

Vybrané grafové algoritmy v jazyce Python

1 Úvod

Tento text vznikl jako doprovodný materiál ke cvičení na Škole programovacích technik pořádané na katedře Informatiky Vysoké školy báňské - Technická univerzita Ostrava. Programovacím jazykem pro cvičení je Python. Každý úkol ve cvičení obsahuje popis zadání, odkaz na kódy ze kterých při řešení budete vycházet a u některých úkolů je součástí také odkaz na externí video, jenž může danou problematiku pomoci přiblížit.

Python

Spouštění vzorových kódů a případné vypracování úkolů v jazyce Python vyžaduje jeho instalaci. Python je dostupný ke stažení pro většinu dnešních platform ¹. Po správném nainstalování Pythonu je možné zkusit vytvořit a spustit první program. K tomuto účelu je možné využít buď příkazový řádek, nebo nějaký vyvojové prostředí typu PyCharm ². Na webu existuje celá řada tutoriálů ³, které ukazují jak Python instalovat a spustit, takže podrobnější popis zde vynecháme.

¹<https://www.python.org/downloads/>

²<https://www.jetbrains.com/pycharm/>

³<https://www.youtube.com/watch?v=N4mEzFDjqtA>

Úkol 1: Načtení grafu

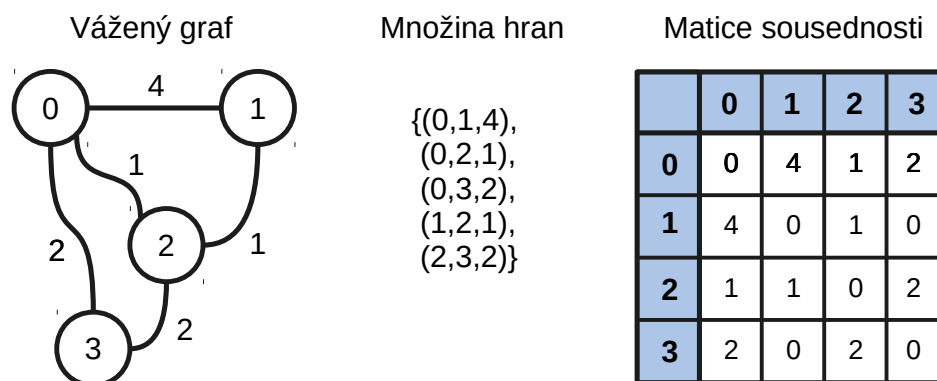
Skript:

https://github.com/RadimBaca/SOPA_cviceni/tree/master/ukol1/mustr.py

Zadaní:

Graf je nyní uložen v souboru jako množina hran ve formátu `vrchol1;vrchol2;vaha`.

Hlavním úkolem je nyní uložit graf jako matici sousednosti ⁴.



Video:

<https://www.youtube.com/watch?v=9C2cpQZVRBA>

Toto video obsahuje podrobnější popis matice sousednosti spolu s popisem jejich výhod a nevýhod.

Úkol 2: Stupeň grafu

Skript:

https://github.com/RadimBaca/SOPA_cviceni/tree/master/ukol2/mustr.py

⁴https://cs.wikipedia.org/wiki/Matice_sousednosti

Zadání:

Graf je nyní reprezentován jako matice sousednosti. Vážený stupeň vrcholu je dán součtem vah jeho hran. Pro spočítání váženého stupně každého vrcholu v grafu využijte Numpy funkci `sum`.

```
rank = matrix.sum(axis=1)
```

Hlavním úkolem v tomto zadání je vypsát prvních deset osob dle jejich váženého stupně. Při vypisování prvních deset osob využijte také mapu `names`, kterou jste načetli ze souboru v rámci předchozího úkolu.

Úkol 3: Karate klub

Skript:

https://github.com/RadimBaca/SOPA_cviceni/tree/master/ukol3/mustr.py

Zadání:

Zachary karate klub. Jde o sociální síť univerzitního karate klubu. Tento karate klub byl studován doktorem Wayne W. Zacharym po dobu tří let (1970-1972). Vrcholy sítě jsou osoby karate klubu a hrany představují skutečnost, že dvě osoby se mezi sebou znají a stýkají i mimo karate klub. V průběhu kdy byl karate klub studován se vystupňoval konflikt mezi administrátorem a instruktorem, který vedl k rozdělení klubu na dva ⁵.

Načtěte graf karate klubu (naleznete v adresáři data github repozitáře) a použijte předchozí analýzu dle vážených stupňů najít dvě nejdůležitější osoby (administrátora a instruktora) v grafu. Tzn. naleznete jejich číslo.

Úkol 4: Průměr grafu a BFS

Data:

https://github.com/RadimBaca/SOPA_cviceni/tree/master/ukol4/mustr.py

Zadání:

⁵https://en.wikipedia.org/wiki/Zachary%27s_karate_club

Průměr grafu je v podstatě průměrná nejkratší vzdálenost v grafu. Dokončete kód jenž bude s pomocí algoritmu Breath-first search ⁶ hledat nejkratší vzdálenost od výchozího vrcholu ke všem ostatním vrcholům v grafu. Algoritmus tedy bude vracet list s těmito vzdálenostmi.

Uveďme ještě základní popis datových struktur v algoritmu, kde V je počet vrcholů v grafu:

- `distances` - list o velikosti V , kde každá pozice udává nejkratší vzdálenost od výchozího vrcholu. Na začátku jsou všechny nastaveny na nula.
- `visited` - list o velikosti V , kde každá pozice uchovává příznak, zda-li byl vrchol navštíven nebo ne. Na začátku jsou všechny nastaveny na `False`.
- `queue` - list dvojic (`vrchol, vzdalenost`). Na začátku je v listu pouze jedna dvojice (`vychodi_vrchol, 0`). Pro vkládání nových dvojic použijte funkci `append`.

Video:

https://www.youtube.com/watch?v=E_V71Ejz3f4

Toto video názorně demonstruje základní ideu algoritmu.

Úkol 5: Floyd-Warshall

Data:

https://github.com/RadimBaca/SOPA_cviceni/tree/master/ukol5/mustr.py

Zadání:

Spočítejte průměr grafu s pomocí Floyd-Warshall algoritmu ⁷.

⁶https://en.wikipedia.org/wiki/Breadth-first_search

⁷https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm

Tento algoritmus je i spolu s pseudokódem a příkladem obstojně popsán na wikipedii ⁸. Tento algoritmus funguje i pro vážené grafy.

Video:

<https://www.youtube.com/watch?v=4OQeCuLYj-4>

Toto video názorně demonstruje základní ideu algoritmu.

Bonusový úkol: Layer of indirection

Pozorný programátor si jistě všiml určitého nedostatku v naší implementaci, která pro určité typy vstupů může vést k nekorektním výsledkům. Počítáme s tím, že vstupní graf indexuje všechny vrcholy s pomocí konečné posloupnosti přirozených čísel $a_n = n$ kde n jde od nuly. Problém tedy nastane jakmile tento předpoklad není splněn a posloupnost nezačíná nulou, nebo dokoce obsahuje 'díry'. Tzn. některá čísla v daném rozsahu se v posloupnosti indexů vrcholů vůbec nevyskytují. První možnost (tzn. indexy nezačínají nulou) se vyskytuje u kolekce karate klubu a toto je lehce řešitelné odečtením jedničky při počítání parametrů. V případě děr je situace komplikovanější a vyžadovala by vytvoření tzv. *layer of indirection* ⁹.

Tato vrstva by se mohla implementovat s pomocí hash map v Pythonu jednoduše *dictionary* ¹⁰, kde bychom uložili dvojice (*poradi_v_matici*, *index_vrcholu*) pro každý vrcholu. Pořadí v matici by už byla posloupnost přirozených čísel, která roste od nuly a neobsahuje mezery.

Zadání:

Cílem tohoto bonusového úkolu je rozšířit naši funkci `read_matrix`, tak aby

- vracela matici, kde nebudou žádné 'díry'
- vracela dictionary, s pomocí které bude možné pro index řádku matice dohledat původní index vrcholu

⁸https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm

⁹<https://en.wikipedia.org/wiki/Indirection>

¹⁰<https://docs.python.org/3/tutorial/datastructures.html#dictionaries>