Stabla

Slajdovi sa predavanja[1]

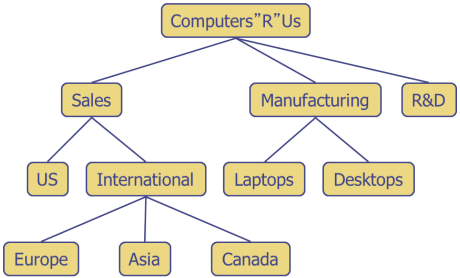
Katedra za informatiku, Fakultet tehničkih nauka, Univerzitet u Novom Sadu

2022.

[1] Po uzoru na materijale sa: `https://github.com/mbranko/asp-slajdovi`

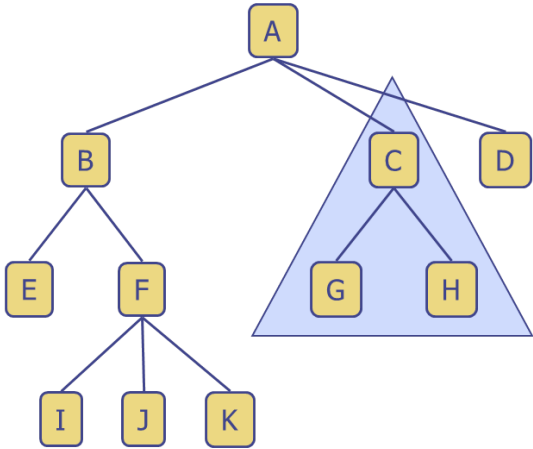
Stablo

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

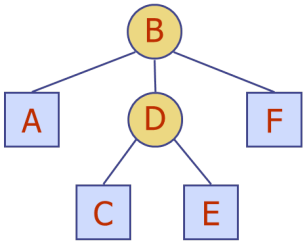


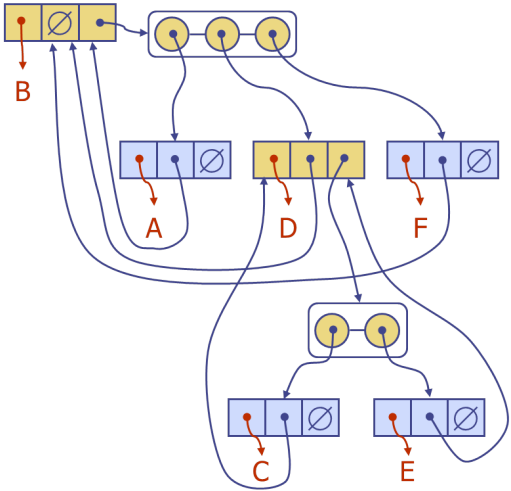
Terminologija

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



Stablo u memoriji





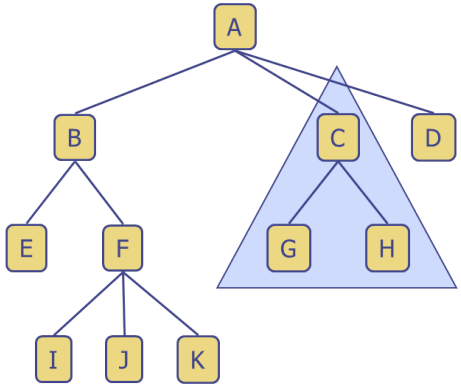
Čvor stabla u Pythonu

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

  def __eq__(self, other):
    return self == other:

  def __ne__(self, other):
    return self != other

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

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

  def add_child(self, el):
    self._children.append(el)
    el._parent = self
```

Dubina

Dubina čvora k može se rekurzivno definisati na sledeći način:

- ako je k koren, dubina je 0
- u suprotnom, dubina čvora k je 1 plus dubina roditelja k



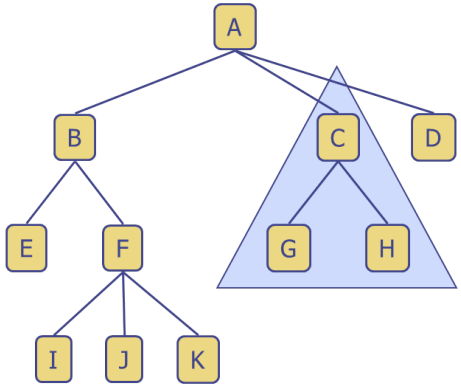
Stablo u Pythonu 1

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

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

  def depth(self, node):
    """
    Metoda izračunava dubinu zadatog čvora.

    Argument:
    - `node`: čvor čija dubina se računa
    """
    if node._parent is None:
      return 0
    else:
      return 1 + self.depth(node._parent)
```

Visina

Visina čvora k može se rekurzivno definisati na sledeći način:

- ako je k list, visina je 0
- u suprotnom, visina čvora k je za 1 veća od maksimuma visina potomaka k



Stablo u Pythonu 2

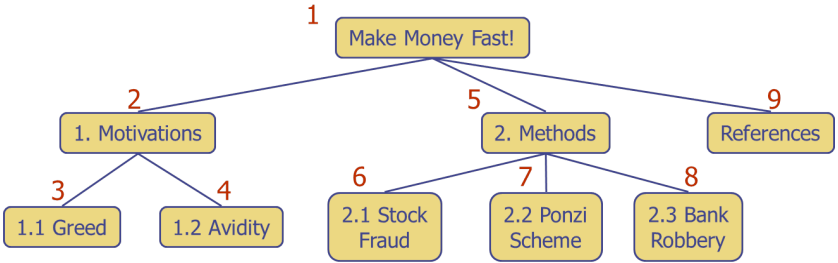
```
def _height(self, node):
  """
  Metoda izračunava visinu podstabla sa zadatim korenom.

  Argument:
  - `node`: koren posmatranog podstabla
  """
  if node.is_leaf():
    return 0
  else:
    return 1 + max(self._height(c) for c in node._children)

def height(self):
  return self._height(self._root)
```

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)



Stablo u Pythonu 3

```
class Tree:
  ...
  def preorder(self, node):
    """
    Preorder obilazak po dubini

    Najpre se vrši obilazak roditelja a zatim svih njegovih potomaka.

    Argument:
    - `node`: čvor od koga počinje obilazak
    """
    if not self.is_empty():
      print(node._data)
      for c in node._children:
        self.preorder(c)
```

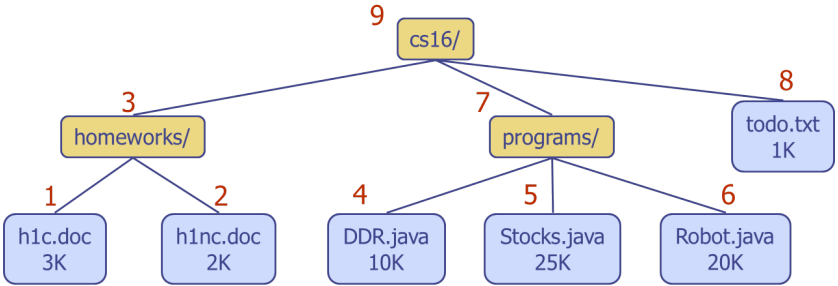
Obilazak stabla po dubini / postorder

postorder(n)

for all dete c od n **do**

postorder(c)

obradi(n)

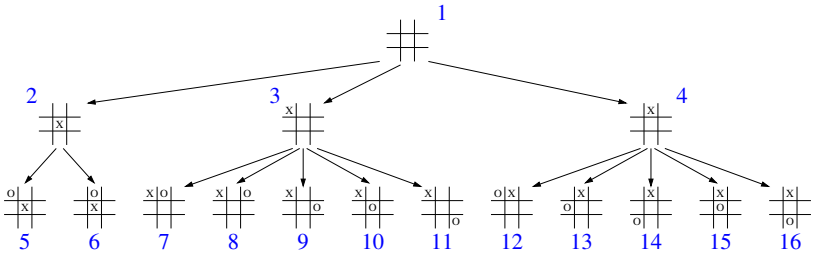


Stablo u Pythonu 4

```
class Tree:
  ...
  def postorder(self, node):
    """
    Postorder obilazak po dubini

    Najpre se vrši obilazak potomaka a zatim i roditelja

    Argument:
    - `node`: čvor od koga počinje obilazak
    """
    if not self.is_empty():
      for c in node._children:
        self.postorder(c)
      print(node._data)
```

Obilazak stabla po širini

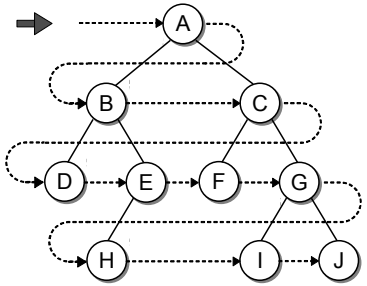
- treba obići sve čvorove dubine d pre nego što se pređe na čvorove dubine $d + 1$
- primer: stablo igre – svi mogući ishodi igre koju igra čovek ili računar; koren je početno stanje igre
- za igru „puta-nula“ (tic-tac-toe)



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($child$)



Stablo u Pythonu 5

```
from queue import Queue

class Tree:
  ...
  def breadth_first(self):
    """
    Metoda vrši obilazak stabla po širini.
    """
    to_visit = Queue()
    to_visit.enqueue(self._root)
    while not to_visit.is_empty():
      el = to_visit.dequeue()
      print(el._data)
      for c in e._children:
        to_visit.enqueue(c)
```