Strukture podataka
Letnji semestar

Binarna Stabla (Binary Trees)

Prof. Dr Leonid Stoimenov
Katedra za računarstvo
Elektronski fakultet u Nišu

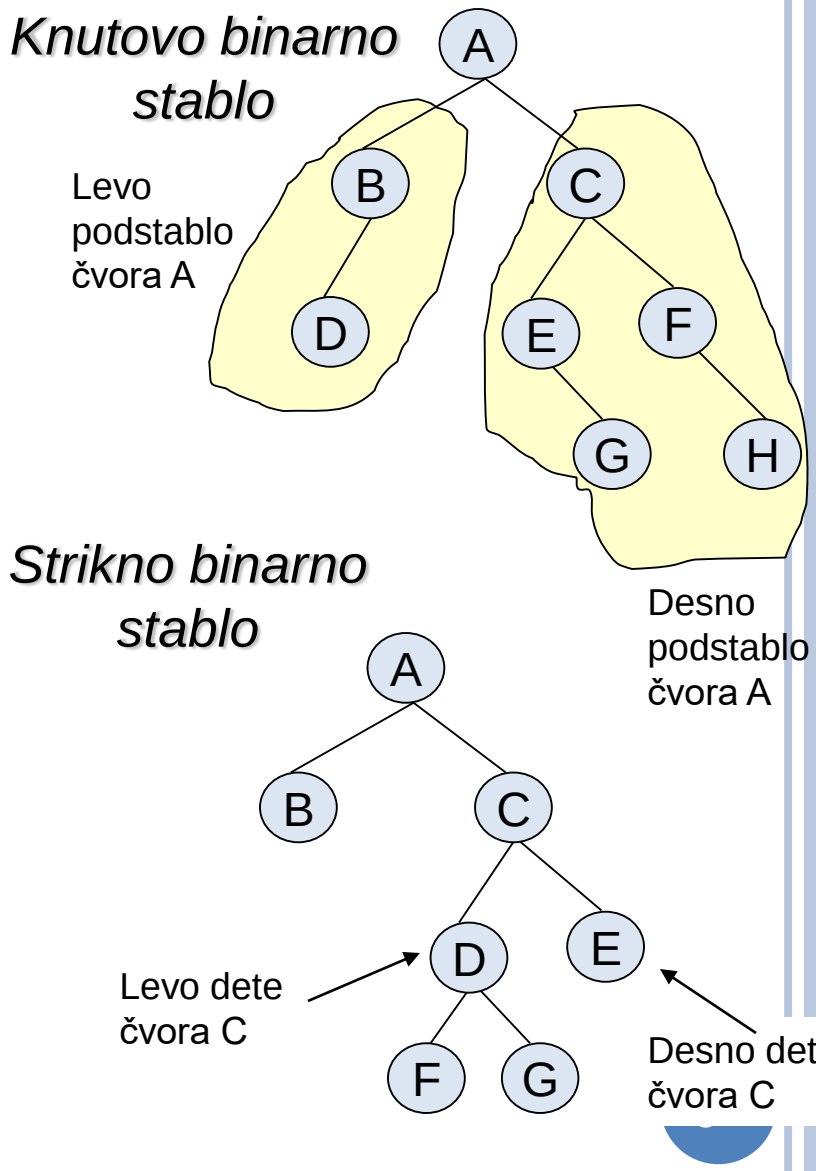
Binarna stabla - Pregled

- Definicija
- Vrste binarnih stabala
- Primeri
- Memorijska reprezentacija
- Implementacija
- Operacije

BINARNA STABLA - DEFINICIJA

- *Binarno stablo* je uređeno stablo gde svaki čvor ima **najviše *dvoje dece***
  - levo dete i desno dete
- Podstabla S1 i S2 čvora *u* nazivaju se **levo** (S1) i **desno** (S2) podstablo, a *u* je njihov **roditelj**
- Ovakvo binarno stablo se takođe naziva ***Knutovo binarno stablo***

- ***Striktno binarno stablo*** - u svakom čvoru ima 0 ili 2 podstabla ***(2-stablo, pravo binarno stablo)***



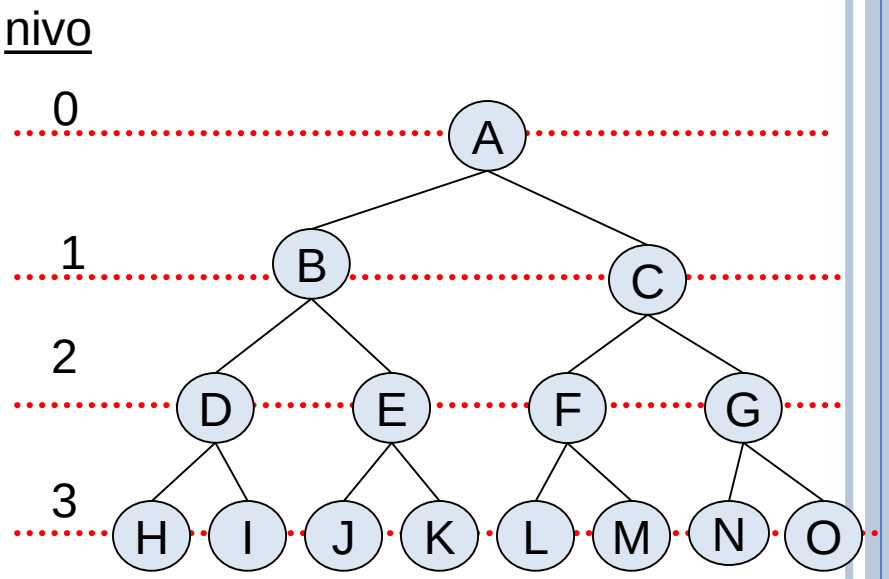
Kompletno binarno stablo

- **Kompletno** ili **potpuno binarno stablo** visine *h* je striktno binarno stablo čiji su svi listovi na nivou *h*
- **Ukupan broj čvorova** u kompletnom binarnom stablu visine *h* je

  $n = 2^0+2^1+2^2+\ldots+2^h$

  $n = \sum 2^j$ za j=0,h

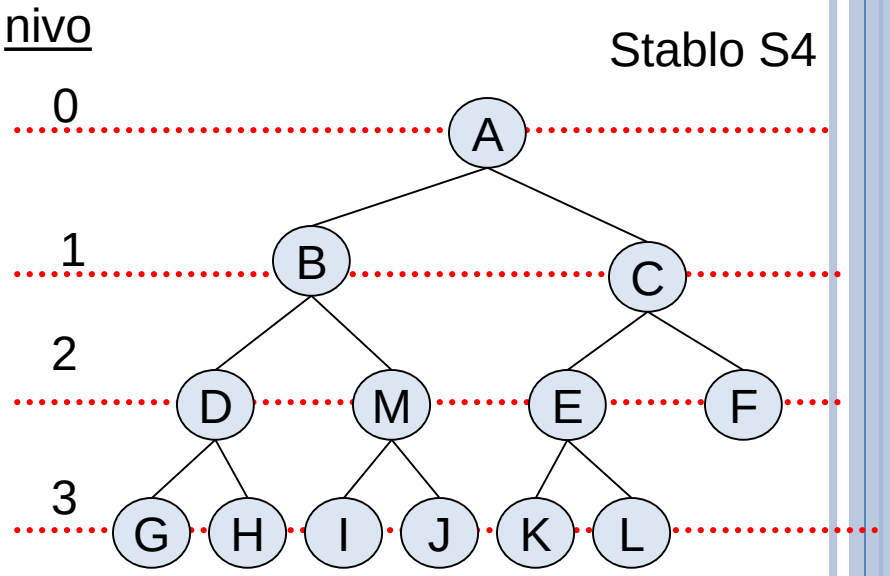
  $n = 2^{h+1} - 1$
- **Broj internih** čvorova: **2^h-1**
- **Broj eksternih** čvorova: **2^h**
- **Visina stabla** od n čvorova: **$h=\log_2(n+1)-1$**

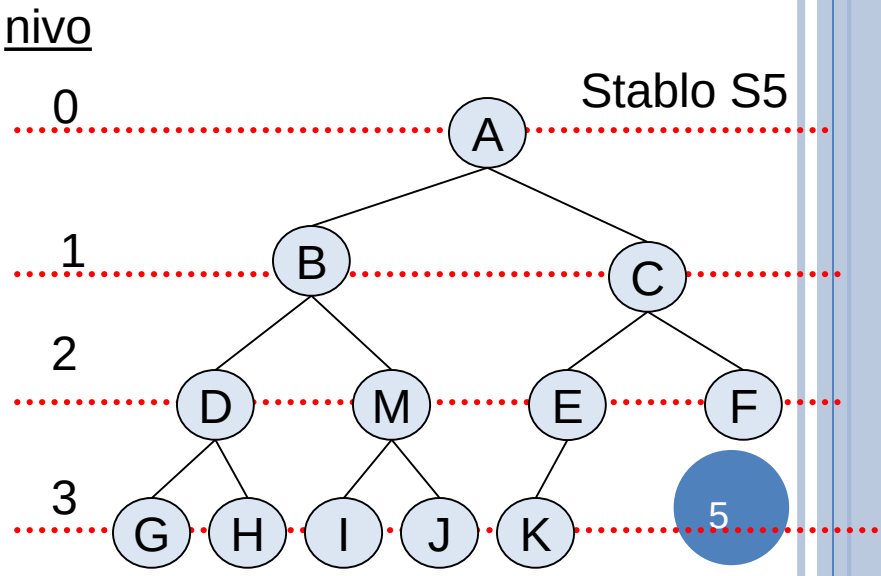


Nivo	Broj čvorova
0	$2^0 = 1$
1	$2^1 = 2$
2	$2^2 = 4$
h=3	$2^3 = 8$
n	**15**

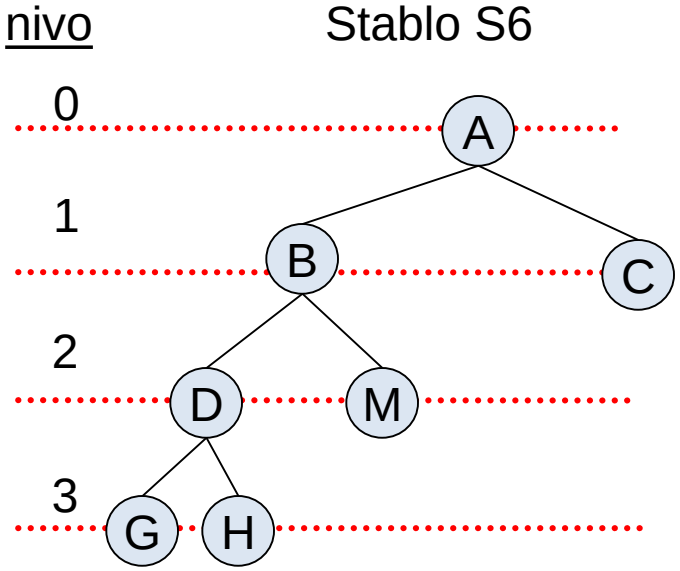
GOTOVO KOMPLETNO BINARNO STABLO

- ***Gotovo kompletno*** ili ***gotovo potpuno binarno stablo*** dubine d je binarno stablo čiji svi čvorovi ispunjavaju sledeća dva uslova:
  1. Svaki čvor *u* na nivou manjem od *d-1* ima dvoje dece
  2. Svaki čvor *u* koji ima desnog potomka na nivou *d* mora imati levo dete i sve leve potomke kao listove na nivou *d* ili mora imati dvoje dece
- **Primer 1:** S4 je gotovo kompletno binarno stablo i to striktno binarno stablo
- **Primer 2:** S5 je gotovo kompletno binarno stablo ali nije striktno binarno stablo

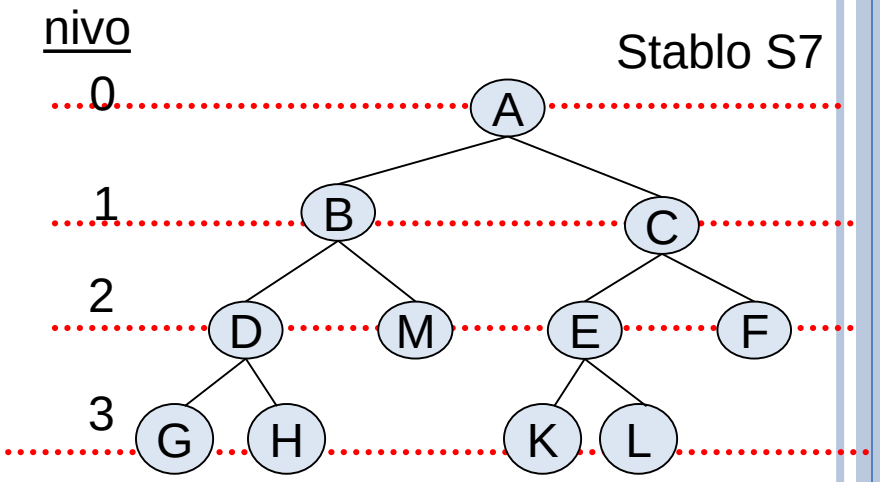




GOTOVO KOMPLETNO BINARNO STABLO

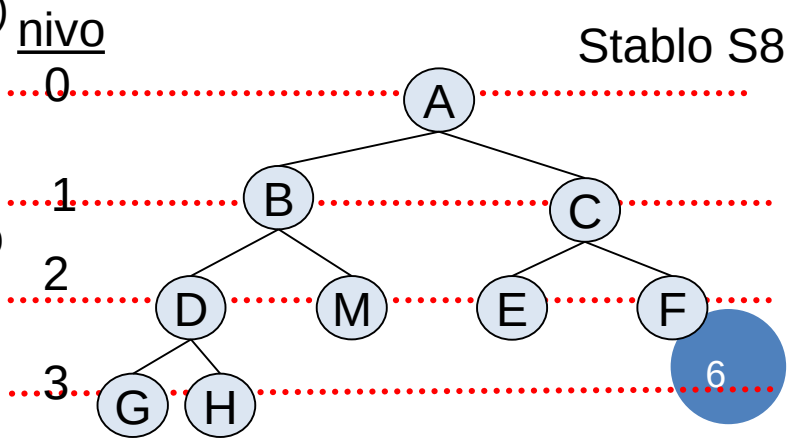


Nije gotovo kompletno binarno stablo
- Ne ispunjava uslov 1 (ima listove na nivoima 1,2 i 3)



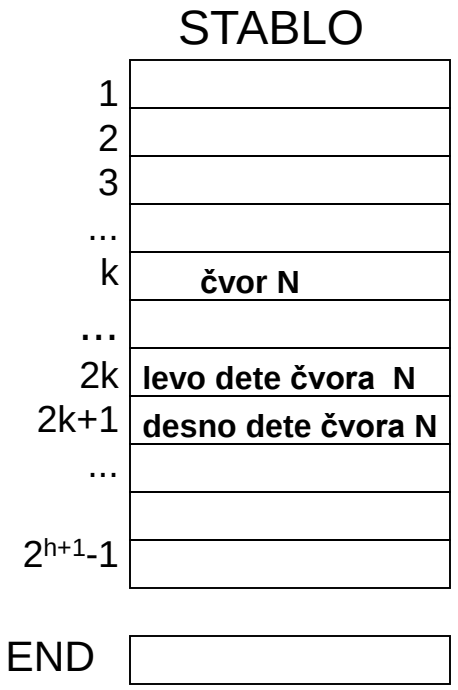
Nije gotovo kompletno binarno stablo
- Ispunjava uslov 1
- Ne ispunjava uslov 2 (K je desni potomak čvora A na nivou 3, a njegov levi potomak M je na nivou 2)

Gotovo kompletno binarno stablo
- Ispunjava oba uslova

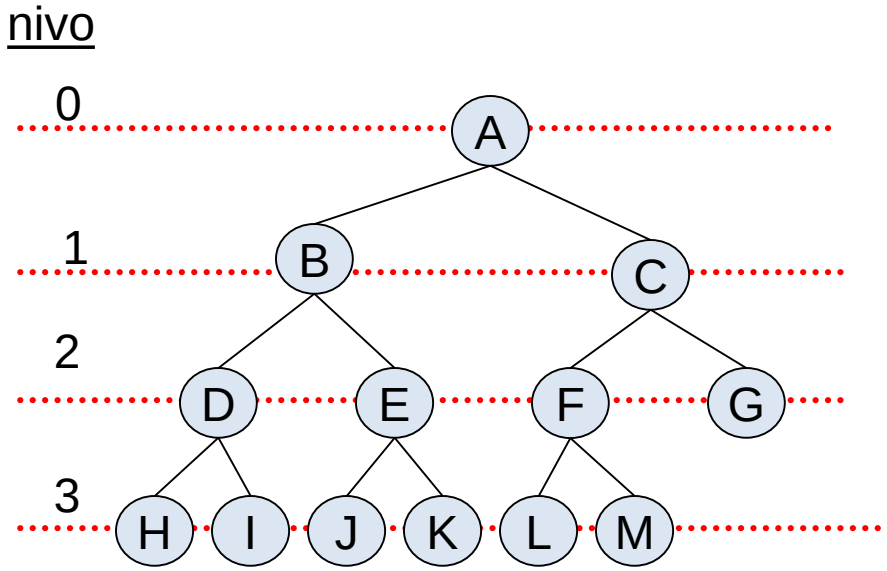


Memorijska reprezentacija binarnog stabla - sekvencijalna

- Koristi se
  - **Jednodimenzionalno polje** STABLO max dužine $2^{h+1}-1$
  - Promenljiva END koja pamti lokaciju poslednjeg čvora stabla
  - STABLO[1] pamti **koren stabla**
  - Ako STABLO [k] pamti čvor N, tada se levo dete čvora N nalazi u STABLO [**2k**], desno dete u STABLO [**2k+1**]
- Ova reprezentacija je pogodna za kompletno ili gotovo kompletno binarno stablo



PRIMER (1)

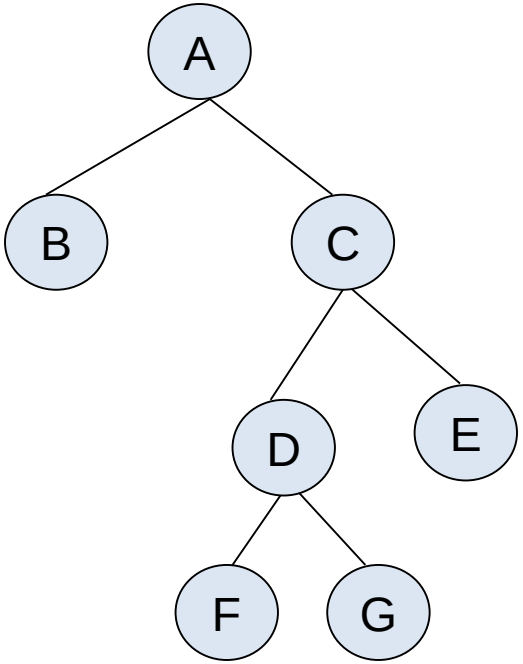


STABLO

1	A
2	B
3	C
4	D
5	E
6	F
7	G
8	H
9	I
10	J
11	K
12	L
13	M
14	
15	

END 13

Primer (2)



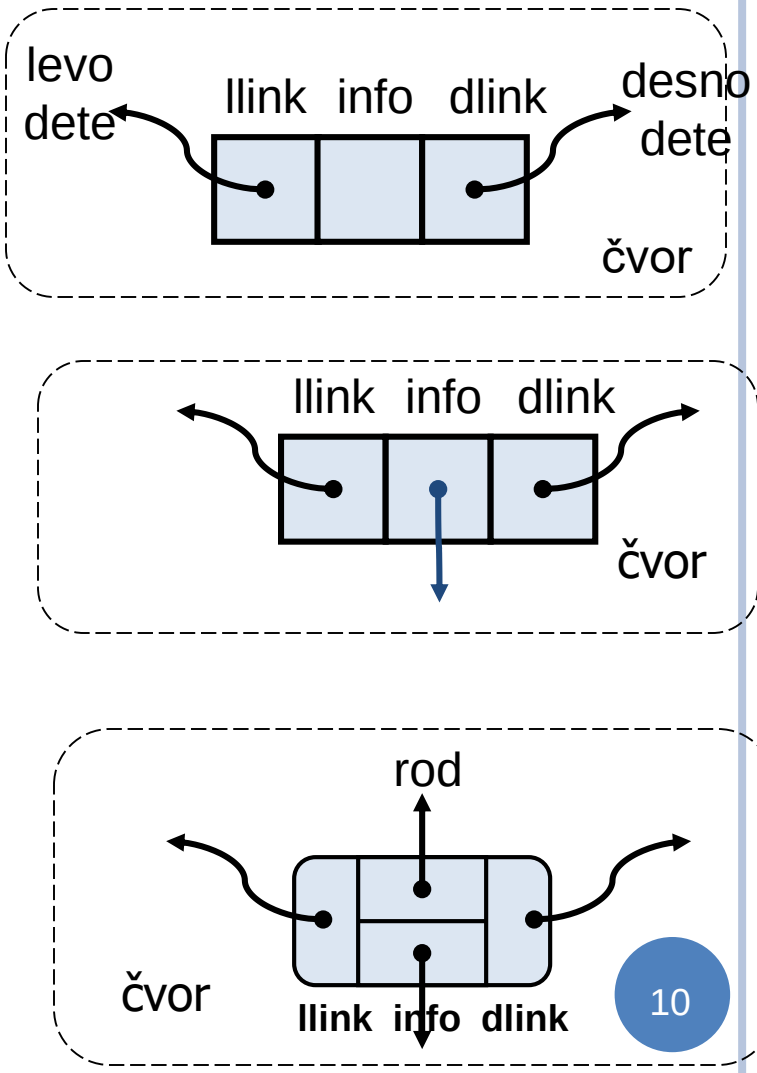
STABLO

1	A
2	B
3	C
4	
5	
6	D
7	E
8	
9	
10	
11	
12	F
13	G
14	
15	

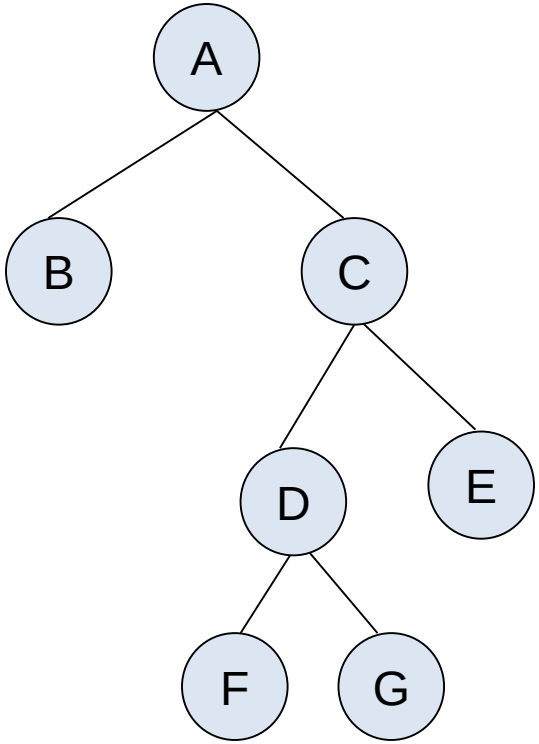
END	13

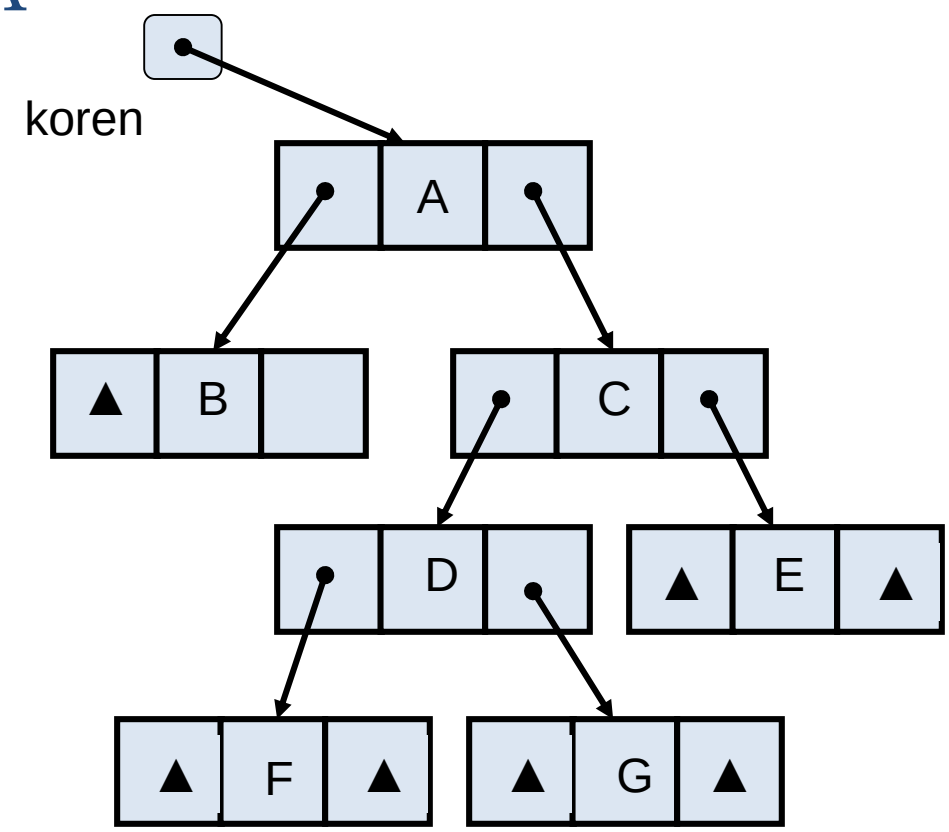
Memorijska reprezentacija binarnog stabla - lančana

- Koristi se dvostrana lančana lista
- **Varijanta 1**
  - INFO – element koji se pamti u čvoru
  - LLINK – pokazivač na levo dete
  - DLINK – pokazivač na desno dete
- **Varijanta 2**
  - INFO – pokazivač na element koji se pamti u čvoru
  - LLINK – pokazivač na levo dete
  - DLINK – pokazivač na desno dete
- **Varijanta 3**
  - Varijanti 1 ili 2 dodat link na roditeljski čvor
- Koristimo varijantu 1



Primer lančane reprezentacije binarnog stabla





Primitivne Operacije

(*p* pokazivač na čvor *nd)*

1. **info(p)** vraća sadržaj čvora *nd*
2. **left(p)** vraća levo dete čvora *nd*
3. **right(p)** vraća desno dete čvora *nd*

4. parent(p) vraća roditeljski čvor čvora *nd*
5. sister(p) vraća brata/sestru čvora *nd*

- Funkcije 2-5 vraćaju null pokazivač ukoliko ne postoji levo dete, desno dete, roditelj ili sestra/brat čvora *nd*

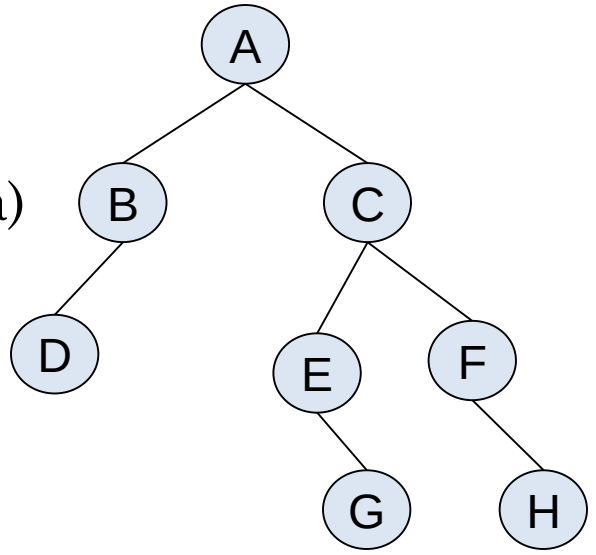
6. isleft(p) vraća vrednost TRUE ako je *nd* levo dete, inače vraća vrednost FALSE
7. isright(p) vraća vrednost TRUE ako je *nd* desno dete, inače vraća vrednost FALSE

Operacije sa binarnim stablima - kompozitne

- Kompozitne operacije
  1. **Obilazak binarnog stabla**
  2. Formiranje binarnog stabla
  3. Umetanje elementa u binarno stablo
  4. Brisanje čvora iz binarnog stabla
  5. Traženje elementa u binarnom stablu
  6. Spajanje dva binarna stabla
  7. ….

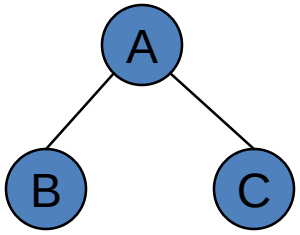
Obilazak stabla

- Kod obilaska binarnog stabla, svaki element se posećuje samo jednom
- Pri poseti jednog elementa/čvora stabla izvršavaju se sve neophodne operacije (kopiranje sadržaja, prikaz, evaluacija operatora i sl).
- Vrste obilaska stabla
  - **Preorder**
  - **Inorder**
  - **Postorder**
  - Level order (Obilazak po nivoima)
- Implementacija
  - Rekurzivna
  - Nerekurzivna

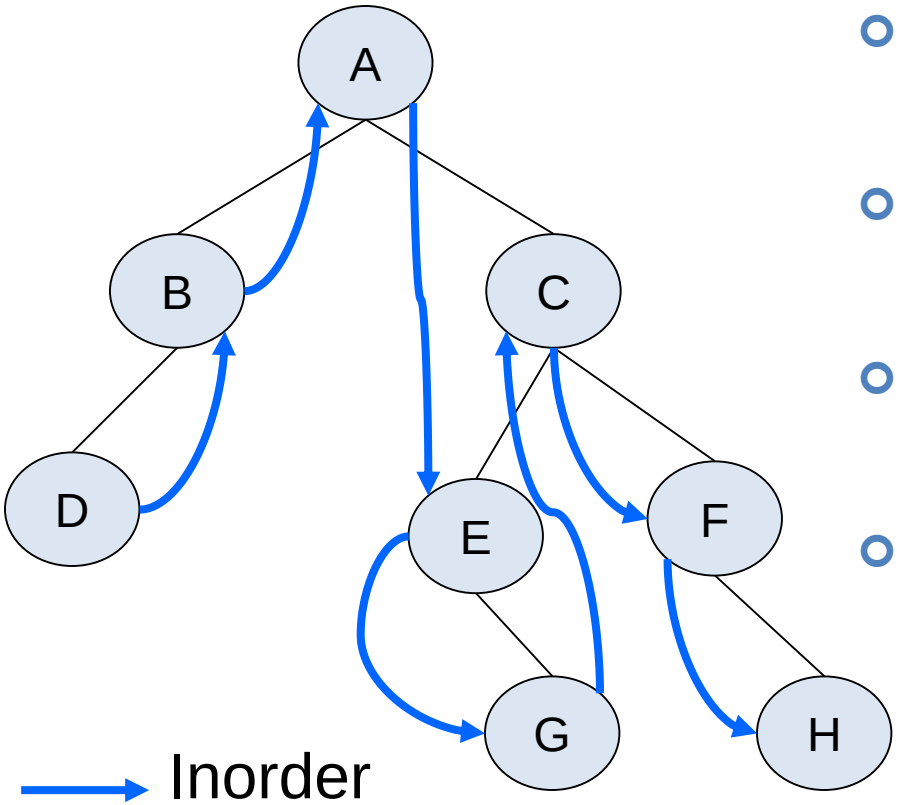


Obilazak binarnog stabla

- **Preorder**
  - Obraditi **koren**
  - Obići **levo** podstablo korena u preorder redosledu
  - Obići **desno** podstablo korena u preorder redosledu
  - A B C
- **Inorder**
  - Obići **levo** podstablo korena u inorder redosledu
  - Obraditi **koren**
  - Obići **desno** podstablo korena u inorder redosledu
  - B A C
- **Postorder**
  - Obići **levo** podstablo korena u postorder redosledu
  - Obići **desno** podstablo korena u postorder redosledu
  - Obraditi **koren**
  - B C A
- Obilazak po **nivoima**
  - Obilaze se čvorovi po **nivoima**, počev od nivoa 0 tj. korena
  - A B C



Primer obilaska binarnog stabla



- Preorder
  - ABDCEGFH
- Inorder
  - DBAEGCFH
- Postorder
  - DBGEHFCA
- Po nivoima
  - ABCDEFGH

PREORDER OBILAZAK STABLA

- Obilazak čvorova stabla na sistematski način
- Čvor se obilazi pre svih njegovih potomaka
- ***Primena:*** štampanje struktuiranog dokumenta

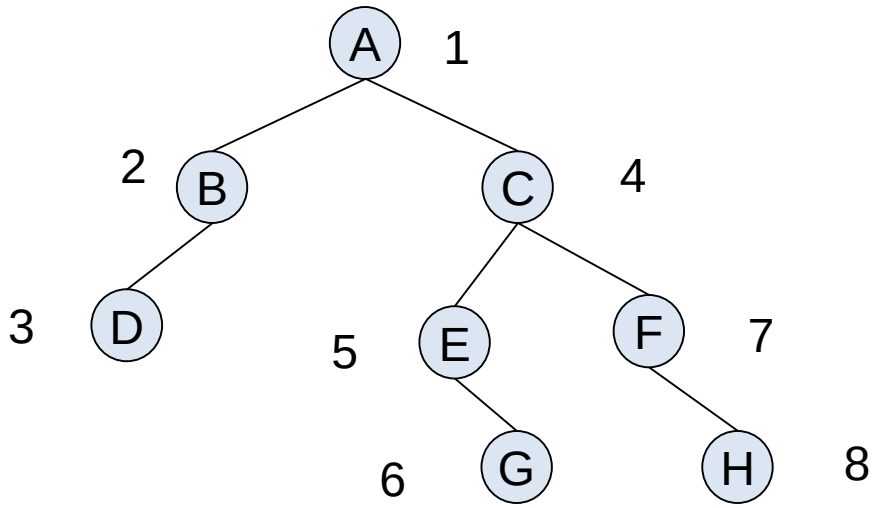
Algoritam S.1. Preorder obilazak

preOrder(v)

visit(v)

preorder (left(v))

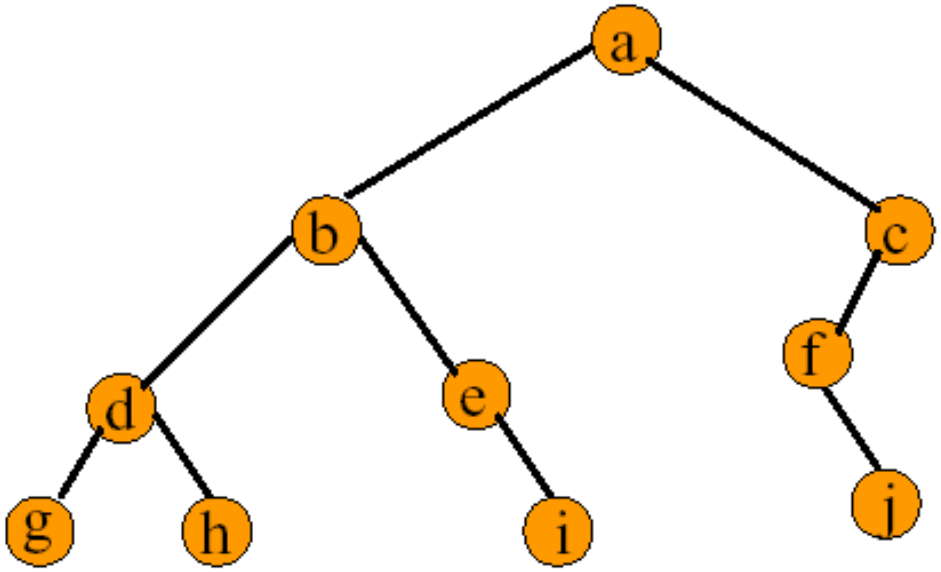
preorder (right(v))



Brojevi 1,2,… označavaju redosled obilaska čvorova

Primer preorder obilaska

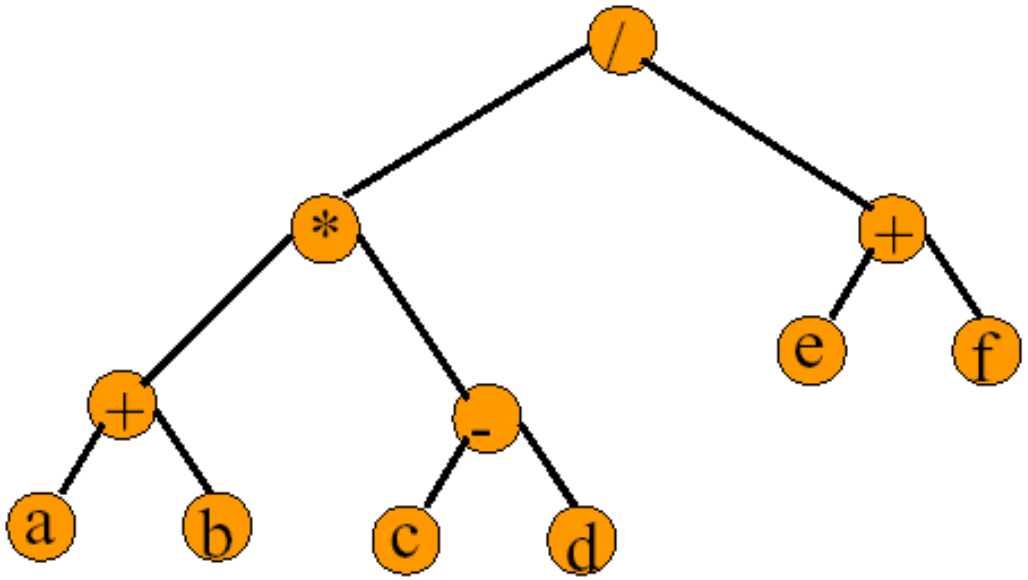
- Visit = print



a b d g h e i c f j

Preorder i artimetički izrazi

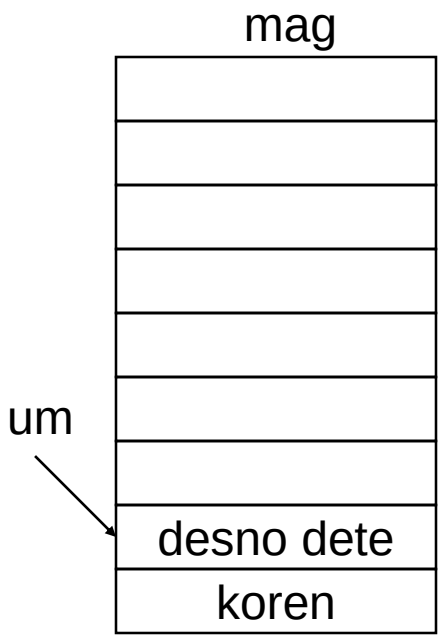
- Prefiks notacija izraza ((a+b)*(c-d))/(e+f)



/ * + a b - c d + e f

NEREKURZIVNI PREORDER

- Korišćenje magacina kao pomoćne strukture



pok - lokacija čvora koji se trenutno obrađuje
mag – pamti čvorove koje treba kasnije obrađivati (desnu decu)
um – vrh (ukazatelj) magacina

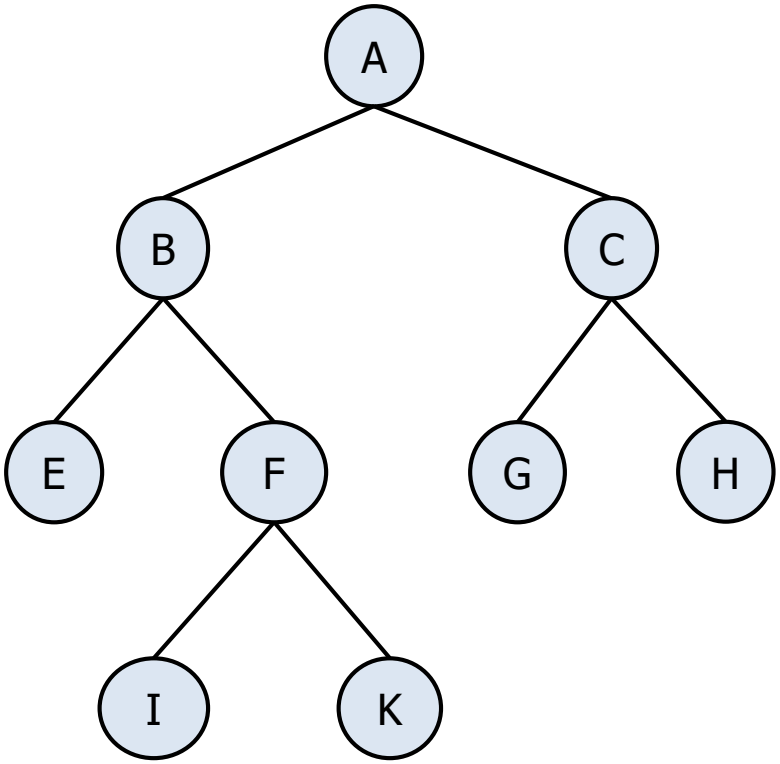
Algoritam S.2. Nerekurzivni preorder

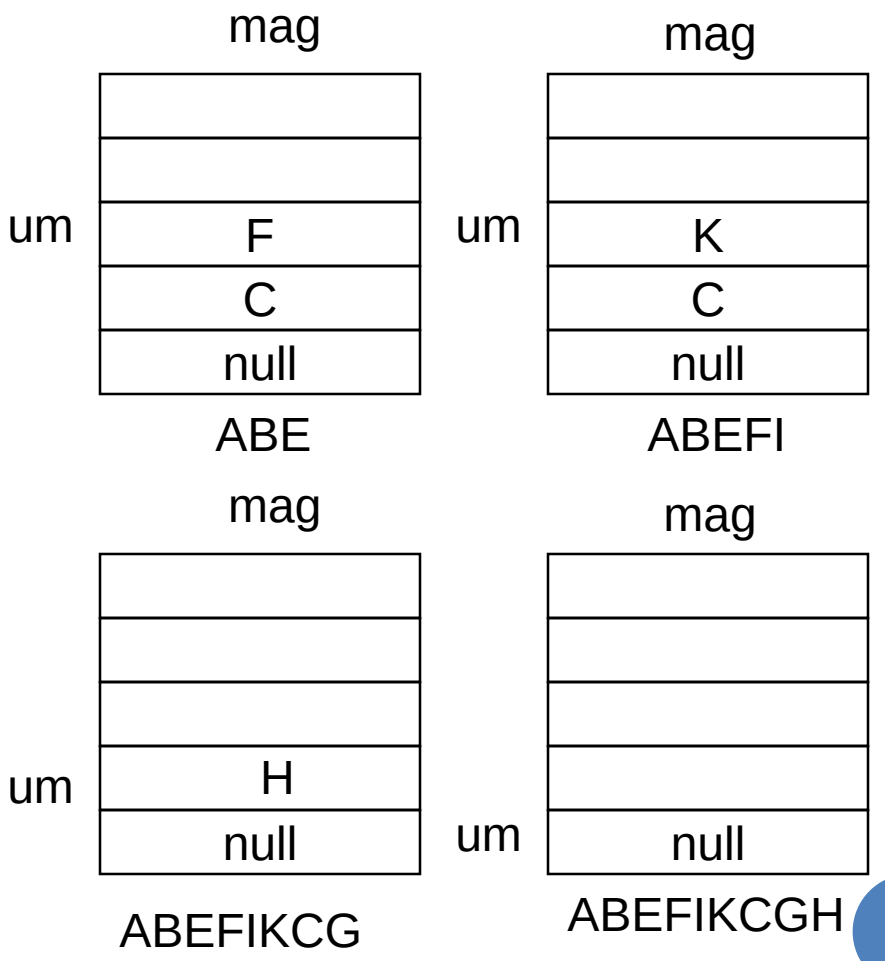
preOrderN(koren)

```
1.  um←1, mag[um] ← null, pok ← koren
    //Inicijalizacija
2.  repeat while (pok < > null)
3.      visit (pok)
4.      //da li postoji desno podstablo
5.      if (right(pok) <> null)
6.        then  {//upis u mag
7.              um ← um+1, mag[um] ← right(pok) }
8.      //da li postoji levo podstablo
9.      if (left(pok) <> null )
10.       then  pok ← left(pok)
11.       else {//čitanje iz magacina
12.         pok ← mag[um], um ← um-1 }
13. endrepeat
14.  end
```

Ilustracija rada nerekurzivnog preorder obilaska

- ABEFIKCGH



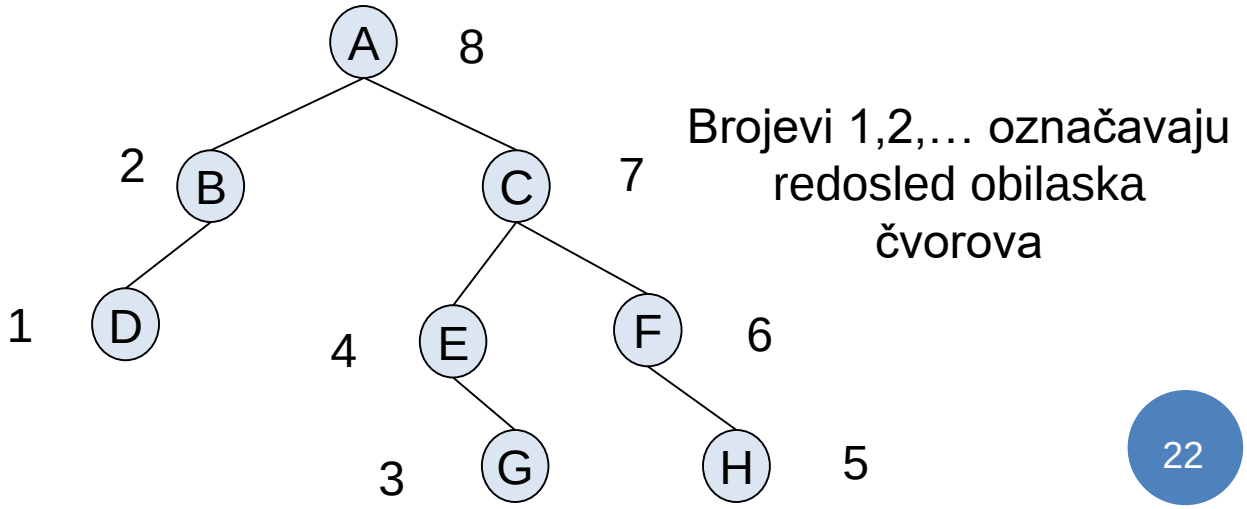


POSTORDER OBILAZAK

- Čvor se obilazi nakon obilaska njegovih potomaka
- **Primena:** izračunavanje prostora koji se koristi za fajlove u direktorijumu i njegovim pod direktorijumima

Algoritam S.3: Postorder obilazak

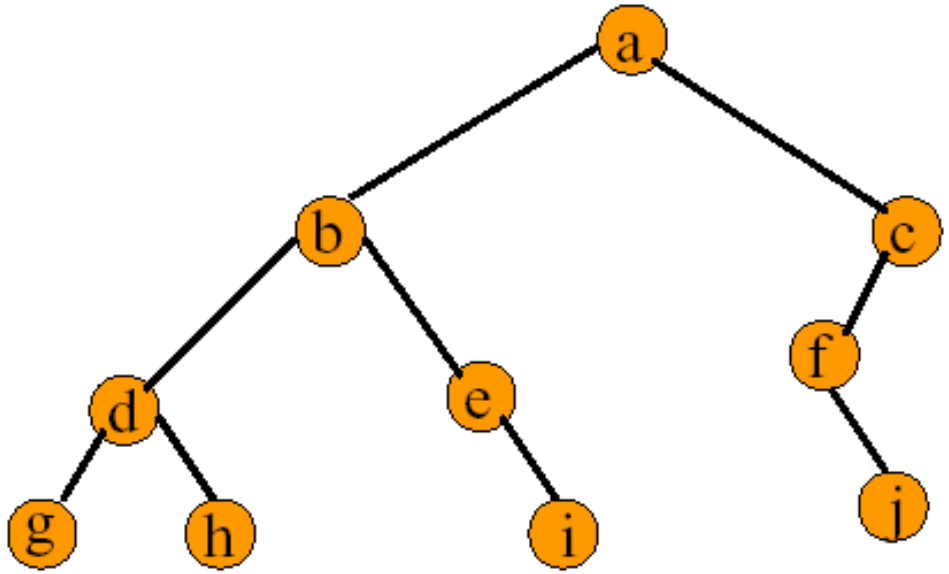
```
postOrder(v)
    postOrder (left(v))
    postOrder (right(v))
    visit(v)
```



Brojevi 1,2,… označavaju redosled obilaska čvorova

Primer postorder obilaska

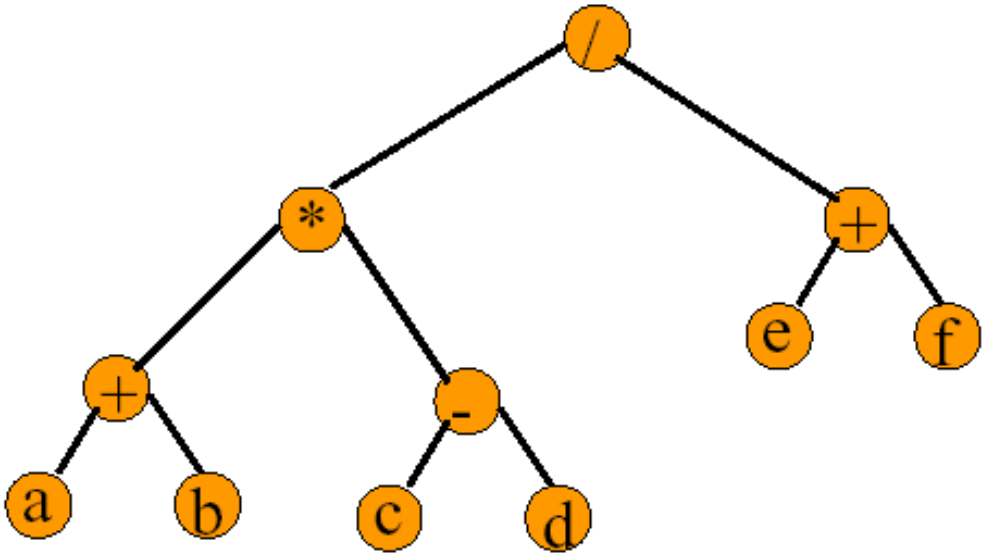
- Visit = print



g h d i e b j f c a

Postorder i artimetički izrazi

- Postfiks notacija izraza ((a+b)*(c-d))/(e+f)



a b + c d - * e f + /

NEREKURZIVNI POSTORDER

- Korišćenje **magacina** kao pomoćne strukture

A) Stablo se obilazi krajnjim levim putem i u magacin upisuju svi čvorovi;

Ako čvor ima desno dete u magacin se upisuje sa predznakom -

B) Povratak uz stablo;

Iz magacina se čita čvor;

Ako je pozitivan obrađuje se

Ako je negativan, povratak na A)

Algoritam S.5. Nerekurzivni postorder

postOrderN(koren)

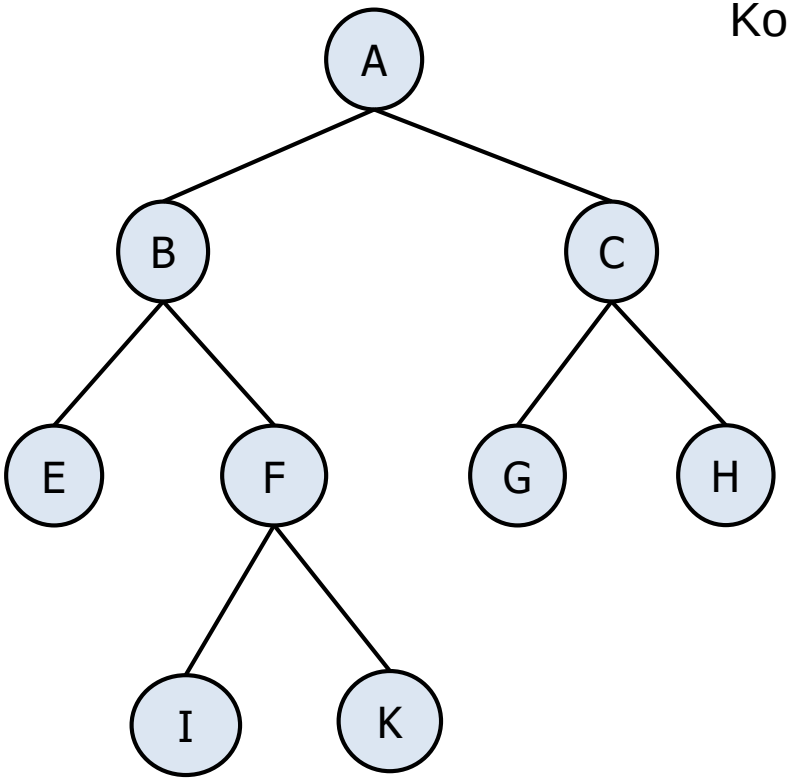
```
1.  um ← 1, mag[um] ← null, pok ← koren //Inicijalizacija
2.  repeat while (pok < > null)
3.    um ← um+1, mag[um] ← pok
4.    if (right(pok) <> null)
5.      then {um ← um+1, mag[um] ← - right(pok) }
6.    endif
7.    pok ← left(pok)
8.  endrepeat
9.  pok ← mag[um], um ← um-1 //čitanje iz mag.
10. repeat while (pok > 0)
11.   visit (pok)
12.   pok ← mag[um], um ← um-1
13. endrepeat
14. if (pok <0)
15.   then      { pok ← -pok, go to 2 }
16. endif
17. end
```

A)

B)

Ilustracija rada nerekurzivnog postorder obilaska

- EIKFBGHCA



Korak A)

um

E
-F
B
-C
A
null

um

I
-K
+F
B
-C
A
null

E

um

+K
F
B
-C
A
null

EI

G
-H
+C
A
null

EIKFB

INORDER OBILAZAK STABLA

- Čvor se obilazi nakon obilaska njegovog levog podstabla i pre obilaska desnog podstabla
- **Primena:** Crtanje binarnog stabla
  - x(v) = inorder rang za v
  - y(v) = dubina za v

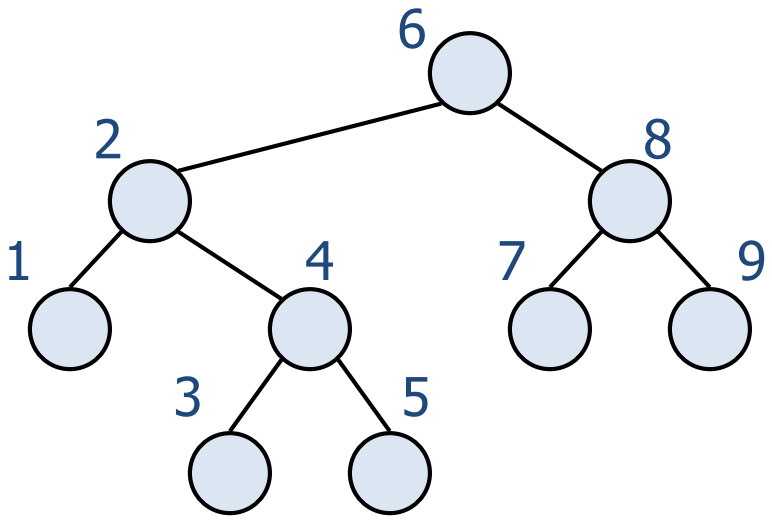
Algoritam S.6. Inorder obilazak

***inOrder*(*v*)**

***inOrder* (*left* (*v*))**

***visit*(*v*)**

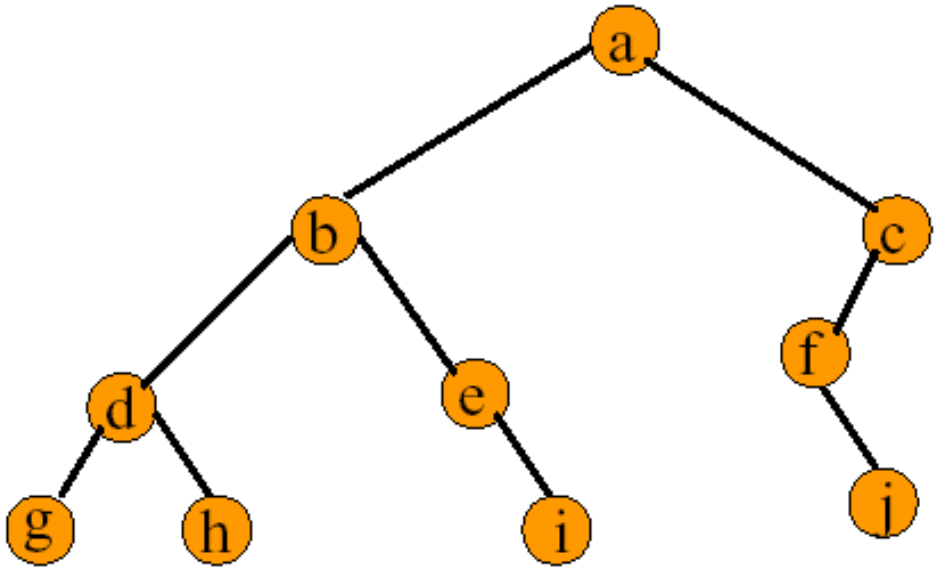
***inOrder* (*right* (*v*))**



Brojevi 1,2,… označavaju redosled obilaska čvorova

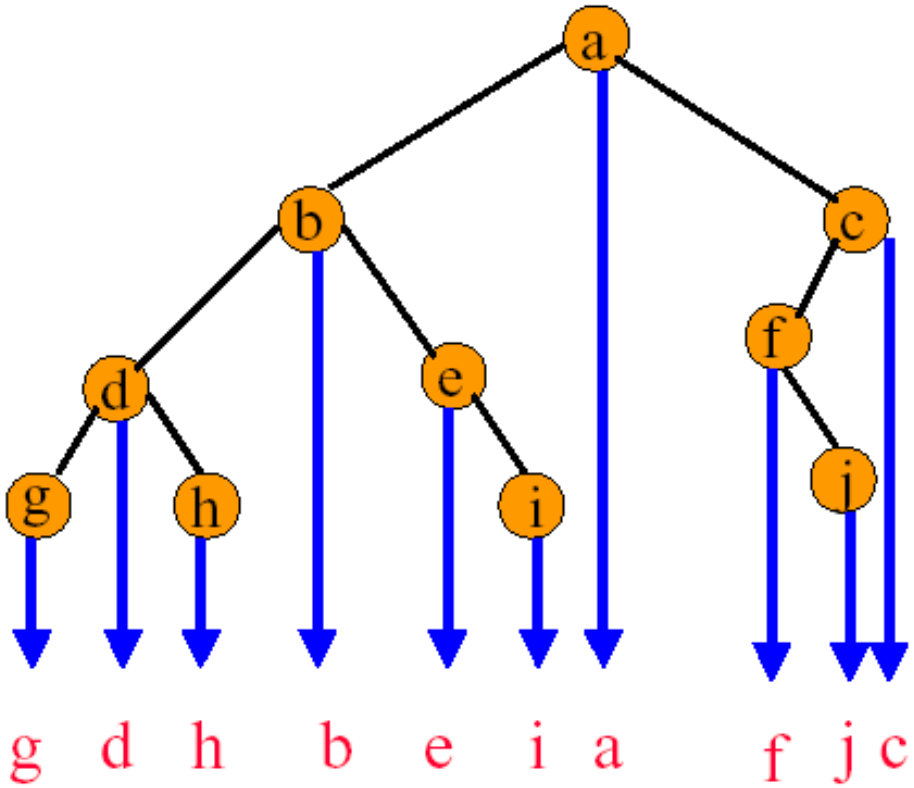
Primer inorder obilaska

- Visit = print



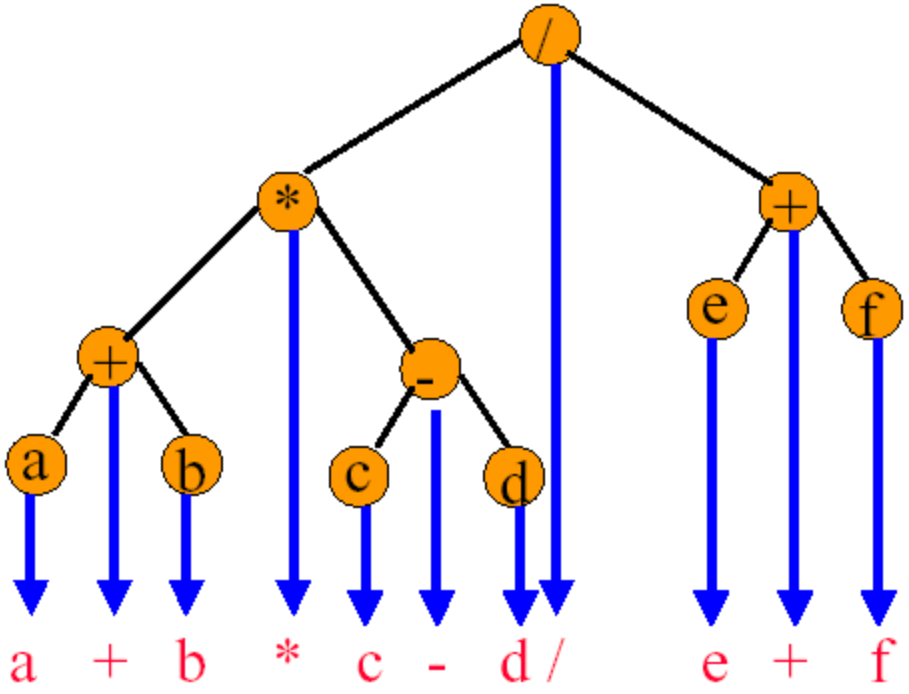
g d h b e i a f j c

Inorder po projekciji (Squishing)



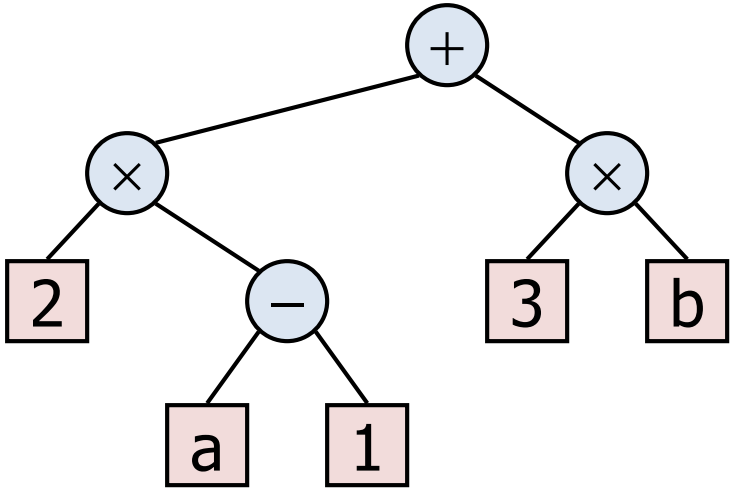
Inorder i artimetički izrazi

- Infiksna notacija izraza ((a+b)*(c-d))/(e+f)
- Da li su neophodne zagrade?



Primena inorder obilaska – prikaz aritmetičkih izraza

- Specijalizacija inorder obilsaka
  - Štampa operand ili operator kod obilaska čvora
  - štampa "(" pre obilaska levog podstabla
  - štampa ")" posle obilska desnog podstabla



Algoritam S.7. Prikaz izraza

```
printExpression(v)
    if isInternal (v)
        print("(")
        inOrder (left(v))
    print(info(v))
    if isInternal (v)
        inOrder (right(v))
        print (")")
```

$((2 \times (a - 1)) + (3 \times b))$

NEREKURZIVNI INORDER

Korišćenje **magacina** kao pomoćne strukture

Algoritam u dva "koraka":

A) Obilazak stabla krajnje levom putanjom, uz upis svakog čvora na koji se naiđe u magacin, sve dok se ne dođe do čvora koji nema levo podstablo

B) Kretanje unazad, uz stablo: čitanje elementa iz magacina i **obrada** sve dok u magacinu postoje elementi.

Obrada: prikaz, ili ako čvor ima desnog potomka *pok* se postavlja da ukazuje na njega, i obrada vraća na korak A)

Algoritam S.8. Nerekurzivni inorder

inOrderN(info, llink, dlink, koren)

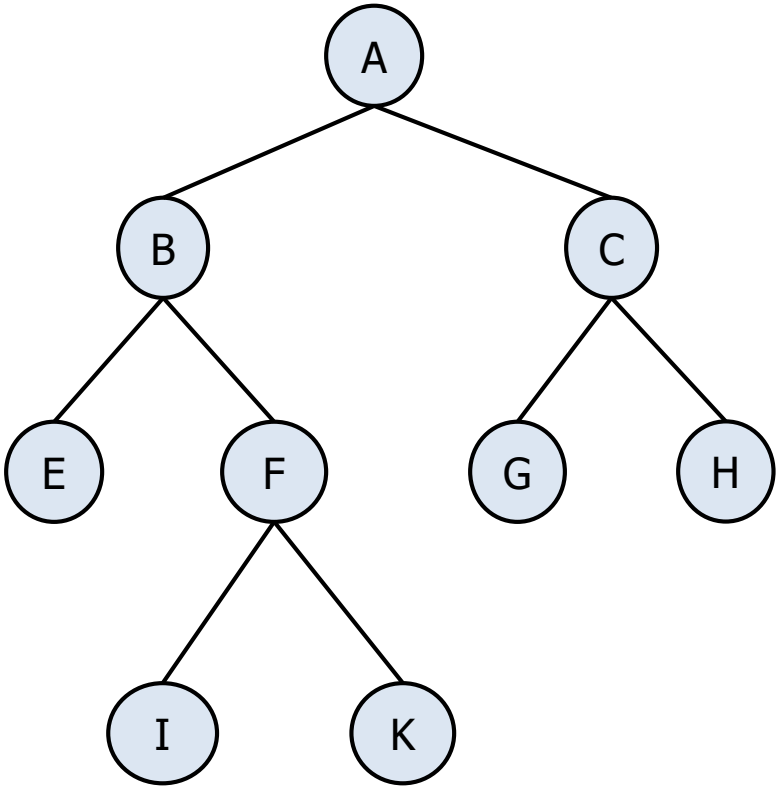
```
1.  um←1, mag[um]←null, pok←koren // Inicijaliz.
2.  repeat while (pok <> null) //po levom podstablu
3.      um ← um+1, mag[um] ← pok
4.      pok ← left(pok)
5.  endrepeat
6.    pok ← mag[um], um ← um-1
7.  repeat while (pok <> null) {  //povratak uz stablo
8.      visit (pok)
9.      if (right(pok) <> null) //postoji desno
    podstablo?
10.       then { pok ← right(pok), go to 2 }
11.     endif
12.   pok ← mag[um], um ← um-1
13. endrepeat
14. end
```

A)

B)

Ilustracija rada nerekurzivnog inorder obilaska

- EBIFKAGCH



um

E
B
A
null

um

I
F
A
null

EB

um

K
A
null

EBIF

um

G
C
null

EBIFKA

Obilazak po nivoima

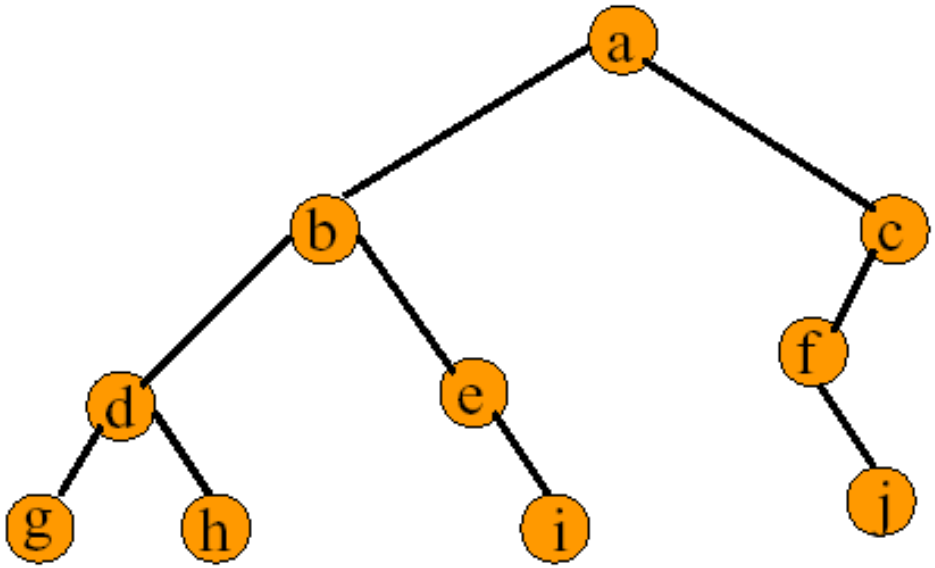
- Obilazi stablo po nivoima, počev od 0-tog, tj od korena stabla
- Koristi pomoćnu strukturu **red** (FIFO)

Algoritam S.9. Obilazak po nivoima

```
levelOrderN(info, llink, dlink, koren)
  pok ← koren
  while (pok <> null)
     visit(pok)
     //dodaj u FIFO red sve potomke čvora
       red_enqueue(left(pok))
       red_enqueue(right(pok))
   //pročitaj iz reda
     pok ← red_dequeue()
      //dequeue treba da vraća null
      // kada je red prazan
  endwhile
end
```

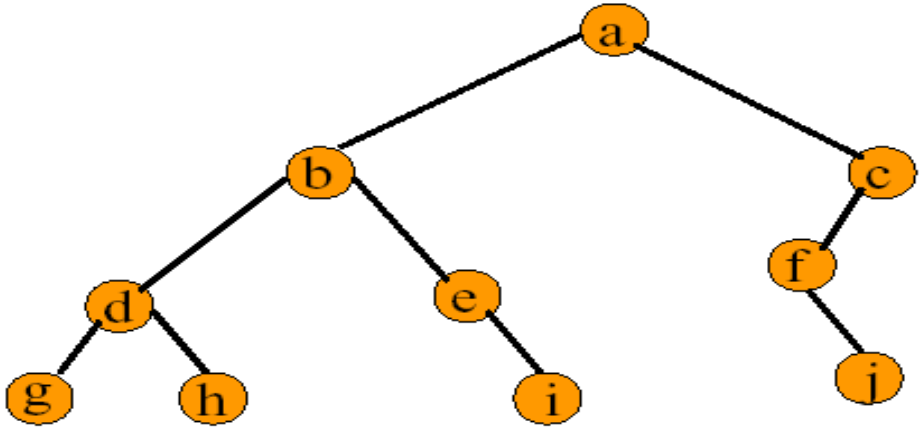
Ilustracija rada obilaska po nivoima

- visit = print



a b c d e f g h i j

OBILAZAK STABLA - PREGLED



- postorder: **ghdiebjfca** ► deca pa koren
- preorder: **abdgheicfj** ► koren pa deca
- inorder: **gdhbeiafjc** ► levo dete, koren, desno dete
  - Ovaj obilazak ima smisla samo za binarno stablo
- po nivoima: abcdefghij ► nivo 0,1,2,...

Rekonstrukcija binarnog stabla – za samostalni rad

- Da li se može rekonstruisati binarno stablo na osnovu dve sekvence dobijene algoritmima za obilazak??
- Da li rekonstrukcija zavisi od datih sekvenci?
- Pokazati na primeru da *preorder* i *postorder* ne definišu na jedinstven način binarno stablo
- Pokazati na istom primeru rekonstrukciju za *inorder* i *preorder*
- Pokazati na istom primeru šta se dešava za *preorder* i *level order*
- Pokazati na istom primeru za *postorder* i *level order*

Primer za vežbu

- *Inorder* i *Preorder* obilazak istog stabla

  Inorder = g d h b e i **a** f j c

  Preorder = **a** b d g h e i c f j

- Koristite kao osnovu *Preorder*, tako što pretražujete s leva u desno; korišćenjem *Inorder* pronalazite u nizu koren (pod)stabla i razdvajate čvorove na levo i desno podstablo
- Primer:
  - **a** je prvi čvor u *Preorder*, odnosno koren stabla;
  - u *Inorder*, **gdhbei** je levo podstablo, a **fjc** desno podstablo

Pitanja, ideje, komentari