Seminář 7 - cvičení

Cílem tohoto cvičení je implementovat vybrané grafové algoritmy a jejich experimentální vyhodnocení. Pro experimenty budeme potřebovat velké grafy, na kterých budeme algoritmy spouštět. Máme několik možností, jak grafy získat:

- Náhodné grafy: Zvolíme počet vrcholů a hran, každou hranu přidáme mezi dva náhodně zvolené vrcholy.
- Pravidelné grafy: Vygenerujeme různé n-rozměrné krychle, pravidelné mřížky, lineárně spojené kliky. Konkrétně budeme generovat regulární grafy.
- Grafy založené na reálných datech:

Naprogramujte rekurzivní i nerekurzivní prohledávání grafu do hloubky. Porovnejte algoritmy, zda prochází uzly ve stejném pořadí. Zda je i v případě nerekurzivního algoritmu každá hrana zpracována právě jednou. Porovnejte také dobu běhu obou implementací.

Vyberte si libovolný další algoritmus nad grafem (hledání nejkratší cesty, minimální kostra grafu, hledání eulerovské nebo hamiltonovské cesty, barvení grafu...) a implementujte ho.

Podívejte se také na následující funkce:

`graph()`, `distances()`, `conncomp()`, `shortestpath()`, `isdag()`

Zvolte si libovolnou reprezentaci grafu.

Náhodný graf

Naprogramujte algoritmus pro vytvoření náhodného grafu. Počty vrcholů a hran bude možné zadat pomocí slideru, který vzhodně omezte.

Regulární graf

n ... vrcholů

k ... stupeň vrcholů

Nutná a postačující podmínka existence regulárního grafu je $n \geq k + 1$ a zároveň $k \cdot n$ je sudé.

Naprogramujte algoritmus pro vytvoření regulárního grafu s `n` uzly stupně `k`. `n` a `k` je možné zadat pomocí slideru. Omezte je vhodně, aby bylo možné graf zkonstruovat.

Zde je jeden z algoritmů. Je možné naprogramovat jiný.

Algoritmus (https://mediatum.ub.tum.de/doc/1315533/document.pdf):

1) Začneme s grafem `G`, který obsahuje n vrcholů a žádné hrany.

2) Opakuj dokud není množina `S` prázdná. `S` představuje množinu dvojic vrcholů grafu `G`, které nejsou sousední a pro které platí, že oba vrcholy mají nejvýše stupeň `k - 1` (označme `d(u)` stupeň vrcholu `u`). Vyber náhodnou dvojici `(u,v)` z `S` s pravděpodobností `(k - d(u))(k - d(v))`. Dvojici přidej do `G`.

3) Pokud je graf regulární, vrať ho, jinak začni s krokem 1.

Reálná data

Například data 'Simple_pairwise-London_tube_map.txt' představující mapu Londýnského metra. Pro načtení dat použijte funkci `readcell()`.

Z načtených dat vytvořte graf. V každém řádku je dvojice názvů stanic, které spolu sousedí.

Pomocí funkce `distance()` kolik stanic jsou od sebe vzdálené stanice `"Barons_Court"` a `"Brent_Cross"`.

Pomocí funkce `shortestpath()` můžeme vypsat všechny stanice. Vypište nejkratší cestu ze stanice `"Barons_Court"` do `"Brent_Cross"`.

Prohledávání grafu do hloubky

Rekurzivní

Naprogramujte rekurzivní funkci na prohledávání grafu do hloubky.

Nerekurzivní

Naprogramujte nerekurzivní funkci na prohledávání grafu do hloubky.

Porovnejte rekurzivní a nerekurzivní funkci. Zda prochází uzly ve stejném pořadí. Porovnejte i dobu výpočtu.

Vyberte si další grafový algoritmus a naimplementujte ho.

Zde budou vaše funkce