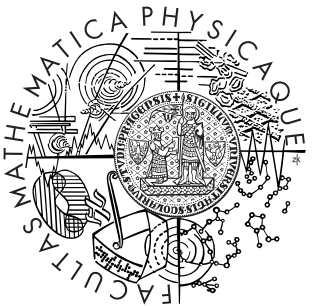
Algoritmy a datové struktury 1

2/2 Z+Zk, NTIN060

pomocné slajdy ke grafovým algoritmům

Pavel Veselý (IUUK)



vesely+ads1@iuuk.mff.cuni.cz

https://iuuk.mff.cuni.cz/∼vesely/vyuka/LS2122/ads1.html

Mosty: opakování

Hrana $e \in G$ je most, pokud $G - e$ má více komponent než G

Mosty: opakování

Hrana $e \in G$ je most, pokud $G - e$ má více komponent než G

- Lemma: e *není* most právě tehdy, když e leží na kružnici
- Pozorování: zpětné hrany nemohou být mosty

Prohledávání do hloubky — kód

function DFSMAIN($G = (V, E)$)
    $\forall v \in V$: stav(v) $\leftarrow$ (N) ▷ Výchozí stav je (N)=nenalezený
    $\forall v \in V$: in(v), out(v) $\leftarrow \infty$
    $T \leftarrow 0$
    for all $\forall v \in V$: stav(v) = (N) **do** ▷ projdeme i nesouvislý graf
        DFSrec(v)
    end for
end function
function DFSREC(v)
    stav(v) $\leftarrow$ (O) ▷ Otevřeme vrchol
    $T \leftarrow T + 1$, in(v) $\leftarrow T$
    for all $\forall vw \in E$: **do**
        if stav(w) = (N) **then**
            DFSrec(w)
        end if
    end for
    stav(v) $\leftarrow$ (Z) ▷ Zavřeme vrchol
    $T \leftarrow T + 1$, out(v) $\leftarrow T$
end function

Mosty: opakování

Hrana $e \in G$ je most, pokud $G - e$ má více komponent než G

- Lemma: e *není* most právě tehdy, když e leží na kružnici
- Pozorování: zpětné hrany nemohou být mosty

Mosty: opakování

Hrana $e \in G$ je most, pokud $G - e$ má více komponent než G

- Lemma: e *není* most právě tehdy, když e leží na kružnici
- Pozorování: zpětné hrany nemohou být mosty

Artikulace

Vrchol $v \in G$ je artikulace, pokud $G - v$ má více komponent než G

Mosty: opakování

Hrana $e \in G$ je most, pokud $G - e$ má více komponent než G

- Lemma: e *není* most právě tehdy, když e leží na kružnici
- Pozorování: zpětné hrany nemohou být mosty

Artikulace

Vrchol $v \in G$ je artikulace, pokud $G - v$ má více komponent než G

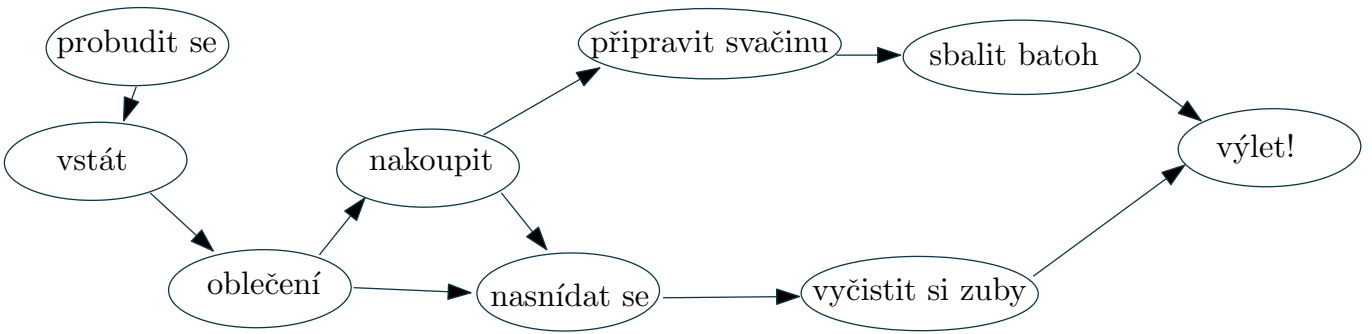
- Lemma: v *není* artikulace právě tehdy, když pro každé dva jeho různé sousedy x a y existuje kružnice obsahující hrany vx a vy

Acyklické orientované grafy

Directed acyclic graphs (**DAG**)

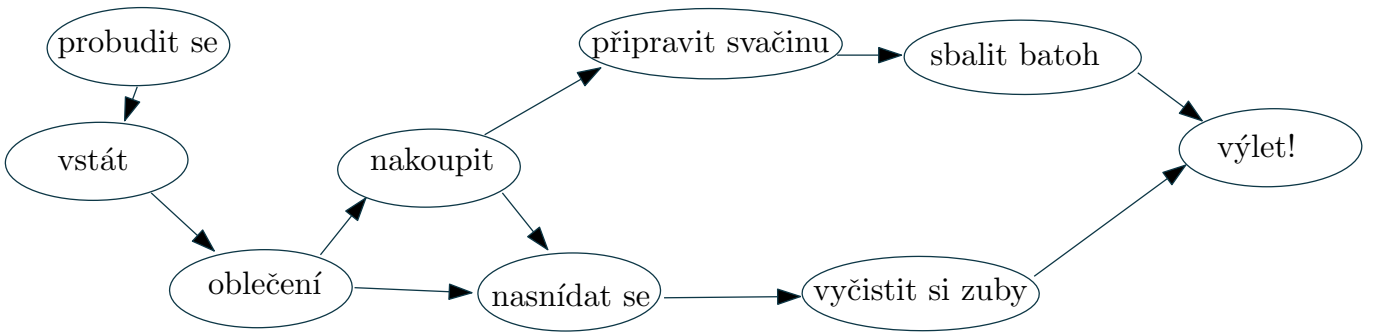
Acyklické orientované grafy

Directed acyclic graphs (**DAG**)



Acyklické orientované grafy

Directed acyclic graphs (**DAG**)



- Plánování pomocí topologického pořadí (uspořádání)

Hledání komponent silné souvislosti (KSS)

Algoritmus Kosaraju-Sharir ('78 a '81)

Ekvivalence: $x \leftrightsquigarrow y$, pokud existuje sled z x do y a sled z y do x

- Třídám ekvivalence $\leftrightsquigarrow$ se říká komponenty silné souvislosti (KSS)
- $\mathcal{C}(G)$ je orientovaný graf KSS, tedy vrcholy jsou KSS $C_1, \ldots, C_k$
  - (C_i, C_j) je hrana v $\mathcal{C}(G)$, pokud $\exists x \in C_i, y \in C_j : xy$ je hrana G
- Lemma: $\mathcal{C}(G)$ je acyklický (DAG)

Hledání komponent silné souvislosti (KSS)

Algoritmus Kosaraju-Sharir ('78 a '81)

Ekvivalence: $x \leftrightsquigarrow y$, pokud existuje sled z x do y a sled z y do x

- Třídám ekvivalence $\leftrightsquigarrow$ se říká komponenty silné souvislosti (KSS)
- $\mathcal{C}(G)$ je orientovaný graf KSS, tedy vrcholy jsou KSS $C_1, \ldots, C_k$
  - (C_i, C_j) je hrana v $\mathcal{C}(G)$, pokud $\exists x \in C_i, y \in C_j : xy$ je hrana G
- Lemma: $\mathcal{C}(G)$ je acyklický (DAG)

Pozorování:

1. existuje ≥ 1 zdrojová KSS a ≥ 1 stoková KSS
2. DFS na libovolném vrcholu stokové KSS C_s navštíví právě C_s

Hledání komponent silné souvislosti (KSS)

Algoritmus Kosaraju-Sharir ('78 a '81)

Ekvivalence: $x \leftrightsquigarrow y$, pokud existuje sled z x do y a sled z y do x

- Třídám ekvivalence $\leftrightsquigarrow$ se říká komponenty silné souvislosti (KSS)
- $\mathcal{C}(G)$ je orientovaný graf KSS, tedy vrcholy jsou KSS $C_1, \ldots, C_k$
  - (C_i, C_j) je hrana v $\mathcal{C}(G)$, pokud $\exists x \in C_i, y \in C_j : xy$ je hrana G
- Lemma: $\mathcal{C}(G)$ je acyklický (DAG)

Pozorování:

1. existuje ≥ 1 zdrojová KSS a ≥ 1 stoková KSS
2. DFS na libovolném vrcholu stokové KSS C_s navštíví právě C_s
3. pro opakovaně spouštěné DFS platí, že poslední uzavřený vrchol je ve zdrojové komponentě
   - Definice: $G^T = (V, \{vu \mid uv \in E(G)\})$ je transponovaný graf
4. KSS G jsou shodné jako KSS G^T, neboli $\mathcal{C}(G^T) = \left(\mathcal{C}(G)\right)^T$

Hledání komponent silné souvislosti (KSS)

Tarjanův algoritmus ('72)

- jen jedno spuštění DFS
- Lemma: každá KSS indukuje v DFS stromu *slabě souvislý* podgraf
  - (důkaz v učebnici)

Hledání nejkratších cest z v_0

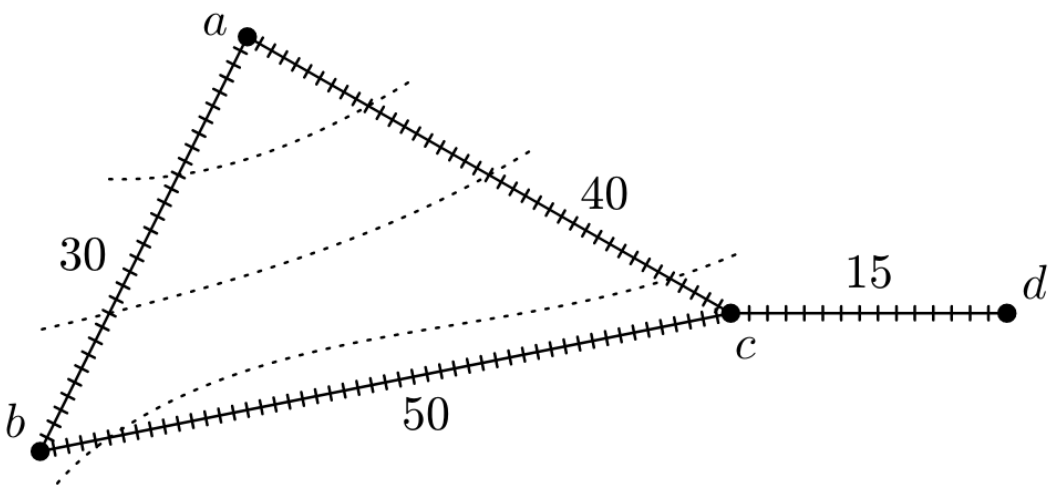
Dijkstrův algoritmus ('56)

- pro grafy bez záporných hran

Hledání nejkratších cest z v_0

Dijkstrův algoritmus ('56)

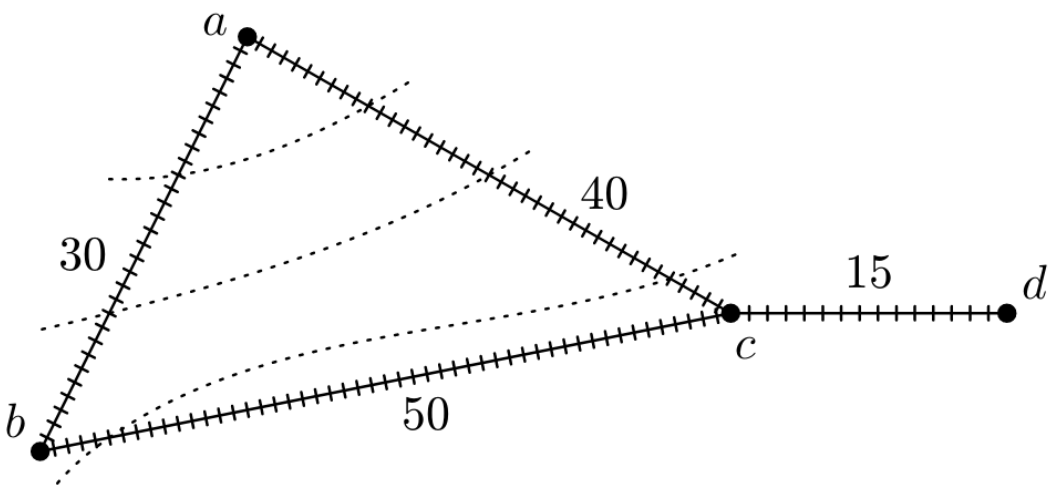
- pro grafy bez záporných hran



Hledání nejkratších cest z v_0

Dijkstrův algoritmus ('56)

- pro grafy bez záporných hran



TODO list:

1. časová složitost (za předpokladu, že každý vrchol uzavřen jen jednou)
2. korektnost pro libovolné nezáporné délky hran
3. co s grafy se zápornými hranami, ale bez záporných cyklů?

Hledání nejkratších cest z v_0

Bellmanův-Fordův algoritmus ('58 a '56; Shimbel '55)

- pro grafy bez záporných cyklů (ale záporné hrany jinak nevadí)

Hledání nejkratších cest z v_0

Bellmanův-Fordův algoritmus ('58 a '56; Shimbel '55)

- pro grafy bez záporných cyklů (ale záporné hrany jinak nevadí)
- Časová složitost $O(n \cdot m)$ — umí se lepší algoritmus?

Hledání nejkratších cest mezi každou dvojicí vrcholů

Floydův-Warshallův algoritmus ('62; Roy '59)

- pro grafy bez záporných cyklů (ale záporné hrany jinak nevadí)

Hledání minimální kostry — 3 základní hladové algoritmy

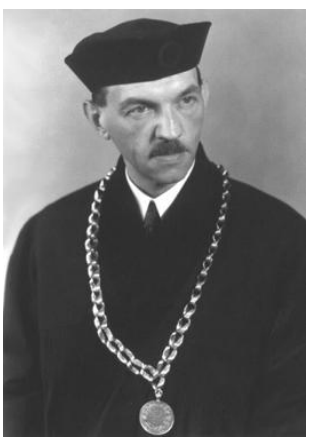
Hledání minimální kostry — 3 základní hladové algoritmy

1. Jarníkův algoritmus [Jarník '30, Prim '57, Dijkstra '59]

- staví min. kostru z jednoho vrcholu

Vojtěch Jarník (1897 – 1970)

- český matematik
- profesor na Karlově Univerzitě
- pomohl založit Československou akademii věd
- knihy o integrálním a diferenciálním počtu
- významné příspěvky v teorii čísel, matematické analýze a kombinatorické optimalizaci



Hledání minimální kostry — 3 základní hladové algoritmy

1. Jarníkův algoritmus [Jarník '30, Prim '57, Dijkstra '59]

- staví min. kostru z jednoho vrcholu

Hledání minimální kostry — 3 základní hladové algoritmy

1. Jarníkův algoritmus [Jarník '30, Prim '57, Dijkstra '59]

- staví min. kostru z jednoho vrcholu

2. Borůvkův algoritmus [Borůvka '26], znovu objeven třikrát

- Borůvkova motivace: konstrukce efektivní elektrické sítě na Moravě
- staví min. kostru paralelně ze všech vrcholů
  - „paralelní Jarník“

Hledání minimální kostry — 3 základní hladové algoritmy

1. Jarníkův algoritmus [Jarník '30, Prim '57, Dijkstra '59]

- staví min. kostru z jednoho vrcholu

2. Borůvkův algoritmus [Borůvka '26], znovu objeven třikrát

- Borůvkova motivace: konstrukce efektivní elektrické sítě na Moravě
- staví min. kostru paralelně ze všech vrcholů
  - „paralelní Jarník“

3. Kruskalův algoritmus [Kruskal '56]

- přidává hrany od nejlehčí po nejtěžší, pokud přidání hrany nevytvoří cyklus

Minimální kostry — co se umí?

- Algoritmus s lineární časovou složitostí pro
  - celočíselné váhy
  - „husté“ grafy, stačí $m/n \geq \log\log\log n$

Minimální kostry — co se umí?

- Algoritmus s lineární časovou složitostí pro
  - celočíselné váhy
  - „husté“ grafy, stačí $m/n \geq \log\log\log n$

- Algoritmus s „průměrně“ lineární časovou složitostí [Karger, Klein, Tarjan '95]
  - Střední hodnota časové složitosti je $O(m)$

Minimální kostry — co se umí?

- Algoritmus s lineární časovou složitostí pro
  - celočíselné váhy
  - „husté“ grafy, stačí $m/n \geq \log\log\log n$
- Algoritmus s „průměrně“ lineární časovou složitostí [Karger, Klein, Tarjan '95]
  - Střední hodnota časové složitosti je $O(m)$
- Optimální deterministický algoritmus v porovnávacím modelu
  - [Pettie & Ramachandran '02]
  - Neznáme však jeho časovou složitost!
  - Nejlepší horní odhad je $O(m\alpha(m, n))$ [Chazelle '00]
  - α je inverze Ackermannovy funkce
  - $\alpha(m, n) \leq 4$ pro všechny *praktické* hodnoty m, n