# 08. Nejkratší cesty. Úloha obchodního cestujícího. Heuristiky a aproximační algoritmy. Metoda dynamického programování. Problém batohu. Pseudo-polynomiální algoritmy

## 1. Nejkratší cesta v grafu

- **sled** je libovolná posloupnost vrcholů a hran jdoucí po sobě při průchodu grafem
- **cesta** je sled bez cyklů
- instance: orientovaný graf G, váhy c : E(G) → R a dva vrcholy s, t∈ V(G)
- pokud nejkratší sled neobsahuje cyklus záporné délky, je zároveň i nejkratší cestou
- pokud graf obsahuje **cyklus se zápornou délkou**, pak jde o NP-obtížný problém

### Trojúhelníková nerovnost

- jestliže graf neobsahuje cyklus se zápornou délkou, pak pro všechny trojice vrcholů i , j , k platí: l(i,j) ≤ l(i,k) + l(k,j).......... l(i,j) je délka nejkratší cesty z i do j
Bellmanova rovnice
- jestliže graf neobsahuje cyklus se zápornou délkou, pak pro všechny trojice vrcholů i , j , k platí: l(i,j) ≤ min{l(i,k) + c(k,j)}

### Podobné úlohy

- **hledání nejdelších cest** se řeší obrácením znamének u délek všech hran. Tím jsme převedli hledání maxima na hledání minima.

- pokud jsou **ohodnocené jen vrcholy** a ne hrany, lze převést na hledání v grafu s ohodnocenými hranami: každý vrchol v původního grafu nahradíme dvojicí vrcholů v1 a v2, spojíme je hranou o délce rovné původní hodnotě vrcholu v hrany, které končily ve vrcholu v přesměrujeme do v1, hrany, které vycházely z vrcholu v budou vycházet z vrcholu v2

### Dijkstrův algoritmus

- vždy vyberu nejlevnější uzel, který není closed, z tohoto uzlu zkouším pinkat do všech OPEN sousedů a zkoušíme, jestli se do nich dostaneme přes tento uzel levněji. Pokud ano, nastaví se nová cena a zapíše se předek na cestě do pole.
- omezení: neumí grafy se záporným ohodnocením hran
- implementace pomocí prioritní haldy, která rychle vrací minimum

$l(s) := 0$; $l(v) := \infty$ pro $v \neq s$; $R := \emptyset$ ;
**while** $R \neq V(G)$ **do**
  Nalezni $v \in V(G) \setminus R$ takový, že $l(v) = \min_{t \in V(G) \setminus R} l(t)$;
  $R := R \cup \{v\}$ ;
  **for** $t \in V(G) \setminus R$ pro které $(v, t) \in E(G)$ **do**
    **if** $l(t) > l(v) + c(v, t)$ **then**
      $l(t) := l(v) + c(v, t)$; $p(t) := v$;
    **end**
  **end**
**end**

- nejkratší cesta se skládá z nejkratších cest
- pokud nás zajímá nejkratší cesta pouze do jednoho cílového vrcholu c, lze skončit, jakmile odebereme vrchol z množiny R
- časová náročnost algoritmu je O(n2), respektive s využitím prioritní fronty O(m + nlogn)

### Bellman-Fordův algoritmus

- dokáže detekovat cykly záporné délky
- časová náročnost algoritmu je O(nm)
- umí si poradit se zápornými hranami

**Vstup**: Orientovaný graf $G$ bez záporných cyklů
váhy $c : E(G) \to \mathbb{R}$ a vrchol $s \in V(G)$.
**Výstup**: Vektory $l$ a $p$. Pro každý $v \in V(G)$ je $l(v)$ délkou nejkratší cesty z vrcholu $s$ a $p(v)$ je předposlední vrchol na této cestě. Pokud $v$ není dostupný z $s$, pak $l(v) = \infty$ a $p(v)$ není definován.

```
l(s) := 0; l(v) := ∞ pro v ≠ s;
for i := 1 to n − 1 do
    for  pro každou hranu (v, t) ∈ E(G) do
        if l(t) > l(v) + c(v, t) then
            l(t) := l(v) + c(v, t); p(t) := v;
        end
    end
end
```

### Floydův algoritmus

- časová náročnost algoritmu je $O(n^3)$
- najde nejkratší cestu mezi všemi dvojicemi uzlů
- graf obsahuje cyklus záporné délky právě když existuje i takové, že $l_{ii} < 0$
- modifikací Floydova algoritmu ($l^0_{ii} = \infty$) lze nalézt (nezáporný) cyklus o minimální délce

**Vstup**: Orient. graf $G$ bez záporných cyklů a váhy $c : E(G) \to \mathbb{R}$.
**Výstup**: Čtvercové matice $l$ a $p$. $l_{ij}$ je délkou nejkratší cesty z vrcholu $i$ do vrcholu $j$. $p_{ij}$ je indexem předposledního vrcholu na této cestě (pokud existuje).

```
l_ij := c((i, j)) pro všechny (i, j) ∈ E(G);
l_ij := ∞ pro všechny (i, j) ∉ E(G) kde i ≠ j;
l_ii := 0 pro všechna i;
p_ij := i pro všechny (i, j);
for k := 1 to n do
    for i := 1 to n do
        for j := 1 to n do
            if l_ij > l_ik + l_kj then
                l_ij := l_ik + l_kj; p_ij := p_kj;
            end
        end
    end
end
```

## 2. Úloha obchodního cestujícího

- cíl: Rozhodnout, zda v grafu G existuje Hamiltonovská kružnice (uzavřená cesta procházející každým vrcholem právě jednou), jejíž váha je minimální
- Hamiltonovská kružnice je NP-úplný problém

- **důkaz**, že TSP je silně NP-obtížný problém:

1. Polynomiální redukcí vytvoříme instanci TSP tak, že každému vrcholu grafu G odpovídá 1 vrchol v úplném neorientovaném grafu Kn.
2. váha hrany {i.j} v Kn je: 1 pokud $\{i,j\} \in E(G)$
3. váha hrany {i.j} v Kn je: 2 pokud $\{i,j\} \notin E(G)$

- jednoduše (v polyn. čase) lze ověřit, že G má Hamiltonovskou kružnici právě když optimální řešení TSP má hodnotu n. Neboli TSP je NP-obtížný.
-pro důkaz, že TSP je silně NP-obtížný musíme dokázat, že neexistuje pseudo-polynomiální algoritmus

### Metrický TSP

- pokud v grafu platí trojúhelníková nerovnost, pak metrický TSP je NP-obtížný

### Heuristika Nejbližší soused

Vstup: Instance (Kn, c) Metrického TSP
Výstup: Hamiltonovská kružnice H
- v každém kroku je vybráno nejbližší doposud nenavštívené město
- není to aproximační algoritmus
- časová náročnost algoritmu je $O(n^2)$

### 2-aproximační algoritmus Dvojitá minimální kostra

- Eulerovská cesta projde každou hranou právě jednou
A / OPT( I(Kn, c) ) <= 2
- časová náročnost algoritmu je $O(n^2)$

**Vstup**: Instance $(K_n, c)$ **metrického TSP**.
**Výstup**: Hamiltonovská kružnice $H$.

1. Nalezneme $T$, **minimální kostru** grafu (MST) $K_n$ s váhami $c$;
2. **Zdvojením hran** $T$ dostaneme multigraf v němž **nalezneme Eulerovskou kružnici** $L$;
3. **Transformujeme Eulerovskou kružnici** $L$ **na Hamiltonovskou kružnici** $H$ **v úplném grafu** $K_n$:
   - vytvoříme posloupnost vrcholů ležících na Eulerovské kružnici $L$;
   - v posloupnosti **vynecháme ty vrcholy, které se v ní již vyskytly**;
   - zbytek tvoří Hamiltonovskou kružnici $H$;

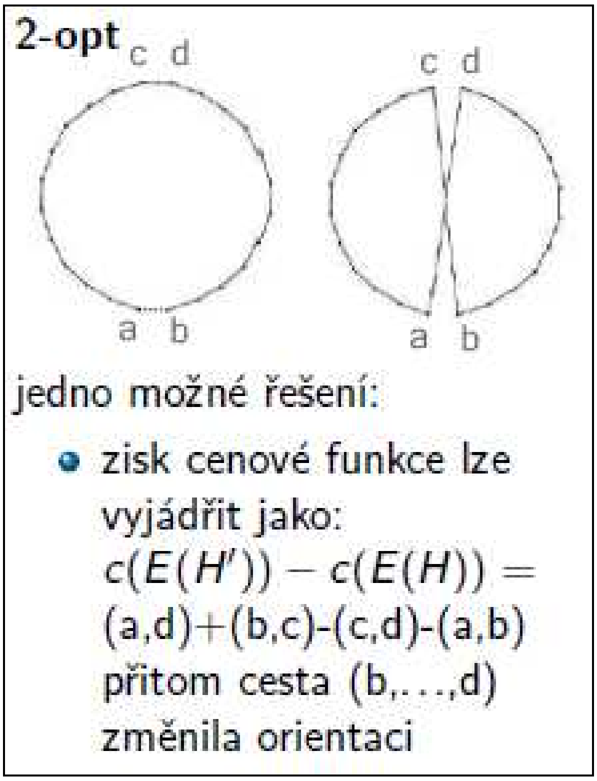
### 3/2-aproximační Christofidesův algoritmus

- časová náročnost algoritmu je $O(n^3)$

### Lokální prohledávání k-OPT

1. použijeme libovolnou Hamiltonovskou kružnici nalezenou např. heuristikou
2. toto řešení zkoušíme vylepšovat pomocí lokálních změn (např. vymazáním dvou hran rozdělíme kružnici na dvě části, které spojíme pomocí jiných hran)

- dovolené jsou jen modifikace, které vylepšují řešení




Obrázek 1 - k OPT

## 3. Problém batohu

- **instance**: n je počet předmětů, $c_i$ jsou ceny předmětů, $w_i$ jsou hmotnosti předmětů a W je nosnost batohu. Cíl: Nalézt podmnožinu S ⊆ {1, . . . , n} takovou, že $\sum_{j \in S} w_j < W$ a $\sum_{j \in S} c_j$ maximální
- jde o NP obtížný problém
- složitost $O(C \cdot n)$, kde **C** je suma všech cen, C ovlivňuje počet sloupců tabulky
- algoritmus pro knapsack je **pseudo-polynomiální**, protože jeho složitost je závislá na C (může být až exponenciálně velké)
- jeden z mála problémů, pro něž existuje aproximační algoritmus s libovolně malou poměrnou odchylkou od optima

- **postup**:
1. začínáme na pozici [0,0], což znamená, že v batohu je nula kilo za nula korun
2. procházíme strom všech možných řešení a ořezáváme ty, které již přerostly stanovenou nosnost batohu
3. pohyb o jednu pozici dolů znamená, že předmět do batohu nedáváme
4. pohyb o jednu pozici dolů a o $w_i$ pozic doprava znamená, že předmět i o váze $w_i$ do batohu přidáme
5. nakonec se projde poslední řádek a najde se poslední největší cena
6. průchodem stromu doleva vzhůru lze zjistit seznam předmětů, které se do batohu vešly

| $x_i$ \ Σ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 (v/w) | 0 | | | | | | | | | | | | |
| 1 (3/3) | 0 | | | 3 | | | | | | | | | |
| 2 (5/6) | 0 | | | 3 | | 6 | | | ~~9~~ | | | | |
| 3 (3/4) | 0 | | | 3 | | 6 | 7 | | ~~10~~ | | | | |
| 4 (1/1) | 0 | 1 | | 3 | 4 | 6 | 7 | 8 | | | | | |

### Fractional knapsack problem

= instance úlohy taková, že předměty lze dělit na části, čili $\sum_{j \in S} x_j \cdot c_j$ je minimální,

$0 <= x_j <= 1$

- **řešení**: seřadit předměty sestupně podle relativní ceny $\frac{c_i}{w_i}$, zaplnit batoh, až už se nic nevejde a doplnit zbytek částí nějakého předmětu

### 2-aproximační algoritmus pro Knapsack

- předměty jsou seřazené sestupně podle relativní ceny $\frac{c_i}{w_i}$

h = min{j∈{1, . . . , n} : $\sum_{i=1}^{j} w_i > w$ }

- výběr lepšího ze dvou řešení {1,..., h-1} a {h} je 2-aproximační algoritmus
- časová náročnost je O(n)