Rotacije stabala

Seminarski rad u okviru kursa
Konstrukcija i analiza algoritama 2
Matematički fakultet

Stanojević Strahinja,
Bajić Ana

januar 2019.

Sažetak

U ovom radu je prikazan je algoritam za rotaciju binarnog pretraživačkog stabla prvo u listu a zatim, koristeći rotacije suprotne početnim u obrnutom redosledu, prevođenje te liste u neko drugo stablo. Ispitana je složenost algoritma i prikazani su dobijeni rezultati.

Ključne reči — rotacije, stabla, algoritam, C++

Sadržaj

1 Postavka problema **2**

2 Algoritam **2**
2.1 Rotacija udesno 3
2.2 Rotacija ulevo 4

3 Složenost algoritma **5**

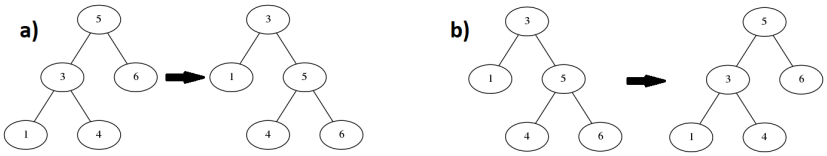
4 Vizuelizacija **6**
4.1 Format tekstualne datoteke za prikaz stabla 8
4.2 Format DOT datoteke 8
4.3 Prikaz pomoću Tkinter biblioteke 9

Literatura **9**

1 Postavka problema

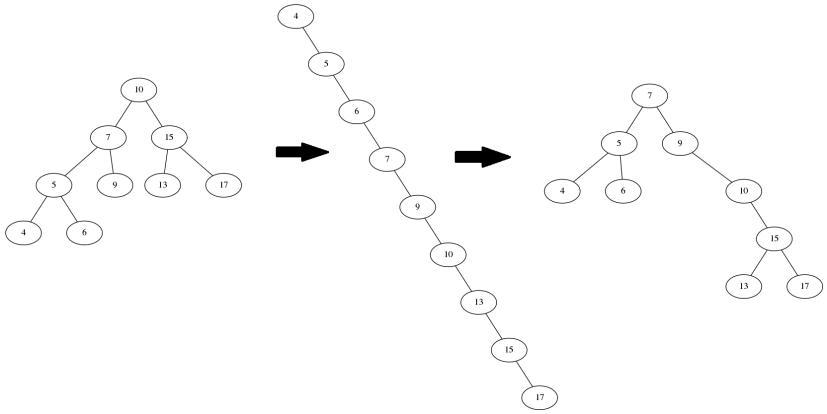
Problem je postavljen na sledeći način:

"Potrebno je proizvoljno binarno pretraživačko stablo T_1 prevesti u drugo proizvoljno binarno pretraživačko stablo T_2 koristeći dve vrste rotacija. Dozvoljene rotacije su prikazane na slici 1."



Slika 1: Dve vrste dozvoljenih rotacija nad stablom. a) Rotacija udesno. b) Rotacija ulevo.

Na slici 2 su prikazani glavni koraci algoritma - primena rotacija dok se stablo ne prevede u listu (tj. u stablo takvo da svaki njegov čvor ima samo desnog sina) a onda primena rotacija dok se ne dovede u formu drugog stabla. U daljem tekstu ćemo nekad pod rotacijom podrazumevati jediničnu rotaciju (sa slike 1) a nekad rotaciju celog stabla (koja predstavlja niz jediničnih rotacija).



Slika 2: Rotacija stabla (a) T_1 u (b) listu a zatim u stablo (c) T_2

2 Algoritam

Algoritam se sastoji iz narednih koraka:

1. Rotacija stabla T_1 udesno kako bi se dobila lista
2. Rotacija liste ulevo kako bi se dobilo stablo T_2

Ovi koraci će biti detaljno objašnjeni u narednim odeljcima, uz propratni kod.

Algoritam je implementiran u programskom jeziku C++ (preciznije, u C++11), u okruženju Qt. Vizuelizacija je urađena u programskom jeziku Python (više o implementaciji i korišćenim bibliotekama u odeljku 4).

2.1 Rotacija udesno

Na slici 3 je prikazan kod za rotaciju stabla udesno, dok se ne dođe do liste (tj. stabla čiji čvorovi imaju samo desnog sina).

Ideja je da se primenjuje rotacija a) sa slike 1 sve dok postoje levi sinovi. Na slici 4 se vidi implementacija ove rotacije. Proverava se postojanje levog sina (dalje ***LS***) i ako on postoji, podiže se za jedan nivo ka korenu stabla (njegov otac (dalje ***OS***) mu postaje desni sin). Levi sin čvora ***LS*** ostaje nepromenjen, a ***OS*** dobija novog levog sina - prethodnog desnog sina čvora ***LS***.

Zatim se zalazi u čvorove koji i dalje imaju leve sinove i oni se razrešavaju na isti način, sve dok se ne stigne do listova.

Naredbe *writeToFile()* ispisuju stablo u tekstualnu datoteku svaki put kad se desi rotacija nekog čvora oko roditelja. Format ispisa je detaljno opisan u odeljku 4.1

```
void Tree::rotateRight()
{
    while (root->left) {
        rotateRightOnce(&root);
        writeToFile();
    }
    Node *rootPom1 = root;
    Node *rootPom2 = root->right;

    while (rootPom1->right) {
        while (rootPom2 && rootPom2->left) {
            rotateRightOnce(&rootPom2);
        }
        rootPom1->right = rootPom2;
        rootPom1 = rootPom1->right;
        if (rootPom2)
            rootPom2 = rootPom2->right;
        writeToFile();
    }
}
```

Slika 3: Kod za rotaciju stabla udesno do liste.

```
void Tree::rotateRightOnce(Node **node)
{
    if ((*node)->left) {
        Node *tmp = (*node)->right;
        Node *nodeTmp = (*node);
        Node *tmpRight = (*node)->left->right;
        (*node) = (*node)->left;
        (*node)->right = nodeTmp;
        nodeTmp->right = tmp;
        nodeTmp->left = tmpRight;
    }
}
```

Slika 4: Kod za rotaciju čvorova udesno.

2.2 Rotacija ulevo

Nakon što je početno stablo prevedeno u listu, tj. dovedeno do stabla čiji čvorovi imaju samo desnog sina, potrebno je tu listu rotirati ulevo tako da se dođe do traženog stabla.

Na slici 5 je prikazan kod za izvođenje ove transformacije. Počinje se od korena traženog stabla i zatim se rekurzivno nastavlja obilazak levog i desnog podstabla. Ako čvor (u daljem tekstu $\boldsymbol{U}$) u traženom stablu ima levog sina, potrebno je izvršiti rotaciju liste ulevo. Kao što se sa dela b) slike 1 vidi, rotacija ulevo se vrši tako što se pronađe čvor $\boldsymbol{U}$ u listi (označimo taj pronađeni čvor sa $\boldsymbol{V}$) a zatim se pronađu svi njegovi preci. Novi koren liste postaje $\boldsymbol{V}$, njegov levi sin postaje podstablo predaka a desni ostaje ostatak liste. Ovaj proces se nastavlja sve dok se ne dobije traženo stablo.

Naredba *writeToFile()* se poziva nakon svake rotacije čvora oko roditelja i u ovom slučaju. Više o ispisu će biti reči u odeljku 4.1

```
void Tree::rotateLeft(Tree &t2)
{
    Node *node2 = t2.root;
    rotateLeftOnce(&root, node2);
}

void Tree::rotateLeftOnce(Node **root, Node *root2)
{
    if (!root2)
        return ;

    if (root2->left) {
        Node *pom = findNode(*root, root2->number);
        Node *pom2 = findLeftSubTree(*root, root2->number);
        *root = pom;
        (*root)->left = pom2;
        writeToFile();
    }
    rotateLeftOnce(&(*root)->left, root2->left);
    rotateLeftOnce(&(*root)->right, root2->right);
}
```

Slika 5: Kod za rotaciju stabla ulevo.

```
Node *Tree::findNode(Node *root, int number)
{
    while (root) {
        if (root->number == number)
            return root;
        root = root->right;
    }

    return nullptr;
}

Node *Tree::findLeftSubTree(Node *root, int number)
{
    Node *pom = root;
    while (root->right) {
        if (root->right->number == number) {
            root->right = nullptr;
            break;
        }
        root = root->right;
    }
    return pom;
}
```

Slika 6: Pomoćne funkcije za rotaciju stabla ulevo.

3 Složenost algoritma

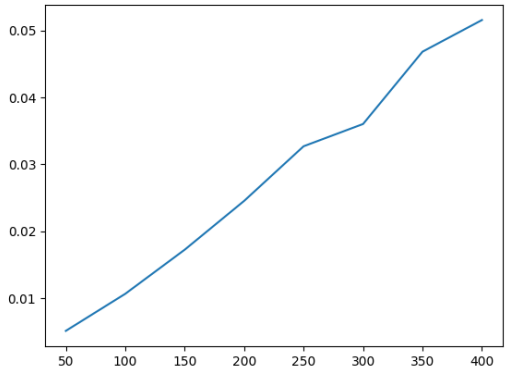
Analizom koda možemo primetiti par jednostrukih petlji koje prolaze kroz čvorove stabla, te su očigledno linearne. Jedino problematično parče koda može biti dvostruka petlja sa slike 3. Međutim, može se zaključiti da se i na tom mestu prolazi samo linearan broj čvorova - svaki put kad se izvrši rotacija levog sina oko oca, smanjuje se broj levih sinova, tako da se ne može desiti da se prođe više od N čvorova, gde N predstavlja ukupan broj čvorova u stablu.

Radi merenja složenosti algoritma, stabla su generisana slučajno - za stablo od N čvorova, slučajno su generisani brojevi iz opsega [1, N]. Za svako N generisano je po hiljadu parova stabala (početno i krajnje). U tabeli 1 su prikazani podaci o vremenu koje je potrebno algoritmu da transformiše jedno slučajno generisano stablo u drugo, takođe slučajno generisano, stablo.

N	ms
50	0.005168
100	0.010696
150	0.017277
200	0.024561
250	0.032709
300	0.036031
350	0.046821
400	0.051543

Tabela 1: Vreme izvršavanja u odnosu na broj čvorova.

Složenost algoritma je linearna po broju čvorova, što se može videti i sa grafika na slici 7. U najviše $\boldsymbol{N}$ rotacija se dolazi do liste a u najviše isto toliko rotacija se vraća nazad.



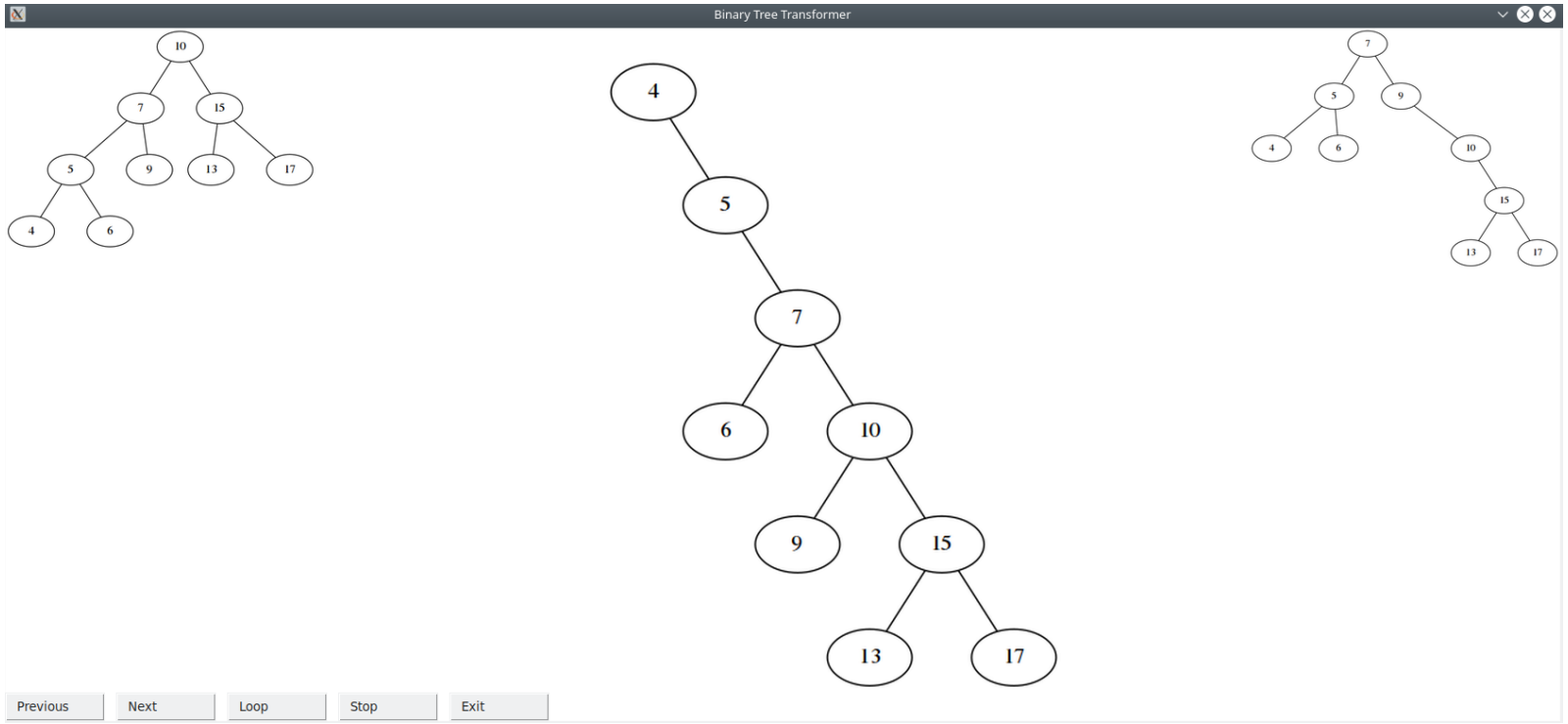
Slika 7: Prikaz vremena izvršavanja algoritma u odnosu na broj čvorova.

4 Vizuelizacija

Vizuelizacija je odrađena u programskom jeziku Python, korišćenjem dve biblioteke - Graphviz i Tkinter. Sastoji se iz tri koraka:

1. ispisivanja stabla u tekstualnu datoteku svaki put kad se desi rotacija čvora oko svog roditelja,
2. kreiranja DOT datoteke na osnovu tekstualne, u formatu koji koristi Graphviz alat,
3. kreiranja PNG datoteka od DOT datoteka koristeći Graphviz alat i njihovo prikazivanje pomoću biblioteke Tkinter.

Na slici 8 je prikazan interfejs. U gornjem levom uglu je slika početnog stabla a u gornjem desnom uglu je slika krajnjeg, željenog stabla. U sredini se smenjuju slike stabala od početnog do krajnjeg, gde svaka prikazuje stanje nakon jedne izvršene rotacije čvora oko roditelja. U donjem levom delu interfejsa se nalaze dugmići za kontrolu prikaza.



Slika 8: Vizuelni prikaz algoritma.

4.1 Format tekstualne datoteke za prikaz stabla

Nakon što se desi jedinična rotacija čvora oko svog roditelja, (celo) stablo se ispisuje u tekstualnu datoteku formata sa slike 9. Prvi broj predstavlja čvor u stablu. Nakon njega slede najviše dva broja a najmanje nijedan broj. Ako se nalaze dva broja, prvi predstavlja levog sina čvora sa početka linije a drugi predstavlja desnog sina. Ako se nalazi samo jedan broj, to znači da čvor sa početka linije ima samo jednog sina, a to da li je levi ili desni se određuje na osnovu toga da li je njegova vrednost manja ili veća od vrednosti roditelja. Odsustvo brojeva nakon čvora govori da je čvor u stvari list i da nema potomke.

```
10 7 15
7 5 9
5 4 6
4
6
9
15 13 17
13
17
```

Slika 9: Prikaz teksutalne datoteke nakon ispisa stabla.

4.2 Format DOT datoteke

Graphviz je alat koji služi za generisanje preglednog prikaza grafova. Format datoteke koji zahteva se može videti na slici 10. Dajemo objašnjenje nekih delova sintakse:

1. ***A -- B*** predstavlja kreiranje grane između čvorova A i B, kao i kreiranje samih čvorova (sem ako nisu ranije uvedeni).
2. ***C [style=invis, width=0, label=“"]*** predstavlja kreiranje novog nevidljivog čvora, koji se koristi da bi se stablo pravilno prikazalo (u suprotnom se potomak stavlja direktno ispod roditelja, umesto malo u levu ili desnu stranu).
3. ***{rank=same A -- C -- B [style=invis]*** **}** predstavlja kreiranje nevidljive grane između čvorova A i C (koji je i sam nevidljiv) i nevidljive grane između čvorova C i B. Uz parametar *rank=same*, ovime se postiže da A i B budu na istom nivou u stablu i da između njih postoji razmak (tj. da A bude levo od svog oca a B desno)

```
graph G{
    5 -- 4
    center0 [style=invis, width=0, label=""]
    5 -- center0 [style=invis]
    5 -- 6
    {rank=same 4 -- center0 -- 6 [style=invis] }
    7 -- 5
    center1 [style=invis, width=0, label=""]
    7 -- center1 [style=invis]
    7 -- 9
    {rank=same 5 -- center1 -- 9 [style=invis] }
    10 -- 7
    center2 [style=invis, width=0, label=""]
    10 -- center2 [style=invis]
    10 -- 15
    {rank=same 7 -- center2 -- 15 [style=invis] }
    15 -- 13
    center3 [style=invis, width=0, label=""]
    15 -- center3 [style=invis]
    15 -- 17
    {rank=same 13 -- center3 -- 17 [style=invis] }
}
```

Slika 10: Prikaz DOT datoteke.

4.3 Prikaz pomoću Tkinter biblioteke

Korišćenjem Tkinter biblioteke moguće je napraviti *canvas* tj. površinu za prikaz slika i dugmića. Omogućava skaliranje slika, njihovo precizno pozicioniranje na površini za prikaz, kreiranje dugmića i vezivanje odgovarajućih poziva metoda za njih. Da bi se omogućila promena slika na klik dugmeta *next* ili *previous* ili automatska promena klikom na dugme *loop*, potrebno je napraviti klasu koja sadrži listu slika i metode za pojedinačno crtanje samo jedne slike (prethodne ili sledeće) kao i metodu koja koristi tajmer za automatsko iscrtavanje slika.

Literatura

[1] M. Spasić, Konstrukcija i analiza algoritama 2, vežbe, 2019.

[2] Dokumentacija za graphviz https://pypi.org/project/graphviz/

[3] Online graphviz alat http://www.webgraphviz.com/

[4] Dokumentacija za tkinter https://wiki.python.org/moin/TkInter