

[\[\[courses:b6b36zal:zadani:10_dijkstra\]\]](#)[Course Ware](#)

Hide Sidebar

Login

Search

Trace: • [10_dijkstra](#)

—Table of Contents

- [10. úkol - Dijkstra](#)
 - [Zadání](#)
 - [Edge](#)
 - [Vertex](#)
 - [Dijkstra](#)
 - [Příklad použití a ukázka](#)
 - [Testování](#)
 - [Bodování](#)
 - [Termín odevzdání](#)

10. úkol - Dijkstra

Vaším desátým úkolem je naimplementovat [Dijkstrův algoritmus](#) a použít ho k hledání nejkratších cest v grafu ([SPT \(shortest path problem\)](#)). Graf dostanete zadán pomocí struktur (vrcholy a hrany) s tím, že hrany jsou kladně ohodnoceny a jsou orientované. V grafu mohou být cykly.

Zadání

Vytvořte soubor `dijkstra.py` (můžete využít šablonu [dijkstra.py](#)) a v něm vytvořte následující třídy:

Můžete použít tuto šablonu: [dijkstra.py](#)

Edge

Třída reprezentující hranu v grafu. Tato třída má následující proměnné:

- `source` - numerický identifikátor vrcholu, v kterém hrana začíná (int)
- `target` - numerický identifikátor vrcholu, v kterém hrana končí (int)
- `weight` - váha hrany (int)

Třída musí mít metodu:

```
__init__(self, source, target, weight)
```

- která je zavolána při vytvoření třídy a přijímá ID zdrojového vrcholu (`source`), ID cílového vrcholu (`target`) a váhu hrany (`weight`). Skript při vytváření hran použije pouze metodu `init` tzn skript nastaví `source`, `target` a `weight`. Nastavení ostatních parametrů (pokud je potřebujete) je již ve vaší režii. Můžete jejich nastavení samozřejmě přidat do metody `init` na tomto objektu, nebo můžete využít inicializace grafu v Dijkstre (`createGraph`).

Vertex

Třída reprezentující vrchol v grafu. Tato třída má následující proměnné:

- `id` - numerický identifikátor vrcholu (int)

- name - jméno vrcholu (String)
- minDistance - minimální vzdálenost, za kterou se lze dostat do tohoto vrcholu (int) z vrcholu, nad kterým byla provedena funkce computePath.
- previousVertex - předchozí vrchol - vrchol, přes který vede cesta do tohoto vrcholu pro minimální cestu (Vertex)
- edges - hrany, které začínají v tomto vrcholu (list of Edge)

Třída musí mít metodu

```
__init__(self, id, name)
```

, která se volá při vytvoření a natavuje vrcholu jeho id (id) a název(name). Skript při vytváření vrcholů použije pouze metodu init tzn skript nastaví id a jméno. Nastavení ostatních parametrů (pokud je potřebujete) je již ve vaší režii. Můžete jejich nastavení samozřejmě přidat do metody init na tomto objektu, nebo můžete využít inicializace grafu v Dijkstre (createGraph).

Dijkstra

Třída reprezentující Dijkstrův algoritmus. Tato třída má pouze jednu proměnnou

- vertexes - vrcholy grafu, nad kterými chceme provádět Dijkstrův algoritmus

Třída má tyto operace:

- createGraph(self, vertexes, edgesToVertexes) - metoda vytvoří graf ze zadaných vrcholů. Vertexes je pole objektů typu Vertex a edgesToVertexes je pole typu Edge.
- getVertexes(self) - metoda vrátí vrcholy, nad kterými je možné provést Dijkstrův algoritmus. TJ vrcholy, které jsou vytvořeny pomocí metody init.
- computePath(self, sourceId) - metoda nalezne nejkratší cesty ze zadaného vrcholu do všech vrcholů v grafu. Metoda nic nevrací, ale po skončení operace by měly mít všechny vrcholy vyplněnou proměnou minDistance, která reprezentuje minimální vzdálenost o zadaného vrcholu.
- getShortestPathTo(self, targetId) - metoda vrátí list vrcholů, přes které vede z vrcholu, nad kterými byla spuštěna operace computePath do vrcholu specifikovaného v targetId.
- resetDijkstra(self) - tato metoda vyresetuje aktuální výsledky po průchodu dijkstrovým algoritmem. Metoda nerozpojí či nezahodí graf, pouze ho vrátí do stavu, v jakém byl před operací computePath

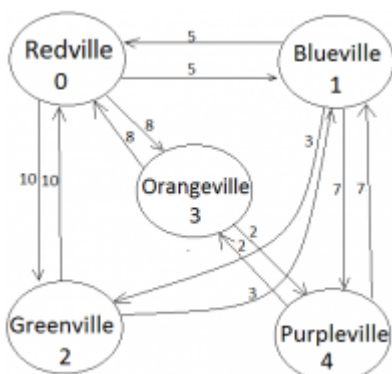
Pozn: proměnné sourceId a targetId jsou číselné. Je nutné si tedy zadaný graf uložit a poté vrcholy najít na základě jejich ID.

Pozn2: Dijkstrův algoritmus počítá s tím, že vzdálenost vrcholů, které nezpracoval je nekonečno. V Pythonu ho prosím reprezentujte takto: float('inf')

Příklad použití a ukázka

Vzhledem k delšímu zápisu příkladu použití je příklad umístěn v souboru [dijkstrause.py](#).

Níže je graf, který by vám měl vzniknout po správné konstrukci přiloženého příkladu:



Operace computePath nad vrcholem Redville našla v grafu nejkratší cesty a proto jsou minimální vzdálenosti z tohoto vrcholu následující: Printing min distance from vertex:Redville

Min distance to:Redville is: 0 - #V Redville začínáme

Min distance to:Blueville is: 5 - # Do Blueville se dostaneme přímou cestou hranou s vahou 5

Min distance to:Greenville is: 8 - # Do Greenville se se dostaneme nejlevněji přes Blueville (5) a poté do Greenville (3)

Min distance to:Orangeville is: 8 - # Do Orangeville se dostaneme nejlevněji přímou cestou hranou s vahou 8

Min distance to:Purpleville is: 10 - # Do Purpleville se dostaneme nejlevněji přes Orangeville (8) a poté do Greenville (2)

Poté resetujeme nastavení Dijkstry a necháme ji spočítat to samé pro další vrchol s ID 1 (Blueville) - doplnění cest je zřejmé

Printing min distance from vertex:Blueville

Min distance to:Redville is: 5

Min distance to:Blueville is: 0

Min distance to:Greenville is: 3

Min distance to:Orangeville is: 9

Min distance to:Purpleville is: 7

V případě, že chceme znát nejkratší cesty do všech vrcholů z vrcholu Blueville tak řešení může vypadat následovně:

Printing min distance from vertex:Blueville

Min distance to:Redville is: 5

Path is: Blueville, Redville

Min distance to:Blueville is: 0

Path is: Blueville

Min distance to:Greenville is: 3

Path is: Blueville, Greenville

Min distance to:Orangeville is: 9

Path is: Blueville, Purpleville, Orangeville

Min distance to:Purpleville is: 7

Path is: Blueville, Purpleville

Výše uvedené výpisy jsou výpisy ze souboru, kde je naznačeno jak si vaši implementaci můžete testovat.

dijkstra.se.py

Dijkstrův algoritmus používá k ukládání vrcholů, které jsou nejbližší k zdrojovému vrcholu prioritní frontu. Použijte ji také - nepoužití prioritní fronty může znamenat, že se vaše SPT budou lišit. Pokud přidáváte do prioritní fronty vrchol, jehož minimální cena se shoduje s některým vrcholem, který již ve frontě je, pak ho vložte před tento vrchol.

Testování

Náš skript bude testování provádět obdobně jako je tomu v referenčním případě s tím rozdílem, že z vaší strany není nutné cokoli vypisovat. Testovací skript si vytvoří instanci Dijkstra() a bude nad ní postupně volat metody, které chce otestovat. Poté si vždy pomocí metody getVertexes() získá aktuální reprezentaci grafu a tu porovná s referenčním řešením.

Bodování je rozděleno do několika částí - viz sekce bodování. V případě 3.bodu jsou vám zadávány náhodně generované grafy. U těchto grafů se může stát, že nebudou spojitě. V případě nespojitého grafu provádíte operace pouze nad částmi, které jsou spojitě.

Kromě správnosti řešení testujeme také jeho výkon. Hranice jsou nastaveny dostatečně vysoko, ale vzhledem k tomu, že jde o první testování výkonu tak bychom vás chtěli požádat aby jste v případě, že vám řešení neprochází na čas napsali na fóru.

K vyhodnocení používáme python verze 3.

Bodování

Úloha je hodnocena osmi body. Body jsou vám udělovány následovně:

1. Funguje vám správně část Dijkstry, která počítá minimální vzdálenosti na základních datech - 2b
2. Funguje vám správně část Dijkstry, která počítá minimální vzdálenosti a nejkratší cestu základních datech - 2b
3. Funguje vám správně část Dijkstry, která počítá minimální vzdálenosti a cesty na náhodných grafech - 4b

Implementujte Dijkstrův algoritmus, použití jiného algoritmu bude hodnoceno 0 body(a to i zpětně)!

Termín odevzdání

Úterní cvičení: 3.1.2017

Páteční cvičení: 6.1.2017

Vždy před začátkem vašeho cvičení.

Aby jste nemuseli na cvičení je nutné odevzdat tento úkol za plný počet bodů. Pro páteční cvičení do 6.12.2016 a pro úterní cvičení do 16.12.2016. Vždy před začátkem vašeho cvičení

[b6b36zal](#)

[cviceni](#)

[zadani](#)

[1_introduction_assignment](#)

[2_python_in_action](#)

[3_calculator](#)

[4_pi](#)

[5_polynoms](#)

[6_data_sorting](#)

[7_showroom](#)

[8_bst](#)

[9_permutations](#)

[10_dijkstra](#)

[cviceni](#)

[kombinovane_studium](#)

[zadani](#)

 [b6b36zal](#)

 [cviceni](#)


 [zadani](#)

 [1_introduction_assignment](#)

 [2_python_in_action](#)

 [3_calculator](#)

 [4_pi](#)

 [5_polynoms](#)

 [6_data_sorting](#)

 [7_showroom](#)

 [8_bst](#)

 [9_permutations](#)

 [10_dijkstra](#)

 [cviceni](#)

 [kombinovane_studium](#)



Groups:

courses/b6b36zal/zadani/10_dijkstra.txt · Last modified: 2016/10/05 22:48 by tomasma5

Login

Back to top