

Rotacije stabala

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

Stanojević Strahinja,
Bajić Ana

26. novembar 2018.

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 — C++, rotacije, stabla, algoritam

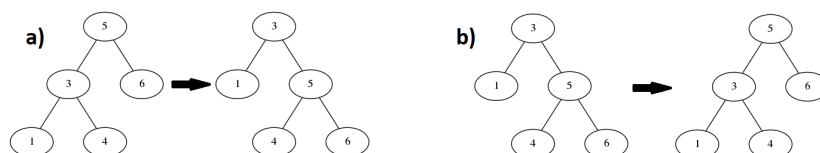
Sadržaj

1	Postavka problema	2
2	Algoritam	2
2.1	Rotacija udesno	3
2.2	Rotacija ulevo	4
3	Složenost algoritma	5
4	Zaključak	5
	Literatura	5

1 Postavka problema

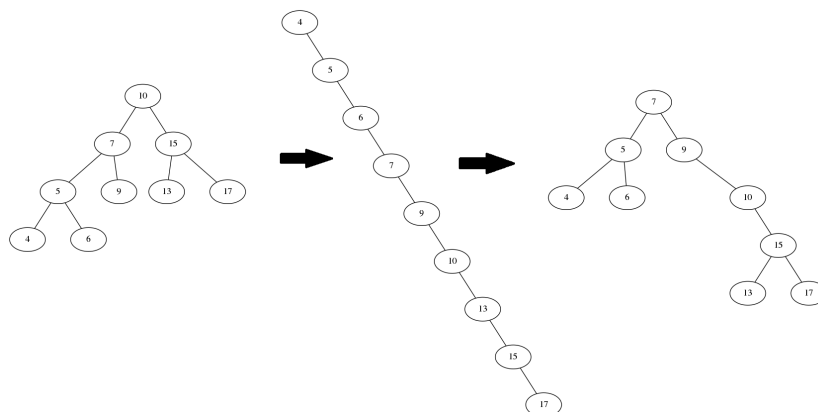
Problem je postavljen na sledeći način:

“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 a onda primena rotacija dok se ne dovede u formu drugog stabla.



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 pratni kod.

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 desne sinove).

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, diže se za jedan 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**.

Naredna *while* petlja zalazi u čvorove koji i dalje imaju leve sinove i razrešava ih na isti način, sve dok ne stigne do listova.

```
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 stabla udesno.

2.2 Rotacija ulevo

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

Na slici 5 je prikazan kod za izvođenje ove transformacije. Ako čvor (dalje U) u stablu koje želimo da dobijemo 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 U u listi (V) a zatim se pronađu svi njegovi preci. Novi koren liste postaje V , njegov levi sin postaje podstablo predaka a desni ostaje ostatak liste. Ovo nastavlja da se ne dobije traženo stablo.

```
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;
    }

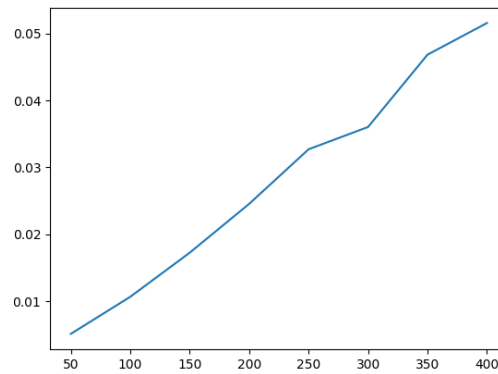
    return nullptr;
}

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

Složenost algoritma je linearna po broju čvorova. U najviše 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.

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.

4 Zaključak