

Stabla

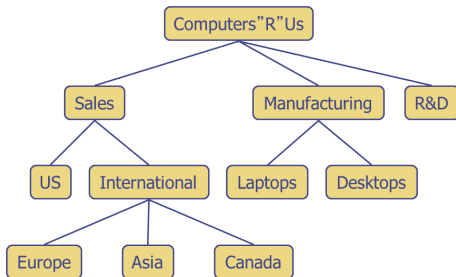
© Goodrich, Tamassia, Goldwasser

Katedra za informatiku, Fakultet tehničkih nauka, Univerzitet u Novom Sadu

2021.

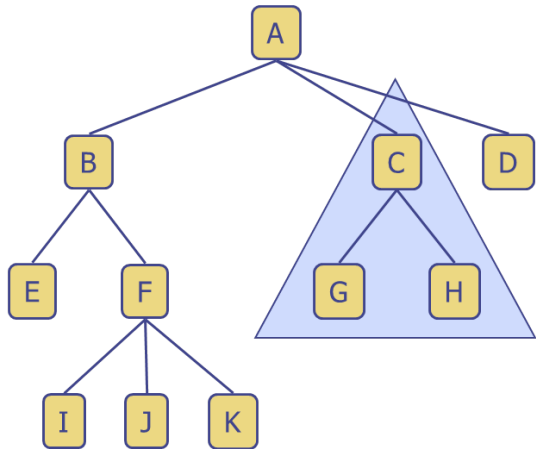
Stablo

- **stablo** je apstraktni model hijerarhijske strukture
- sastoji se od čvorova koji su u vezi **roditelj/dece**
- svaki čvor ima najviše jednog roditelja; tačno jedan čvor nema roditelja
- čvor ima nula ili više dece



Terminologija

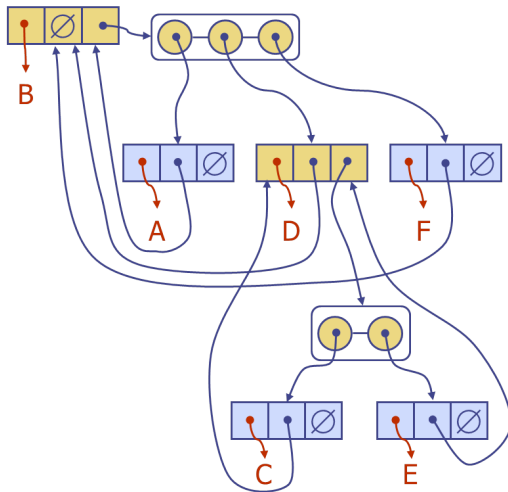
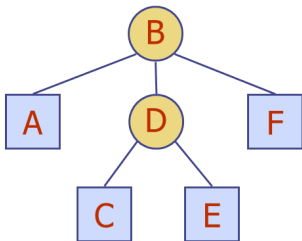
- **koren** (root): jedini čvor bez roditelja
- **unutrašnji čvor**: čvor sa bar jednim detetom
- **spoljašnji čvor/list** (leaf): čvor bez dece
- **predak**: roditelj, deda, praded, ...do korena
- **dubina čvora**: broj predaka
- **visina stabla**: najveća dubina
- **potomak**: dete, unuče, praunuče, ...
- **podstablo**: čvor stabla i njegovi potomci



Stablo ATP

- opšte metode:
 - int `len()`
 - boolean `is_empty()`
 - iterator `nodes()`
- metode za pristup podacima:
 - node `root()`
 - node `parent(n)`
 - iterator `children(n)`
 - int `num_children(n)`
- metode za ispitivanje čvorova:
 - boolean `is_leaf(n)`
 - boolean `is_root(n)`
- ažuriranje sadržaja:
 - element `replace(n, o)`

Stablo u memoriji



Čvor stabla u Pythonu

```
class TreeNode:
    def __init__(self):
        self._element = None
        self._parent = None
        self._children = []

    def __eq__(self, other):
        # kako utvrđujemo da li su dva čvora jednaka?

    def __ne__(self, other):
        # kako utvrđujemo da li su dva čvora različita?

    def is_root(self):
        return self._parent == None

    def is_leaf(self):
        return len(self._children) == 0
```

Stablo u Pythonu 1

```
class Tree:
    def __init__(self):
        self._root = None

    def is_empty(self):
        return self._root == None

    def depth(self, node):
        if node._parent is None:
            return 0
        else:
            return 1 + self.depth(node._parent)
```

Obilazak stabla

- obilazak **po dubini** (depth-first): obiđi čvor i njegove potomke pre braće
 - **preorder**: prvo čvor pa deca
 - **postorder**: prvo deca pa čvor
- obilazak **po širini** (breadth-first): obiđi čvor i njegovu braću pre potomaka
 - obilazak „po generacijama“ u stablu

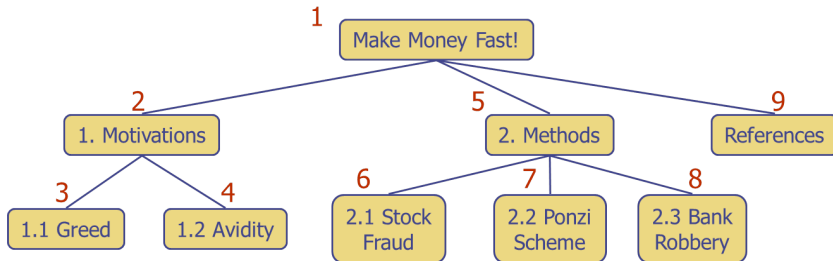
Obilazak stabla po dubini / preorder

preorder(n)

 obradi(n)

for all dete c od n **do**

 preorder(c)



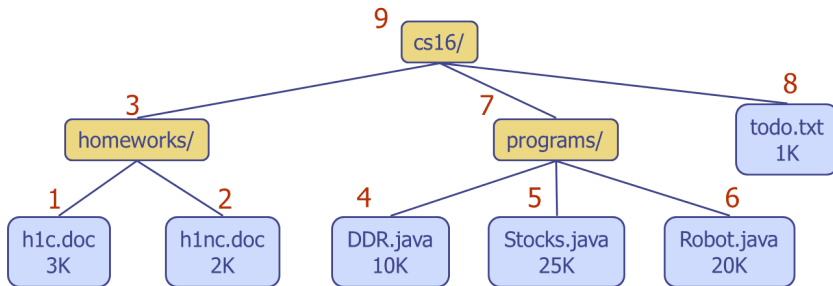
Obilazak stabla po dubini / postorder

`postorder(n)`

for all dete c od n **do**

`postorder(c)`

`obradi(n)`



Stablo u Pythonu 2

```
class Tree:
    ...
    def preorder(self, func):
        self._preorder(self._root, func)

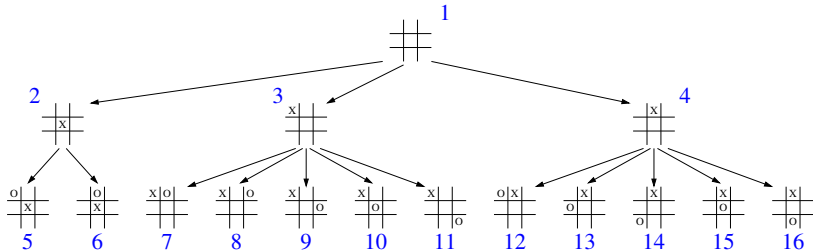
    def postorder(self, func):
        self._postorder(self._root, func)

    def _preorder(self, node, func):
        func(node)
        for child in node._children:
            self._preorder(child, func)

    def _postorder(self, node, func):
        for child in node._children:
            self._postorder(child, func)
        func(node)
```

Obilazak stabla po širini

- treba obići sve čvorove dubine d pre nego što se pređe na čvorove dubine $d + 1$
- primer: stablo igre – svi mogući ishodi igre koju igra čovek ili računar; koren je početno stanje igre
- za igru „puta-nula“ (tic-tac-toe)



Obilazak stabla po širini

breadth_first(*root*)

napravi novi prazan red *Q*

Q.add(*root*)

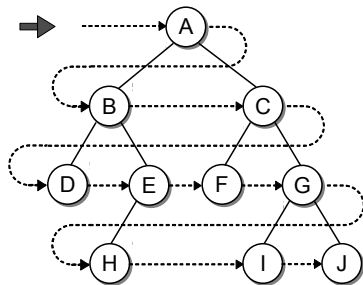
while *Q* nije prazan **do**

node \leftarrow *Q*.dequeue()

 obradi(*node*)

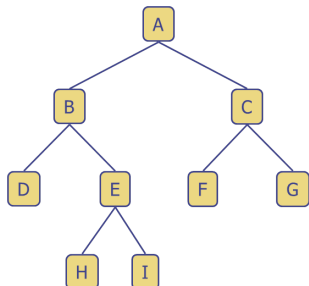
for all *child* dete od *node* **do**

Q.enqueue(*child*)



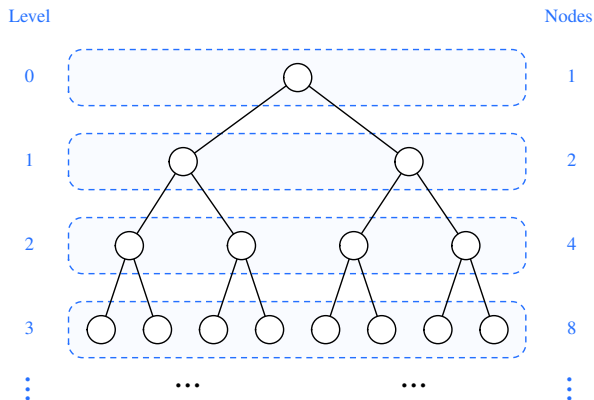
Binarno stablo

- stablo za koje važi:
 - svaki čvor ima najviše dvoje dece
 - svako dete je označeno kao **levo dete** ili **desno dete**
 - levo dete po redosledu prethodi desnom detetu
- levo podstablo – levo dete kao koren
- desno podstablo – desno dete kao koren
- **pravilno** binarno stablo: svaki čvor ima 0 ili 2 deteta



Osobine binarnog stabla

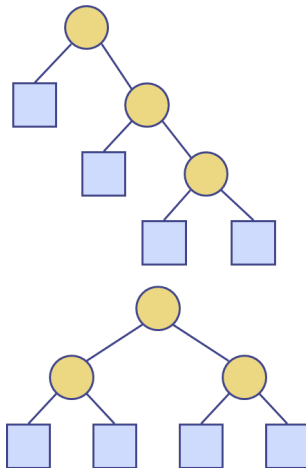
- nivo stabla d ima najviše 2^d čvorova
- broj čvorova po nivou raste eksponencijalno



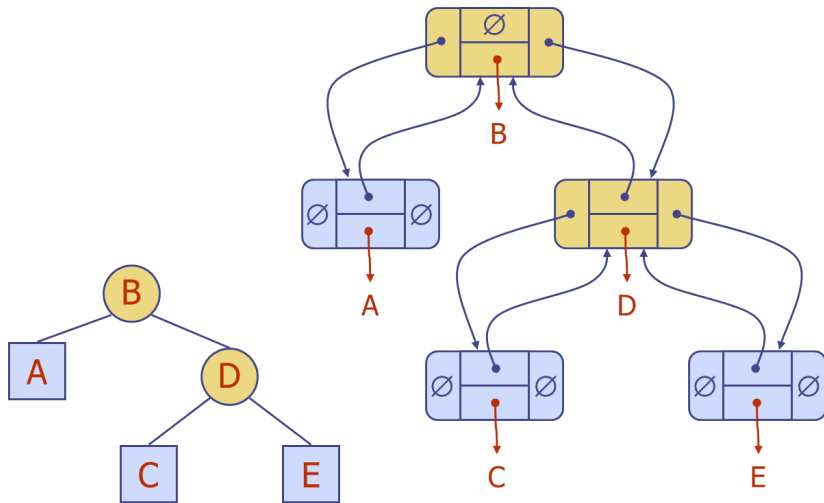
Osobine binarnog stabla

- n – broj čvorova
- e – broj listova
- i – broj internih čvorova
- h – visina

- $e = i + 1$
- $n = 2e - 1$
- $h \leq i$
- $h \leq (n - 1)/2$
- $e \leq 2^h$
- $h \geq \log_2 e$
- $h \geq \log_2(n + 1) - 1$



Binarno stablo u memoriji / čvorovi i reference



Binarno stablo u memoriji / pomoću niza

- rang čvora:

- $\text{rang}(\text{root}) = 1$

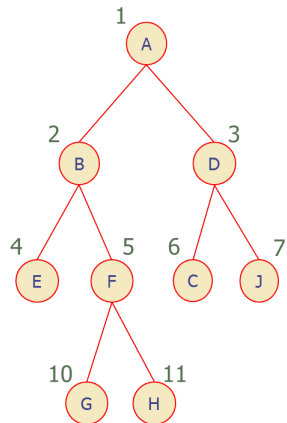
- za levo dete:

$$\text{rang}(\text{node}) = 2 \cdot \text{rang}(\text{parent})$$

- za desno dete:

$$\text{rang}(\text{node}) = 2 \cdot \text{rang}(\text{parent}) + 1$$

- čvor v se smešta u $A[\text{rang}(v)]$



Obilazak binarnog stabla / inorder

inorder(n)

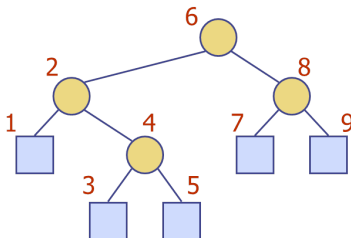
if n ima levo dete **then**

 inorder(levo dete)

 obradi(n)

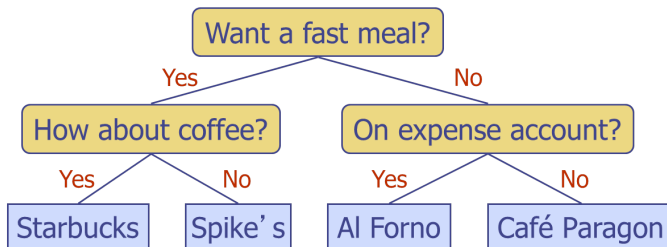
if n ima desno dete **then**

 inorder(desno dete)



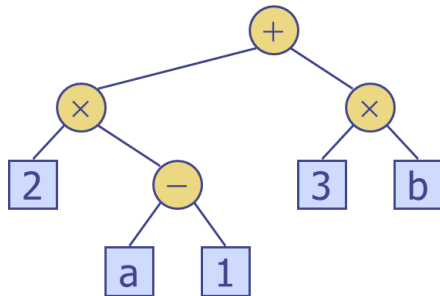
Stabla odlučivanja

- binarno stablo strukturirano prema procesu odlučivanja
- unutrašnji čvorovi – pitanja sa da/ne odgovorima
- listovi – odluke
- primer: gde za večeru?



Stablo aritmetičkih izraza

- binarno stablo kreirano na osnovu aritmetičkog izraza
- unutrašnji čvorovi – operatori
- listovi – operandi
- primer: $2 * (a - 1) + 3 * b$



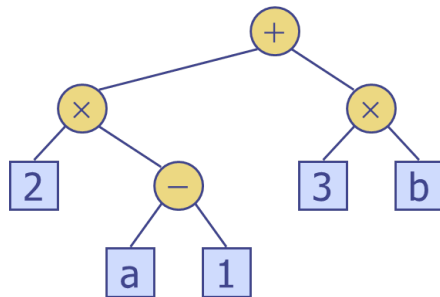
Ispisivanje aritmetičkih izraza

- specijalni slučaj **inorder** obilaska

`printExpr(n)`

```

if  $n$  ima levo dete then
    print("(")
    printExpr(levo dete)
print( $n$ )
if  $n$  ima desno dete then
    printExpr(desno dete)
print(")")
    
```



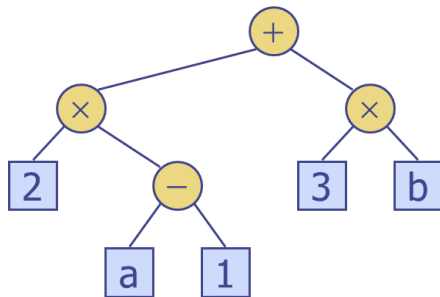
Izračunavanje aritmetičkih izraza

- specijalni slučaj **postorder** obilaska

$\text{evalExpr}(n)$

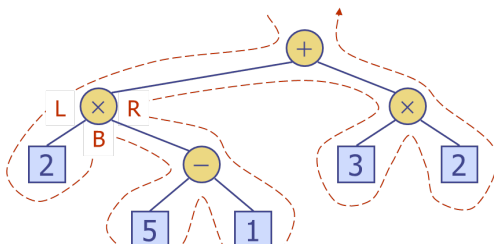
```

if  $n$  je list then
  return  $n.\text{element}$ 
else
   $x \leftarrow \text{evalExpr}(n.\text{left})$ 
   $y \leftarrow \text{evalExpr}(n.\text{right})$ 
   $\diamond \leftarrow \text{operator u } n$ 
  return  $x \diamond y$ 
    
```



Ojlerov obilazak stabla

- opšti postupak za obilazak stabla
- preorder, inorder, postorder su specijalni slučajevi
- posmatramo grane stabla kao zidove koji uvek moraju da nam budu sa leve strane prilikom kretanja
- svaki čvor se poseti tri puta
 - sa leve strane (preorder)
 - sa donje strane (inorder)
 - sa desne strane (postorder)



Crtanje stabla

- treba odrediti (x, y) koordinate čvorova stabla
- $x(n)$: broj čvorova posećenih pre čvora n u **inorder** obilasku
- $y(n)$: dubina čvora n

