Tartalomjegyzék

Gráfok

Játszunk el ezekkel a parancsokkal:

```
G = Graph({0: [1,4,5], 1: [2,6], 2: [3,7], 3: [4,8], 4: [9], 5: [7, 8], 6: [8,9], 7: [9]})
G.show()
G.degree()
G.degree(5)
G.edges()
G.neighbors(5)
G.add_edge(6,4)

graph = DiGraph({8:[3, 10], 3:[1, 6], 6:[4,7], 10:[14], 14:[13]})
```

Majd írjuk meg a DFS algoritmust. Az algoritmussal feszítõfát keresünk. Az algoritmus kiindul a G gráf egy adott v csúcsából, megnézi az összes szomszédját, majd az egyiken továbbmegy, ha azon végigért akkor néz egy másikat és így tovább. A lényeg, hogy minden csúcsot csak egyszer érintsünk. Ennek a megvalósítása közel sem triviális. A wikipedián leírt kód jó, de egy kis magyarázatra szorul:

A csúcsokat cimkézzük, háromféle jelölés van: semmilyen, érintett és bejárt. Ha egy csúcson még nem jártunk akkor semmilyen. Ha érintettük, de még nem biztos hogy minden szomszédját is, akkor érintett. Ha biztosan érintettük minden szomszédját is, akkor bejárt. Értelemszerûen az algoritmus addig fut amíg minden csúcs bejárt nem lesz.

Példa

Az ötlet a következõ:

- adott csúcsbol indulunk, ez alapból érintett
- kiválasztunk egy szomszédot
- õ mostantól érintett
- majd ennek a szomszédnak is kiválasztjuk egy még nem érintett szomszédját és így tovább
- ha nem tudunk nem érintett szomszédot kiválasztani, akkor az adott csúcsot bejártnak jelöljük
- ezek után az algoritmus "http://wiki.math.bme.huvisszamegy"http://wiki.math.bme.hu az útvonalon és nézi a már érintett csúcsok még nem érintett csúcsait
- rekurzióval ez elég egyszerû

Próbáljátok ki a korábban definiált gráfra és egy random gráfra is:

```
randomGraph = graphs.RandomGNP(9, 0.3)
```

Az elsõ paraméter a csúcsok száma, a második, hogy mekkora valószínûsége legyen egy élnek 2 csúcs között.

Irányított gráfok

Próbáljuk ki a következõ parancsokat:

```
d1 = digraphs.RandomDirectedGNP(9, 0.3)

d1.show()

d1.neighbors_in(2)

d1.neighbors_out(2)
```

Vegyes függvények

Szószámláló, szótár segítségével

- Készíts függvényt ami a bemenetként adott szöveget feldolgozza olyan módon, hogy a bemenetként adott szótárba minden,

a szövegben szerepl? szót az el?fordulásának számával beilleszti a szótárba.

- Segítség:

```
s = "Volt egyszer egy kiskutya"
L = s.split(' ')
```

- **L** értéke:

```
["Volt", "egyszer", "egy", "kiskutya"]
```

- Tehát a split metódus feldarabolja a stringet a megadott karakter mentén.

Gyök számítás intervallum felezéssel

- Készíts függvényt ami a bemenetként kapott kifejezés gyökét próbálja megtalálni a szintén paraméterként megadott intervallumban.
- Az algoritmus annyi, hogy az adott intervallumot kettévágjuk, vesszük mindkét végének a négyzetét és megnézzük melyikbe esik a szám aminek keressük a négyzetét. Majd ezzel az intervallummal folytatjuk ugyanígy, amígy egy kis epszilon szám alá nem megy az intervallum hossza (pl 10^(-5)). Ekkor pedig visszaadjuk az intervallum közepét pl.

Szigetek megszámlálása

- Készíts függvényt, ami a bemeneteként kapott mátrixban, a szomszédos 1-eseket szigetnek véve visszaadja a térképen található szigetek számát.
- A matrix eleme 1 vagy 0 lehet.
- A mátrix mindig azonos oldalhosszúságú, de bármekkora lehet.
- A szomszédos elemek csak a vízszintesen vagy függ?legesen szomszédos elemeket értjük, átlósan nem értelmezett.

- Példa bemenet: [[0,1,1,0],[0,1,0,0],[0,0,0,1],[0,0,1,1]]