7. Teorie grafů

1. Význam

- velice elegantní nástroj pro systematické zpracování dat z reálného světa
- počítačová implementace je většinou jednoduchá (BFS a DFS)
- jedná se v základu o matematickou strukturu z diskrétní matematiky (práce s množinami)
- zakladatel Leonhard Euler v 18. století
- graf G je množina uspořádaných dvojic, vrcholů V a hran E; $G = (V, E)$
- rozlišujeme neorientované a orientované - to jsou diagramy
- stupeň daného vrcholu označuje počet jeho sousedů

2. Důležité grafy

- úplný graf: každý je propojen s každým, resp. $V = v_1, v_2, \dots, v_n$ a $E = \binom{n}{2}$, značíme K_n
- úplný binární graf: $V = A \cup B \wedge \emptyset = A \cap B$ a $E = \{\{a, b\}, a \in A \wedge b \in B\}$, značíme $K_{n,m}$
- cyklický graf: $V = v_1, v_2, \dots, v_n$ a $E = \{\{v_i, v_{i+1}\}, i \in \{1, \dots, n-1\}\} \cup \{v_1, v_n\}$, značíme C_n
- cesta: $V = \{v_1, v_2, \dots v_n\}$ a $E = \{\{v_i, v_{i+1}\}, i \in \{1, \dots, n-1\}\}$, značíme P_n
- úplný biparitní graf: otázka rovinnosti grafu $K_{3,3}$, tři domy, tři studny, propojit všechny cesty, aniž by se křížily, řešení na torusu

3. Reprezentace grafů

- **matice sousednstí**, vytvoříme matici $n \times n$ tak, že na i-tém řádku a j-tém sloupci zapíšeme 1 pakliže existuje hrana grafu mezi vrcholy v_i a v_j, nulu, pakliže vrchol neexistuje
  - v čase $\mathcal{O}(n)$ jsme schopni zjistit, jestli dva vrcholy spolu sousedí
  - zabere prostor $\mathcal{O}(n^2)$
- **seznam sousedů**, pro každý vrchol vytvoříme seznam jeho sousedů, seznamů tedy bude n
  - reprezentace zabírá prostor $\mathcal{O}(n + m)$, kde m je počet vrcholů
  - dále se dají seznamy komprimovat, takže potom hovoříme o komprimovaných seznamech

4. Vyhledávací algoritmy

- základní algoritmy, které jsou schopné určitým způsobem procházet graf, a tím pádem v něm hledat

A) BFS - prohledávání do šířky

- jedná se o nejjednodušší
- breadth-first search
- na vstupu dostaneme konečný graf, BFS jej postupně prochází, a to tak, že první zvolí vrchol v_0, zjistíme jeho následníky a ty projdeme, poté jejich následníky atd.

- rozlišujeme tři typy stavů vrcholu: nenalezené (algoritmus o nich vůbec neví, zatím), otevřené (jsou ty, které víme, že existují, ale neznáme jejich sousedy), uzavřené (již jsme prozkoumali, není třeba se vracet)
- takže první prohlásíme v_0 za otevřený, ostatní jsou ještě uzavřené, po prozkoumání otevřeme sousedy a uzavřeme v_0 a tak dále
- BFS se vždy zastaví, je to dáno tím, že se prošlé vrcholy uzavřou a těch je konečné množství
- pakliže je vrchol v_i dosažitelný z vrcholu v_0, pak jej BFS nalezne, pokud je nedosažitelný zůstane nenalezený
- můžeme díky BFS zjistit, jestli je graf souvislý
- vrstvy grafu: díky postupné posloupnosti můžeme definovat vrstvy grafu, tudíž n-tá vrstva grafu obsahuje právě ty vrcholy, které byly uzavřeny v n-té iteraci BFS
- časová složitost je $\mathcal{O}(n+m)$

B) DFS - prohledávání do hloubky

- depth-first search
- je založen na podobné filozofii jako BFS, ale vrcholy zpracovává rekurzivně, neboli kdykoliv narazí na dosud nenalezený vrchol, otevře jej a zavolá se rekurzivně na všechny jeho nenalezené následníky, následně původní vrchol uzavře a vrátí se z rekurze
- jinými slovy DFS pracuje tak, že si vytváří konečné cesty (vedoucí buď do vrcholu, který má pouze jednu hranu, nebo do vrcholu již označeného jako otevřený), poté vrcholy, které se na cestě nacházejí označí za uzavřené, ale jenom ty, které neposkytují další cesty
- to znamená, že pokud DFS doběhne a projde všechny vrcholy, pak je graf spojitý, právě ty vrcholy, které jsou z v_0 dosažitelné
- pracuje v čase $\mathcal{O}(n+m)$, v prostoru $\mathcal{O}(n+m)$
- jednoduše se dá představit průběh DFS na stromovém grafu, jednoduše vrcholy, které leží na kořenu, který právě objevujeme jsou otevřené, ale zatím nezavřené