FIKS 8. Ročník, 2. kolo, úloha 3

Prohlídkové okruhy – řešení

Předpoklady

Přiložené je praktické řešení v pythonu *okruhy.py***. Ke správné funkci programu je potřeba do stejné složky vložit soubor se vstupem, jménem** *input.txt.* **Ve stejné složce program vytvoří/přepíše soubor** *output.txt* **s řešením. Řešení využívá dvě knihovny, tj.** *os* **a** *collections* **(obě jsou obsaženy v základní instalaci Pythonu)**

Teorie postupu:

Úloha je jednoduše řešitelná pomocí Breadth-first search (prohledávání do šířky). Vzhledem k tomu, že v jednom zadání je více dotazů, ale všechny se týkají grafu vytvořeného ze stejného zadání, uděláme si pro řešení *class,* které v *__init__* funkci dáme dvojce hran, ze kterých sestrojíme orientovaný graf. Dáme jí pak funkci, ve které se budeme ptát, zda v grafu existuje cesta mezi vrcholy dotazu, to nám vrátí buď *true* („Cesta existuje“) nebo *false* („Cesta neexistuje“).

Popis kódu:

Soustředím se pouze na class Solve a funkce v ní, ostatní funkce slouží pro zpracování vstupu.

[7-33] ***class Solve*** obsahuje dvě funkce, tj. rezervovaná metoda *__init__*, která vytvoří orientovaný graf ze vstupu a *doesExist*, což je funkce pro řešení jednotlivých dotazů.

[9-13] ***__init__*****:** při inicializaci a vytvoření objektu pomocí *Solve* musíme dodat seznam hran [9], ze kterých tato funkce poté vytvoří orientovaný graf pomocí modulu *defaultdict* z knihovny *collections* [11]. Poté se projde každá dvojce dodaných hran a zapíše se do grafu se začátkem jako klíčem, a s koncem jako hodnotou [12,13].

[15-33] ***doesExist:*** je metoda, která v grafu vytvořeném v *__init__* použije BFS, aby zjistila, zda existuje cesta mezi vrcholy dotazu. Nejprve se vytvoří set jménem *done* [17], do kterého budeme vkládat vrcholy, které jsme už navštívili a nejsou koncovým bodem a také proměnná *nodes*, do které vložíme začáteční vrchol dotazu [19]. Pak začneme while loop s podmínkou, že proměnná nodes není prázdná [21]. V loopu si založíme proměnnou *v* jako zrovna řešený vrchol a odstraníme jí z *nodes* [23], abychom jí neřešili vícekrát a nevytvořili tak nekonečný loop. Zkontrolujeme, jestli vrchol *v* odpovídá cíli dotazu. V případě že ano, cesta existuje a metoda vrací *true [24,25].* Pokud ne, přidáme vrchol do seznamu již navštívených vrcholů [27] a pomocí for loopu si zjistíme všechny vrcholy do kterých z něj vedou cesty [29]. Pokud tyto vrcholy nejsou v již navštívených, připojíme je k *nodes.* [30,31]. Ve chvíli, kdy na řádku 23 odstraníme vrchol, který je jediným vrcholem v *nodes*, není řešením dotazu a nenavazují na něj žádné nenavštívené body, *nodes* by šel do další iterace prázný, čímž nebude splněna podmínka while loopu a ten se ukončí. Metoda *doesExist* tedy vrátí *false* [33], protože cesta neexistuje.

Čas a paměť:

Časová a paměťová náročnost řešení dotazu se orientuje podle náročností BFS, tj. O(n) přičemž n je počet křižovatek v zoo, paměťová náročnost je O(n+k), kde n je počet křižovatek a k je počet cest mezi křižovatkami.

Testováno na Windows 11, Python verze 3.9.4 64-bit