ALGORITMI I STRUKTURE PODATAKA

RAČUNSKE VEŽBE - TERMIN BR. 11 - BINARNA STABLA PRETRAŽIVANJA

ALDINA AVDIĆ, DIPL. INŽ. - apljaskovic@np.ac.rs

RAČUNARSKA TEHNIKA, SOFTVERSKO INŽENJERSTVO, INFORMATIKA I MATEMATIKA

Binarno stablo pretraživanja

- Binarno stablo pretraživanja (BST), poznato i kao sortirano binarno stablo, je binarno stablo zasnovano na čvorovima, gde svaki čvor ima uporedljivi ključ (sa dodeljenom vrednošću) i zadovoljava uslov da je vrednost svakog čvora veća od vrednosti svakog čvora u njegovom levom podstablu i manja od vrednosti svakog čvora u njegovom desnom podstablu.
- Svaki čvor ima najviše dva deteta. Svako dete mora da bude ili list (nema nijedno dete) ili koren još jednog binarnog stabla pretrage. BSP je dinamička struktura podataka, i veličina BSP-a je ograničena samo količinom slobodne memorije u operativnom sistemu. Najveća prednost binarnog stabla pretrage je da ostaje uređeno, što omogućava brže vreme pretrage nego većina drugih struktura.

Binarno stablo pretraživanja

- Osnovna svojstva binarnog stabla pretrage:
  - Levo podstablo čvora sadrži samo čvorove koje imaju manju vrednost od njega.
  - Desno podstablo čvora sadrži samo čvorove koje imaju veću vrednost od njega.
  - Levo i desno podstablo moraju takođe biti binarna stabla pretrage.
  - Svaki čvor može imati najviše 2 deteta.
  - Ne smeju postojati duplikati čvorova.
  - Postoji jedinstven put od korena do svakog čvora.
- Najveća prednost binarnog stabla pretrage u odnosu na ostale strukture podataka je da algoritmi sortiranja i algoritmi pretrage kao npr. pretraga u dubinu (in-order) mogu biti veoma efikasni.

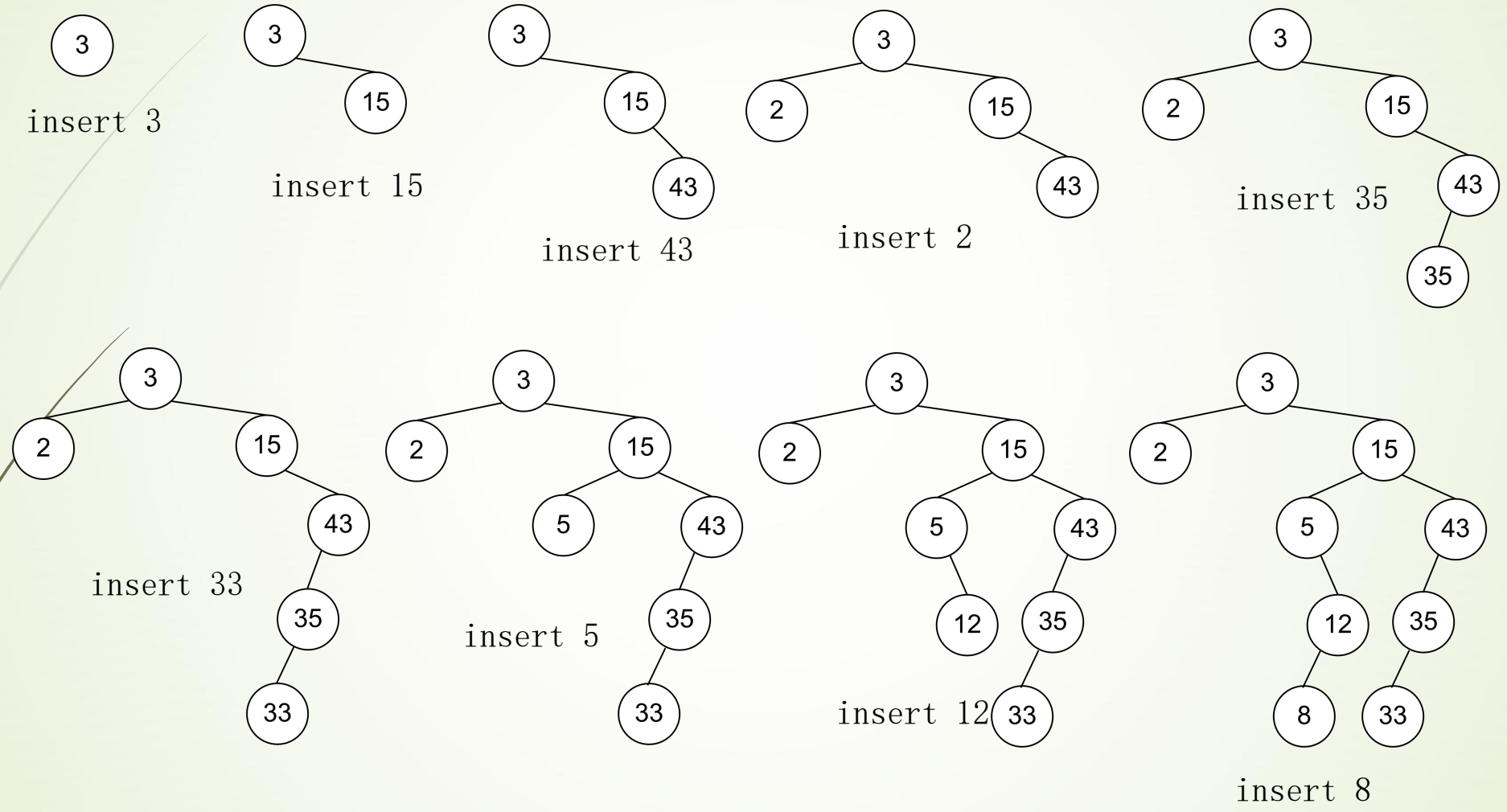
Zadatak 1

U binarno stablo pretraživanja umetnuti redom čvorove sa vrednostima ključeva 3, 15, 43, 2, 35, 33, 5, 12, 8. Prikazati izgled stabla nakon svakog umetanja.

Kako bi izgledao redosled ispisivanja vrednosti ključeva, ako se stablo obiđe u *inorder* poretku?

Navesti prednosti stabla binarnog pretraživanja nad metodama pretraživanja vektora (linearnim i nelinearnim).

Zadatak 1 - Rešenje



Zadatak 1 - Rešenje

- Inorder obilaskom stabla binarnog pretraživanja bi se uvek dobili ključevi uređeni po neopadajućem poretku.

- Prednosti stabla:
- jednostavno umetanje i brisanje ključeva
- relativno mala vremenska složenost
  O(h) - u najboljem slučaju O(log n), u najgorem O(n)

7

Zadatak 2

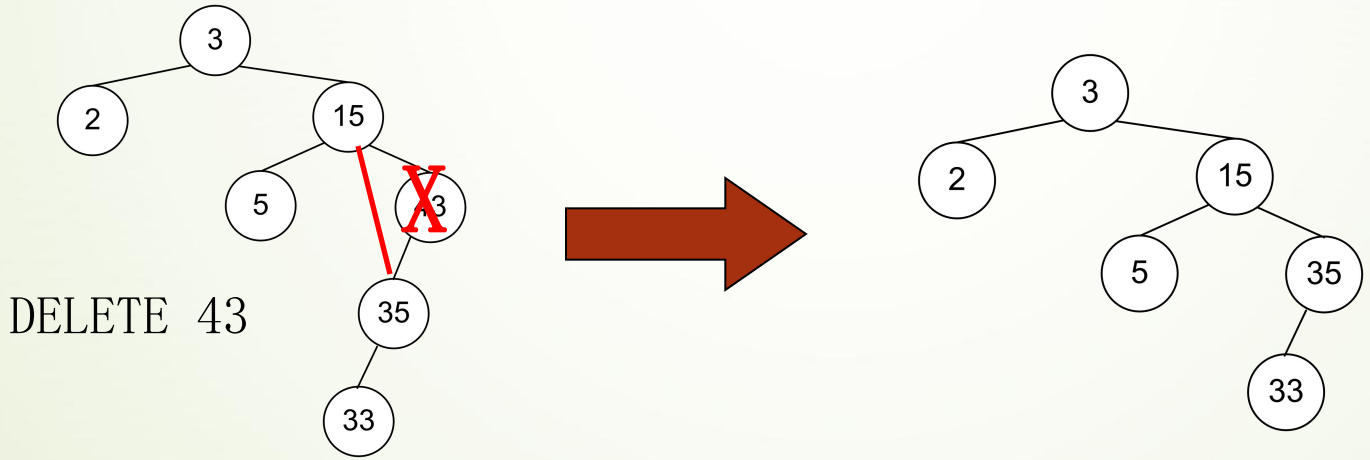
- Navesti i ilustrovati sve slučajeve brisanja čvorova iz binarnog stabla pretraživanja.
- Iz binarnog stablo pretraživanja iz prethodnog zadatka, nakon umetanja svih čvorova, redom se brišu čvorovi 8, 15, 5. Nacrtati izgled stabla nakon svakog brisanja.

Zadatak 2 - Rešenje

Mogući slučajevi brisanja čvorova iz stabla binarnog pretraživanja:

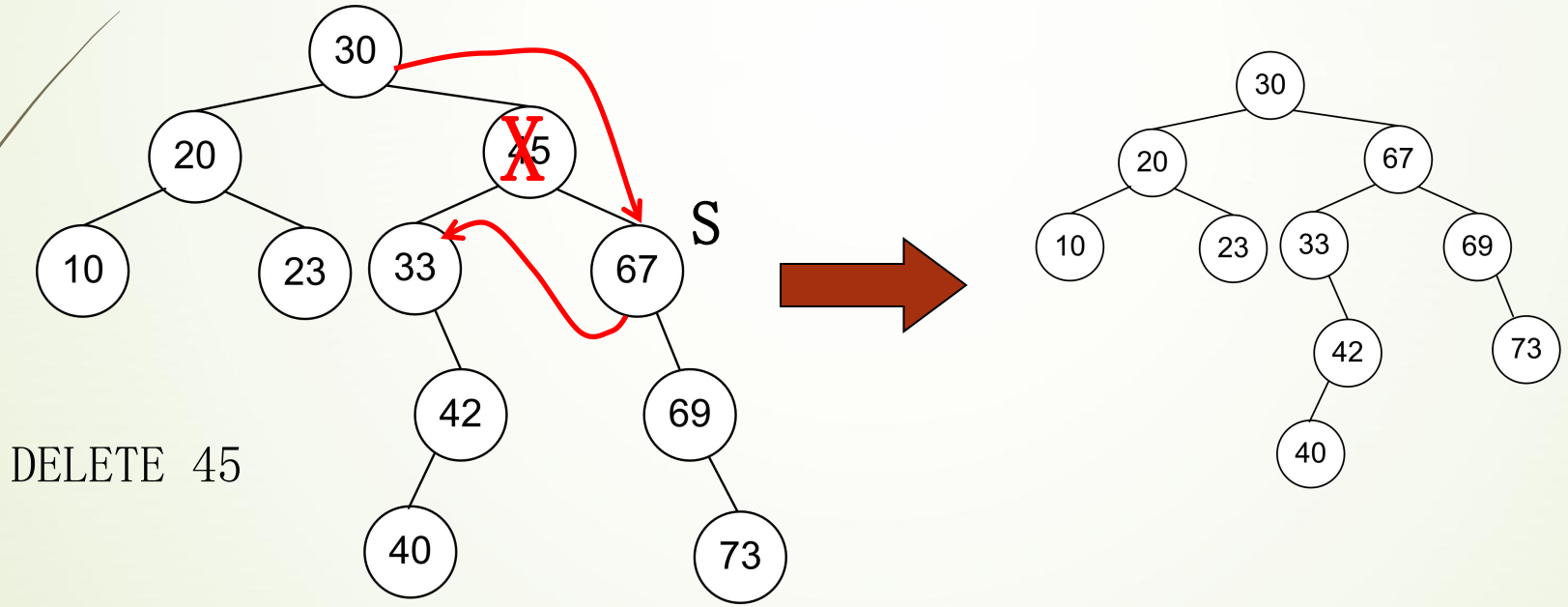
- Čvor koji se briše nema potomke:
  _odgovarajući pokazivač roditeljskog čvora postaje NIL

- Čvor koji se briše ima samo jednog direktnog potomka:
  _direktni potomak preuzima mesto čvora koji se briše



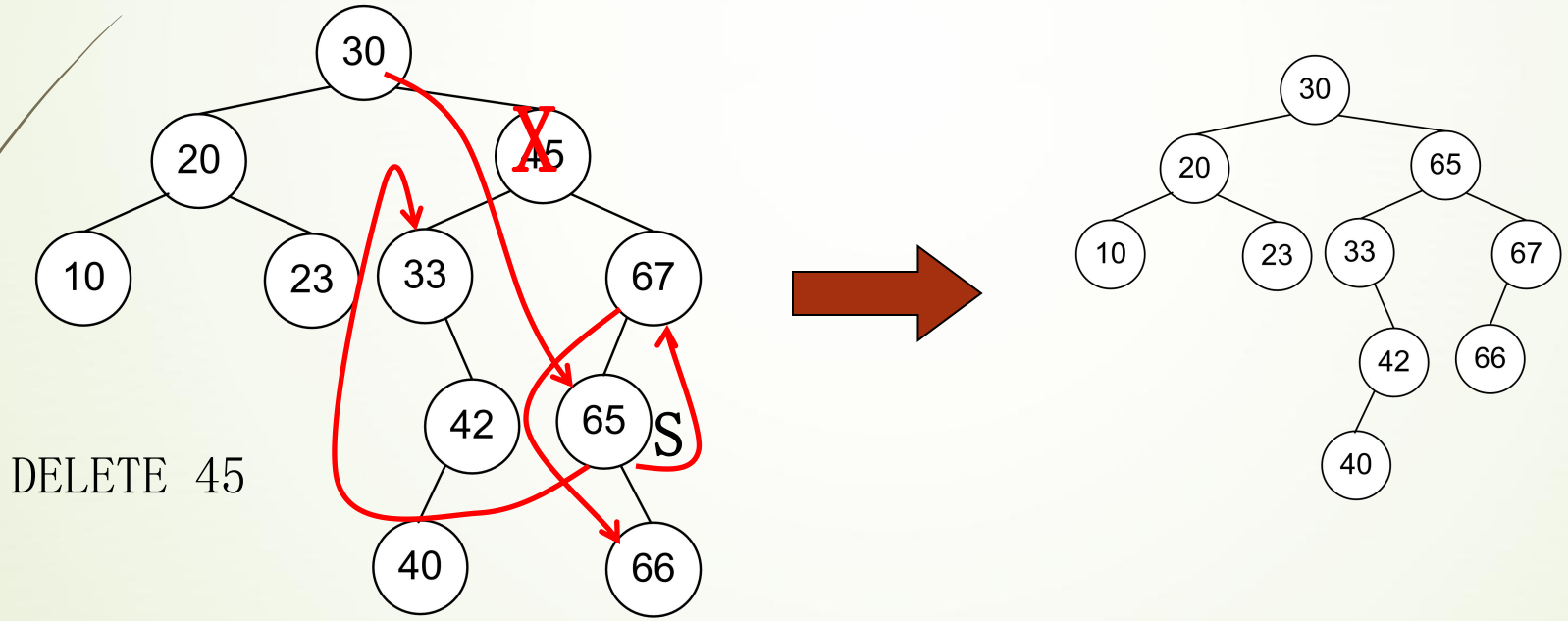
Zadatak 2 - Rešenje

- Čvor koji se briše ima oba sina:
  _pronalazi se sledbenik S čvora koji se briše;
  - roditelj S je čvor koji se briše:
    S zauzima mesto roditelja (preuzima njegovo levo podstablo)

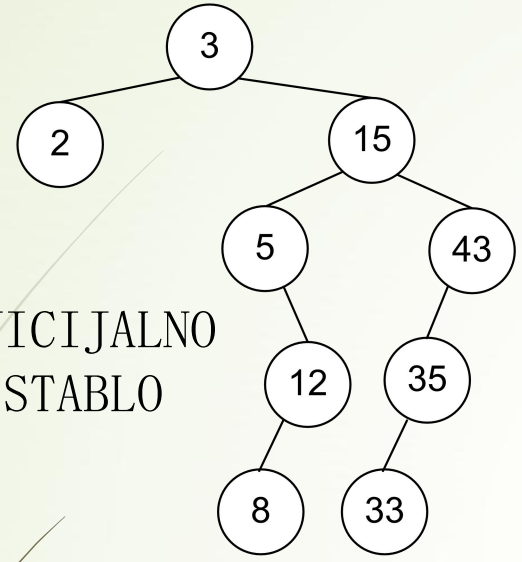


Zadatak 2 - Rešenje

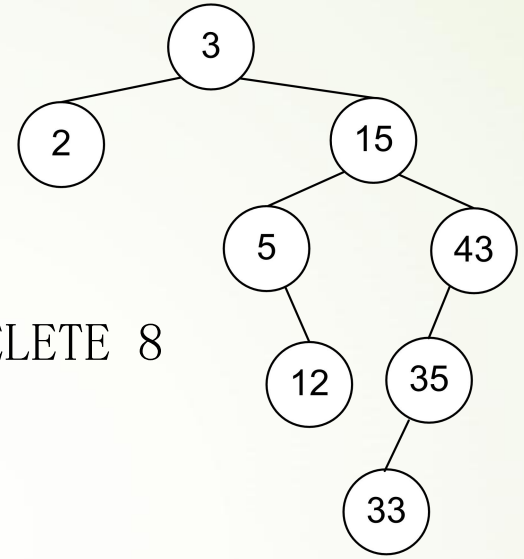
- Čvor koji se briše ima oba sina:
  _pronalazi se sledbenik S čvora koji se briše;
  - roditelj S nije čvor koji se briše:
    eventualni desni potomak čvora S postaje levi potomak čvora roditelja čvora S, a S zauzima mesto čvora koji se briše



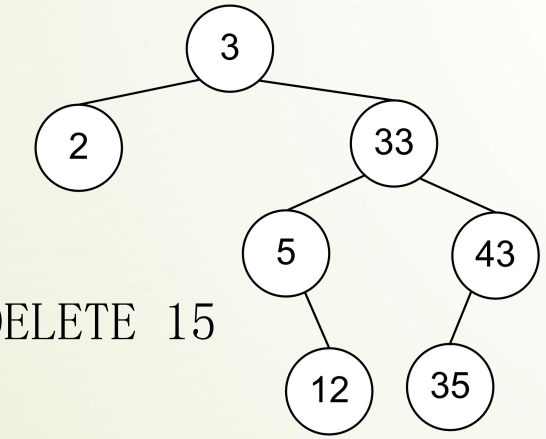
Zadatak 2 - Rešenje



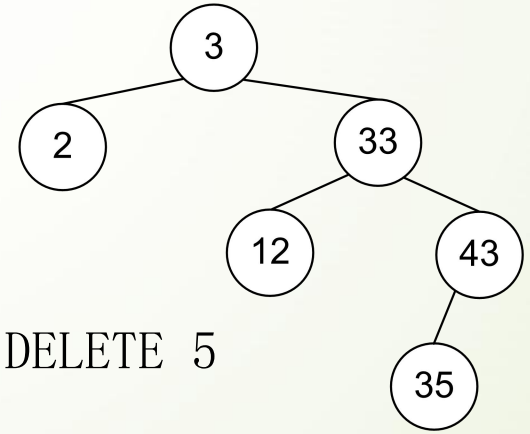
INICIJALNO STABLO



DELETE 8



DELETE 15



DELETE 5

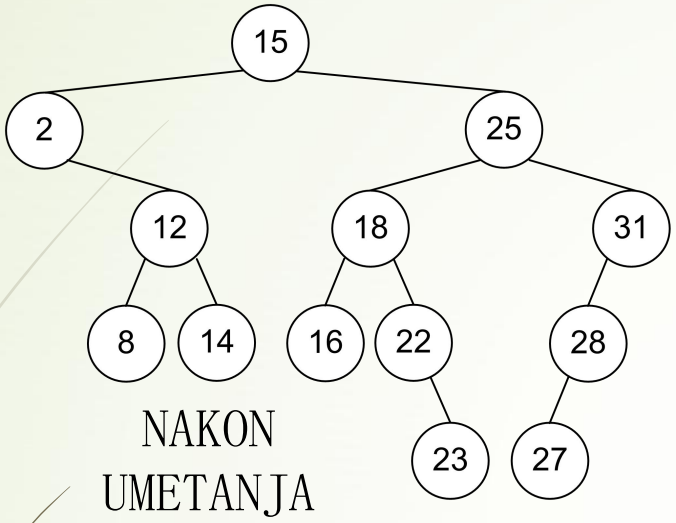
Zadatak 3

U binarno stablo pretraživanja se najpre redom umeću ključevi 15, 25, 18, 31, 16, 2, 12, 14, 8, 22, 23, 28, 27.

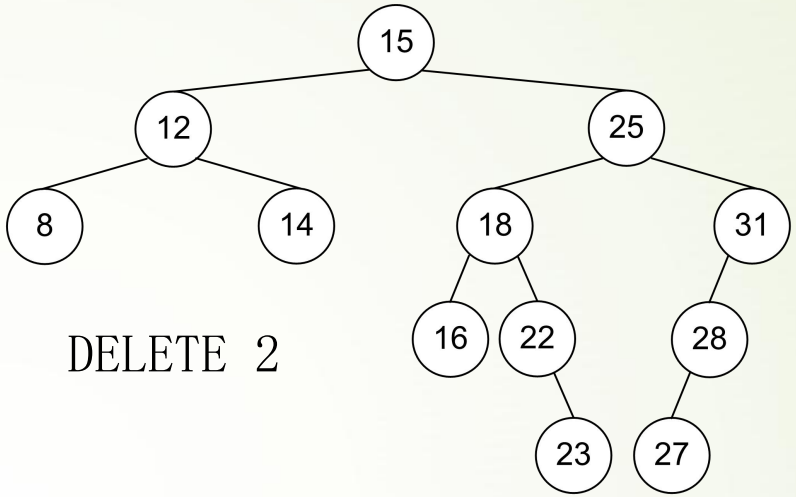
Zatim se redom brišu ključevi 2, 25, 15, 16, 18, 22, 23

Prikazati izgled stabla nakon svakog umetanja i brisanja.

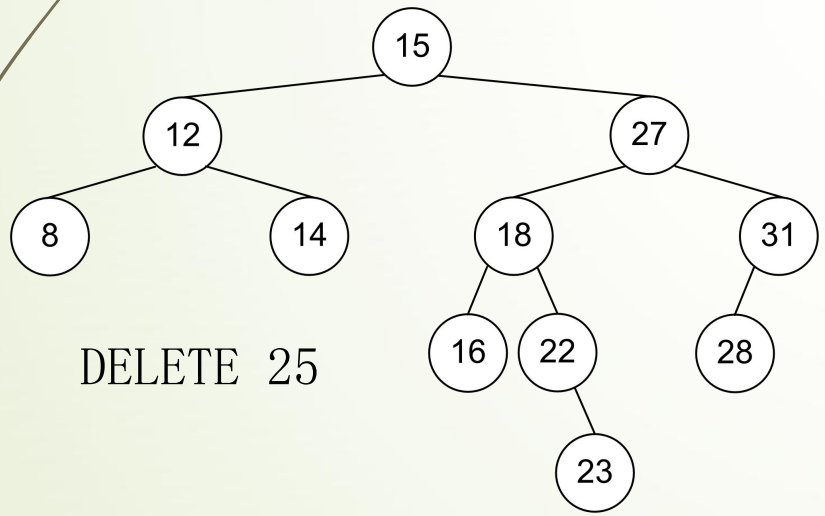
Zadatak 3 - Rešenje



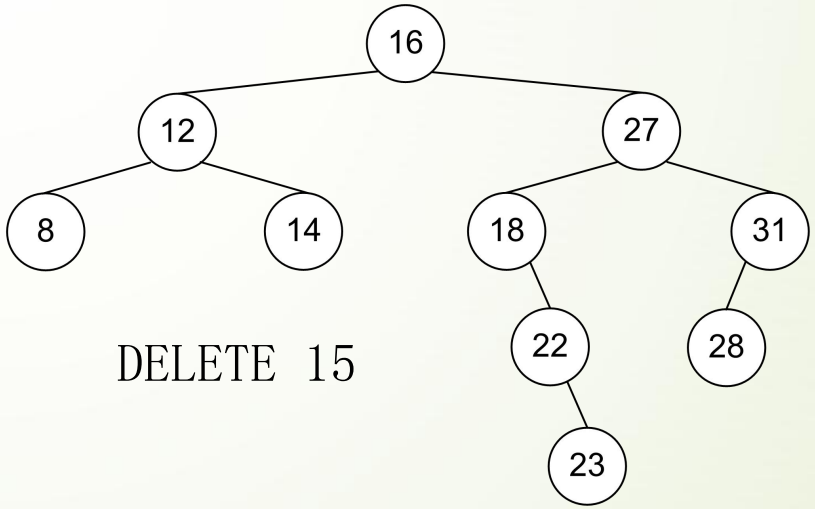
NAKON UMETANJA



DELETE 2

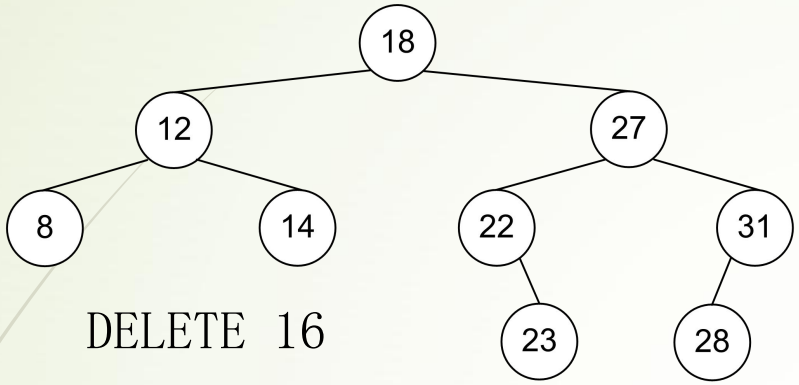


DELETE 25

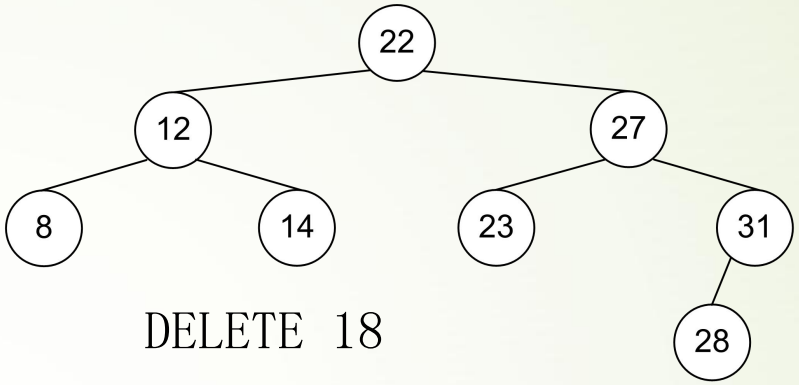


DELETE 15

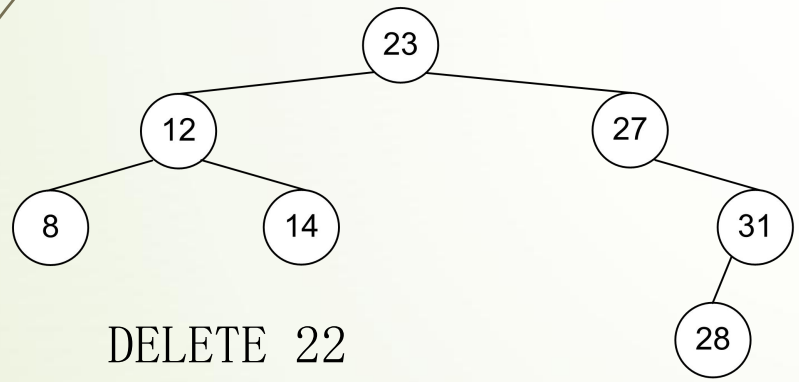
Zadatak 3 - Rešenje



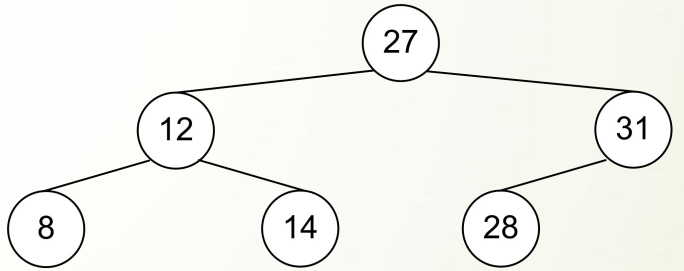
DELETE 16



DELETE 18



DELETE 22



DELETE 23

Zadatak 4

Binarno stablo pretraživanja B se dobija tako što se u njega redom umeću ključevi dobijeni *inorder* obilaskom binarnog stabla pretraživanja A.

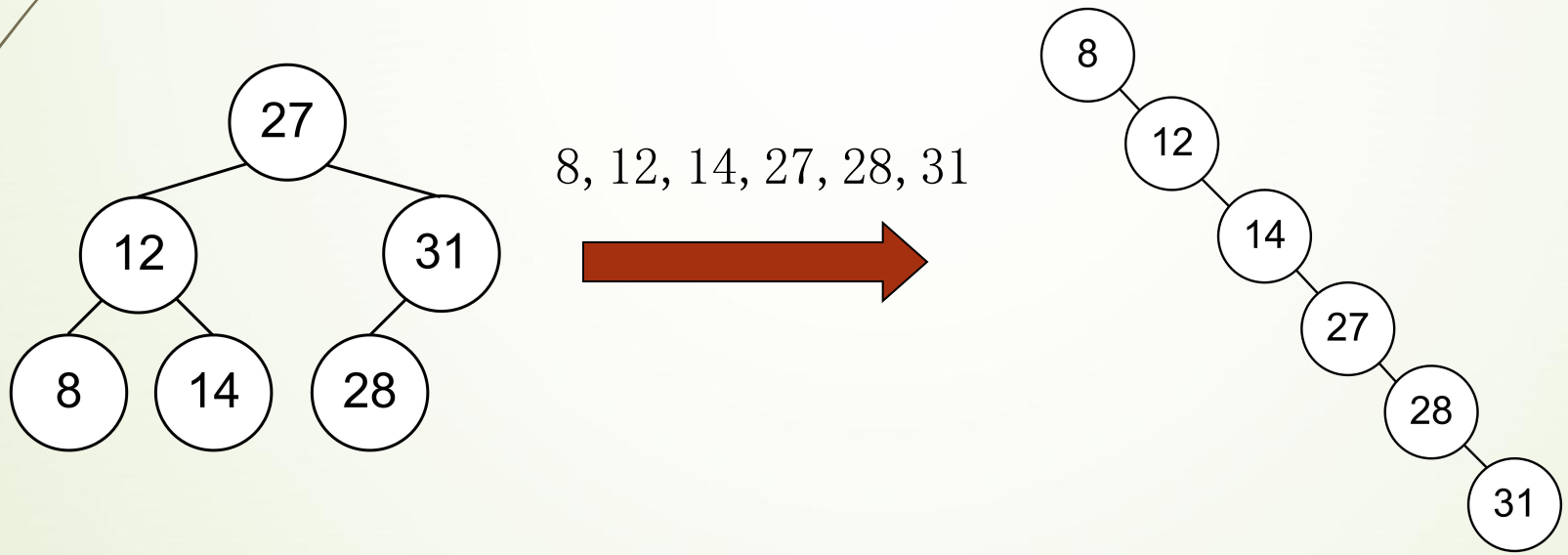
Komentarisati osobine stabla B (kompletnost, balansiranost).
Proceniti vremensku složenost pretrage proizvoljnog ključa u stablu B.

Predložiti redosled umetanja ključeva u stablo B da bi se poboljšale performanse pretrage.

Zadatak 4 - Rešenje

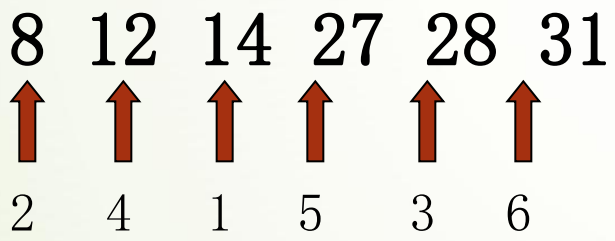
Inorder obilaskom stabla se pristupa ključevima u neopadajućem redosledu.

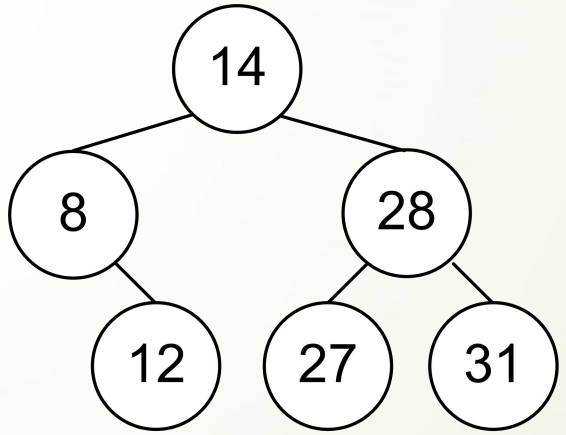
Umetanje ključeva datim redosledom bi napravilo nebalansirano stablo kod kojeg svaki čvor (izuzev lista) ima samo desnog potomka - svodi se na ulančanu listu.



Zadatak 4 - Rešenje

Binarno stablo pretraživanja ima najbolje performanse kada je balansirano. Balansiranost bi se postigla ako bi se najpre izvršilo umetanje ključa na sredini niza ključeva dobijenog *inorder* obilaskom, a zatim ključeva na polovini intervala između početka i sredine niza, odnosno sredine i kraja niza, itd.





Binarno stablo pretraživanja - reprezentacija u C-u

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
    int key;
    struct node *left, *right;
};
```

```
// A utility function to create a new BST node
struct node *newNode(int item)
{
    struct node *temp =  (struct node *)malloc(sizeof(struct node));
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}
```

Binarno stablo pretraživanja - reprezentacija u C-u

```
void inorder(struct node *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d \n", root->key);
        inorder(root->right);
    }
}
```

```
struct node* insert(struct node* node, int key)
{
    /* If the tree is empty, return a new node */
    if (node == NULL) return newNode(key);

    /* Otherwise, recur down the tree */
    if (key < node->key)
        node->left  = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);

    /* return the (unchanged) node pointer */
    return node;
}
```

Binarno stablo pretraživanja - reprezentacija u C-u

```
struct node* search(struct node* root, int key)
{
    // Base Cases: root is null or key is present at root
    if (root == NULL || root->key == key)
       return root;

    // Key is greater than root's key
    if (root->key < key)
       return search(root->right, key);

    // Key is smaller than root's key
    return search(root->left, key);
}
```

```
struct node * minValueNode(struct node* node)
{
    struct node* current = node;

    /* loop down to find the leftmost leaf */
    while (current->left != NULL)
        current = current->left;

    return current;
}
```

Binarno stablo pretraživanja - reprezentacija u C-u

```
struct node* deleteNode(struct node* root, int key)
{
if (root == NULL) return root;

if (key < root->key)
        root->left = deleteNode(root->left, key);

    else if (key > root->key)
        root->right = deleteNode(root->right, key);

    else
    {
        if (root->left == NULL)
        {
            struct node *temp = root->right;
            free(root);
            return temp;
        }
```

Binarno stablo pretraživanja - reprezentacija u C-u

```
int main()
{
    /* Let us create following BST
              50
           /     \
          30      70
         /  \    /  \
       20   40  60   80 */
    struct node *root = NULL;
    root = insert(root, 50);
    insert(root, 30);
    insert(root, 20);
    insert(root, 40);
    insert(root, 70);
    insert(root, 60);
    insert(root, 80);

    // print inoder traversal of the BST
    inorder(root);
```

Test

1. Šta je binarno stablo pretraživanja?
2. Koja su njegova svojstva?
3. Objasniti umetanje u BSP?
4. Objasniti 4 slučaja brisanja kod BSP?
5. Šta se dobija inorder obilaskom BSP?
6. Kako se implementira BSP u C-u?
7. Da li se razlikuju obilasci od obilazaka binarnog stabla?
8. Objasniti pretragu kod BST?
9. Objasniti koja je razlika u odnosu na dodavanje novog čvora kod običnog binarnog stabla?
10. Napisati f-ju u C-u za kreiranje novog čvora BST.

Test

- Test poslati do 18.05.2020. u 14h na mejl apljaskovic@np.ac.rs prema uputstvima sa sajta univerziteta

Hvala na pažnji!