Analiza tehnika za reorganizaciju samopodešavajućih stabala

Svetlana Štrbac-Savić
Visoka škola elektrotehnike i računarstva strukovnih studija
Beograd, Srbija
cecasmail@yahoo.com

Milo Tomašević
Elektrotehnički fakultet
Beograd, Srbija
mvt@etf.rs

Sadržaj—**U ovom radu se razmatra jedna posebna vrsta stabala binarnog pretraživanja – samopodešavajuća ili *splay* stabla. Ona su izrazito pogodna u slučajevima povećane vremenske lokalnosti pristupa i vrše rekonstrukciju stabla pri svakoj operaciji. Predmet ovog rada je analiza operacija za održavanje ovog stabla koje se nazivaju *splaying* tehnikama. Detaljno su opisane i upoređene dve osnovne *splaying* tehike: odozdo na gore i odozgo na dole. Navedene su i neke modifikacije *splaying* tehnika koje su predložene u cilju optimizacije performansi, pre svega smanjenja broja rotacija u odnosu na osnovne tehnike. Na kraju, data je i procena performansi *splay* stabala korišćenjem postupka amortizovane analize, a navedeni su i neki rezultati eksperimentalne komparativne evaluacione analize ovih stabala sa drugim stabima binarnog pretraživanja.**

Ključne reči- stabla binarnog pretraživanja; splay stabla; splaying tehnike; (key words)

I. UVOD

Stabla binarnog pretraživanja predstavljaju jednu od osnovnih struktura za pretraživanje u operativnoj memoriji [1]. Postoji više vrsta ovih stabala sa ciljem da se postigne što manja visina i što bolja balansiranost, koja garantuje logaritamsku složenost operacija. Za razliku od njih postoje i stabla binarnog pretraživanja kod kojih nije cilj optimalni balans, već se struktura podršava tako da se maksimalno iskoristi vremenska lokalnost koja je jedna od osnovnih karakteristika u pristupu podacima. To su samopodešavajuća ili *splay* stabla koja su predložili Sleator i Trajan [2]. Ova stabla se reorganizuju pri svakoj operaciji, pa i čak i onim neinvazivnim kao što je pretraživanje, da bi se obezbedio brz pristup skoro pristupanim ključevima. Logika ove reorganizacije je da se čvorovi kojima se češće pristupa dovode bliže korenu stabla. Ovo ima efekta kod uniformnog pristupa ključevima, npr. kada se najčešće pretražuje neki opseg vrednosti, što je u praksi izraženo za veliki broj primena.

Tehnike kojima se vrši reorganizacija stabla nazivaju se *splaying* tehnikama. U ovom radu će u drugoj i trećoj sekciji biti detaljno opisane dve osnovne *splaying* tehnike, i to odozgo na dole i odzdo na gore. Zatim će u četvrtoj sekciji biti opisane modifikovane tehnike, koje se primenjuju u cilju smanjenja broja rotacija, a samim tim i poboljšanjem performansi operacija. Na kraju će u petoj sekciji biti detaljno analizirana performansa *splay* stabala u vidu vremenske složenosti operacija korišćenjem jednog posebnog postupka, amortizovane analize složenosti.

II. *SPLAYING* ODOZDO NA GORE

Prvo će biti opisan postupak u kome se rotacije vrše od listova ka korenu stabla po putu pristupa, tj. metoda odozdo na gore. Prednost ovog metoda je u olakšanoj realizaciji u odnosu na metodu *splayinga* odozgo na dole.

Pre samog *splaying-a* mora se prvo naći željeni čvor, a to se ostvaruje metodom pretraživanja za binarno stablo pretraživanja.

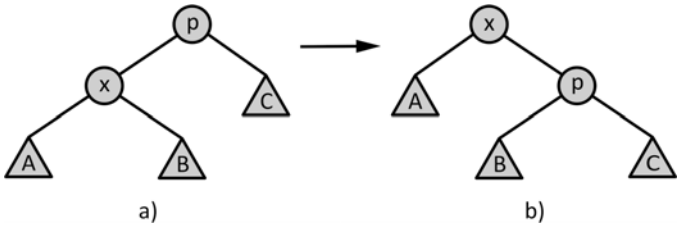
Neka je *x* čvor (ali ne koren) na putu po kom se vrši rotacija i neka je *p* roditelj čvora *x,* a *g* roditelj čvora *p*. Postoje tri moguća koraka:

1) Ako je roditelj čvora *x* tj. *p* koren stabla, onda se samo vrši odgovarajuća (leva ili desna) rotacija ova dva čvora. Ova operacija se naziva **cik** i grafički je prikazana na Sl. 1. Ovaj korak je isti kao jednostruka rotacija koja se koristi na primer kod AVL stabala [3].

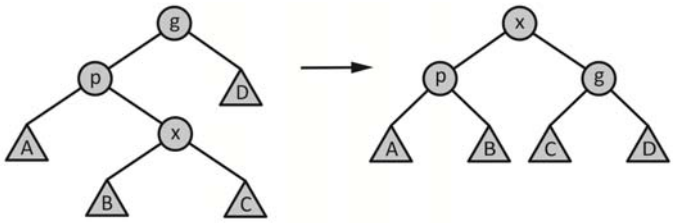
2) *x* je desni potomak *p*, a *p* levi potomak *g* ili obrnuto. Ovde se primenjuje dvostruka rotacija. Ova operacija se naziva ***cik-cak*** i prikazana je na Sl. 2.

3) *x* i *p* su oba levi potomci, ili oba desni potomci svojih roditelja (*p* i *g*). Ovde se vrši transformacija stabla prikazana na Sl. 3. tj. dve jednostruke rotacije u istu stranu - u ovom slučaju u desno. Ova operacija se naziva ***cik-cik***.

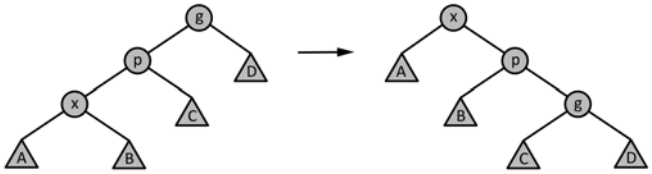
Splaying odozdo na gore se obavlja tako što se prvo nađe tražena vrednost ključa ili mesto na kom bi trebala ona da bude posle umetanja. Primenom opisanih koraka traženi čvor se dovodi u koren stabla, ako je pretraga bila uspešna. Ako ključ nije nađen, poslednji čvor (ne null) kome je pristupljeno pre završenja pretrage se dovodi u koren stabla.



Slika 1. Cik korak



Slika 2. Cik-cak korak



Slika 3. Cik-cik korak

Opisana tehnika se koristi pri operaciji pretraživanja, što znači da se pretraživanje vrši tako što je ili željena vrednost u korenu stabla ili nije nađena. Što se tiče operacija umetanja i brisanja, one počinju operacijom pretraživanja za zadatu vrednost. Kada je u pitanju umetanje, ako je nađena tražena vrednost, operacija se završava, a ako ne, željena vrednost se umeće u koren stabla posle neuspešne pretrage. Operacija uklanjanja čvora iz stabla se obavlja tako što se čvoru koji treba ukloniti iz stabla T prvo koristeći operaciju pretraživanja zasnovanu na *splaying* tehnici. Ova operacija postavlja željeni čvor u koren stabla T. Brisanjem čvora u korenu stabla, dobijamo dva podstabla T_L i T_D (levo i desno podstablo). U narednom koraku se pronalazi maksimalni element podstabla T_L i ovaj element se nizom odgovarajućih rotacija dovodi u koren podstabla T_L. Ovaj element nema desnog potomka. Operacija brisanja će biti završena tako što će T_D postati desni potomak čvora u korenu podstabla T_L. Postupak brisanja može se obaviti i simetrično - posle nađenog željenog čvora pronalazi se minimum desnog podstabla, pa taj minimum postaje novi koren, a njegovo levo podstablo postaje T_L [4].

III. 4. SPLAYING ODOZGO NA DOLE

Ukratko će biti opisan i postupak tehnika *splaying* koji se vrši od korena ka listovima, jer je u nekim primenama ovaj postupak ocenjen kao efikasniji. Prednost ove tehnike je što zahteva samo jedan obilazak stabla pri pretraživanju i reorganizaciji stabla. U ovom postupku se koriste dva pomoćna stable: levo L i desno D. Ova podstabla su prazna na početku, a tokom *splayinga* se odgovarajući ključevi ili podstabla spajaju sa odgovarajućim stablom L ili D, dobijenim u prethodnom koraku.

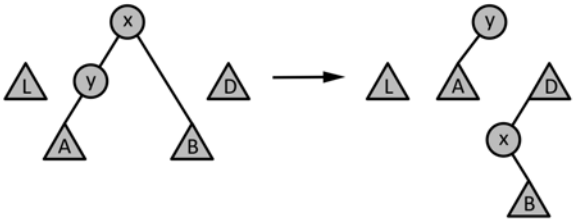
Neka je X trenutni koren stabla, Y ključ kome smo takođe pristupili i neka su inicijalizovana dva prazna pomoćna stabla: levo L i desno D. Stablo se obilazi tako što se pristup vrši paralelno u dva čvora na putu pristupa. U zavisnosti od vrednosti ključeva čvorova kojima smo pristupili postoje tri situacije:

1) Ako je Y traženi ključ, levo (desno) dete od X, tada se X i njegovo desno (levo) podstablo, dodaju u pomoćno stablo D (L), kao najlevlje (najdesnije) podstablo, Sl. 4.

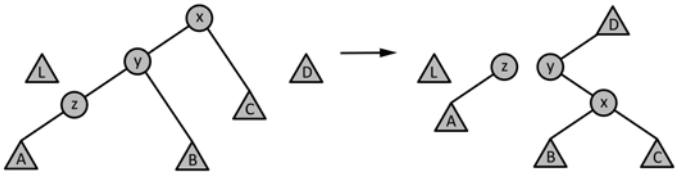
2) Ako je traženi čvor levo-levo (desno-desno) od X, tada se prvo izvodi rotacija u desno (levo) oko čvora Y, pa se podstablo Y ubacuje u R (L), kao najlevlje podstablo, Sl. 5.

3) Ako je traženi čvor levo-desno (desno-levo) od X, onda se primenjuju sledeća dva koraka. Prvo se X i njegovo desno (levo) podstablo nadovežu na R (L), a onda se srednjem stablu, vrši rotacija ulevo (udesno) i čvor Z postaje koren, Sl. 6.

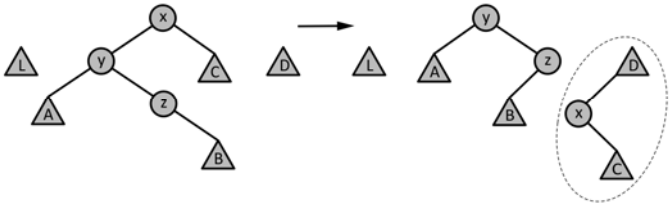
Na kraju se vrši spajanje pomoćnih stabala sa središnjim stablom čiji je ključ u korenu tražena vrednost Sl. 7, tako što se trenutno levo i desno podstablo korena središnjeg stabla spajaju sa L i D kao njihova respektivno, najdesnija, tj najlevlja stabla, a zatim se L i D spajaju sa korenom. Detaljan opis ove tehnike dat je u [2].



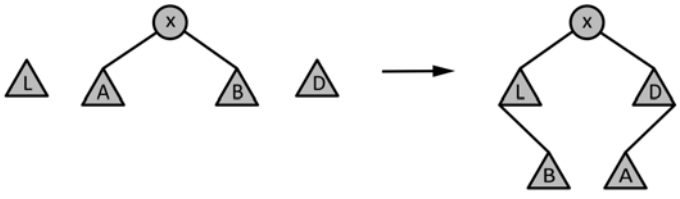
Slika 4. *Splaying* odozgo na dole – situacija 1.



Slika 5. *Splaying* odozgo na dole – situacija 2



Slika 6. *Splaying* odozgo na dole – situacija 3



Slika 7. Spajanje glavnog sa pomoćnim podstablima

IV. DRUGE TEHNIKE SPLAYINGA

Sami autori [2] favorizuju tehniku odozgo na dole, zato što nema potrebe da se pamte adrese roditeljskih čvorova, pa je prostorna složenost kod ove tehnike optimalnija, i zato što se celi proces izvršavanja operacije i reorganizacije vrši u jednom prolazu. Oni ovu tehniku smatraju efikasnijom i u smislu vremenske složenosti, što nije tačno za neke primene, a to je pokazano u radu [5] za testove rađene nad velikim tekstualnim kolekcijama.

Pored osnovnih *splaying* tehnika opisanih u prethodnom delu rada, predložene su i druge tehnike u cilju optimizacije performansi *splay* stabala. Pre svega, cilj kod svih predloženih rešenja je da se smanji broj rotacija i na taj način smanji cena operacija, a da se, po prestanku reorganizacije stabla, ono ostavi u što optimalnijoj topologiji u smislu cene pristupa čvorovima koji se često traže. Ovde će biti navedene tri tehnike.

Prvu su predložili još Sleator i Tarjan u [2] i ta tehnika se naziva polu-podešavajuća tehnika. Ovde se uvodi vrednost *m* koja predstavlja broj operacija koje se izvršavaju po klasičnom *splaying-u*. Tada se prekida sa reorganizacijom stabla, jer se smatra da je u *m* koraka stablo reorganizovano tako da češće korišćene vrednosti nalaze pri vrhu. Što je *m* veće, to je ova tehnika sličnija *splaying-u*, a što je *m* manje to se ova tehnika približava standardnom stablu binarnog pretraživanja.

Druga modifikacija je polu-splaying. Ovu ideju detaljno objašnjava Furer [6]. On predlaže da se vrednost kojoj se pristupa ne dovodi u koren stabla već da se ona pomeri samo za neki nivo, na primer za jedan nivo ili pola ukupnog puta, itd. Što je veći broj nivoa koji čvor menja, to je ovo tehnika približnija klasičnom *splaying-u*.

Treća tehnika uvodi brojač pristupa čvoru za svaki čvor i *splaying* se vrši dok se ne dostigne neka, unapred zadata, vrednost brojača. Ovo je slična ideja kao kod *Randomized* stabala [7], samo što nema prioriteta, već se uvodi brojač pristupa. Navedena tehnika je detaljno opisana u [5] .

V. ANALIZA PERFORMANSI

Splay stabla se ne zasnivaju ni na kakvom visinskom balansiranju koje ograničava najgori slučaj, pa u najgorem slučaju mogu da imaju linearnu vremensku složenost *O(n)*. To može biti prihvatljivo samo ako se ovaj, najgori slučaj, događa retko. Međutim, ako se posmatra duga sekvenca operacija nad jednim *splay* stablom, pogotovo u slučaju vremenske lokalnosti pristupa ključevima, vremenska složenost čak i najgoreg slučaja je dobra i reda *O(log n)*, što će u nastavku biti analitički dokazano.

Amortizovana analiza daje procenu cene najgoreg slučaja duže sekvence događaja. Za razliku od analize vremenske složenosti algoritma najgoreg slučaja, gde se posmatra jedna operacija ili jedna situacija i posmatra se koliko posla je potrebno da se ona izvrši, kod amortizovane analize uzima se u obzir dugačka sekvenca događaja, a ne izolovana pojedinačna operacija.

Prosečna vremenska složenost se, takođe, razlikuje od amortizovane analize jer se kod nje posmatra prosečna složenost u odnosu na sve moguće ulaze, kao i da je analiza svakog događaja nezavisna u odnosu na naredni ulaz. Kod amortizovane analize ovi događaji su uzajamno zavisni. Tako da jedan događaj u sekvenci može uticati na cenu narednih događaja. Može se desiti da jedna operacija bude složena i skupa za izvođenje, ali da ostavlja strukturu podataka, nad kojom se operacije vrše, u stanju u kom je naredna operacija mnogo kraća.

Sledi dokaz amortizovane analize *splay* stabala. U tom cilju biće prvo date neophodne definicije.

Definicija 1

Za svaki korak *i splaying* procesa i svaki čvor *x* u *T*, rang u *i*-tom koraku se definiše na sledeći način:

$$r_i(x) = \lg\left|T_i(x)\right|.$$

Ova rang funkcija se, otprilike, ponaša kao idealna visina. Ona zavisi od broja čvorova u podstablu čiji je koren *x*, a ne od njegove visine i pokazuje kolika bi visina bila kad bi stablo bilo balansirano.

Ako je *x* list, onda $\left|T_i(x)\right|$=1, pa $r_i(x) = 0$. Koren stabla ima najveći rang u stablu.

Količina posla koju algoritam mora da odradi da bi se pristupilo čvoru ili ubacio čvor u podstablo zavisi od visine podstabla, pa se iz tog razloga rangira koren podstabla. Kreditna funkcija će biti tako definisana tako da velika i visoka stabla imaju velike vrednosti kreditne funkcije, a niska i mala stabla manje vrednosti, jer se količina rada *splaying* tehnikom povećava sa visinom stabla.

Definicija 2

Za svaki čvor *x*∈*T* i posle svakog koraka *i splaying* postupka, čvor *x* ima kredit jednak vrednosti rang funkcije u *x* tj. $r_i(x)$.

Vrednost kredita stabla je jednaka zbiru kredita pojedinačnih čvorova stabla

$$c_i = \sum_{x \in T_i} r_i(x).$$

Ako je stablo prazno ili sadrži samo jedan čvor, onda je njegov kredit 0. Sa porastom broja čvorova u stablu, povećava se i kredit stabla.

Definicija 3

Amortizovana kompleksnost a_i koraka *i splaying* postupka definiše se na sledeći način

$$a_i = t_i + c_i - c_{i-1},$$

pri čemu je t_i stvarna cena *i*-tog koraka, dok je c_i-c_{i-1} promena kredita u *i*-tom koraku. [8]

Lema 1

Neka su α, β *i* γ pozitivni realni brojevi za koje važi $\alpha + \beta \le \gamma$, tada je

$$\log_2 \alpha + \log_2 \beta \le 2\log_2 \gamma - 2$$

Dokaz:

Kako je

$$\left(\sqrt{\alpha} + \sqrt{\beta}\right)^2 \ge 0$$
$$\left(\alpha - 2\sqrt{\alpha\beta} + \beta\right) \ge 0$$
$$\sqrt{\alpha\beta} \le \frac{\alpha + \beta}{2}$$

Na osnovu pretpostavke $\alpha + \beta \le \gamma$, dobija se

$$\sqrt{\alpha\beta} = \frac{\gamma}{2}$$

Kvadriranjem i logaritmovanjem tako dobijene nejednačine dobija se

$$\log_2 \alpha + \log_2 \beta \le 2\log_2 \gamma - 2$$

što je i trebalo dokazati.

Lema 2

Ako je *i*-ti korak *splaying* procesa **cik-cik** ili **cak-cak** korak u čvoru *x*, tada njegova amortizovana složenost a_i zadovoljava sledeću nejednakost

$$a_i < 3\left(r_i(x) - r_{i-1}(x)\right).$$

Dokaz:

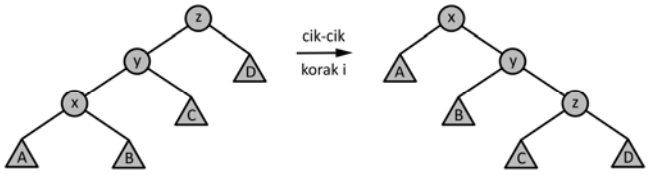
Posmatrajmo slučaj ilustrovan na Sl. 8.

Stvarna složenost dvostruke rotacije ulevo i udesno su dve jedinice i samo se visine podstabala čiji su koreni *x*, *y* i *z* menjaju ovim korakom. Tako se svi c_i potiru sa c_{i-1} sem ovih u sledećoj jednačini:

$$a_i = t_i + c_i - c_{i-1}$$

$$= 2 + r_i(x) + r_i(y) + r_i(z) - r_{i-1}(x) - r_{i-1}(y) - r_{i-1}(z)$$
$$= 2 + r_i(y) + r_i(z) - r_{i-1}(x) - r_{i-1}(y) \qquad (1)$$

$r_i(x)=r_{i-1}(z)$, jer imaju isti broj grana u podstablu.



Slika 8. *i*-ti korak splayinga je cik-cik

Neka je $\alpha = \left|T_{i-1}(z)\right|$, $\beta = \left|T_i(z)\right|$ i $\gamma = \left|T_i(x)\right|$.

Sa Sl. 8 ovog slučaja uočava se da $T_{i-1}(x)$ sadrži komponente *x, A* i *B*; $T_i(z)$ sadrži komponente *z*, *C* i *D*; $T_i(x)$ isto sadrži sve ove komponente (i komponentu *y*).

Kako je $\alpha+\beta\le\gamma$, iz leme 1 sledi

$$r_{i-1}(x) + r_i(z) \le 2r_i(x) - 2 \text{, tj.}$$
$$2r_i(x) - r_{i-1}(x) - r_i(z) \ge 0.$$

Dodavanjem poslednje nenegativne vrednosti prethodne nejednačine jednačini (1) za a_i dobija se:

$$a_i \le 2r_i(x) - 2r_{i-1}(x) + r_i(y) - r_{i-1}(y).$$

Pre koraka *i*, *y* je roditelj *x*, pa važi $\left|T_i(y)\right| > \left|T_{i-1}(x)\right|$. Posle koraka *i*, *x* postaje roditelj *y*, pa je $\left|T_i(x)\right| > \left|T_i(y)\right|$. Logaritmovanjem ovih nejednačina dobija se $r_{i-1}(y)>r_{i-1}(x)$ i se $r_i(y)>r_i(x)$.

Lema 3

Ako je *i*-ti korak *splaying* procesa **cik-cak** ili **cak-cik** korak u čvoru x, tada njegova amortizovana složenost a_i zadovoljava sledeću nejednakost

$$a_i < 2\left(r_i(x) - r_{i-1}(x)\right).$$

Lema 4

Ako je *i*-ti korak *splaying* procesa **cik-cak** ili **cak-cik** korak u čvoru *x*, tada njegova amotizovana složenost a_i $x \in T$ i posle svakog koraka *i splaying* postupka, čvor *x* ima kredit jednak vrednosti rang funkcije u *x* tj.

$$a_i < 1 + \left(r_i(x) - r_{i-1}(x)\right).$$

Dokaz leme 3 sličan je dokazu leme 2, a dokaz leme 4 neposredno sledi, tako da neće biti navedeni.

Teorema 1

Amorizovana cena umetanja ili pronalaženja čvora u *splay* stablu sa *n* čvorova ne prelazi *1+3ln n* koraka ka vrhu željenog čvora.

Dokaz:

Da bi odredili cenu operacije umetanja ili pronalaženja elementa, neophodno je sabrati pojedinačne cene svih *splaying* koraka prilikom obavljanja operacije. Ako je u toku operacije obavljeno *m splaying* koraka, tada najviše jedan korak (poslednji) može biti jednostruka rotacija ulevo ili jednostruka rotacija udesno (cik odnosno cak koraci), na koje se odnosi lema 4, dok svi ostali koraci zadovoljavaju granice date lemama 2 i 3.

Ukupna cena koja se sada dobija je:

$$\begin{aligned}
&\sum_{i=1}^{m} a_i = \sum_{i=1}^{m-1} a_i + a_m \\
&\le \sum_{i=1}^{m-1} \left(3r_i(x) - 3r_{i-1}(x)\right) + \left(1 + 3r_m(x) - 3r_{m-1}(x)\right) \\
&= 1 + 3r_m(x) - r_0(x) \\
&\le 1 + 3r_m(x) \\
&= 1 + 3\log_2 n.
\end{aligned}$$

Što je i trebalo dokazati.

Lema 5

Ukupna stvarna cena i ukupna amortizovana cena niza od *m* operacija nad *splay* stablom povezane su sledećom relacijom

$$\sum_{i=1}^{m} t_i = \left(\sum_{i=1}^{m} a_i\right) + c_0 - c_m$$

Dokaz:

$$\sum_{i=1}^{m} t_i = \sum_{i=1}^{m} \left(a_i + c_{i-1} - c_i\right) = \left(\sum_{i=1}^{m} a_i\right) + c_0 - c_m .$$

U poslednjoj sumi c_{i-1} i c_i se potiru i ostaju samo nulti i m-ti *c*, čime je tvrđenje dokazano.

Na kraju, biće povezane stvarna i amorizovana cena niza operacija umetanja i pronalaženja čvora. Primenom leme 5 dobija se da se stvarna cena niza *m splay* koraka razlikuje od amortizovane za samo c_0- c_m gde su c_0 i c_m stanja kredita počenog i krajnjeg stabla. Ako stablo nikada nema više od *n* čvorova, tada je kredit svakog čvora uvek između 0 i *ln n*. Iz toga sledi da je kredit stabla, kako početnog, tako i posle izvršenih *m* operacija, u intervalu 0 do *n ln n*. Iz toga sledi sledeća teorema:

Teorema 2

Ukupna složenost niza od *m* operacija umetanja ili pronalaženja čvora u *splay* stablu čiji je broj čvorova u svakom trenutku manji od *n* je

$$m\left(1 + \ln n\right) + n \ln n$$

koraka ka vrhu željenog čvora. [9]

Dokaz teoreme 2 sledi iz teoreme 1 i prethodnog razmatranja.

U teoremi 2 svaki *splay* korak se računa kao dva koraka ka vrhu, izuzev rotacije ulevo ili rotacije udesno (*cik* koraka) u poslednjem koraku koji se računaju kao jedan korak ka vrhu.

Kod ovih stabala operacija može, u najgorem slučaju, imati vremensku složenost *O(n)*, što može biti prihvatljivo samo ako se ovaj, najgori slučaj, ne događa često. Teorema 2 dokazuje da je složenost niza operacija umetanja i pretraživanja *O(*log *n)*. Iako ovde najgori slučaj nije *O(log n)* po operaciji, prednost je što nema loših ulaznih sekvenci.

Detaljna eksperimentalna analiza performansi *splay* stabala uz korišćenje pogodno izabranog sintetičkog modela radnog opterećenja izvršena je u [10]. Ona je potvrdila rezultate teorijske analize i pokazala da su u slučaju izražene vremenske lokalnosti vrednosti ova stabla efikasnija od visinski balansiranih AVL i crveno-crnih stabala. Ona su najefikasnija kada je niz ključeva po kome se vrši pretraga sortiran. U ovom slučaju njihove performanse su loše za jednu operaciju, ali kako raste broj operacija to se i njihova efikasnost u odnosu na visinski balansirana stabla povećava. *Splay* stabla su efikasnija od navedenih stabala i u slučajevima u kojima je opseg pretraživanih ključeva manji od 10% u odnosu na početno stablo bez obzira u kom intervalu se ti ključevi nalaze i nezavisno od njihovog redosleda i broja ponavljanja vrednosti. Kao i slučaju sortiranih vrednosti ključeva, i u ovom slučaju sa porastom broja operacija cena ovih operacija opada, što kod balansiranih predstavnika nije slučaj.

VI. ZAKLJUČAK

Splay stabla predstavljaju posebnu vrstu samopodešavajućih stabla binarnog pretraživanja koja za razliku od ostalih ne ograničavaju najgori slučaj, pa on može da ide i do *O(n),* što može biti prihvatljivo samo ako se to retko događa. Amortizovana analiza duge serije operacija pokazuje logaritamsku složenost što ova stabla čini konkurentnim sa skoro balansiranim stablima binarnog pretraživanja, a da pri tom za razliku od većine skoro balansiranih vrsta stabala, nemaju potrebu za čuvanjem dodatnih informacija kao što su visina kod AVL, boja kod crveno-crnih stabala, itd.

Splay stabla su pogodna za primene gde verovatnoća pristupa ključevima nije uniformna i pogotovo daju odlične rezultate kada je izražena vremenska lokalnost (npr. algoritmi keširanja, dealokacije nepotebih objekata, itd.). Ovo je posledica reorganizacije stabla koja se vrši pri svakoj operaciji, što ponekad neopravdano povećava cenu operacije, a da u amortizovanom smislu to nije opravdano.

Od dve osnovne *splaying* tehnike pogodnijom se smatra ona koja ide odozgo na dole. Predložena su i različita rešenja za optimizaciju, koja u zavisnosti od primene mogu da daju bolje rezultate.

[1] Milo Tomašević, *Algoritmi i strukture podataka,* Akademska misao, 2011.

[2] Daniel Dominic Sleator and Robert Endre Trajan, "Self–adjusting Binary Search Trees", Journal of the Association for Computing Machinery, Vol. 32, No. 3, July 1985, pp. 652 -686, 1985.

[3] M. Adelson – Velski and E. M. Landis, "An Algorithm for the Organization of Informations", Soviet Mathematics Doklady. 3: 1259-1263, 1962.

[4] Weiss Mark Allen, *Data Structures and Algorithm Analysis in C*, Addison – Westley, 1997.

[5] Hugo E. Williams, Justin Zobel and Steffen Heinz, "Self-adjusting trees in practise for large text collections", Software - Practice and Experience, pp. 925-939, 2001.

[6] M. Furer, "Randomized splay trees", In ACM-SIAM Symposium on Descrete Algoritms, pp: 903-904, Baltimore, MD, 1999.

[7] R. Seidel and C.R. Aragon, "Randomazed search trees", Algoritmica, 16:464-497, 1996.

[8] Robert Endre Tarjan, "Amortized Computational Complexity", SIAM Journal on Algebraic and Discrete Methods, Vol. 6, No. 2, April 1985, pp. 306-318, 1985.

[9] R. Cole, "On the Dynamic Finger Conjecture for Splay Trees", Part II: The Proof. SIAM Journal on Computing 30, pages 44-85, 2000.

[10] Svetlana Štrbac-Savić, "Uporedna analiza performansi skoro balansiranih stabala binarnog pretraživanja", magistarska teza, Elektrotehnički fakultet , Beogradu, 2009.

ABSTRACT

This paper considers a special kind of binary search trees – self-adjusting, splay trees. They are especially convenient in case of increased temporal locality of accesses. To this end, splay trees are reorganized on each operation, even on search. In this paper we analyze splaying techniques for tree maintenance. Two basic techniques are presented: top-down and bottom-up splaying. Some advanced splaying techniques proposed to decrease the number of rotation are also mentioned here. Finally, the paper brings the performance analysis of the splay trees, as well as some finding of experimental comparison with other kind of binary search trees.

AN ANALYSIS OF THE SPLAYING TECHNIQUES IN SELF-ADJUSTING TREES

Svetlana Štrbac-Savić
Milo Tomašević