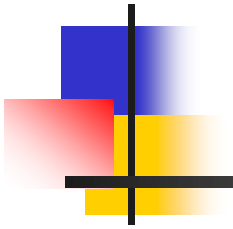
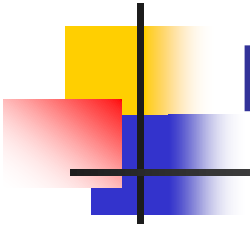


PROGRAMIRANJE II

P-08: Nelinearne strukture podataka

prof. dr **Dražen Brđanin**
2023/24



P-08: Nelinearne strukture podataka

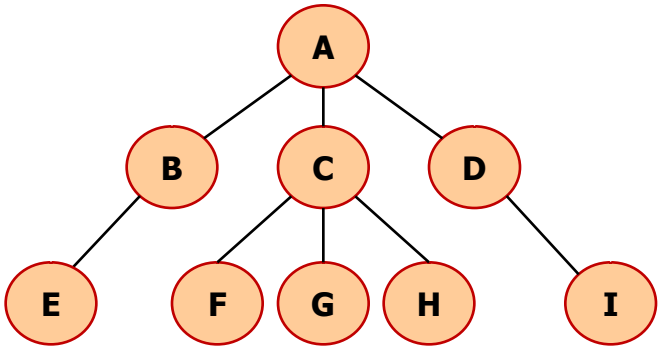
- **Sadržaj predavanja**
  - Osnovni pojmovi o nelinearnim strukturama podataka
  - Osnovne nelinearne strukture
    - stablo
    - binarno stablo
      - sekvencijalna i ulančana reprezentacija
      - operacije (dodavanje, obilazak, brisanje)

Nelinearne strukture podataka

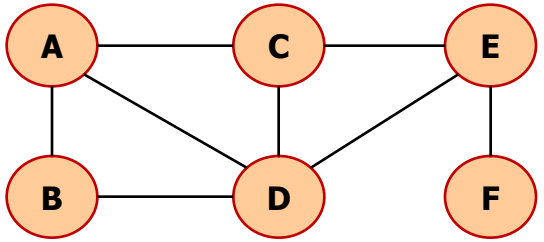
- **Nelinearna struktura podataka**:
  - kolekcija podataka u kojoj elementi mogu da imaju više neposrednih prethodnika (predaka) i/ili više neposrednih sljedbenika (nasljednika)
  - **svaki element u kolekciji može biti povezan sa više od druga dva elementa**

- **Vrste nelinearnih struktura:**

stablo



graf

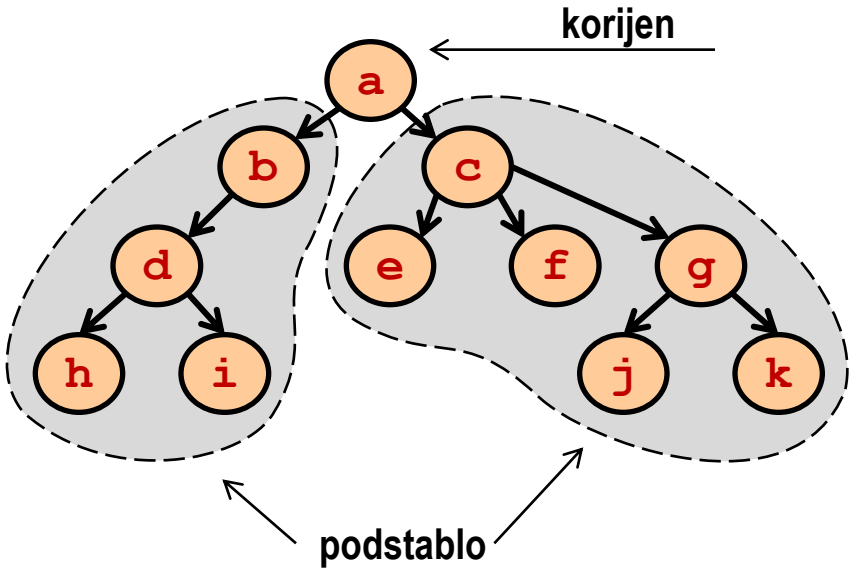


Stablo je nelinarna struktura u kojoj nema zatvorenih (cikličkih) putanja – ne postoji mogućnost da se obilaskom povezanih čvorova napravi "krug".

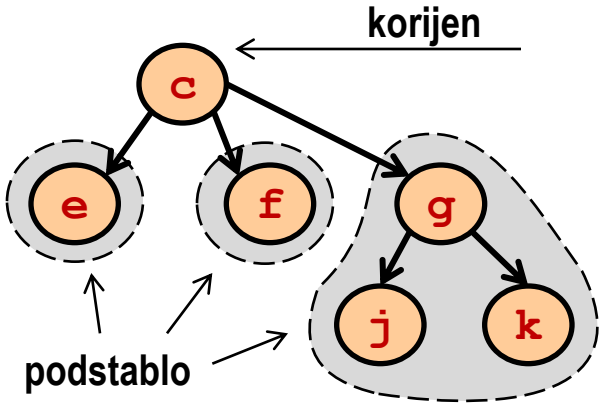
- **Memorijska reprezentacija:**
  - **ulančana (češće)**
  - **sekvencijalna**

Stablo

- **STABLO (eng. TREE)** = **konačan skup čvorova sa sljedećim svojstvima**:
  - postoji specijalan čvor koji se naziva **korijen** (eng. *root*)
  - ostali čvorovi raspoređeni su u disjunktne podskupove koji su stabla – i nazivaju se **podstabla**



Definicija stabla je rekurzivna.
Svako podstablo također je stablo.
To znači da su i lijevo i desno podstablo (na slici) također stabla. Npr.



Stablo – terminologija

stepen čvora (eng. *node degree*):
broj podstabala koji kreću iz tog čvora
npr. stepen(a)=2, stepen(c)=3

list (eng. *leaf*) = terminalni čvor
skup terminalnih čvorova: T={h, i, e, f, j, k}
skup neterminalnih čvorova: N={a, b, c, d, g}

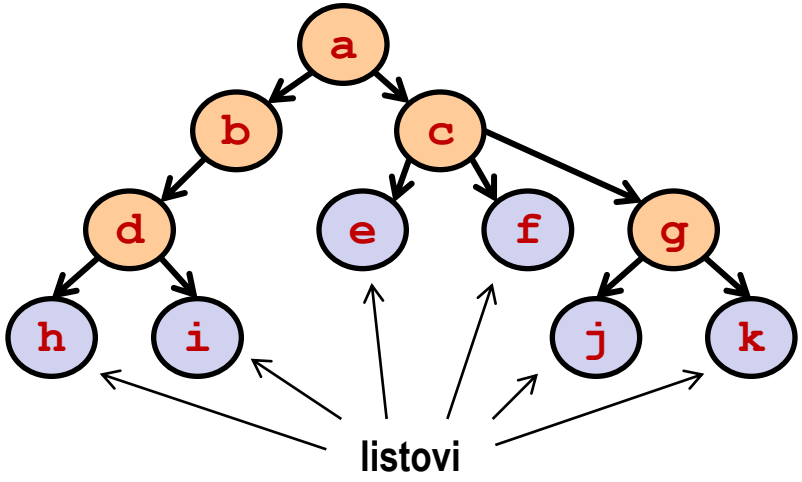
djeca/sinovi (eng. *children*) čvora=
korijeni podstabala datog čvora

dati čvor naziva se roditelj/otac (eng. *parent*)

npr. djeca(a)={b, c}, djeca(c)={e, f, g}
npr. roditelj(b)=a, roditelj(j)=k

preci čvora = svi čvorovi kroz koje se mora proći od korijena stabla do datog čvora
npr. preci(h)={d, b, a}, preci(e)={c, a}

potomci čvora = čvorovi u stablu kojem je dati čvor korijen
npr. potomci(b)={d, h, i}



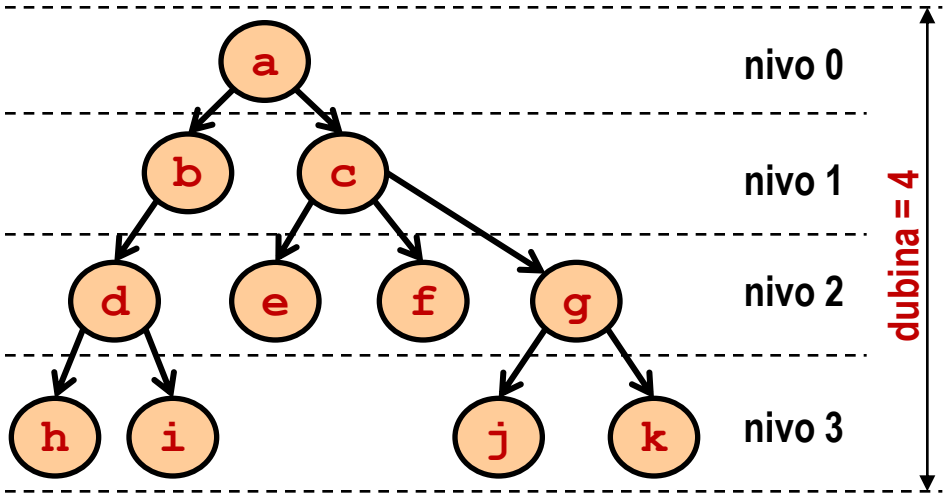
brat čvora = čvor koji ima istog roditelja
npr. braća(e)={f, g}, braća(d)=∅

Stablo – terminologija (nastavak)

stepen čvora (eng. *node degree*):
broj podstabala koji kreću iz tog čvora
npr. stepen(a)=2, stepen(c)=3

stepen stabla(eng. *tree degree*) =
maksimalni stepen čvorova u datom stablu

npr. stepen stabla u primjeru je 3, jer je to najveći stepen čvorova u datom stablu, tj. stepen(c)=3
stepen(stablo)=2 ↔ binarno stablo



nivo čvora (eng. *node level*):

nivo(korijen)=0
nivo(djeca(roditelj))=nivo(roditelj)+1

npr. nivo(a)=0, nivo(b)=1, nivo(d)=2, ...

alternativno se može uzeti da je nivo(korijen)=1

dubina stabla (eng. *tree depth*) =
ukupan broj nivoa, tj. maksimalni nivo čvorova (uvećan za 1)

npr. dubina stabla u primjeru je 4

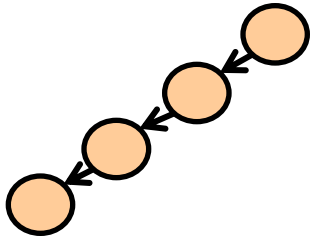
Binarno stablo

Binarno stablo: stepen(stablo)=2

stepen svakog čvora nije veći od 2 (svaki čvor može da ima nijednog, jednog ili najviše dva djeteta)

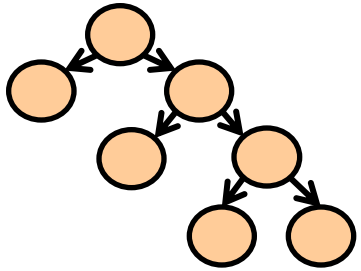
degenerisano (*skewed*) stablo

stablo sa jednim čvorom po nivou



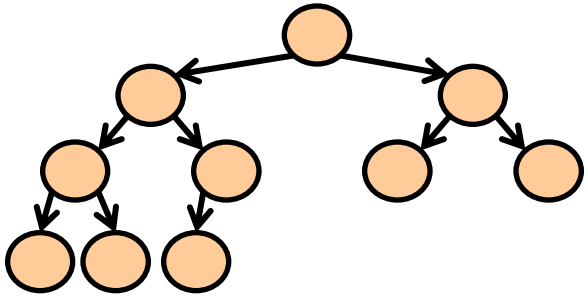
puno (*full*) stablo

svaki čvor grananja ima oba djeteta



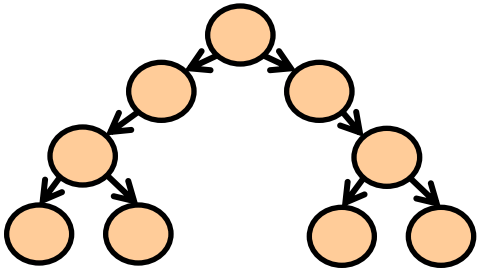
skoro kompletno stablo

svi nivoi su puni, osim posljednjeg u kojem su čvorovi maksimalno lijevo



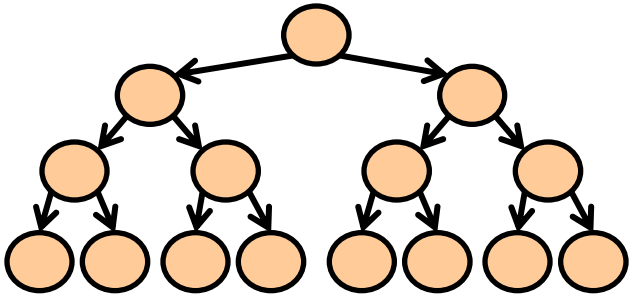
savršeno (*perfect*) stablo

svi listovi su na istom nivou

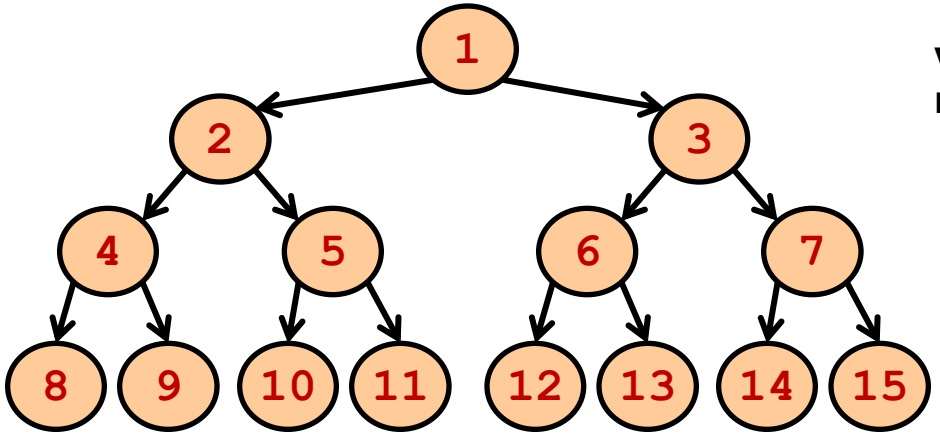


kompletno (*complete*) stablo

svi nivoi su puni



Binarno stablo



veze između čvorova (osnov za sekvencijalnu reprezentaciju binarnog stabla):

lijevi sin: $2i$, $2i \leq n$

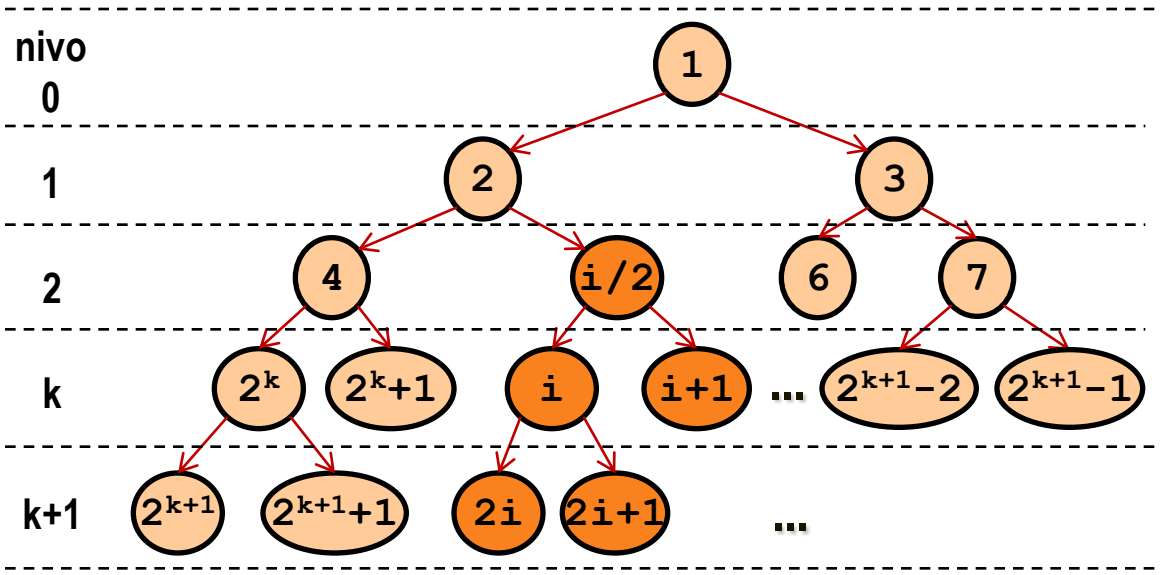
desni sin: $2i+1$, $2i+1 \leq n$

otac: $\lfloor i/2 \rfloor$

maksimalan broj čvorova:

na nivou k: $n_{max(k)}=2^k$

u stablu dubine m: $n_{max}=2^m-1$



Sekvencijalna reprezentacija stabla

veze između čvorova (osnov za sekvencijalnu reprezentaciju binarnog stabla):

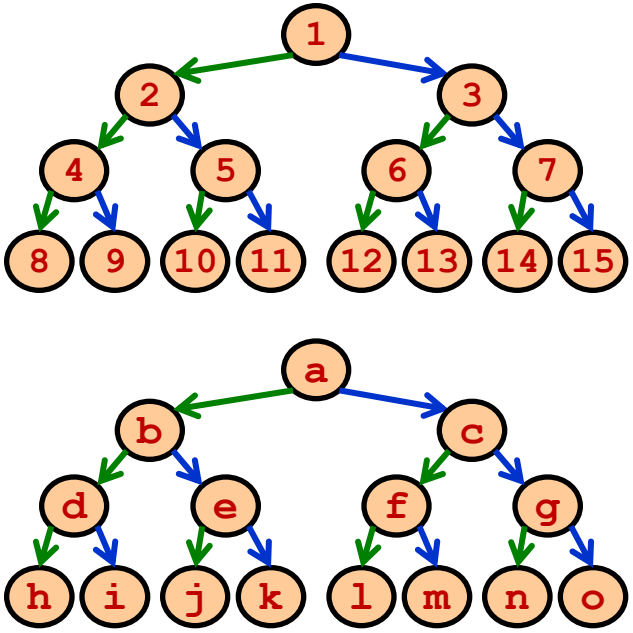
lijevi sin: $2i$, $2i \leq n$

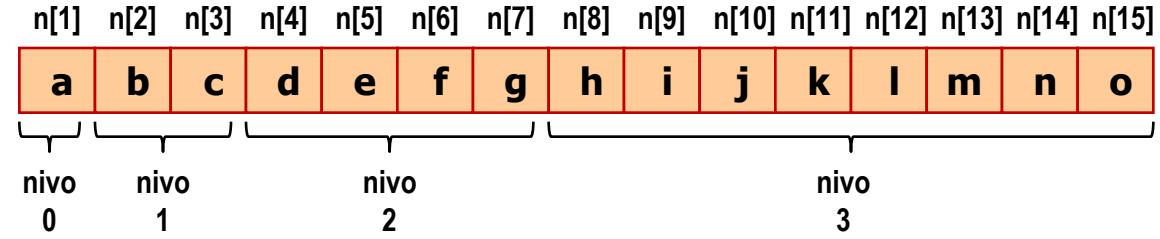
desni sin: $2i+1$, $2i+1 \leq n$

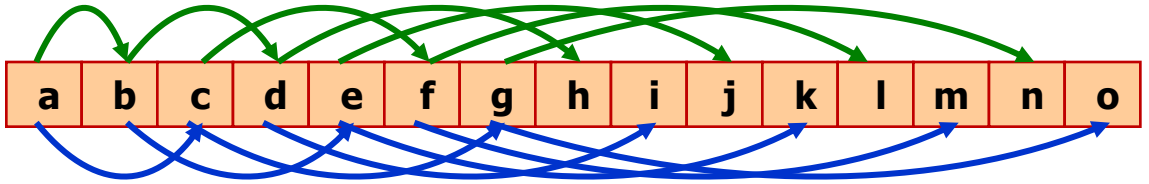
otac: $\lfloor i/2 \rfloor$

Niz omogućava veoma jednostavnu reprezentaciju binarnog stabla, jer se veze između roditelja i djece lako reprezentuju rasporedom čvorova u elemente sa odgovarajućim indeksima.

Radi jednostavnije reprezentacije, indeksi idu od 1.

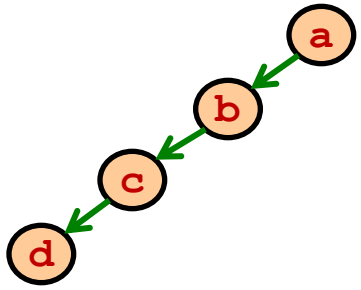


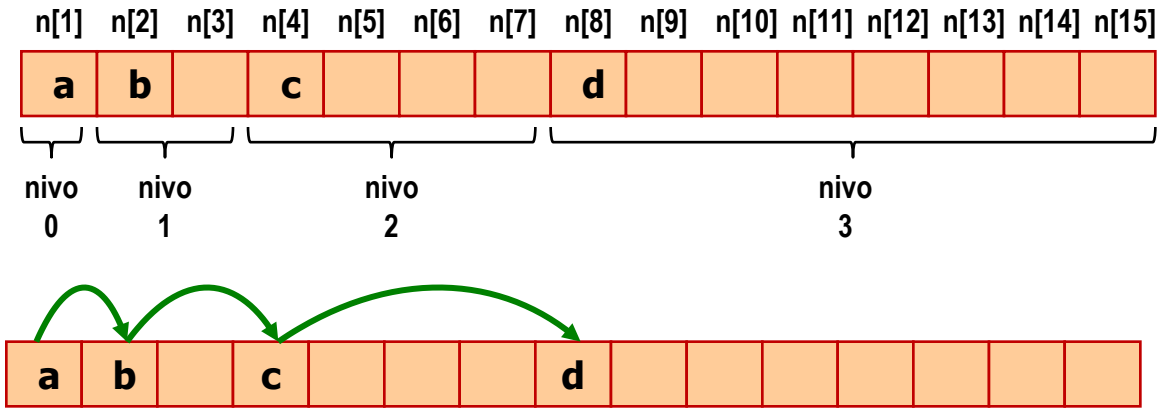




Sekvencijalna reprezentacija stabla

Sekvencijalna reprezentacija degenerisanog stabla





Degenerisano stablo je najgori slučaj

Ako je dubina=k, popunjeno je k elemenata niza od ukupno 2^k-1

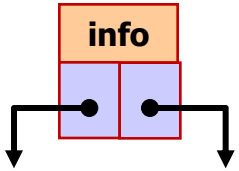
Za k=4, popunjenost: 4/15

Mane sekvencijalne reprezentacije:

- promjena strukture stabla (ubacivanje i izbacivanje čvorova) može da nameće brojna pomjeranja elemenata u nizu
- često niska popunjenost (pogotovo u slučaju degenerisanog stabla)

Ulančana reprezentacija stabla

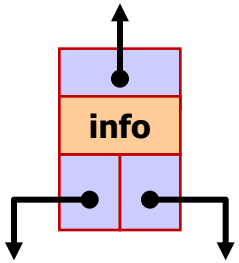
- **Ulančana reprezentacija**
  - dinamička alokacija
  - pristup elementima je indirektan (pokazivači)



- osnovni element: **ČVOR** (eng. *node*)
  - informacioni sadržaj
  - pokazivači

```
typedef struct node {
                        <tip> info;
                        struct node *left_child;
                        struct node *right_child;
                      } NODE;
```

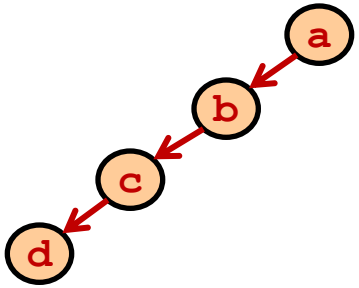
- **Ponekad postoji i pokazivač prema roditeljskom čvoru**

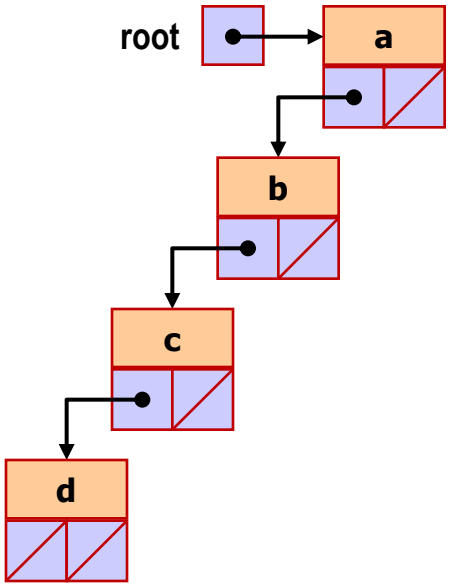



```
typedef struct node {
                        <tip> info;
                        struct node *left, *right, *parent;
                      } NODE;
```

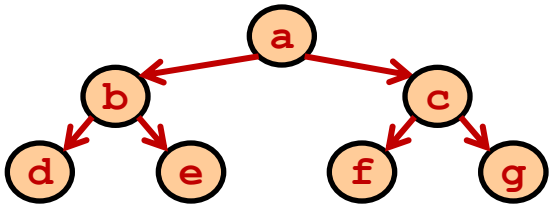
Ulančana reprezentacija stabla

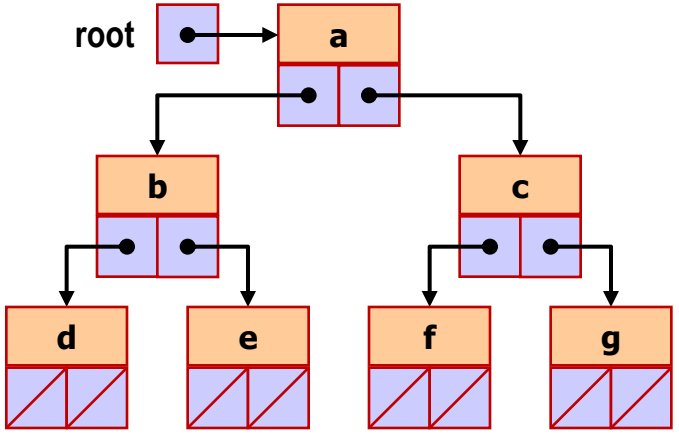
Ulančana reprezentacija degenerisanog stabla





Ulančana reprezentacija kompletnog stabla

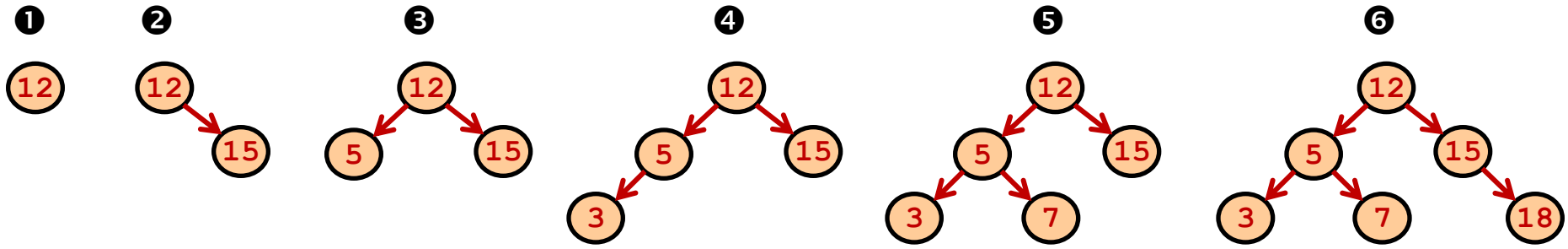




Stablo binarnog pretraživanja

- **Stablo binarnog pretraživanja/pretrage (*binary search tree* - BST)**:
  - često se naziva sortirano / uređeno stablo (na osnovu ključa u informacionom dijelu čvora)
  - **Dodavanje novog čvora započinje pretragom od korijena i poređenjem ključeva**
  - **Ako je ključ čvora (koji se dodaje) manji od ključa u tekućem čvoru, poređenje se nastavlja sa lijevim djetetom**
  - **Ako je ključ čvora (koji se dodaje) veći od ključa u tekućem čvoru, poređenje se nastavlja sa desnim djetetom**
  - **Ako tekući čvor nema zahtijevanog potomka (lijevog, odnosno desnog), novi čvor se dodaje na mjesto nedostajućeg potomka**

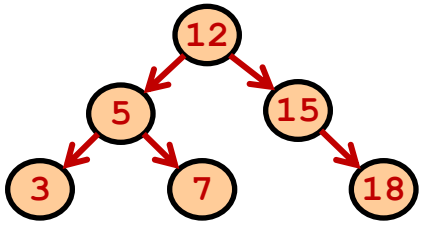
Primjer: Formirati stablo binarnog pretraživanja na osnovu sljedećeg niza: 12, 15, 5, 3, 7, 18



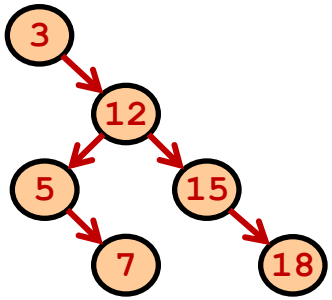
Stablo binarnog pretraživanja

STRUKTURA STABLA ZAVISI OD POLAZNOG RASPOREDA ELEMENATA U NIZU
(različiti polazni rasporedi u nizu rezultuju različtim stablom).

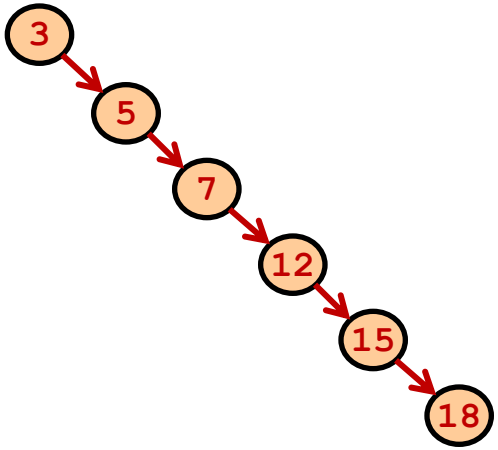
Primjer 1: 12, 15, 5, 3, 7, 18



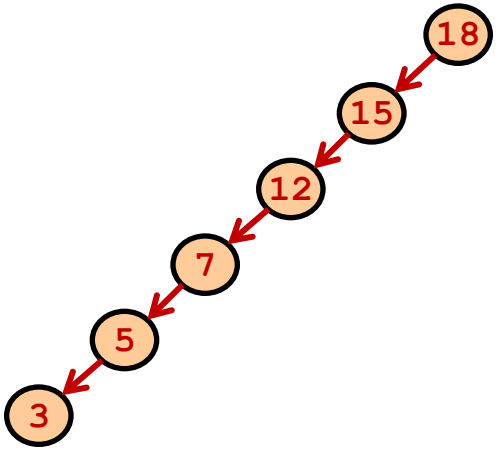
Primjer 2: 3, 12, 15, 5, 7, 18



Primjer 3: 3, 5, 7, 12, 15, 18

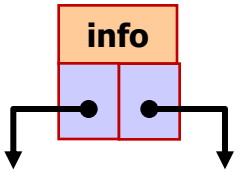


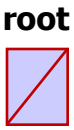
Primjer 4: 18, 15, 12, 7, 5, 3



Stablo binarnog pretraživanja

Formiranje stabla binarnog pretraživanja






```
typedef struct node
{
   <tip> info;
   struct node *left, *right;
} NODE;
```

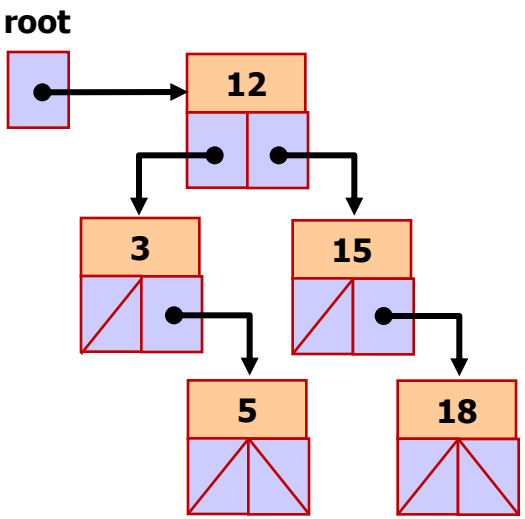
```
/* formiranje novog cvora */
NODE *newNode(<tip> info)
{
   NODE *q = (NODE*) malloc(sizeof(NODE));
   if (q==NULL) return NULL;
   q->left = q->right = NULL;
   q->info = info;
   return q;
}
```

```
/* primjer formiranja stabla */
int main()
{
   int niz[] = {12, 3, 15, 5, 18, 5};
   NODE *root = NULL;
   //...

   return 0;
}
```

Stablo binarnog pretraživanja

Formiranje stabla binarnog pretraživanja

```
/* dodavanje cvora u BST */
NODE *add(NODE *node, <tip> info)
{
  if (node == NULL)
     return newNode(info);
  if (info < node->info)
     node->left = add(node->left, info);
  else if (info > node->info)
     node->right = add(node->right, info);
  return node;
}
```



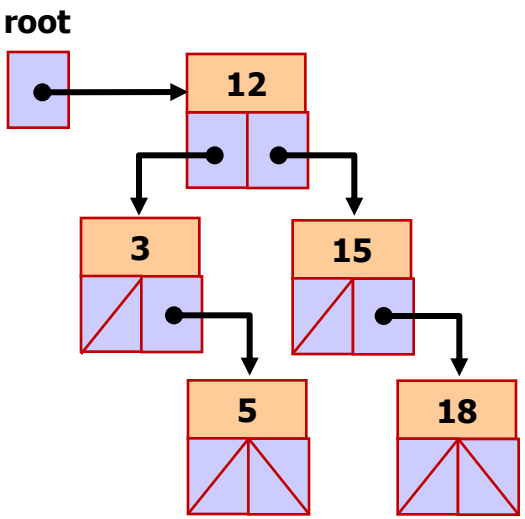

```
/* primjer formiranja stabla */
int main()
{
   int niz[] = {12, 3, 15, 5, 18, 5};
   NODE *root = NULL;
   for (int i=0; i<6; i++)
      root = add(root, niz[i]);
   return 0;
}
```

Stablo binarnog pretraživanja

Pretraživanje stabla binarne pretrage

```
/* pretrazivanje BST */
NODE *searchNode(NODE *node, <tip> kljuc)
{
   if (node == NULL)
      return NULL;
   else
      if (kljuc == node->info)
         return node;
      else
         if (kljuc < node->info)
            return searchNode(node->left, kljuc);
         else
            return searchNode(node->right, kljuc);
}
```

Funkcija searchNode() pronalazi i vraća pronađeni čvor, odnosno NULL ako čvor nije pronađen



Ako pokazivač ne pokazuje na čvor – traženi podatak nije pronađen

Ako informacioni dio čvora sadrži traženi ključ – traženi podatak je pronađen

Ako je traženi ključ manji od ključa u čvoru – rekurzivno se provjerava lijevo podstablo

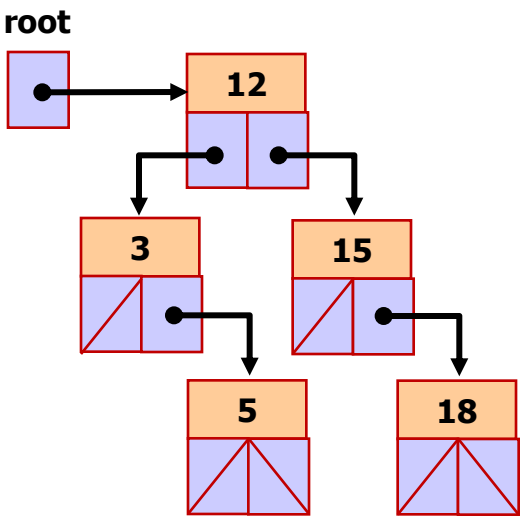
Ako je traženi ključ veći od ključa u čvoru – rekurzivno se provjerava desno podstablo

Stablo binarnog pretraživanja

Pretraživanje stabla binarne pretrage

```
/* pretrazivanje BST */
NODE *searchNode(NODE *node, <tip> kljuc)
{
   if (node == NULL)
      return NULL;
   else
      if (kljuc == node->info)
         return node;
      else
         if (kljuc < node->info)
            return searchNode(node->left, kljuc);
         else
            return searchNode(node->right, kljuc);
}
```

Funkcija searchNode() pronalazi i vraća pronađeni čvor, odnosno NULL ako čvor nije pronađen



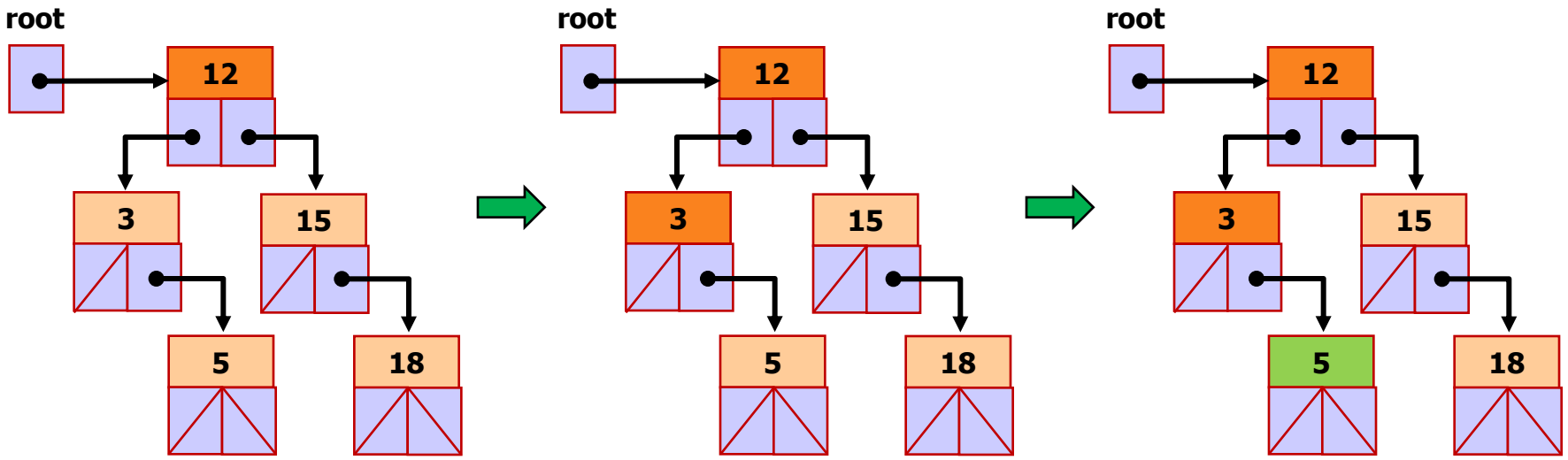

```
/* primjer pretrazivanja BST */
...
for (int i=1; i<100; i++)
   if (searchNode(root,i))
      printf("%d ", i);
...
```

```
3 5 12 15 18
```

Stablo binarnog pretraživanja

Pretraživanje stabla binarne pretrage

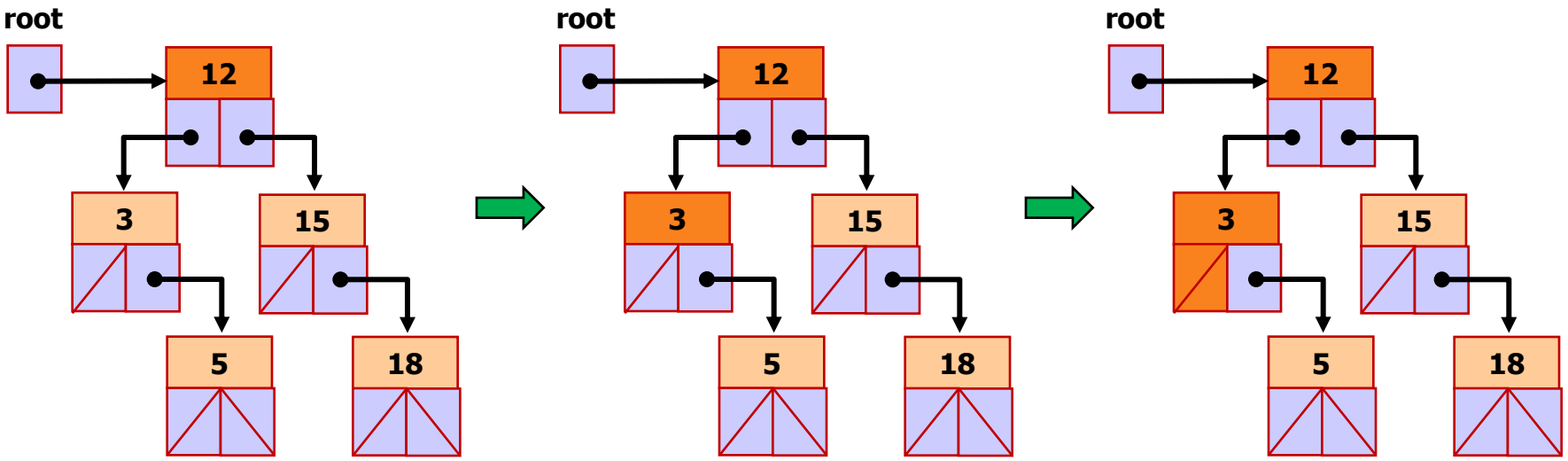
```
/* primjer uspjesne pretrage BST */
searchNode(root,5)
```



Stablo binarnog pretraživanja

Pretraživanje stabla binarne pretrage

```
/* primjer neuspjesne pretrage BST */
searchNode(root,2)
```



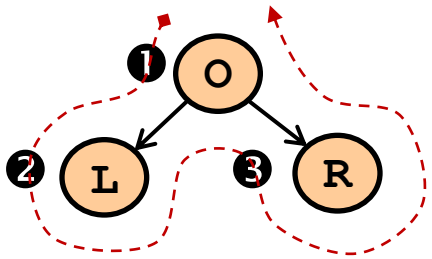
Obilazak stabla

Obilazak (eng. *traversal*) stabla:

- sistematična procedura kojom se svaki čvor u stablu obiđe (posjeti) samo jednom
- uobičajeni načini obilaska:
  - **po dubini**: **pre-order, in-order, post-order**

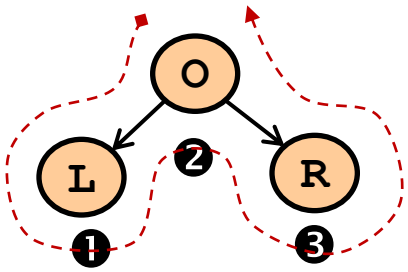
order = LEFT - RIGHT

Pre-order
(OTAC → left → right)



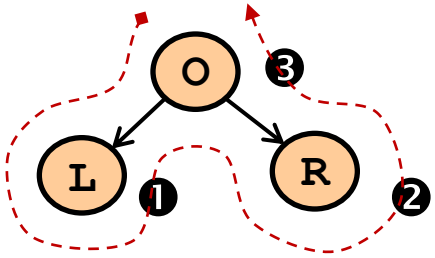
O-L-R

In-order
(left → OTAC → right)



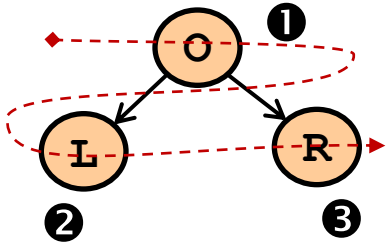
L-O-R

Post-order
(left → right → OTAC)



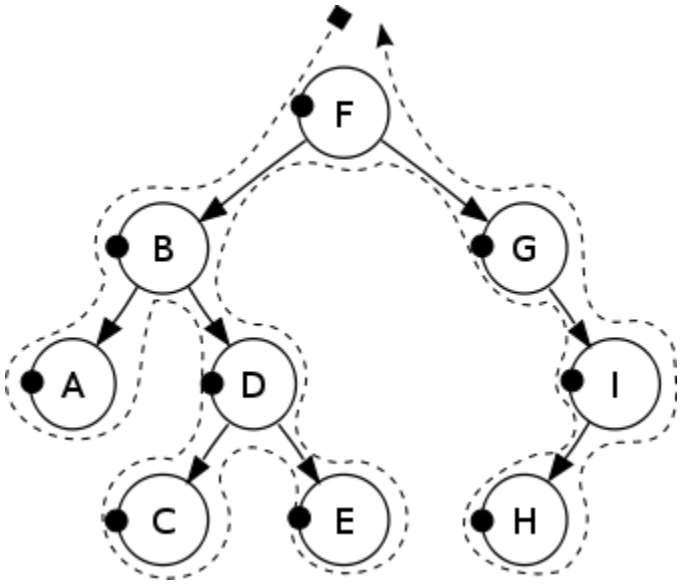
L-R-O

  - **po širini** (nivo po nivo)

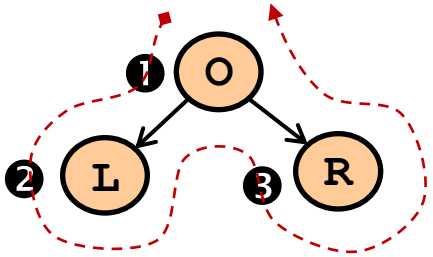


Pre-order obilazak stabla

Pre-order obilazak (OTAC → left → right)



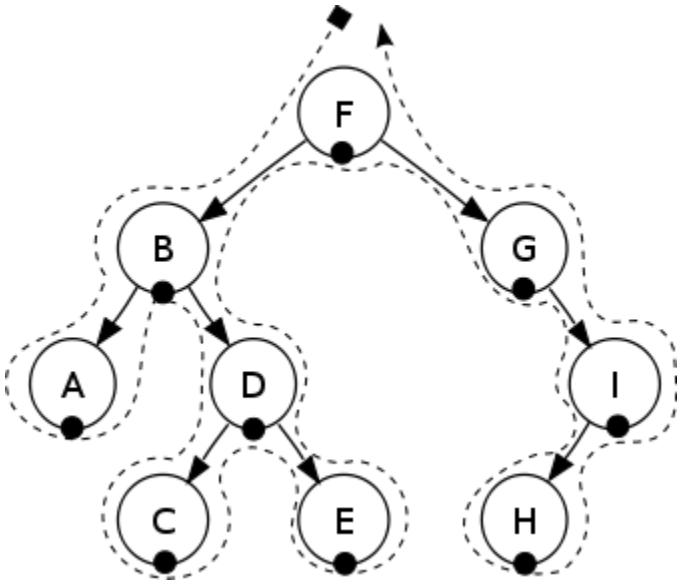
Pre-order: F, B, A, D, C, E, G, I, H



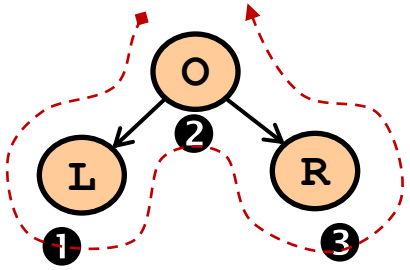

```
/* rekurzivni pre-order obilazak stabla */
void preorder(NODE *node)
{
   if (node != NULL)
   {
      /* obrada oca */
      ...

      /* obilazak podstabala */
      preorder(node->left);
      preorder(node->right);
   }
}
```

In-order obilazak stabla

In-order obilazak (left → OTAC → right)



In-order: A, B, C, D, E, F, G, H, I

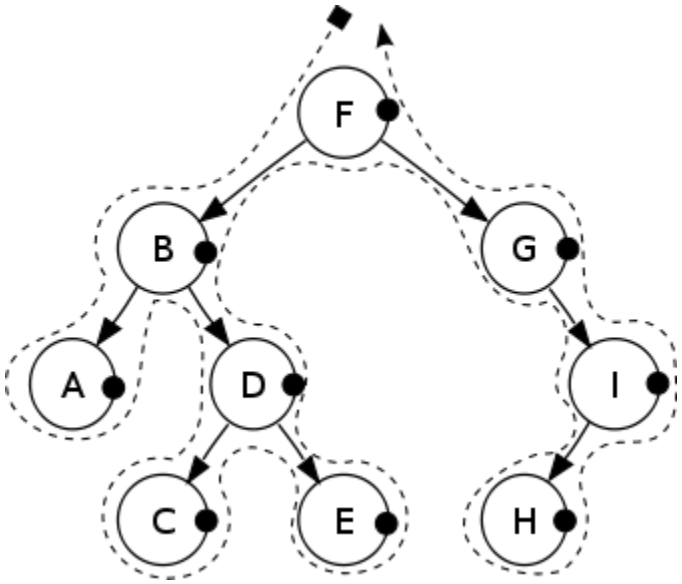



```
/* rekurzivni in-order obilazak stabla */
void inorder(NODE *node)
{
   if (node != NULL)
   {
      inorder(node->left);

      /* obrada oca */
      ...

      inorder(node->right);
   }
}
```

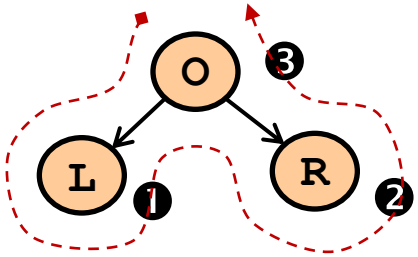
Ako je stablo sortirano (kao što je BST), in-order obilaskom stabla dobija se sortirani raspored čvorova.

Post-order obilazak stabla

Post-order obilazak (left → right → OTAC)



Post-order: A, C, E, D, B, H, I, G, F

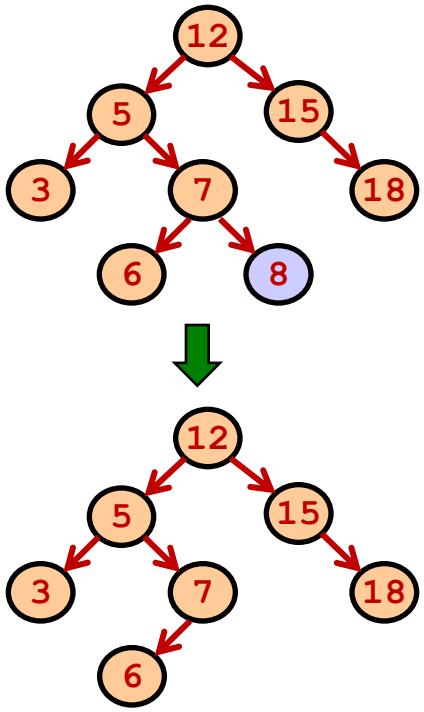



```
/* rekurzivni post-order obilazak stabla */
void postorder(NODE *node)
{
   if (node != NULL)
   {
      postorder(node->left);
      postorder(node->right);

      /* obrada oca */
      ...
   }
}
```

Brisanje čvora iz stabla

Tipične situacije:

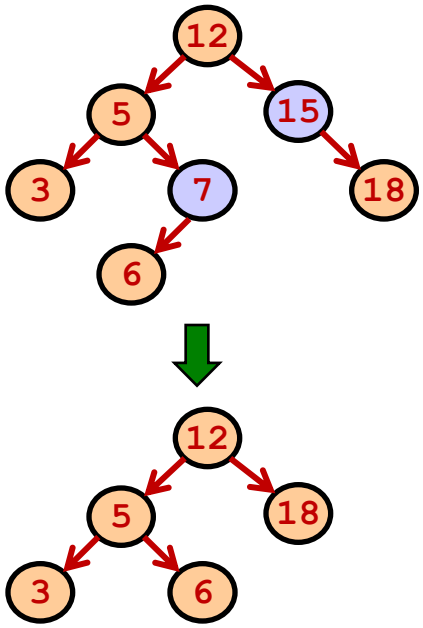
terminalni čvor



najjednostavniji slučaj

Treba obrisati dati čvor, a pokazivač u roditeljskom čvoru postaviti na NULL

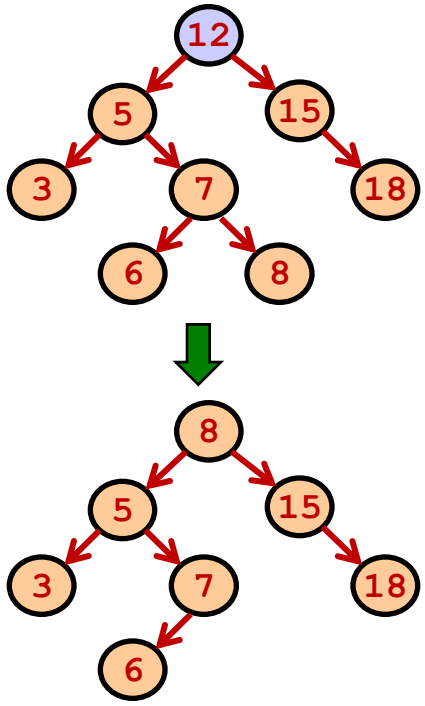
čvor sa jednim djetetom



slično brisanju čvora iz liste

Treba obrisati dati čvor, a pokazivač u roditeljskom čvoru postaviti da pokazuje na sina obrisanog čvora

čvor sa dva djeteta



različite tehnike za brisanje

npr. u dati čvor iskopirati informacioni sadržaj čvora sa najvećom vrijednošću u lijevom podstablu i obrisati taj terminalni čvor

(ili najmanji iz desnog podstabla)

Brisanje čvora iz stabla

```
NODE *deleteNode(NODE *node, char kljuc)
{
   if (node == NULL) return NULL;
   if (kljuc < node->info)    // pretrazivanje i brisanje u lijevom podstablu
      node->left = deleteNode(node->left, kljuc);
   else
      if (kljuc > node->info)  // pretrazivanje i brisanje u desnom podstablu
         node->right = deleteNode(node->right, kljuc);
      else    // brisanje datog cvora
         if (!node->left && !node->right) // ako nema oba sina
           {  free(node); return NULL; }
         else if (node->left == NULL)   // ako nema lijevog sina
           {  NODE *q = node->right; free(node); return q;  }
         else if (node->right == NULL)   // ako nema desnog sina
           {  NODE *q = node->left; free(node); return q;  }
         else   // ako ima oba sina
           {
              NODE *max = node->left;
              while (max->right) max=max->right;
              node->info = max->info;
              node->left = deleteNode(node->left, max->info);
           }
   return node;
}
```