TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

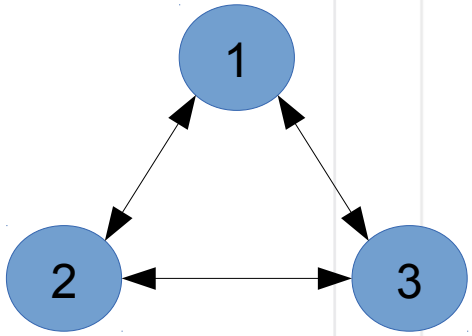
Programmeerimise süvendatud algkursus ITI0140

2014

Teemad

- Graafi esitus (andmestruktuur)
- Laiuti otsing (Breadth-First search)

Graafi esitus (sõnastik-hulk)



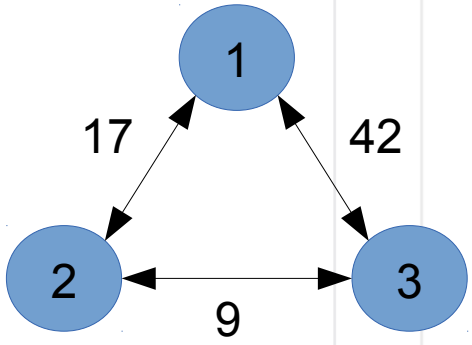
Hariliku graafi puhul (ilma kaaludeta servad) võib võtme-väärtuspaari väärtus olla määratud hulgana (set).

Näide:

```
unweighted = {1: set([2, 3]), 2: set([1, 3]), 3: set([2, 1])}
print(unweighted)
```

```
>> {1: {2, 3}, 2: {1, 3}, 3: {1, 2}}
```

Graafi esitus (sõnastik-sõnastik)



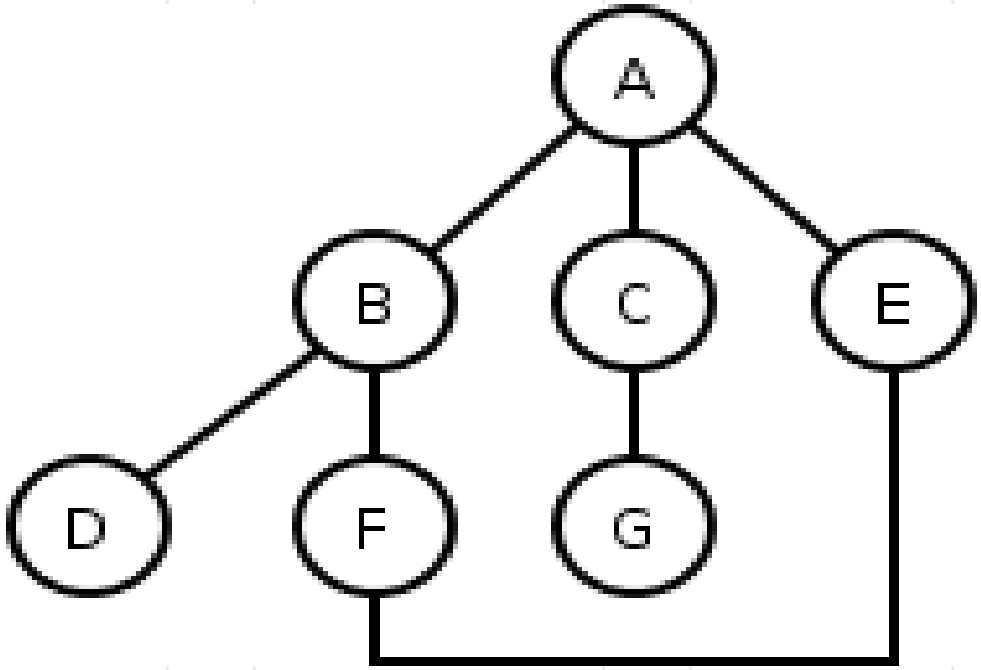
Kaalutud graafi puhul on otstarbekas defineerida naabrid omakorda sõnastikuna, mille väärtused on kaalud.

Näide:

```
weighted = {1: {2: 17, 3: 42}, 2: {1: 17, 3: 9}, 3: {1: 42, 2: 9}}
print(weighted)
```

```
>> {1: {2: 17, 3: 42}, 2: {1: 17, 3: 9}, 3: {1: 42, 2: 9}}
```

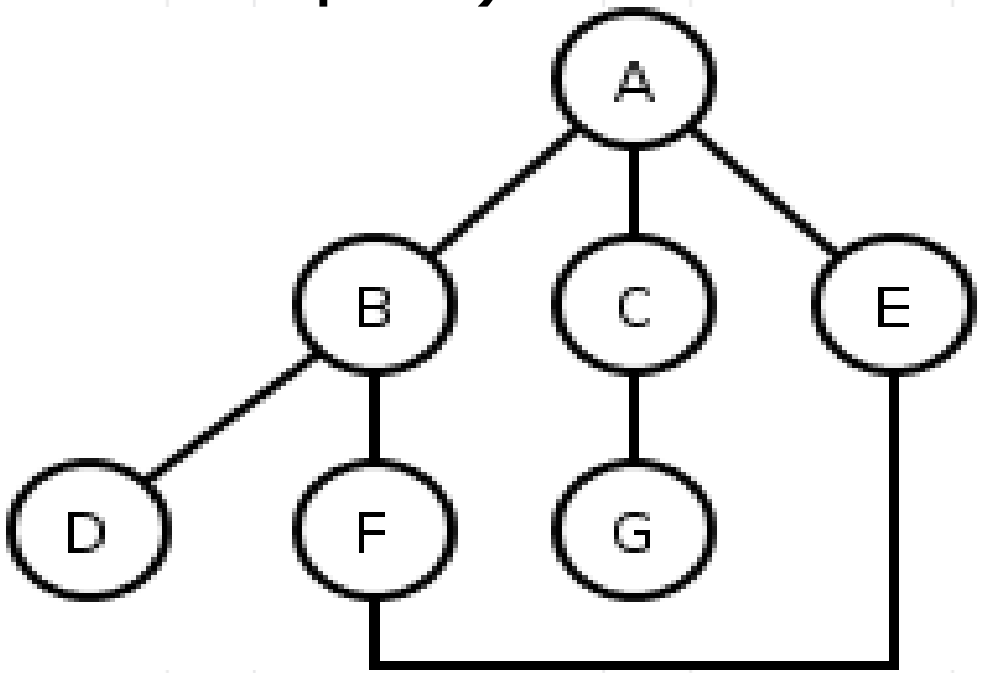
Laiuti otsing (ilma mäluta)



Kontrollitavate tippude järjekord samm-sammult:

1) A: [B, C, E]
2) B: [C, E, A, D, F]
3) C: [E, A, D, F, A, G]
4) E: [A, D, F, A, G, A, F]
5) A: ...

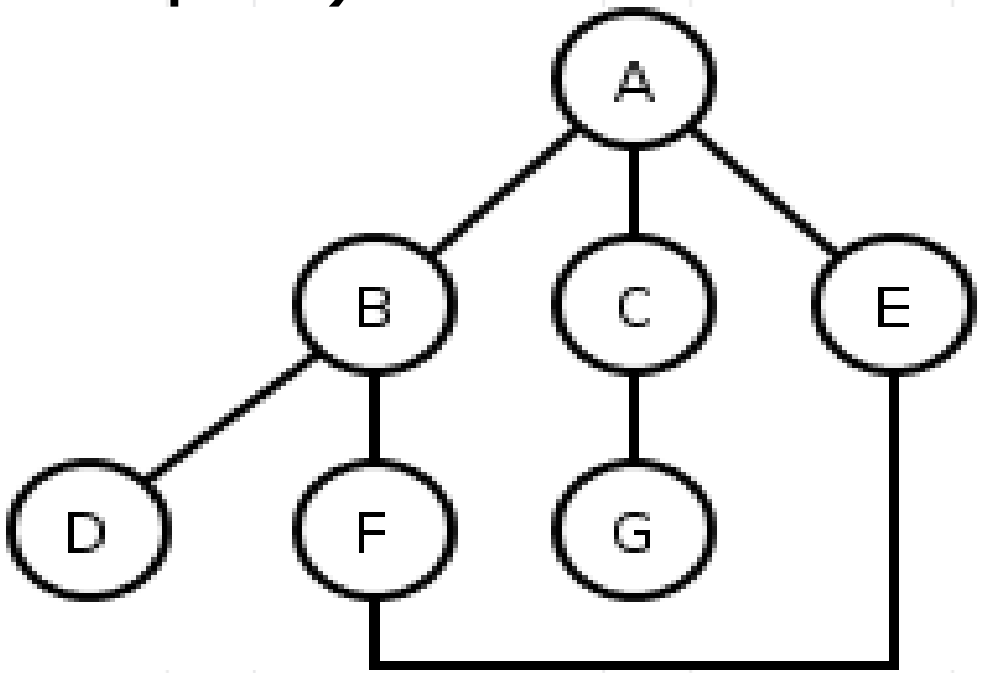
Laiuti otsing (mäluga, külastatud tipud)



Kontrollitavate tippude järjekord samm-sammult:

1) A: [B, C, E] visited = [A]
2) B: [C, E, D, F] visited = [A, B]
3) C: [E, D, F, G] visited = [A, B, C]
4) E: [D, F, G, F] visited = [A, B, C, E]
5) D: ...

Laiuti otsing (mäluga, nähtud tipud)



Kontrollitavate tippude järjekord samm-sammult:

0) seen = [A]
1) A: [B, C, E] seen = [A, B, C, E]
2) B: [C, E, D, F] seen = [A, B, C, E, D, F]
3) C: [E, D, F, G] seen = [A, B, C, E, D, F, G]
4) E: [D, F, G] seen = [A, B, C, E, D, F, G]
5) D: ...

BFS

http://en.wikipedia.org/wiki/Breadth-first_search

Ülesanne

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

Implementeerige kolm laiuti otsingu algoritmi:

1) Kasutatakse järjekorda (*queue*) järgmise tipu valikuks ja ei jäta meelde tippe, mis on juba läbi käidud või mida on juba nähtud.

2) Kasutatakse järjekorda järgmise tipu valikuks ja hulka (*set*), et jätta meelde tipud, mis on juba läbi käidud.

3) Kasutatakse järjekorda järgmise tipu valikuks ja hulka, et jätta meelde tipud, mida on juba nähtud ja mis on järjekorda mingil hetkel lisatud.

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.)