

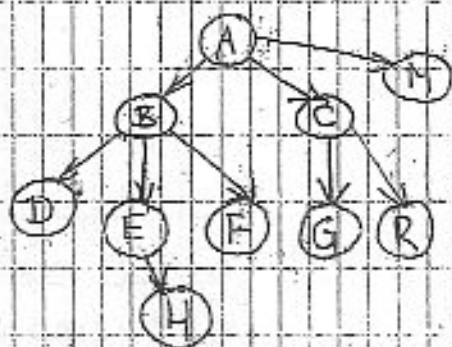


STRUKTURA PO DATAKA

Ignacio

1. Stablo → implementacija

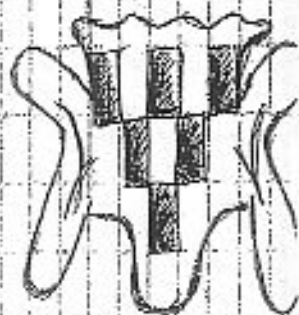
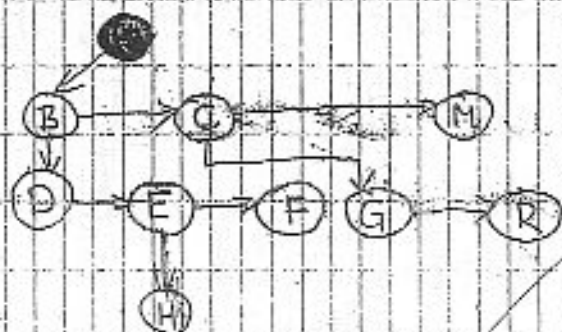
Jedna od mogućih implementacija je da svaki "roditelj" sadrži pokazivač na svako svoje "dijete".
 To dobro funkcionira kod npr. binarnih stabala, ali kod običnih stabala broj čvorova može eksponencijalno rasti. Stoga se koristi druga metoda. Svaki roditelj sadrži pokazivač na jedno svoje dijete, te pokazivač na jednog brata. Tako umjesto:



strukt Stablo

int element; // npr.
 strukt Stablo * dijete;
 strukt Stablo * brati;

umamo:



2. Procedura za upis direktorija

Koristimo rekursiju.

```
void listDir ( direktorij ili file D, int dubina )
{
    if ( D postoji kao direktorij ili file )
    {
        ispiši na ekran ( D, dubina );
        if ( D je direktorij )
            za svako dugme C, D, ... listDir ( C, dubina + 1 );
    }
}
```

dubina nam koristi koliko tabova trebamo upisati, tako da mena bude "umučena".

Ovo je preorder upis. Za postorder, nešto kao ovo:

```
void listDir ( direktorij ili file D, int dubina )
{
    if ( D postoji kao direktorij ili file )
    {
        if ( D je direktorij )
            za svako dugme C, D, ... listDir ( C, dubina + 1 );
        ispiši na ekran ( D, dubina );
    }
}
```


3. Procedura za ispis veličine direktorija

```
int RecDirectory (direktori ili file D)
```

```
{  
    int Total;
```

```
    Total = 0;
```

```
    if ( D je direktorij ili file)
```

```
    {
```

```
        if ( D je direktorij)
```

```
        {  
            za svako djetelo C D,
```

```
                Total += RecDirectory (C);
```

```
            Total ++;
```

```
        }  
        else Total = File size (D);
```

```
    }  
    return Total;
```

```
}
```

4. Binarno stablo je stablo koje se sastoji od korijena, lijevog i desnog podstabla. T_l i T_r kojeg čvor ima najviše 2 djeteta.

```
struct cvorstabla
```

```
{
```

```
    int element;
```

```
    struct cvorstabla *lijevo;
```

```
    struct cvorstabla *desno;
```

```
}
```

```
struct cvorstabla * NapraviPravno (struct cvorstabla *T)
```

```
{
```

```
    if (T != NULL)
```

```
    {
```

```
        NapraviPravno (T->lijevo);
```

```
        NapraviPravno (T->desno);
```

```
        free(T);
```

```
    }
```

```
    return NULL;
```

```
}
```

```
struct cvorstabla * Nadi (struct cvorstabla *T, int x)
```

```
{
```

```
    if (T == NULL) return NULL;
```

```
    if (x < T->element) return Nadi (T->lijevo, x);
```

```
    if (x > T->element) return Nadi (T->desno, x);
```

```
    else return T;
```

```
}
```



```
struct tnode * Nadimax (struct tnode * T) {
```

```
    if (T == NULL) return NULL;
```

```
    if (T->desno != NULL) return T;
```

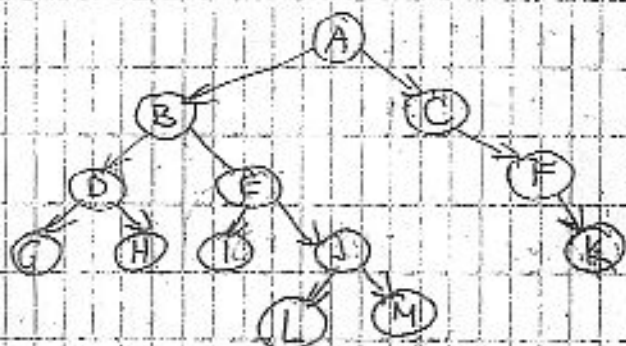
```
    else
```

```
        return Nadimax (T->desno);
```

```
}
```

Znači, za binarno stablo je specifično da pri unosu podatka, gledamo prvo i ako je element koji unosimo manji od elementa čuša, nastavljamo pretraživanje u lijevom podstablu. Ako, naiđemo na element onda ne radimo ništa, a ako dođemo do NULL onda uvrstimo taj element. Shodno za "veći element".

5. Inorder, preorder, postorder



inorder: G, H, D, I, L, M, J, E, B, C, F, K

preorder: A, B, D, G, H, E, I, J, L, M, C, F, K

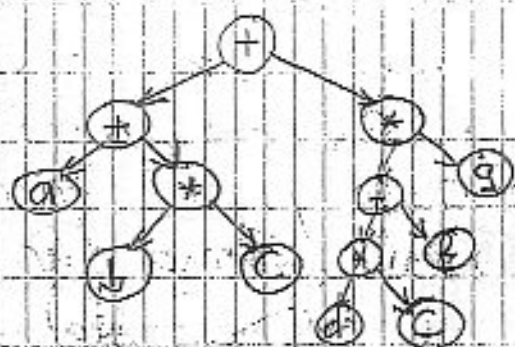
postorder: G, H, D, I, L, M, J, E, B, K, F, C, A

preorder - prvo se obradi čvor, zatim podstabla s lijeva na desno

postorder - prvo se obrade stabla s lijeva na desno, zatim se obradi čvor

inorder - lijevi čvor - desno

$$6. (a + b * c) + ((d * e + f) * g)$$



Jedan od načina je da rekursivno predstavimo
 lijevo i desno podstablo i, primijenimo operator
 u čvoru na lijevo i desno podstablo, s tim
 da ono u podstablu stavljamo u zagrade.
 Drugi način je da primijenimo postorder
 obradu i dobijemo postfix izraz.

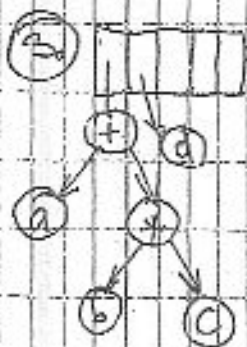
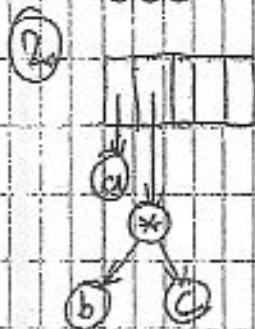
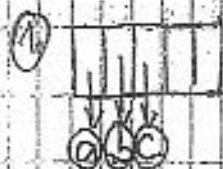
$abc * + dc * f + g * +$

[Handwritten signature]

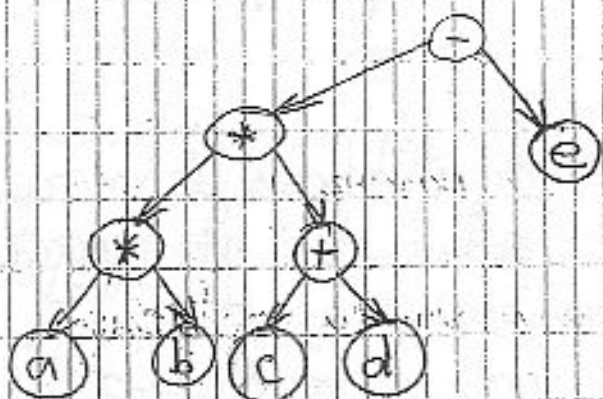
7. Stablo proračuna generiramo iz postfix izraza uz pomoć stoga.

Ako je simbol operand stvorimo stablo s jednim čvorom i u stogu smjestimo pokazivač na njega.
 Ako je simbol operator uzmemo zadnja dva pokazivača sa stoga i stvorimo novo stablo čiji je korijen taj operator, a lijevi i desni pokazivači pokazuju na ono što pokazuju ti dva pokazivača.

mpot. $a b c * + d$



8. Prefix, infix i postfix za dabo



infix: $a*b*c*d+e-$

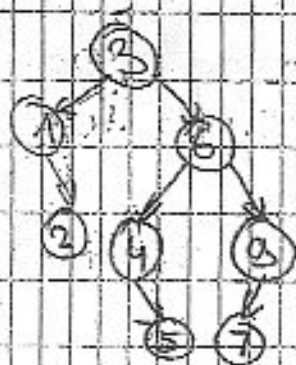
prefix: $+*+a*b+c*d\ e$

postfix: $a\ b\ * \ c\ d\ + \ * \ e \ -$

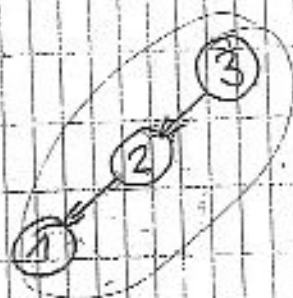
9. AVL stable: 3, 1, 4, 6, 9, 2, 5, 7



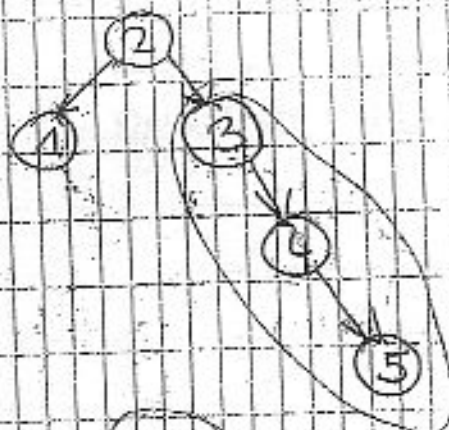
формативна структура



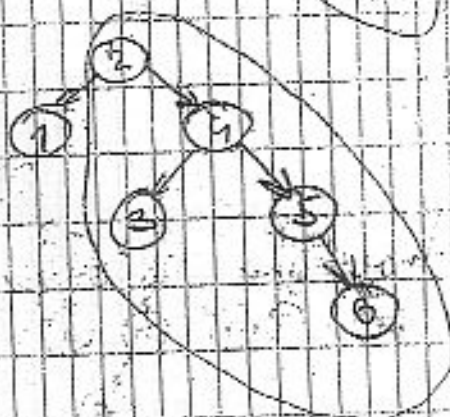
5. b) 3, 2, 1, 4, 5, 6, 7



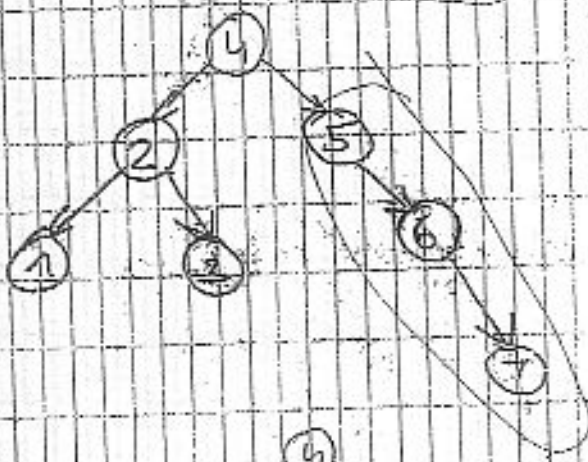
jednostruka rotacija



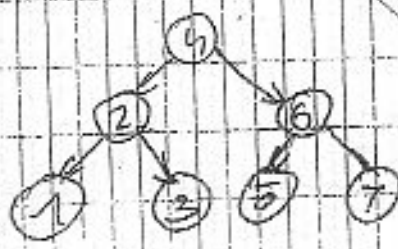
jednostruka rotacija



jednostruka rotacija

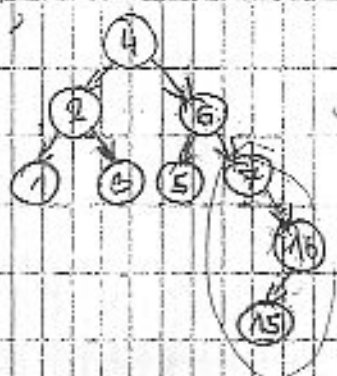


jednostruka rotacija

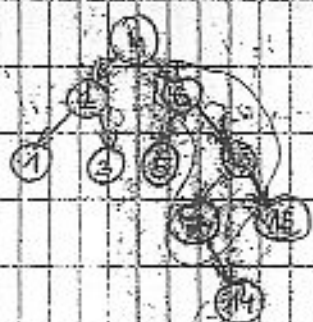


11

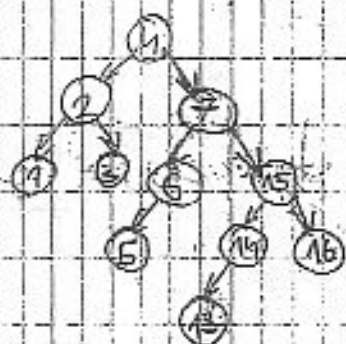
9. c) 13, 2, 1, 4, 5, 6, 7, 16, 15, 14, 13, 12, 11, 10, 9, 8, 25, 24, 23, 22



dvostruka rotacija



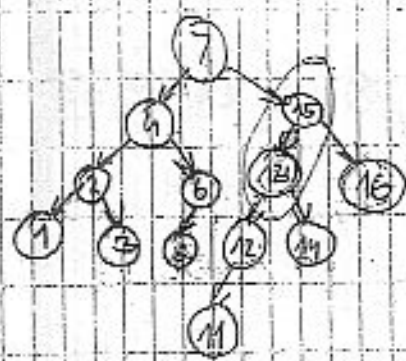
dvostruka rotacija



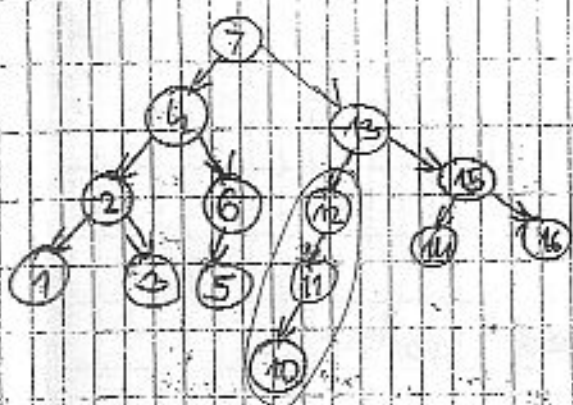
jednostruka rotacija



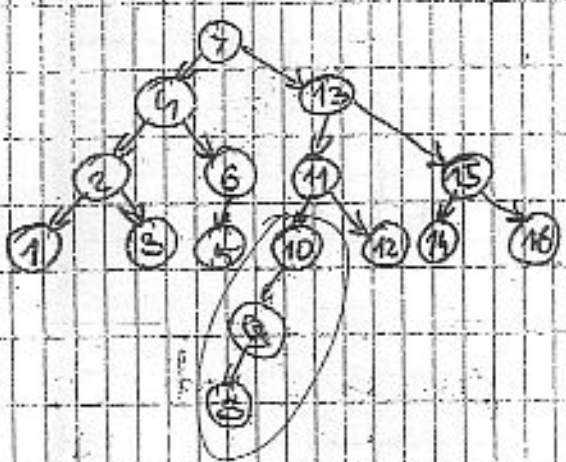
jednostruka rotacija



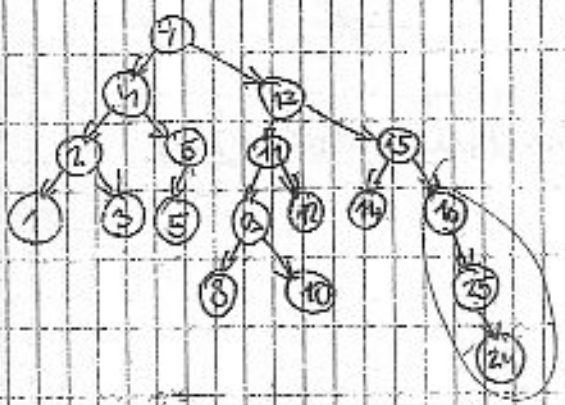
jednostruka rotacija



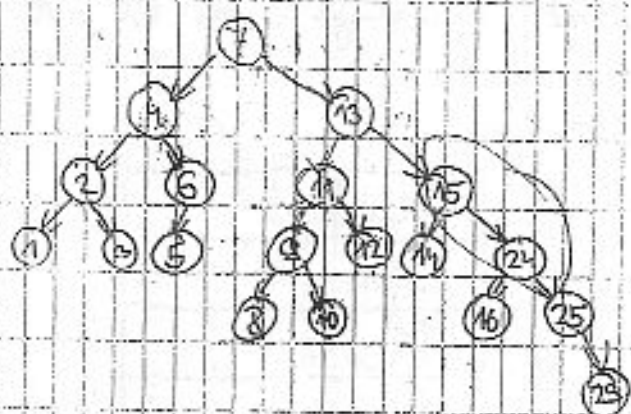
jednostruka rotacija



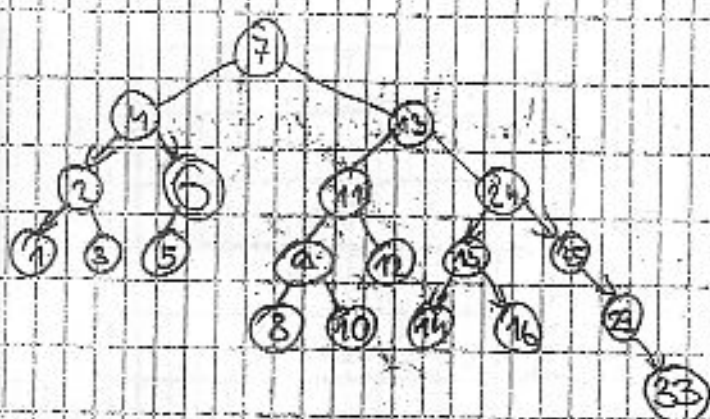
jednostruka rotacija



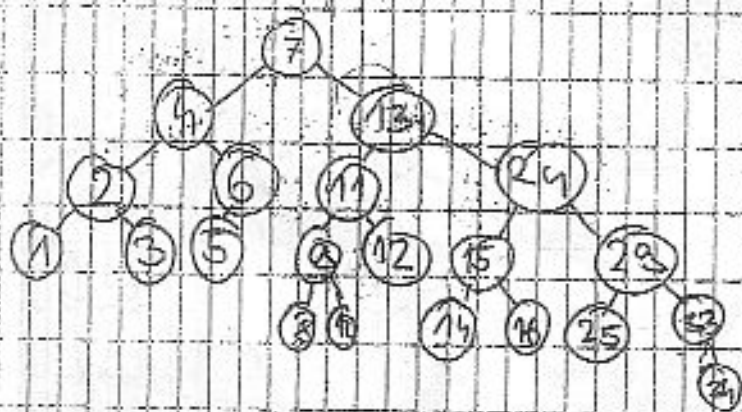
dvostruka rotacija



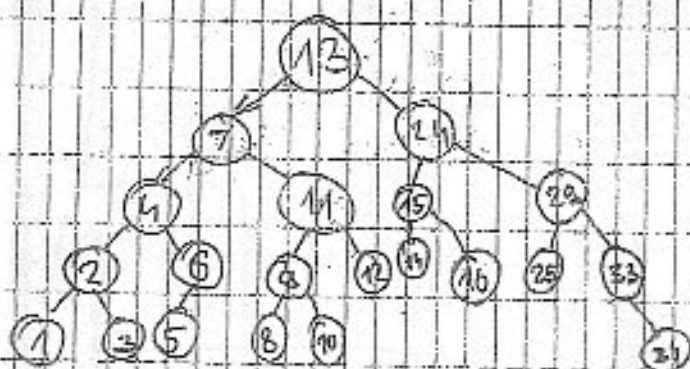
jednostruka rotacija



jednostruka rotacija



jednostruka rotacija



14

15

10. Zbraj prva dva broja (ASCII vrijednost) podijeljeno sa 10
ostatak je mesto u tablici

Kvadratično isprkavanje $hash_2(x) = 7 - (x \bmod 7)$

Split $\rightarrow 195 : 11 \dots 8$

Zadar $\rightarrow 487 : 11 \dots 0$

Sibenik $\rightarrow 188 : 11 \dots 1$

Dubrovnik $\rightarrow 185 : 11 \dots 9$

Ryeka $\rightarrow 187 : 11 \dots 0$

Pula $\rightarrow 197 : 11 \dots 10$

Plöče $\rightarrow 188 : 11 \dots 1$

Trogir $\rightarrow 198 : 11 \dots 0$

0

Zadar

1

Sibenik

2

Plöče

3

Trogir

4

~~Plöče~~

5

~~Plöče~~

6

~~Plöče~~

7

Ryeka

8

Split

9

Dubrovnik

10

Pula

