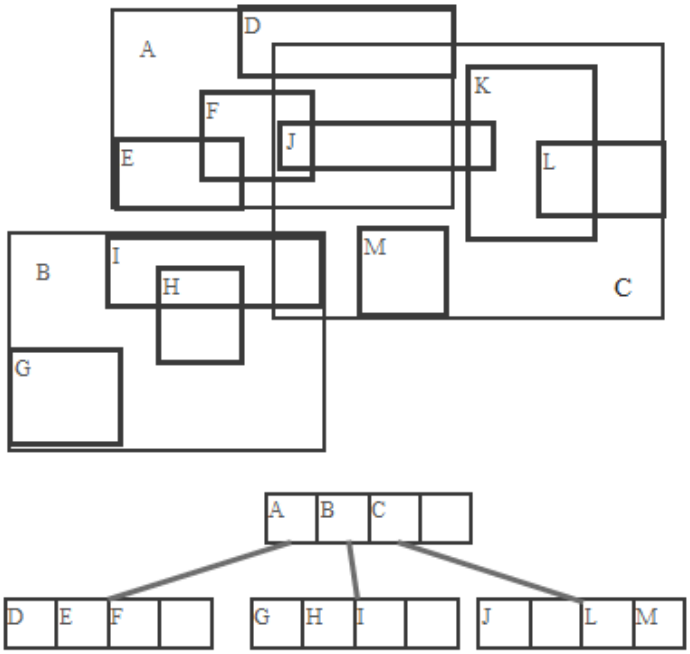
Osnovna verzija R-stabla definisana je sledećim uslovima [4]:

- U listovima se može nalaziti između m i M podataka, ($m \leq \frac{M}{2}$), osim ako je koren.
- U listovima se čuvaju podaci i MBR-ovi tih podataka.
- Unutrašnji čvor sadrži između m i M sinova, osim ako je koren.
- Unutrašnjem čvoru odgovara MBR koji sadrži sve MBR svojih sinova.
- Koren ima najmanje dva sina, osim ako je list. Tada može sadržati samo jedan element.
- Svi listovi se nalaze na istoj dubini.

Čvorovi stabla su strukture dva tipa, zavisno od toga da li je čvor list ili unutrašnji čvor. Struktura lista [7] je niz čiji je svaki element oblika $(I, identifikator\,objekta)$. Identifikator objekta se odnosi na objekat koji se smešta u strukturu, a I predstavlja MBR, koji obuhvata taj objekat. $I=(I_0, I_1, \ldots, I_{n-1})$, n je broj dimenzija u kojima radimo, a svaki I_i je interval koji sadrži projekciju objekta u i-toj dimenziji. Na primer, za n=2 to je interval oblika $[(a,c),(b,d)]$. Struktura unutrašnjeg čvora je niz elemenata oblika $(I, pokazivač)$, gde je pokazivač pokazivač na sina čvora, a I predstavlja MBR koji obuhvata sve MBR sinova čvora.

17

Primer R-stabla (slika 4), sa parametrima m=2, M=4. U primeru, unutrašnjem čvoru A odgovara MBR koji sadrži MBR-ove njegovih sinova E, F i D.



Slika : Primer strukture jednog R – stabla

3.4. Traženje, u[illegible]

Unošenje novog ključa u stablo je slično kao kod B^+-stabla. Nova vrednost se unosi u list, ako je list pun, treba ga „pocepati“. Ako je potrebno, cepanje čvorova se propagira ka korenu.

Pseudo kod [8]:

Upis u R-stablo (podatak E, koren RN)

/*Upisuje se podatak E u stablo sa korenom RN*/

Pretražiti stablo od korena RN do odgovarajućeg lista. Na svakom nivou proći kroz sve elemente.

for svaki element unutrašnjeg čvora **do**

odabrati sledbenika L takvog da je potrebno minimalno proširenje L.mbr da bi obuhvatio i E.mbr

if ima više takvih sledbenika

odabrati onog čiji MBR ima najmanju površinu

18

if odabrani list L može da primi E

Uneti E u L

Ažurirati sve MBR-ove na putu od korena do L, proširiti ih tako da obuhvate i E.mbr

else // L je pun

Pozvati algoritam za cepanje lista.

if isto povećanje je potrebno za oba lista

odabrati onaj čiji MBR ima manju površinu

if tokom dodele ostane K ključeva nedodeljenih, a u jednom od dva lista ima m-K ključeva

dodeliti sve ključeve tom listu, bez obzira na prethodni kriterijum

/*Da bi oba lista imala bar m ključeva*/

Ažurirati sve čvorove na putu od korena do listova L1 i L2 tako da njihovi MBR-ovi obuhvataju i nove čvorove L1 i/ili L2

Izvršiti cepanje unutrašnjih čvorova ako je potrebno

U slučaju cepanja korena, kreirati novi koren i povećati visinu stabla

Za razlaganje lista na dva lista može se uložiti manji ili veći napor. Nekoliko mogućih rešenja su:

- **Algoritam linearne složenosti**. Neka je S skup koji sadrži sve ključeve iz L i novi ključ E. Odabrati dva ključa e1 i e2 iz S takva da je njihova razdaljina veća od razdaljine svaka dva druga ključa iz S. Kreirati dva nova lista L1 i L2, e1 smestiti u L1, a e2 u L2. Ispitati preostale ključeve iz S i svaki ubaciti u L1 ili L2 u zavisnosti od toga čiji MBR će trebati manje da se uveća da bi ga pokrio.
  Algoritam je linearan po broju ključeva u skupu S.
- **Algoritam kvadratne složenosti**. Neka je S skup koji sadrži sve ključeve iz L i novi ključ E. Inicijalno se odabiraju dva ključa e1 i e2 iz S takva da ako bi se napravio MBR takav da obuhvata ova dva čvora, prazan prostor MBR-a (prostor unutar MBR koji nije obuhvaćen samim ključevima) bi bio maksimalan. Kreirati dva nova lista L1 i L2, e1 smestiti u L1, a e2 u L2. Ostale ključeve ubacivati jedan po jedan. U svakom koraku za sve ključeve koji su preostali u S računa se proširenje d1 potrebno da bi se

19

taj čvor uneo u L1 i proširenje d2 potrebno da bi se taj ključ uneo u L2. Bira se ključ kod koga je razlika d1-d2 najveća i unosi se u list koji zahteva manje proširenje.
Složenost ovog algoritma je kvadratna po broju ključeva u skupu S. Rešenje koje daje ne mora biti i najbolje rešenje, ali je dovoljno dobro, pa Guttman u [4] savetuje da se koristi baš ovaj algoritam kao dobar kompromis složenosti i kvaliteta rešenja.

- **Algoritam eksponencijalne složenosti**. Proveriti sve moguće varijante grupisanja ključeva i odabrati najbolju.
  Ovaj algoritam iako daje najbolje rešenje ima složenost $2^{|S|-1}$, već pod predpostavkom da S ima oko 50 ključeva ova složenost je neprihvatljivo velika.

Algoritam brisanja kod R-stabala se značajno razlikuje od onog u B-stablima, gde se problem nedovoljnog broja ključeva u listu rešava spajanjem braće. Kod B-stabala je to moguće jer braća sadrže uzastopne ključeve, što u R-stablima nije slučaj. Iako bi moglo da se realizuje brisanje sa idejom spajanja braće, prednost dole predstavljenog algoritma je ta što koristi algoritam umetanja koji teži da očuva dobre osobine drveta.

Pseudo kod [8]:

Algoritam Brisanje (podatak E, koren RN)

/*Briše se ključ E iz stabla sa korenom RN*/

if RN je list

Pretraziti sve ključeve RN dok se ne nađe E.mbr

else //RN je unutrašnji čvor

20

Pronaći sve ključeve čiji MBR-ovi sadrže E.mbr. Pratiti odgovarajuća podstabla do lista L koji sadrži E.

Obrisati E.

endif

Pozvati algoritam **Sažimanje stabla(L)**

if koren ima samo jednog sledbenika

Ukloniti koren.

Postaviti sledbenika za koren.

endif

Algoritam Sažimanje stabla (čvor L)

Postaviti X:=L.

Neka je S skup čvorova koji će biti obrisani. Inicijalno je prazan.

while X nije koren

Neka je P roditelj čvora X i neka je E ključ u P koji odgovara sledbeniku X.

if X sadrži manje od m ključeva

Ukloniti E iz P.

Ubaciti X u S.

endif

Postaviti X:=P.

endwhile

Ubaciti sve ključeve čvorova koji se nalaze u S.

Algoritam pretrage je sličan onom kod B-stabala. Unosi se MBR i stablo se pretražuje sa ciljem pronalaženja svih podataka čiji MBR se preklapa sa zadatim MBR.

Pseudo kod [7]:

21

Algoritam Pretraga(MBR E, koren RN)

/*Traže se ključevi u stablu sa korenom RN čiji MBR se preklapa sa E*/

if RN nije list

za sve sinove Ki iz RN proveriti da li se Ki.mbr preklapa sa E

if ima preklapanja

Pozvati algoritam **Pretraga(MBR E, koren Ki)**

endif

else //RN jeste list

za sve elemente Ki iz RN proveriti da li se Ki.mbr preklapa sa E

if ima preklapanja

Vratiti Ki kao jedno rešenje

endif

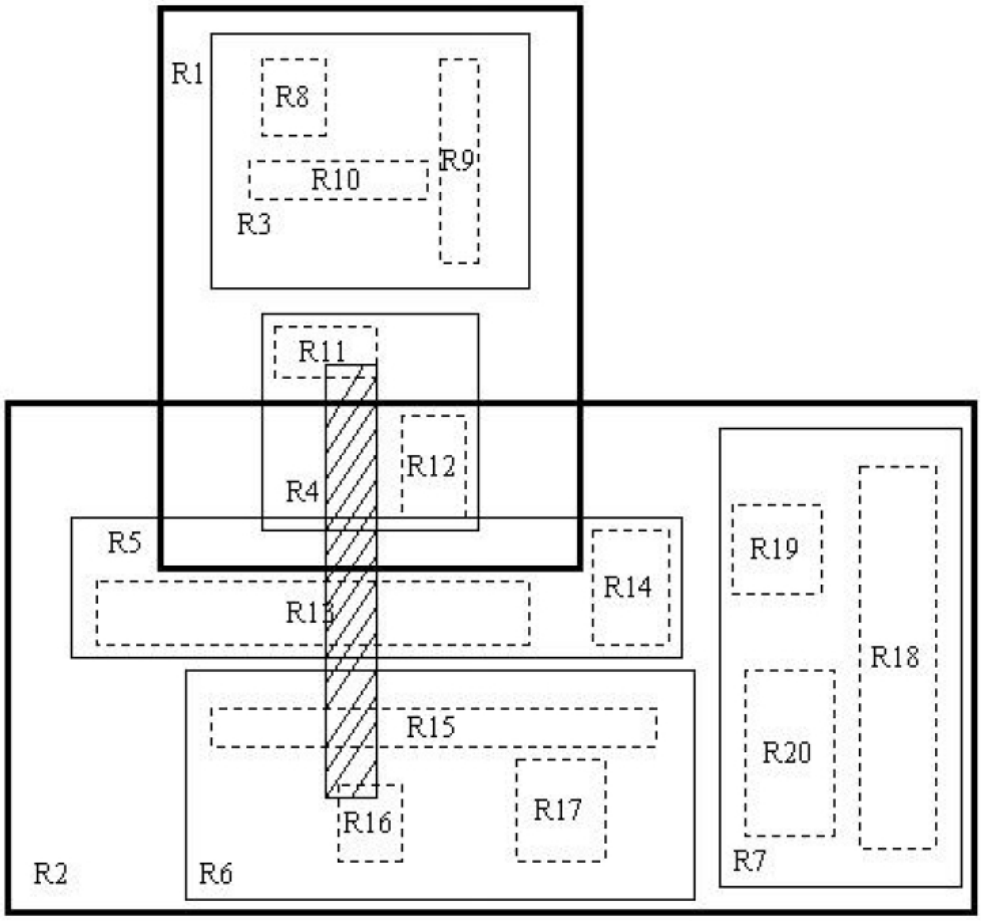
endif

Primer:

Na slici 5 nalazi se primer podele prostora podataka na MBR-ove, a na slici 6 njemu odgovarajuće R-stablo. Pretražuje se šrafirani pravougaonik. Na slici 6 zaokruženi čvorovi predstavljaju one čvorove koje je algoritam pretrage obišao, a ključevi u listovima koji su označeni strelicom predstavljaju rešenja.

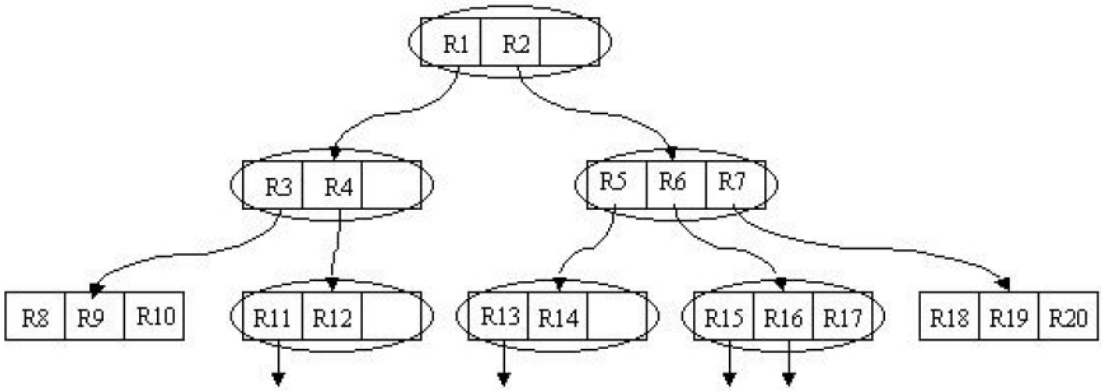
Algoritam kreće od korena i zaključuje da se traženi pravougaonik preklapa sa MBR oba sina u čvoru (R1 i R2). Iz R1 nastavlja u njegovog sledbenika i proverava oba njegova ključa (R3 i R4), međutim do preklapanja dolazi samo sa R4, pa ne obilazi sledbenika R3. Iz R4 nastavlja do lista u kome se traženi pravougaonik preklapa samo sa pravougaonikom R11, pa se on vraća kao jedan od rezultata, i pretraga u ovom podstablu se završava. Analogno se vrši pretraga i u drugom podstablu na koje pokazuje ključ R2.

22



Slika : Primer podataka grupisanih u MBR-ove.

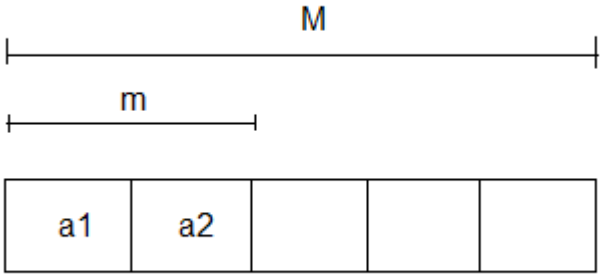
23



Slika : R-stablo koje odgovara podacima prikazanim na slici 5.

3.6. Izbor vrednosti m i M

Promenljiva m označava minimalan dozvoljen broj podataka u čvoru, a M maksimalan dozvoljen broj podataka u čvoru. Slika 7 prikazuje jedan primer čvora sa parametrima i njegove parametre m=2 i M=5.



Slika : Prikaz parametara M i m

24

Od promenljive m zavisi broj čvorova i složenost operacija u stablu. Maksimalan broj čvorova u stablu je $\lceil \frac{N}{m} \rceil + \lceil \frac{N}{m^2} \rceil + 1$, gde N predstavlja broj objekata u stablu. Visina stabla, a samim tim i složenost pretraživanja stabla je $|\log mN|$ -1, gde je N broj podataka. Promenljiva M se bira u zavisnosti od kapaciteta diska. Odabir ovih parametara je veoma bitan. Ako je m malo, ređe dolazi do situacije da nakon brisanja imamo premalo podataka u čvoru i da moramo da reorganizujemo stablo. Sa druge strane što je M manje, visina stabla je veća, pa samim tim i složenost pretraživanja.

4. Programska realizacija i rezultati

Operacije sa R-stablom realizovane su u programu R-stablo. U okviru programa realizovana je struktura podataka R-stablo konkretizovana za upotrebu pretraživanja zgrada po geografskoj lokaciji. Implementiran je i interfejs koji olakšava rad sa tom strukturom.

Za razvoj aplikacije korišćen je programski jezik C++ i višeplatformski aplikacioni interfejs Qt creator za razvoj grafičkog okruženja programa.

4.1. Organizacija programa

25

Celine u programu mogu se podeliti u dve grupe. Jedna grupa sadrži sve celine vezane za strukturu R-stabla i one predstavljaju suštinu programa. Druga grupa su celine vezane za interfejs i za iscrtavanje na interfejsu. U ovom smislu opis programa je i podeljen na dve celine. Prvo ide detaljan opis implementacije strukture podataka R-stablo, a tek zatim objašnjenje kako funkcionišu funkcije koje su vezane za interfejs.

Kako je R-stablo zapravo skup pametno organizovanih i povezanih čvorova, napravljena je apstraktna klasa Node (engleski čvor), koja služi kao bazna klasa za dva tipa čvora koji se pojavljuju u stablu: INode (skraćeno od engleskog Internal Node, što znači unutrašnji čvor) i Leaf (list na engleskom). Ova dva tipa čvora su razdvojena zbog toga što se razlikuju u strukturi (unutrašnji čvorovi samo usmeravaju pretragu, listovi čuvaju podatke), a samim tim i po implementaciji određenih metoda. S obzirom na to da su metode klasa INode i Leaf koje nasleđuju od Node međusobno isprepletane opis i objašnjenje tih metoda u daljem tekstu je odvojen od opisa samih klasa i svaka metoda se paralelno objašnjava za INode i Leaf.

Kao osnovni podatak MBR ima svoju klasu u koju su izdvojene sve funkcije potrebne za rad sa njim. Klasa Node ima pristup klasi MBR.

Podatak koji se čuva u stablu je entitet za sebe pa samim tim ima svoju klasu TreeData.

4.1.1. Klasa MBR

Klasa MBR sadrži pre svega podatke koji ga definišu u prostoru: x i y koordinate donjeg levog temena – xDL, yDL (gde DL označava skraćenicu od engleskog Down Left, dole levo); x i y koordinate gornjeg desnog temena – xUR, yUR (gde UR označava skraćenicu od engleskog Up Right, gore desno). Prilikom kreiranja novog objekta klase MBR proverava se da xDL bude manje od xUR i analogno za y koordinate. Ukoliko se kreira novi objekat klase MBR u trenutku kada se ne znaju njene dimenzije za sve četiri koordinate treba zadati vrednost 0. U metodama klase MBR vodi se računa o ovom slučaju koji ne predstavlja

26

objekat visine i širine 0, već specijalan slučaj. Osim toga svaki MBR ima svoj jedinstveni identifikator id, koji se računa automatski u konstruktoru tako što se statički pamti (preko idGen) koji je to po redu napravljen MBR.

Klasa MBR može da izračuna svoju površinu pozivom funkcije getP(), potrebno povećanje u slučaju unošenja novog MBR u postojeći - checkInsert(MBR *m) i površinu preseka sa drugim MBR – checkIntersect(MBR *m). Klasa zna da li se seče sa drugim MBR - intersect(MBR *m) i ume da se proširi ako se od nje zahteva da prihvati novi MBR – insert(MBR *m). Ove metode su implementirane u skladu sa objašnjenjem datim u glavi 3.2. Takođe klasa ume da proveri da li je jednaka nekom drugom MBR – areEqual(MBR *m), što se radi proverom jednakosti njihovog jedinstvenog identifikatora.

```
class MBR
{
private:
    float xDL;
    float yDL;
    float xUR;
    float yUR;
    static int idGen;
    int id;
public:
    MBR(float xd, float yd, float xu, float yu);
    ~MBR();
    float getxDL();
    float getxUR();
    float getyDL();
    float getyUR();
    int getId();
    void resetId();
    bool areEqual(MBR *m);
    bool intersect(MBR *m);
    float checkIntersect(MBR *m);
    float checkInsert(MBR *m);
    void insert(MBR *m);
    float getP();
    void clear();
};
```

4.1.2. Klasa TreeData

Klasa TreeData služi za smeštanje i obradu podataka koji se smeštaju u stablo. Prilikom unosa jednog podatka, odnosno skloništa, kreira se novi objekat klase TreeData i on

27

se popunjava svim potrebnim informacijama. S obzirom na to da se ovi podaci čuvaju u programu samo da bi mogli da se ispišu kada se u toku pretrage nađu skloništa koja odgovaraju korisnikovim parametrima, ova klasa nema veliki broj metoda. Osim gettera i settera, postoje još samo dve metode. Metod printData unosi tekstualni podatak u skup s. Taj tekstualni podatak sastoji se od podataka koje TreeData čuva nadovezanih jedan na drugi onako kako će oni biti ispisani. Metod areEqual proverava jednakost dva podatka prema njihovim jedinstvenim identifikatorima, i koristi se prilikom brisanja podatka, za verifikaciju podatka koji se briše, da se ne bi obrisao pogrešan podatak.

```
class TreeData
{
private:
    int id;
    QString buildingName;
    QString buildingType;
    QString address;
    QString city;
    QString state;
    QString zip;
    QString county;
    QString contact;
    QString phoneNumber;
public:
    TreeData(int _id, QString bN, QString bT, QString add, QString cy,
QString st, QString zi, QString cou, QString cont, QString pN);
    ~TreeData();
    int getId();
    QString getBuildingName();
    QString getBuildingType();
    QString getAddress();
    QString getCity();
    QString getState();
    QString getZip();
    QString getCounty();
    QString getContact();
    QString getPhoneNumber();
    void setData(int _id, QString bN, QString bT, QString add, QString cy,
QString st, QString zi, QString cou, QString cont, QString pN);
    void printData(set<QString> &s);
    bool areEqual(TreeData *d);
};
```


28

4.1.3. Klase Node i njene podklase

Klasa Node je apstraktna klasa i nema informacije o podacima, već samo o metodama koje implementiraju njene podklase. Ovakva polimorfna struktura čvorova je veoma zgodna jer je većina metoda rekurzivna i u trenutku rekurzivnog poziva se ne zna da li će se metod pozvati nad unutrašnjim čvorom ili nad listom, što se lako rešava tako što se svaka metoda poziva nad klasom Node, a o tome nad kojom podklasom(INode ili Leaf) će biti pozvan metod vodi računa kompajler u toku samog izvršavanja programa.

```
class Node
{
public:
    virtual void setParent(Node *n) = 0;
    virtual MBR *getMBR() = 0;
    virtual void insert(MBR *m, TreeData *d) = 0;
    virtual void find(MBR *m, set<int> &s) = 0;
    virtual void erase(MBR *m, TreeData *d) = 0;
    virtual void split() = 0;
    virtual void colectData(set< pair<MBR *, TreeData *> > &s) = 0;
    virtual void eraseTree() = 0;
    virtual void updateMBR() = 0;
};
```

I unutrašnji čvor i list čuvaju pokazivač na roditelja (INode *parent) koji može biti samo tipa INode, jer listovi nemaju sinove pa samim tim ne mogu biti roditlji. Ovaj podatak je potreban za kretanje kroz stablo od lista ka korenu što se koristi uglavnom za potrebe ažuriranja podataka u precima.

Unutrašnji čvor sadrži i niz parova (MBR, pokazivač na sina) što je implementirano kao mapa parova (MBR *, Node *). Osim toga INode čuva i informaciju o svom MBR koji obuhvata sve MBR njegovih sinova.

```
typedef map<MBR *, Node *> Element;
class INode: public Node
{
private:
    Element nodeElements;
    INode *parent;
    MBR *mbr;
public:
      INode(INode *_parent) {};
      ~INode(){};
      Element getNodeElements(){};
      virtual MBR *getMBR(){};
      virtual void setParent(Node *n) {};
      virtual void eraseTree(){};
      virtual void insert(MBR *m, TreeData *d) {};
```


29

```
        virtual void find(MBR *m, set<int> &s) {};
        virtual void erase(MBR *m, TreeData *d) {};
        virtual void split(){};
        void setParentsElements(){};
        virtual void updateMBR(){};
        void eraseElement(Node *n, bool remove=true) {};
        void insertElement(Node *n) {};
        void removeNode(){};
        virtual void collectData(set< pair<MBR *, TreeData *> > &s) {};
};
```

List osim pokazivača na roditelja sadrži i niz parova (MBR, podatak) što je implementirano kao mapa parova (MBR *, TreeData *), takođe svaki list pamti i informaciju o svom MBR koji obuhvata MBR svih podataka na koje taj list pokazuje.

```
typedef map<MBR *, TreeData *> DataElement;
class Leaf: public Node
{
private:
    DataElement leafElements;
    INode *parent;
    MBR *mbr;
public:
        Leaf(INode *_parent) {};
        ~Leaf(){};
        DataElement getLeafElements(){};
        virtual MBR *getMBR(){};
        virtual void setParent(Node *n) {};
        virtual void eraseTree(){};
        virtual void insert(MBR *m, TreeData *d) {};
        virtual void find(MBR *m, set<int> &s) {};
        virtual void erase(MBR *m, TreeData *d) {};
        virtual void split(){};
        virtual void updateMBR(){};
        void removeLeaf() {};
        virtual void collectData(set< pair<MBR *, TreeData *> > &s) {};
};
```

4.1.4. Metoda insert()

Metoda insert poziva se nad korenom stabla prilikom unosa novog podatka d koji ima MBR m.

Ukoliko se stablo sastoji samo od jednog čvora – korena, onda je koren tipa list i metoda insert se poziva nad listom. U suprotnom insert se poziva nad unutrašnjim čvorom.

30

Metoda insert za unutrašnji čvor:

Za svaki MBR sina računa se koliko proširenje je potrebno da prihvati MBR podatka koji se unosi. Izračunavanje se vrši pozivom metode klase MBR checkInsert i oduzimanjem od toga površine MBR sina da bi se dobila površina samog proširenja, a ne proširenog MBR sina. Nad sinom koji zahteva najmanje proširenje, metoda se poziva rekurzivno. Ukoliko ima više takvih sinova odabira se onaj sa najmanjom površinom.

Metoda insert za listu:

Kada se rekurzivnim pozivima dođe do lista stabla, podatak i njegov MBR se smeštaju u taj list. Ukoliko list ne prevazilazi svoj kapacitet zadat parametrom M, ažurira se njegov MBR i rekurzivno ka korenu od lista vrši se ažuriranje MBR svakog pretka. Ovo se izvršava pozivom metode updateMBR, koji će biti opisan kasnije. Ukoliko se unošenjem novog podatka list prepunio, odnosno sadrži više od M podataka, on treba da se pocepa. Za to pozivamo metodu split, koja će takođe biti objašnjena kasnije.

4.1.5. Metoda *split*()

Metoda split je veoma kompleksna i za sobom povlači pozivanje mnogih drugih funkcija koje su napravljene kao pomoćne da bi split mogao da se izvršava. Do pozivanja ovog metoda dolazi kao što je u prethodnoj sekciji 4.2.4. objašnjeno iz metode insert.

Cilj je zamentiti postojeći čvor koji se prepunio sa dva nova čvora. Elemente iz starog čvora treba rasporediti u nove na što praktičniji način. Kao što je u glavi 3.4. objašnjeno postoje tri varijante različite složenosti. Po preporuci kreatora R-tabala Gutmana [4], u implementaciji sam odabrala varijantu kvadratne složenosti.

Pošto se metoda split uvek prvi put poziva nad listom, a tek potom tokom ažuriranja predaka tog lista može doći do pozivanja splita nad unutrašnjim čvorom, u ovoj glavi prvo će biti opisan split nad listom.

Metoda split za listu:

31

Na samom početku metode kreiraju se dva nova lista čiji roditelj je isti kao kod postojećeg koji se cepa. Zatim se dvostrukom for petljom za svaka dva elementa postojećeg lista ispituje razdaljina njihovih MBR:

```
currP=(it->first)->checkInsert(jt->first)-(it->first)->getP()-
(jt->first)->getP()+(it->first)->checkIntersect(jt->first);
```

Razdaljina dva MBR – currP računa se kao prazan prostor koji ostaje prilikom proširenja jednog MBR drugim. Odnosno pozivom metode klase MBR checkInsert računa se površina dobijena proširivanjem jednog MBR drugim. Zatim pošto površina samih MBR nije od važnosti, oduzimaju se od dobijene vrednosti površine oba MBR koja se spajaju, pozivom metode getP klase MBR. Može da se desi da se data dva MBR seku, u kom slučaju površina preseka bi se dva puta oduzela od površine MBR koji ih oba obuhvata. Zato na kraju dodajemo površinu njihovog preseka pozivom metode checkIntersect klase MBR.

Čvorovi koji su najudaljeniji razdvajaju se. Jedan od njih se unosu u jedan od dva novokreirana lista, a drugi u drugi novokreirani list. Ovo se obavlja pozivanjem metode insert nad novim listovima. Kako je ranije objašnjeno, ovim se automatski ažurira i MBR tih listova.

Sledeći korak je smeštanje preostalih elemenata postojećeg lista u nove tako da na kraju MBR novih listova bude što manji.

Za svaki preostali element računa se koliko proširenje je potrebno da novi listovi prihvate njegov MBR. Odabira se onaj element kod koga je razlika proširenja potrebnog da se ubaci u jedan list i proširenja potrebnog da se ubaci u drugi list najveća. Ovo izračunavanje se vrši sledećom formulom:

```
dist=(mbr_e1->checkInsert(it->first)-mbr_e1->getP()-
(it->first)->getP()+mbr_e1->checkIntersect(it->first))-
(mbr_e2->checkInsert(it->first)-mbr_e2->getP()-(it->first)->getP()+
mbr_e2->checkIntersect(it->first));
```

Razlika proširenja – dist se računa tako što se izračuna prethodno opisan currP za jedan list i novi potencijalni element i od njega se oduzme currP za drugi list i novi potencijalni element. Ukoliko je dist pozitivan znači da je potrebno veće proširenje de se novi element ubaci u prvi list, pa samim tim njega unosimo u drugi list i obratno. U slučaju da je dist=0, odnosno oba lista zahtevaju isto proširenje da bi prihvatila novi element, on se unosi u onaj sa manjom površinom. Tokom ovog procesa mora da se vodi računa o tome da se ne desi da se većina elemenata ubaci u jedan list, a da drugi list ostane nedovoljno popunjen. Ukoliko se desi da je

32

preostalo tačno toliko elemenata koliko treba da se jedan od listova popuni do minimalne dozvoljene količine, treba sve elemente smestiti u njega.

Na kraju kada su se svi elementi rasporedili treba ažurirati pretke. Ukoliko list nema predaka, odnosno on je koren treba napraviti novi koren koji će biti unutrašnji čvor i u njega ubaciti pokazivače na novokreirane listove koji postaju njegovi jedini sinovi. Takođe u novim listovima treba postaviti pokazivače na njihovog novog roditelja – korena. Ukoliko pak pocepani list ima roditelja, iz roditelja treba obrisati njega i uneti nove listove. Ove operacije obavljaju se pozivom metode insertElement i eraseElement, koje takođe rade rekurzivno, ali u smeru ka korenu i koje će biti objašnjene malo kasnije.

Metoda split za unutrašnji čvor:

Metoda split nad unutrašnjim čvorom izvršava se na veoma sličan način kao i nad listom sa ključnom razlikom da se na početku kreiraju dva nova unutrašnja čvora, a ne lista i da se u obzir sve vreme uzimaju MBR sinova umesto MBR podataka. Postoji još jedna bitna razlika, a to je da se na samom kraju metode moraju promeniti pokazivači na roditelja svih sinova koji su se prebacili iz starog čvora u nove čvorove. Za ovo je zadužena metoda setParentsElements koja se poziva nad novim čvorovima. Ona prolazi kroz sve elemente mape čvora i svakom sinu menja pokazivač na roditelja na čvor nad kojim je metoda pozvana.

4.1.6. Metoda *insertElement*()

Ova metoda postoji samo u unutrašnjem čvoru i postoji da bi se mogao novi čvor ubaciti u unutrašnji čvor, kao što je ranije napomenuto, prilikom ažuriranja predaka, u metodi split. Metoda insert za ovo nije pogodna, jer ona unosi podatak, a ne čvor i samim tim može da unese samo u list, dok kroz unutrašnje čvorove samo prolazi u potrazi za odgovarajućim listom.

Metoda insertElement prima čvor, a samim tim i njegov MBR koji se čuva u čvoru. U mapu nodeElements unosi dati čvor i njemu odgovarajuć MBR. Ukoliko čvor može da primi novi element prelazi se na ažuriranje njegovog MBR. Ukoliko je čvor prepunjen on se cepa pozivom metoda split.

33

4.1.7. Metoda *eraseElement*()

Kao što je ranije napomenuto ova metoda je pomoćna i poziva se ako se uklanja čvor iz drveta i treba obrisati pokazivač na njega iz njegovog roditelja. Kao i insertElement i ova metoda postoji samo u unutrašnjem čvoru, jer list nema sinova, pa samim tim i ne može da se briše pokazivač na sina. Ukoliko uklanjanjem čvora i njegovog MBR iz mape NodeElements broj elemenata u čvoru ostaje iznad zadatog parametrom m, treba ažurirati njegov MBR. U suprotnom čvor treba ukloniti pozivom metode removeNode. Metod prima identifikator remove, koji je inicijalno postavljen na true. Ukoliko je postavljen na false, čvor se ne uklanja iako ima manje od m elemenata, jer indikator ukazuje na to da će već u sledećem koraku biti u njega unešen novi element. Ovaj specifičan slučaj se javlja u funkciji split, kada se jedan element briše, ali se na njegovo mesto smeštaju dva nova.

4.1.8. Metoda *removeNode*()

Metoda removeNode je metoda unutrašnjeg čvora i poziva se ukoliko je potrebno obrisati unutrašnji čvor, do čega dolazi kada broj elemenata u čvoru padne ispod dozvoljene granice zadate parametrom m. Razlikujemo dva slučaja za koje ova metoda treba različito da radi.

Prvi slučaj je ukoliko čvor koji se uklanja nije koren. U tom slučaju pozivamo metodu collectData koja prolazi kroz celo podstablo tog čvora i pamti sve parove (podatak, MBR) koji se nalaze u listovima ovog podstabla, a koji treba da se sačuvaju, jer se metodom removeNode želi obrisati samo taj čvor nad kojim je pozvan, ne i podaci koji su posredno vezani za njega. Nakon što su svi podaci sačuvani u jedan skup iz roditelja čvora briše se pokazivač na njega pozivom metode eraseElement, čime se iz drveta obrisao sam čvor i celo njegovo podstablo. Zatim se, iz skupa u koji smo ih sačuvali, jedan po jedan pozivom metode insert ponovo unose svi podaci koji su na ovaj način uklonjeni iz drveta.

Drugi slučaj je kada se metoda removeNode pozvala nad korenom. Korenu je dozvoljeno da ima najmanje dva elementa nezavisno od toga na koliko je postavljen parametar m. U ovom slučaju, ako koren ima bar dva elementa, removeNode se ne izvršava. Ako je broj elemenata korena pao ispod dva, odnosno ima samo jedan element, taj element postaje novi koren.

34

4.1.9. Metoda *removeLeaf*()

Ova metoda se poziva ukoliko broj elemenata u listu padne ispod dozvoljene parametrom m. Iako radi sličnu stvar kao i removeNode, ove dve metode se pozivaju u različitim slučajevima i rade na različite načine, pa su zato razdvojene. Metoda removeLeaf poziva metodu collectData koja čuva sve podatke preostale u tom listu koji ne treba da se obrišu iz stabla. Zatim briše pokazivač na njega iz roditelja pozivom metode eraseElement i ponovo ubacuje podatke sačuvane u skupu u stablo metodom insert. Ukoliko je list nad kojim je ova metoda pozvana u isto vreme i koren, metoda se ne izvršava.

4.1.10. Metoda *collectData*()

Kao što je ranije napomenuto ova metoda služi da pokupi sve podatke u određenom podstablu i da ih smesti u zadati skup s. Nad unutrašnjim čvorovima ova metoda se samo poziva rekurzivno za sve sinove unutrašnjeg čvora. Kada stigne do lista, svi elementi iz mape LeafElements kopiraju se u zadati skup s.

4.1.11. Metoda *updateMBR*()

I nad unutrašnjim čvorom i nad listom ova metoda radi isto. Obriše MBR datog čvora, postavljajući sve njegove koordinate na nula, što je kako je objašnjeno specijalan slučaj za klasu MBR, a zatim prolazeći kroz mapu čvora, ponovo ubacuje sve MBR koji se u mapi nalaze. Iako ovo deluje neefikasno, uzimajući u obzir da svaki čvor ima maksimalno M elemenata, a da je parametar M uglavnom broj do sto, složenost ove operacije nije neprihvatljiva. Drugi razlog ovakve implementacije je taj što ukoliko se ažuriranje poziva zbog brisanja elementa iz MBR, smanjenje MBR tako da ne obuhvata zadati ne daje tačno rešenje jer se u tom trenutku ne zna gde se sve MBR koji se briše preseca sa drugima koji se ne brišu.

4.1.12. Metoda *find*()

35

Metoda virtual void *find*(MBR *m, set<int> &s) se poziva kada se u stablu traže svi podaci čiji MBR se seče sa zadatim m.

Metoda find za unutrašnji čvor:

U slučaju unutrašnjeg čvora traže se sinovi čiji MBR se seče sa zadatim. Za razliku od metode insert gde se metoda rekurzivno pozivala samo nad jednim sinom, ovde se find rekurzivno poziva nad svim sinovima koji ispunjavaju zadati uslov. Za proveru da li se MBR seku, poziva se metoda klase MBR checkIntersect.

Metoda find za listu:

Za sve MBR podataka koji se nalaze u listu proverava se da li se seku sa traženim. Ako da, podatak vezan za taj MBR se unosi u skup s. Može se desiti da skup s sadrži više podataka, ali to i jeste cilj, jer je moguće da ima više podataka koji odgovaraju datom kriterijumu.

4.1.13. Metoda *erase*()

Ukoliko korisnik želi da obriše podatak iz drveta poziva se ova metoda. Metoda erase prima MBR koji služi da se pretraga za elementom koji se želi obrisati usmerava. Ukoliko je taj MBR nepoznat program sam zadaje najveći mogući MBR.

Metoda erase za unutrašnji čvor:

Metoda nad unutrašnjim čvorom proverava da li se MBR nekog sina preseca sa zadatim i poziva rekurzivno metodu nad tim sinom. Kada se u listu naiđe na traženi podatak postavlja se flag, da bi unutrašnji čvorovi znali da je dalja pretraga nepotrebna.

Metoda erase za listu:

U listu metoda proverava za svaki MBR podatka da li seče sa zadatim, ako da, proverava se da li je podatak vezan za taj MBR odgovarajuć. Ako jeste, podatak i njegov MBR se brišu iz lista. Ukoliko je broj elemenata u listu i dalje dovoljan, ažurira se MBR samog lista. U suprotnom poziva se metoda removeLeaf.

4.1.14. Metoda *eraseTree*()

36

Ova metoda se poziva ukoliko korisnik želi da obriše celo stablo. U unutrašnjim čvorovima, metoda se prvo poziva rekurzivno za sve sinove, a zatim se po povratku iz rekurzije briše i sam čvor. Dodatno u unutrašnjem čvoru idGen klase MBR koji služi da broji dotle napravljen broj objekata klase MBR, postavlja se ponovo na nulu.

U listu nije potreban dalji rekurzivan poziv pa se samo briše list.

4.2. Organizacija pomoćne komponente u programu

Osim klasa koje su vezane za samu strukturu podataka R-stabla, postoji pet funkcija, po jedna za svako dugme na interfejsu. Svaka funkcija zadužena je za konkretnu akciju nad stablom i svaku tu akciju izvršava pozivom odgovarajuće metode nad korenom, koji je globalna promenljiva u programu.

- on_read_from_form_clicked() – funkcija učitava podatke unešene preko interfejsa, smešta ih u odgovarajuće klase i poziva funkciju insert() za unošenje novog podatka u stablo;
- on_read_from_file_clicked() – funkcija učitava podatke iz fajla, liniju po liniju. Svaki podatak smešta u odgovarajuću klasu i zatim poziva funkciju insert() za unošenje novog podatka u stablo;
- on_find_clicked() – funkcija učitava zadate koordinate, smešta ih u novi objekat klase MBR i zatim poziva metodu stabla find() za taj MBR, koja pretražuje stablo. Rezultat pretrage ispisuje;
- on_erase_element_clicked() – funkcija učitava koordinate i jedinstveni identifikator skloništa koje je korisnik uneo. Poziva erase() metodu stabla koja briše podatak koji odgovara učitanim parametrima;
- on_remove_tree_clicked() – funkcija poziva metodu stabla eraseTree(), koja briše celo stablo.

Za svaku funkciju važi da nakon što se završi akcija nad stablom koju je pozvala, ukoliko je odabrana opcija Iscrtavanje, poziva funkciju za iscrtavanje podataka u stablu.

Navedene funkcije koriste biblioteke map i set iz STL (skraćenica od engleskog Standard Template Library) biblioteke, zatim QFileDialog, qfile, fstream i qtextstream koje se koriste za učitavanje podataka iz fajla i qMessageBox potreban za pokretanje prozora kojim se korisnik informiše u slučaju greške.

37

Za crtanje napravljena je klasa widget koja koristi ugrađene funkcije biblioteke QWidget. Sa ostatkom programa ova klasa komunicira preko klase mainwindow ugrađenim mehanzmom.

4.3 Primena programa R-stablo

Svrha programa R-stablo je čuvanje i pretraživanje lokacije zgrada od društveno korisnog značaja. Pod lokacijom se u ovom slučaju podrazumevaju geografske koordinate i adresa. Koordinate koristi program za smeštanje i pretraživanje baze podataka, a adresa i ostale informacije vezane za instituciju, služe korisniku. Ovako korisnici programa R-stablo, nakon učitavanja odgovarajućih podataka, imaju brz i lak pretraživač, a da pri tom kao rezultat pretrage dobijaju podatke koji su lako čitljivi za čoveka.

Za svaku instituciju, osim i X i Y koordinate, vezane su sledeće informacije:

- Jedinstveni identifikator
- Naziv
- Tip zgrade
- Adresa
- Grad
- Država
- Poštanski kod
- Okrug
- Ime kontakt osobe
- Broj telefona

U programu su razdvojene komponente za učitavanje koordinata i ostalih podataka. Koordinate su tipa: real (realan broj) i one su jedine relevantne informacije stoga obavezno moraju biti zadate. Sve dodatne informacije o podacima, u programu se pamte kao tekstualni podaci i po njima ne može da se vrši pretraga. Zahvaljujući ovakvoj organizaciji u slučaju da program treba da se prilagodi drugoačijem tipu baze podataka, treba promeniti samo klasu TreeData u koju se dodatni podaci smeštaju i treba promentiti način učitavanja, što podrazumeva promenu interfejsa (polja za unos dodatnih informacija) i dve funkcije zadužene za učitavanje: on_read_from_form_clicked() i on_read_from_file_clicked().

Primeri baza podataka koji mogu biti korišćeni:

38

- baza podataka skloništa
- baza podataka privatnih škola
- baza podataka domova zdravlja
- baza podatka biblioteka
- baza podataka doma za stare

Nakon unošenja podataka, upiti treba da budu oblika oblika: Koje institucije se nalaze na prostoru koji obuhvata MBR sa koordinatama od (X_1,Y_1) do (X_2,Y_2).

4.4. Upotreba programa R-stablo

U ovom poglavlju detaljno je opisano kako se upotrebljava program R-stablo koji je napisan uz ovaj rad. Poglavlje kreće od same osnove: pokretanja programa, zatim prolazi kroz detalje interfejsa i objašnjava čemu služe opcije i kako ih treba koristiti. Izgled i upotreba programa zavise od baze podataka koja je korišćena, stoga ovo poglavlje sadrži i deo koji opisuje detaljno i bazu podataka i izgled fajla koji sadrži podatke.

4.4.1 Prevođenje programa

Besplatan Qt sa ugrađenim kompajlerom može da se pronađe na zvaničnoj web stranici http://qt-project.org/downloads#qt-creator. Ovde je korišćena verzija Qt 5.1.1 for Windows 32-bit (MinGW 4.8, OpenGL, 666 MB) .

Kada se alat instalira, program se može otvoriti na dva načina:

- Duplim klikom na ikonicu sa ekstenzijom .pro, tipa Qt Project file
- Odabiranjem opcije Open file or project iz padajućeg menia File, otvara se pretraživač u kome treba pronaći folder u kome se nalazi program i odabrati fajl sa ekstenzijom .pro, tipa Qt Project file.

Kada se u Qt-u otvori program, sa leve strane treba odabrati opciju Projects, ukoliko je izabrana opcija Shadow build, treba je isključiti. Zatim pritiskom na zelenu strelicu u donjem levom uglu prozora, pokrenuti program.

39

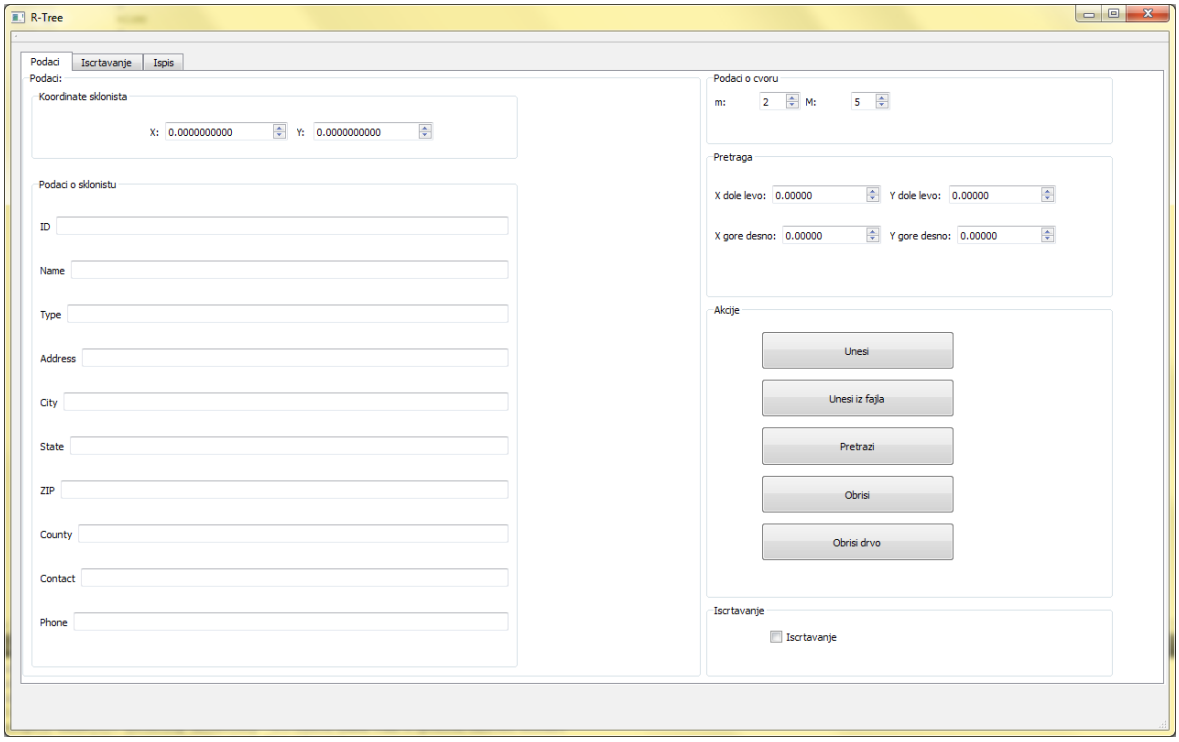
4.4.2 Testni podaci

U svrhu testiranja, u program su učitani realni podaci iz baze podataka o skloništima u Severnoj Karolini, koji se mogu pronaći i besplatno preuzeti sa internet stranice: http://data.nconemap.com/geoportal/catalog/search/browse/browse.page. Ova baza podataka je odabrana jer je lako dostuna, dobro organizovana, laka za upotrebu, a odgovarajuća primeni stabla.

4.4.3. Korišćenje interfejsa

40

Nakon pokretanja programa otvara se interfejs za rad sa programom (Slika 8).



Slika : Interfejs programa

Interfejs se sastoji od tri stranice:

- Podaci – stranica koja predstavlja kontrolnu tablu za upravljanje programom;
- Iscrtavanje – stranica za vizuelno prikazivanje podataka;
- Ispis – stranica za ispisivanje rezultata akcija. Pre svega služi za ispisivanje rezultata pretrage, ali se tu ispisuju i poruke korisniku o uspešnosti akcije koju je pokrenuo.

Prilikom pokretanja programa, otvara se interfejs na stranici pod nazivom Podaci.

Sa leve strane interfejsa na stranici Podaci nalazi se deo sa poljima za unos potrebnih podataka vezanih za sklonište.

Odsek za unos institucija sastoji se od dve sekcije:

1) Podaci o x i y koordinatama zgrade.

41

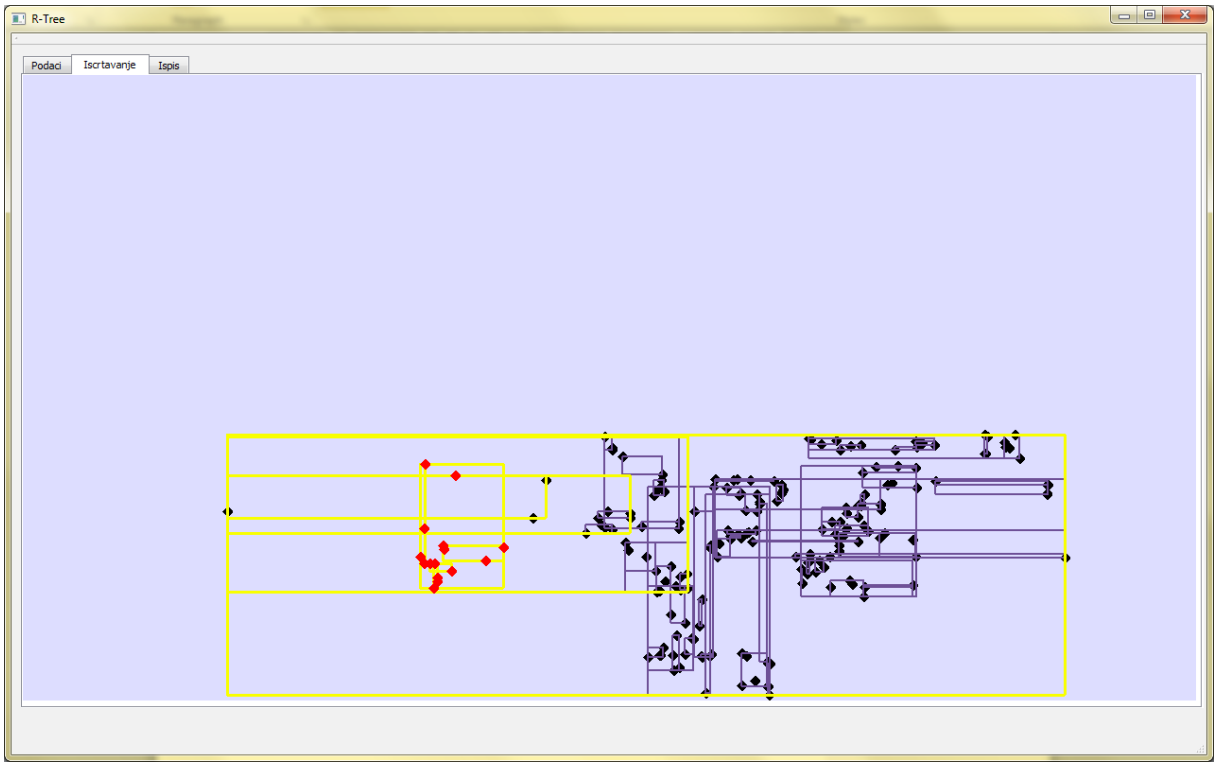
2) Podaci o samoj instituciji. Redom: jedinstveni identifikator, ime, tip, adresa, grad, država, poštanski kod, okrug, ime osobe koja se može kontaktirati, kontakt telefon.

Sa desne strane nalaze se kontrole vezane za stablo. Ovaj deo se sastoji od četiri sekcije:

1) Podaci o čvoru je sekcija koja sadrži dva polja za unos, celobrojnog tipa. Ova dva polja: m i M označavaju parametre m i M objašnjene u glavi 3.6. Po pokretanju programa parametri su namešteni na m=2, M=5, ali se mogu promeniti. Prilikom unosa prvog podatka u stablo, menjanje ovih parametara se na dalje onemogućava, jer ti parametri moraju ostati jedinstveni za sve čvorove u stablu. Kada se celo stablo obriše menjanje ovih parametara ponovo postaje moguće sve do početka kreiranja nekog novog stabla.
2) Pretraga sadrži četiri polja za unos decimalnih brojeva i odnose se na odabir MBR koji se pretražuje. Kako je rečeno u tački 3.1 MBR je definisan:
   a) x i y koordinatama svog donjeg levog temena, što se može se odrediti u poljima x dole levo, y dole levo;
   b) x i y koordinatama svog gornjeg desnog temena, što se može odrediti u poljima x gore desno, y gore desno.
3) Sekcija Akcije sadrži pet dugmeta kojima se bira željena akcija:
   a) Dugme „Unesi“ unosi u stablo par: podatak unešen u prethodnoj sekciji u polje za podatak i MBR koji se asocira sa unetim podatkom, a koji korisnik treba da definiše u prethodnoj sekciji određenoj za MBR;
   b) Dugme „Unesi iz fajla“ otvara dijalog kojim korisnik može da učita fajl iz kog će se podaci i njihovi MBR učitati. Izgled takvog fajla biće opisan kasnije;
   c) Dugme „Obriši“ briše podatak odabran u prethodnoj sekciji iz drveta. Da bi podatak koji treba da se obriše mogao lakše da se pronađe poželjno je uneti u prethodnoj sekciji MBR koji će da usmerava protragu, odnosno MBR koji se seče sa MBR podatka koji se briše;
   d) Dugme „Pretraži“ očekuje da korisnik u prethodnoj sekciji odabere MBR koji želi da se pretraži u stablu. Kao rezultat pretrage u tekstualno polje koje se nalazi u sledećoj sekciji kontrolne table ispisuje sve podatke čiji MBR se seku sa zadatim.
   e) Dugme „Obriši stabo“ briše do tada napravljeno stablo i resetuje polja za unos na interfejsu.
4) Poslednja sekcija sadrži samo jedno polje: „Iscrtavanje“. Ukoliko je ovo polje označeno, svaka akcija koja se pokrene nad stablom rezultovaće i iscrtavanjem rezultata akcije. Inicijalo, ovo polje nije označeno i preporučuje se da se ne označi ukoliko se akcije vrše nad velikim brojem podataka jer će to znatno usporiti program.

42

Druga stranica rezervisana je za iscrtavanje. Tu se, ukoliko je na kontrolnoj tabli omogućena opcija Iscrtavanje, vizualno prikazuju rezultati akcija vršenih nad stablom. Slika 9 prikazuje izgled ove stranice nakon pretrage.



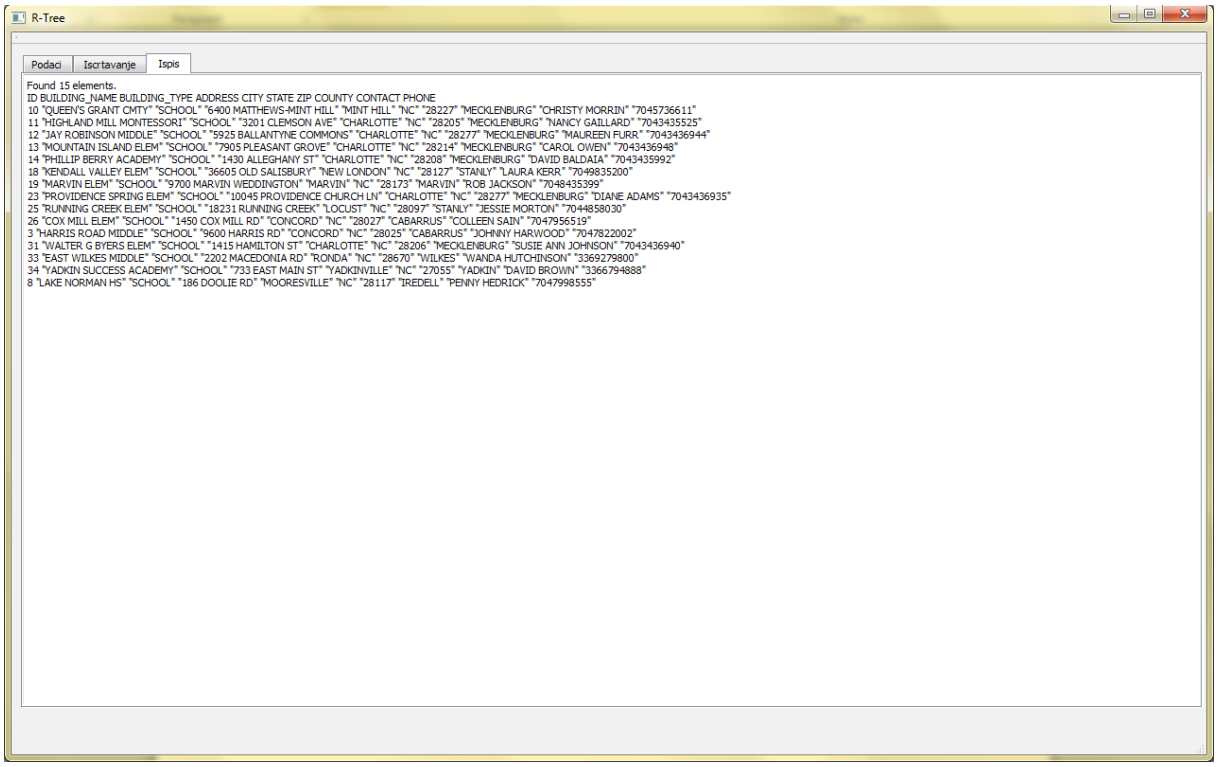
Slika : Ilustracija stranice Iscrtavanje, nakon pretrage nad podacima iz baze podataka skloništa u Severnoj Karolini

Kvadratići crne boje predstavljaju MBR koji su asocirani sa konkretnim podatkom u stablu. Pravougaonici oivičeni tamno ljubičastom bojom providne unutrašnjosti označavaju MBR čvorova stabla.

Prilikom, unosa, pretrage i brisanja žutom bojom se oivičavaju MBR čvorova koji su bili posećeni u procesu. Crvenom bojom označeni su podaci koji su učestvovali u pokrenutoj akciji.

Treća stranica programa služi za ispis. Prilikom unošenja i brisanja ovde se ispisuju poruke korisniku, a prilikom pretrage, ovde se mogu videti informacije o skloništima koja odgovaraju korisnikovom upitu. Na slici 10 može se videti rezultat pretrage, koja odgovara pretrazi prikazanoj na slici 9.

43



Slika : Ilustracija sekcije za ispis nakon pretrage

4.4.4. Izgled fajla sa podacima

Kao što je u prethodnoj glavi rečeno program nudi učitavanje podataka iz fajla. Taj fajl treba da sadrži podatke razdvojene zarezima. Redosled podataka je veoma važan i treba da bude: jedinstveni identifikator, ime, tip, adresa, grad, država, poštanski kod, okrug, ime osobe koja se može kontaktirati, kontakt telefon. Na kraju idu x, zatim y koordinate. Ove koordinate treba da budu okružene znacima navoda i da se decimalno deo od celog dela razdvaja zarezom. Razlog za ovako nekonvencionalno učitavanje koordinata je taj što se izvozom iz baze podataka dobio ovakav fajl, pa je program morao tome da se prilagodi.

Primer:

```
23,"PROVIDENCE SPRING ELEM","SCHOOL","10045 PROVIDENCE CHURCH
LN","CHARLOTTE","NC","28277","MECKLENBURG","DIANE
ADAMS","7043436935","447445,172549","150930,167502125"
```


44

Svaki novi unos od prethodnog treba da bude razdvojen novim redom.

Ne sme da bude nedostajućeg podatka. Ukoliko podatak ne postoji treba ga pisati kao NA (eng. Not Accessible).

5. Zaključak

U ovom radu predstavljena je implementacija R-stabla koja može da čuva i pretražuje zgrade od društveno korisnog značaja po geografskim koordinatama.

Program sadrži deo u kome se vizuelno prikazuje prostorni raspored podataka i MBR svakog čvora u stablu. Pri operacijama sa stablom dinamički se prikazuje i aktivnost čvorova pri zadatoj naredbi. Jedan mogući pravac daljeg razvoja ovog rada bio bi dodavanje vizuelnog prikaza stabla koji bi se paralelno menjao i označavao sa već postojećim prikazom podataka. U tom slučaju ovaj rad bi mogao da posluži kao pogodan alat za edukaciju. Korisnik bi jasno mogao da vidi šta se u kom trenutku i kako dešava u stablu što bi znatno olakšalo razumevanje funkcionisanja R-stabla. Ovakav alat bi bio veoma koristan, jer je R-stablo veoma složena struktura, koju je teško razumeti i nekome ko nema znanja vezano za strukture podataka, ali i za onoga ko je navikao na klasična stabla koja rade sa jednodimenzionim podacima.

Drugi mogući pravac daljeg rada može da bude promena tipa podataka koji se čuvaju u stablu, dok god ti podaci između ostalog sadrže i informaciju o lokaciji u nekom dvodimenzionalnom prostoru, koji ne mora da bude prostor u geografskom smislu. Ovde je predstavljena konkretna varijanta programa koja služi za pretraživanje zgrada i program je testiran nad bazom podataka skloništa u Severnoj Karolini. Ovi podaci su odabrani, jer jasno ilustruju svrhu R-stabla. Uz ovakve izmene, program bi mogao da posluži kao brz i dobar alat za pretraživanje bilo kakvih podataka u dve dimenzije.

45

6. Literatura

[1] D. Comer, „The ubiquitous B-tree,“ *ACM computing surveys*, 1979.

[2] L. Bentle / H. Friedman, „Data structures for range searching,“ *Computing surveys*, 1979.

[3] Y. Manolopoulos, A. Nanopoulos, N. Papadopoulos / Y. Theodoridis, „R-trees have grown everywhere,“ *ACM computing surveys*, 2002.

[4] A. Guttman, „R-trees a dynamic index structure for spatial searching,“ *ACM computing surveys*, 1984.

[5] G. Pavlović Lažetić, „http://poincare.matf.bg.ac.rs/~gordana//projektovanjeBP/B+stablo.pdf,“.

[6] A. Yzelman, „R-trees an efficient structure for spatial data management,“ 2007.

[7] A. Kemper, „R-tree,“ *Algorithms and datastructures for database systems*, 2003.

[8] Y. Manolopoulos, A. Nanopoulos, A. Papadopoulos / T. Y., R-trees:theory and applications, Thessaloniki: Springer, 2005.

46