TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Programmeerimise süvendatud algkursus ITI0140

2014

Teemad

- Pinu (ingl *stack*)
- Rekursioon (ingl *recursion*)
- Sügavuti otsing (ingl *depth-first search*)

Pinu

Pinu on selline andmestruktuur, kuhu elemente lisatakse ja eemaldatakse ühest otsast, n-ö pinu tipust (ingl *top*).

Pinu operatsioonideks on meil: elemendi lisamine (*push*), elemendi eemaldamine (*pop*), oleku kontroll (kas pinu on tühi?)

Pinu võib implementeerida Pythonis näiteks järjendit kasutades.

Pinu näide

Pythonis ei ole järjendil funktsiooni push, vaid append

```
stack = []
stack.append("first") # "push" operation
print(stack)
stack.append(123) # "push" operation
print(stack)
element = stack.pop() # "pop" operation
print(element)
print(stack)
element = stack.pop() # "pop" operation
print(True if len(stack) == 0 else False) # "empty" check
element = stack.pop() # "pop" operation
```

```
>> ['first']
['first', 123]
123
['first']
True
Traceback (most recent call last):
  File "C:\Workspace\Python340\stack.py", line 12, in <module>
    element = stack.pop() # "pop" operation
IndexError: pop from empty list
```

N-ö “püütonlik stiil” oleks

```
print(False if stack else True)
```

Rekursioon

Rekursioon on viis lahendada ülesandeid muutes neid aina lihtsamaks, jõudes lõpuks baasjuhuni. Baasjuhu ja vahepeal läbitud sammude tulemustest kombineeritakse esialgse ülesande lahendus.

Rekursioon programmeerimises tähendab funktsiooni iseenda väljakutsumist muudetud argumentidega.

Rekursiooni näide

Klassikalised rekursiooni näited on matemaatikast tuntud faktoriaal ja Fibonacci arvujada.

Faktoriaal: 6! = 1*2*3*4*5*6 = 720

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(6))
```

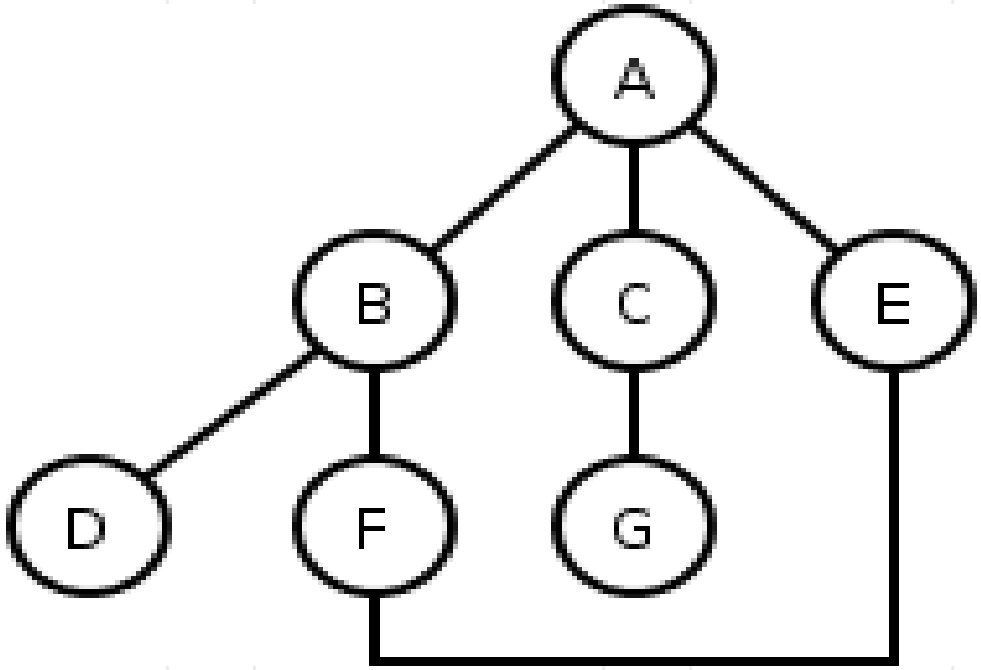
```
>> 720
```

6 * factorial(5)
6 * 5 * factorial(4)
6 * 5 * 4 * factorial(3)
6 * 5 * 4 * 3 * factorial(2)
6 * 5 * 4 * 3 * 2 * factorial(1)
6 * 5 * 4 * 3 * 2 * 1 * factorial(0)
6 * 5 * 4 * 3 * 2 * 1 * 1

Naljana defineeritakse rekursiooni näiteks nii:

rekursioon vt rekursioon

Sügavuti otsing



Kontrollitavate tippude järjekord samm-sammult:

1) A → B
2) B → D
3) B → F (B → D?)
4) F → E
5) E → A
6) A → C (A → B?)

Ülesanne

Genereerige NetworkX paketi abil juhusliku iseloomuga graaf, kasutades juhuslikkuse algväärtustamiseks mingil moel oma matriklinumbrit. Teisendage saadud graaf sõnastiku (*dictionary*) kujule.

Implementeerige kaks sügavuti otsingu algoritmi:

1) Iteratiivne algoritm, mis kasutab pinu (*stack*) andmetüüpi järgmiste tippude meelespidamiseks.

2) Rekursiivne algoritm

Iga otsingualgoritm peaks tagastama läbivaadatud tippude arvu.

Sooritage implementeeritud otsingutega N katset.

Selleks valige juhuslikult alg- ja lõpptipp ning sooritage otsingud. Loendage, mitu tippu iga algoritm summarselt läbi vaatas ja kuvage tulem. Korrake katseid erinevate graafidega (erinev tippude hulk, erinev kaarte hulk jne.)