Laboratorul 8

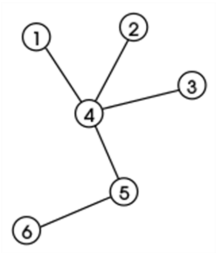
Formatori:
Tutor: Militaru Mihai-Adrian
Tutor: Dragomir Titian-Cornel
+8

Data de începere a cursului:
25.09.2023
Utilizatori înscriși
Calendar
Note

Laboratorul 8

Arbori

Un arbore e un graf neorientat conex și fără cicluri. Arborii reprezintă grafurile cele mai simple ca structură din clasa grafurilor conexe, ei fiind și cei mai frecvent utilizați în practică.



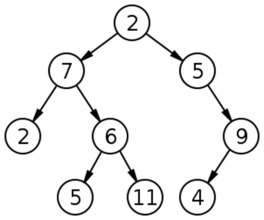
img: http://en.wikipedia.org/wiki/File:Tree_graph.svg

La un arbore, muchiile poartă denumirea de ramuri. Un arbore cu n noduri are n - 1 ramuri.

Arbori cu rădăcină

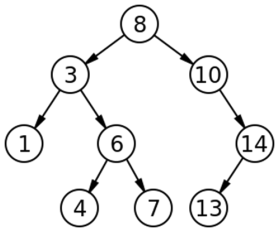
De obicei identificăm un nod anume numit rădăcina, și orientăm muchiile în același sens față de rădăcină. Orice nod în afară de rădăcină are un unic părinte. Un nod poate avea mai mulți copii (fii). Nodurile fără copii se numesc noduri frunză.

Arbori binari
Un arbore binar este un arbore în care fiecare nod are cel mult doi copii, identificați ca fiul stâng și fiul drept.



Arbori binari de căutare

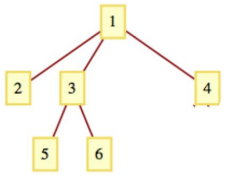
Arborii binari de căutare sunt arbori binari care memorează valori sortate în ordine. Pentru fiecare nod: subarborele stâng are valori mai mici decât rădăcina, iar subarborele drept are valori mai mari decât rădăcina.



Căutarea în arborele binar de căutare se face eficient, din puțini pași. Căutarea se face recursiv, comparând mereu elementul căutat cu rădăcina subarborelui curent: dacă sunt egale am găsit elementul în arbore, dacă elementul căutat e mai mic decât rădăcina curentă, se continuă căutarea în subarborele stâng, iar dacă elementul căutat e mai mare decât rădăcina curentă, se continuă căutarea în subarborele drept.

Reprezentarea unui arbore în Python

1. Reprezentarea unui arbore oarecare - exemplu

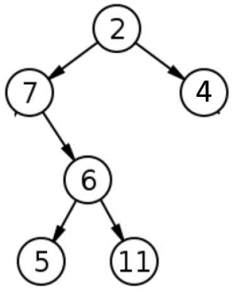



```
arbore_oarecare = { "valoare": 1, " copii":
            [
                { "valoare": 2, " copii": []},
                { "valoare": 3, " copii":
                    [
                        { "valoare" :5, " copii": []},
                        { "valoare" :6, " copii": []}
                    ]
                },
                { "valoare": 4, "copii": []}
            ]
        }
```

2. Reprezentarea unui arbore binar

Un arbore binar îl putem reprezenta recursiv ca un dicționar cu 3 perechi: valoare, arbore stâng și arbore drept.

```
arbore =  {"value": None, "left": None, "right": None}
```



```
binary_tree = { "value" : 2, "left":
           {
               "value": 7, "left": None, "right":
               {
                   "value": 6, "left":
                   {
                       "value": 5, "left": None, "right": None
                   }, "right":
                   {
                       "value":11, "left": None, "right": None
                   },
               },
           }, "right":
               {
                   "value": 4, "left": None, "right": None
               }
       }
```

Parcurgerea arborilor binari

1. Parcurgerea în preordine (rădăcină, subarbore stâng, subarbore drept)

```
def rsd(tree):
    if (tree != None):
        return [tree["value"]] + rsd(tree["left"]) + rsd(tree["right"])
    else:
        return []

print(rsd(binary_tree))
```

2. Parcurgerea în inordine (subarbore stâng, rădăcină, subarbore drept)

```
def srd(tree):
    if (tree != None):
        return  srd(tree["left"]) + [tree["value"]] + srd(tree["right"])
    else:
        return []

print(srd(binary_tree))
```

3. Parcurgerea în postordine (subarbore stâng, subarbore drept, rădăcină)

```
def sdr(tree):
    if (tree != None):
        return  sdr(tree["left"]) + sdr(tree["right"]) + [tree["value"]]
    else:
        return []

print(sdr(binary_tree))
```

Adăugarea unui nod nou la un părinte și o poziție (stânga sau dreapta):

```
def adaugare_nod_pozitie(parinte, nod_nou, pozitie):
    if (parinte[pozitie] == None):
        parinte[pozitie] = nod_nou
    return parinte

binary_tree["left"]=adaugare_nod_pozitie(binary_tree["left"], {"value": 100, "left": None, "right": None}, "left")
print(rsd(binary_tree))
```

Adăugarea unui nod nou în arbore binar de căutare:

```
def adaugare_nod(tree, nod_nou):
    if (tree == None):
        return nod_nou
    if (nod_nou["value"]<tree["value"]):
        tree["left"] = adaugare_nod(tree["left"], nod_nou)
    else:
        tree["right"] = adaugare_nod(tree["right"], nod_nou)
    return tree

print(rsd(adaugare_nod(binary_tree,{"value": 1, "left": None, "right": None})))
```

Ștergerea unui nod (sau subarbore) de la un anumit părinte dat ca parametru:

```
def stergere_nod(parinte, valoare_nod):
    if (parinte["left"]["value"] == valoare_nod):
        parinte["left"] = None
    elif(parinte["right"]["value"] == valoare_nod):
        parinte["right"] = None

stergere_nod(binary_tree["left"], 100)
stergere_nod(binary_tree, 5)
print(rsd(binary_tree))
```

◄ Cod din curs - Arbori in Python

Sari la...

Exerciții - Săptămâna 8 ►

Contactați serviciul de asistență