AG1 – BI-SPOL-4

Základní pojmy teorie grafů. Grafové algoritmy: procházení grafu do šířky a do hloubky, určení souvislých komponent, topologické uspořádání, vzdálenosti v grafech, konstrukce minimální kostry a nejkratších cest v ohodnoceném grafu.

Obsah

1 Základní pojmy **2**
- 1.1 Graf . . . 2
- 1.2 Doplněk . . . 2
- 1.3 Isomorfismus . . . 2
- 1.4 Vrcholy . . . 2
- 1.5 Podgraf . . . 2

2 Typy grafů **4**
- 2.1 Uplný graf K_n . . . 4
- 2.2 Úplný k-partitní graf . . . 4
- 2.3 Cestat P_m . . . 4
- 2.4 Kružnice C_n . . . 4
- 2.5 Hvězda S_n . . . 4

3 Procházení grafu **4**
- 3.1 Vzdálenost . . . 4
- 3.2 DFS . . . 4
- 3.3 BFS . . . 4

4 Souvislost **4**
- 4.1 Souvislý graf . . . 4
- 4.2 Souvislá komponenta . . . 5

5 Topologické uspořádání grafu **5**
- 5.1 Definice . . . 5
- 5.2 TopSort . . . 5

6 Ohodnocený graf **5**
- 6.1 Minimální kostra . . . 5
- 6.2 Hledaní nejkratšíí cesty . . . 6

1 Základní pojmy

1.1 Graf

Neorientovaný graf je uspořádaná dvojice (V, E), kde

- V je neprázdná konečná množina vrcholů,
- E je množina hran.

Množina všech možných hran: $\binom{V}{2}$**.**
Platí tedy $E \subseteq \binom{V}{2} \subseteq 2^V$, kde 2^V je množina všech podmnožin množiny V.

Nechť G **je graf.** Pak:

- $V(G)$ značí jeho množinu vrcholů a $|V(G)|$ velikost této množiny
- $E(G)$ značí jeho množinu hran a $|E(G)|$ velikost této množiny

Dále, **nechť** $e = \{u, v\}$ **je hrana grafu G.** Pak:

- u a v jsou **koncové vrcholy**
- oba koncové body jsou si na grafu G navzájem **sousedy**
- oba koncové body jsou **incidentní** s hranou e

1.2 Doplněk

Doplěk $\overline{G}$ grafu $G = (V, E)$ je graf $(V, \binom{V}{2} \backslash E)$.

(a) Graf G (b) Graf $\overline{G}$

1.3 Isomorfismus

Grafy G a H jsou **isomorfní**, právě tehdy, když existuje zobrazení $f : V(G) \rightarrow V(H)$, kde f je bijekce a pro každou dvojici vrcholů u a v z $V(G)$ platí, $\{u, v\} \in E(G)$ právě tehdy, když $\{f(u), f(v)\} \in E(H)$.

Automorfismus grafu G je isomorfismus, grafu G se sebou samým. (ukazuje symetrie grafu)

1.4 Vrcholy

Stupeň vrcholu v v grafu G je počet hran, které vrchol v obsahují a značíme jej $deg_G(v)$.

Otevřené okolí vrcholu v v grafu G je množina všech sousedů vrcholu v a značíme jej $N_G(v)$.

Uzavřené okolí vrcholu v v grafu G je $N_G(v) \cup \{v\}$ a značíme jej $N_G[v]$.

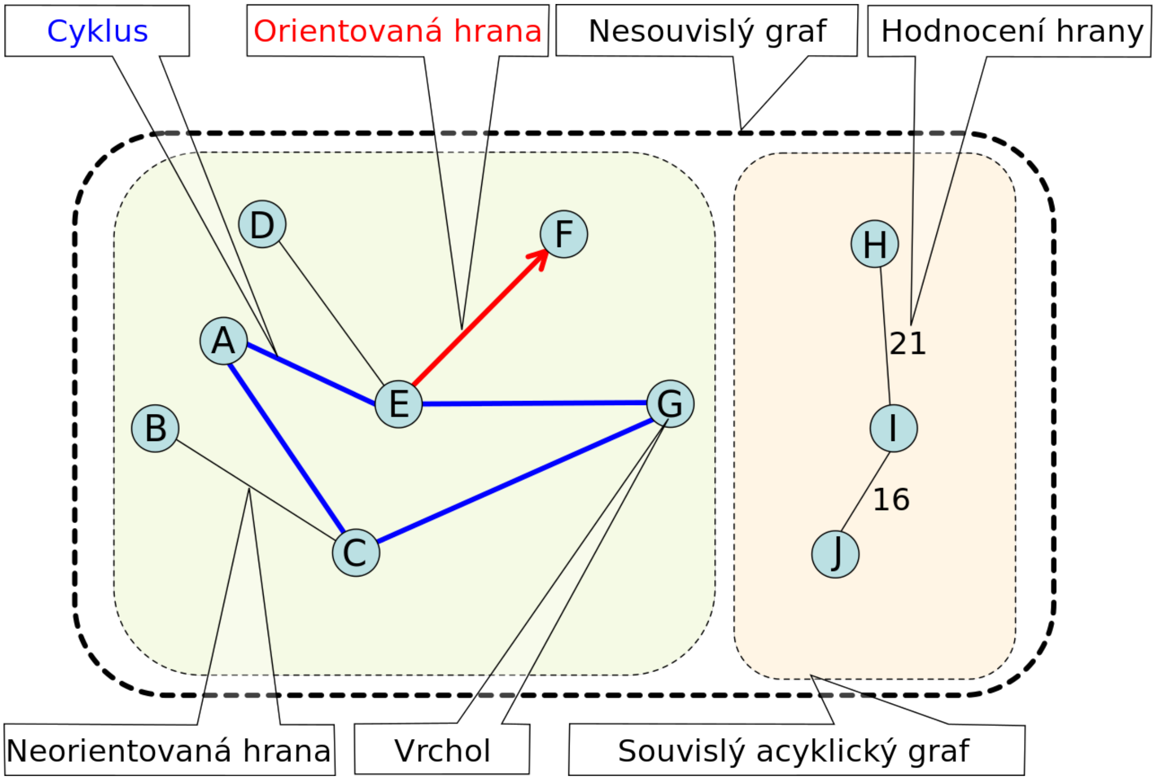
Regulární graf, je graf, ve kterém mají všechny vrcholy stejný stupeň.

Princip sudosti $\sum_{v \in V} deg_G(v) = 2|E|$

1.5 Podgraf

Graf H je **podgrafem** grafu G, když $V(H) \subseteq V(G)$ a $E(H) \subseteq E(G)$; tuto skutečnost značíme $H \subseteq G$.

Graf H je **indukovaným podgrafem** grafu G, když $V(H) \subseteq V(G)$ a $E(H) = E(G) \cap \binom{V(H)}{2}$; tuto skutečnost značíme $H \leq G$.



2 Typy grafů

2.1 Uplný graf K_n

Graf, kde jsou všechny vrcholy spojeny hranou se všemi ostatními vrcholy.

Nechť $n \geq 1$.
Úplný graf na n vrcholech K_n je graf $(V, \binom{V}{2})$, kde $|V| = n$.

2.2 Úplný k-partitní graf

Graf, rozdělený na k skupin, kde je každý vrchol spojen hranou se všemi vrcholy ze všech ostatních skupin, ale není spojen s žádným vrcholem ze své vlastní skupiny.

Nechť $\forall i\{1, ...k\} : n_i \geq 1$.
Úplný k-partitní graf $K_{n1,n2,...,nk}$ je graf $(\cup_{i=1}^{k} V_i, E)$,
kde $\forall i, j\{1, ..., k\}, i \neq j : V_i \cap V_j = , |V_i| = n_i$
a $E = \{\{x, y\}|\exists i, j \in \{1, ..., k\}, i \neq j : xV_i, yV_j\}$,
neboli $E = \binom{\cup_{i=1}^{k} V_i}{2} \setminus \cup_{i=1}^{k} \binom{V_i}{2}$.

2.3 Cestat P_m

Graf, který má m hran, $m + 1$ vrcholů a tvoří cestu.

Nechť $m \geq 0$.
Cesta délky m (s m hranami) P_m **je graf**
$(\{0, ..., m\}, \{\{i, i + 1\}|i \in \{0, ..., m - 1\}\})$.

2.4 Kružnice C_n

Graf, který ma n vrcholů i hran a na všechny vrcholy navazují právě dvě hrany.

Nechť $n \geq 3$.
Kružnice délky n C_m je graf
$(\{1, ..., n\}, \{\{i, i + 1\}|i \in \{1, ..., n - 1\}\} \cup \{\{1, n\}\})$.

2.5 Hvězda S_n

Uplný bipartitní graf, kde je v první partitě právě jeden vrchol a ve druhé alespoň jeden vrchol.

Nechť $n \geq 1$.
Hvězda s n paprsky S_n je graf $K_{1,n}$.

3 Procházení grafu

3.1 Vzdálenost

Vzdálenost $d(u, v)$ dvou vrcholů u a v v (orientovaném) grafu G je délka nejkratší (orientované) cesty v G z vrcholu u do vrcholu v. Pokud z u do v žádná cesta neexistuje, definujeme $d(u, v) = \inf$.

3.2 DFS

Prohledávání do hloubky

3.3 BFS

Prohledávání do šířky

4 Souvislost

4.1 Souvislý graf

Graf G je **souvislý**, pokud pro každé dva vrcholy u, v v grafu G existuje u-v-cesta.

```
Algoritmus DFS_graf (graf G, vrchol v):
pro každý vrchol u  V (G):
    stav(u) := nenalezený
DFS(v)

DFS (vrchol v):
Když stav(v) není nenalezený
    return
stav(v) := otevřený
Pro každého souseda u vrcholu v:
    DFS(u)
stav(v) := uzavřený
```

```
Algoritmus BFS(G, s):
pro každý vrchol v  V (G):
    stav(v) := nenalezený
    D(v) := P(v) := undef
stav(s) := otevřený
D(s) := 0
Q := fronta obsahující s
Dokud je fronta Q neprázdná:
    Odeber začátek fronty Q, označ ho v
    Pro všechny sousedy w vrcholu v:
        Pokud stav(w) = nenalezený:
        stav(w) := otevřený
        D(w) := D(v) + 1
        P(w) := v
        přidej w do fronty Q
    stav(v) := uzavřený
```

4.2 Souvislá komponenta

Indukovaný podgraf H grafu G je souvislou komponentou, pokud je souvislý a neexistuje žádný souvislý podgraf F, $F \neq H$, grafu G takový, že $H \subseteq F$. (Souvislá komponenta je tedy v inkluzi maximální souvislý podgraf grafu G).

5 Topologické uspořádání grafu

5.1 Definice

Topologické uspořádání orientovaného acyklického grafu $G = (V, E)$ je takové pořadí vrcholů $v_1, v_2, ..., v_n$ grafu G, že pro každou hranu $(vi, vj) \in E$ platí $i < j$.

5.2 TopSort

6 Ohodnocený graf

6.1 Minimální kostra

Nechť $G = (V, E)$ je souvislý neorientovaný graf a $w : E \to R$ váhová funkce, která přiřazuje hranám čísla – jejich váhy. Váhovou funkci můžeme přirozeně rozšířit na podgrafy: Váha $w(H)$ podgrafu $H \subseteq G$ je součet vah jeho hran. Kostra je minimální, pokud má mezi všemi kostrami nejmenší váhu.

```
Algoritmus TopSort(orientovaný G)
(1) Q je prázdná fronta
(2) δ[] = pole vstupních stupňů vrcholů G,
          na počátku vynulované
(3) Pro každou hranu (u,v) ∈ E(G):
(4)      δ[v]++
(5) Vlož do Q všechny vrcholy z s δ[z] = 0
(6) Dokud fronta Q není prázdná:
(7)      Odeber prvek z ze začátku fronty Q
(8)      Vypiš z
(9)      Pro každou hranu (z,w) vedoucí ze z:
(10)          δ[w]--
(11)          Pokud δ[w] = 0, zařaď w do Q
(12) Pokud nebyly zpracovány všechny vrcholy,
(13)      graf G obsahuje orientovaný cyklus
```

```
Algoritmus MinKostraJarník (G = (V,E), w : E → ℝ
prostá)
(1) v0 := libovolný vrchol grafu
(2) T := strom obsahující vrchol v0 a žádné hrany
(3) Dokud existuje hrana {u,v} taková,
                     že u ∈ V(T) a v ∉ V(T):
(4)     Přidej nejlehčí takovou hranu spolu s v do T
(5) Vrať T
```

```
Algoritmus MinKostraKruskal
                  (G = (V,E), w : E → ℝ prostá)
(1) Uspořádej hrany podle vah: w(e1) < ... < w(em)
(2) T := (V,∅)
(3) Pro i = 1,...,m opakuj:
(4)     označ u,v krajní vrcholy hrany ei
(5)     Pokud u a v leží v různých komponentách lesa T:
(6)          T := T + ei
(7) Vrať T
```

6.2 Hledaní nejkratšíí cesty

Dijkstrův: *předpokládá nezáporné ohodnocení hran*

Bellmanův-Fordův: *předpokládá neexistenci záporných cyklů v grafu*

Algoritmus Dijkstra(G, v_0)

Pro všechny vrcholy v:
    stav(v) := nenalezený
    $h(v)$:= $+\infty$
stav(v_0) := otevřený
$h(v_0)$:= 0
Dokud existují nějaké otevřené vrcholy:
    Vyber otevřený vrchol v, jehož $h(v)$ je nejmenší.
    Pro všechny následníky w vrcholu v:
        Pokud $h(w) > h(v) + \ell((v, w))$:
            $h(w)$:= $h(v) + \ell((v, w))$
            stav(w) := otevřený
            $P(w)$:= v
    stav(v) := uzavřený
Vrať pole vzdáleností h a pole předchůdců P

Algoritmus Bellman-Ford(G, v_0)

Pro všechny vrcholy v:
    stav(v) := nenalezený; $h(v)$:= $+\infty$
stav(v_0) := otevřený; $h(v_0)$:= 0
Vlož v_0 do fronty
Dokud je fronta neprázdná:
    Vyjmi první vrchol z fronty a označ ho v
    Pro všechny následníky w vrcholu v:
        Pokud $h(w) > h(v) + \ell((v, w))$:
            $h(w)$:= $h(v) + \ell((v, w))$
            Pokud stav$(w) \neq$ otevřený
                Přidej w do fronty
            stav(w) := otevřený
            $P(w)$:= v
    stav(v) := uzavřený
Vrať pole vzdáleností h a pole předchůdců P