Diskrétní matematika

Petr Kovář
petr.kovar@vsb.cz

Vysoká škola báňská – Technická univerzita Ostrava

zimní semestr 2022/2023
DiM 470-2301/01, 470-2301/03*, 470-2301/05

O tomto souboru

Tento soubor je zamýšlen především jako pomůcka pro přednášejícího. Řadu důležitých informací v souboru nenajdete, protože přednášející je říká, ukazuje, případně maluje na tabuli. Přednášky jsou na webu k dispozici, aby studenti mohli snadno dohledat probíraná témata z přednášek, které zameškali.

Pro samostatné studium doporučuji skripta:

- M. Kubesa: Základy diskrétní matematiky, výukový text
- P. Kovář: Úvod do teorie grafů, výukový text

Pro přípravu ke zkoušce a písemkám doporučuji cvičebnici:

- P. Kovář: Cvičení z diskrétní matematiky, sbírka příkladů

Vše na http://homel.vsb.cz/~kov16/predmety_dm.php

Přehled přednášky

Kapitola Vzdálenost a metrika

- motivace
- vzdálenost v grafu
- výpočet metriky
- vzdálenost v ohodnocených grafech
- hledání nejkratší cesty

Motivace

V mnoha praktických aplikacích má smysl v grafu „měřit“ vzdálenosti.

V grafu, který interpretuje silniční síť, je velmi přirozené ptát se

„Jak daleko je z vrcholu (místa) u do vrcholu (místa) v?“

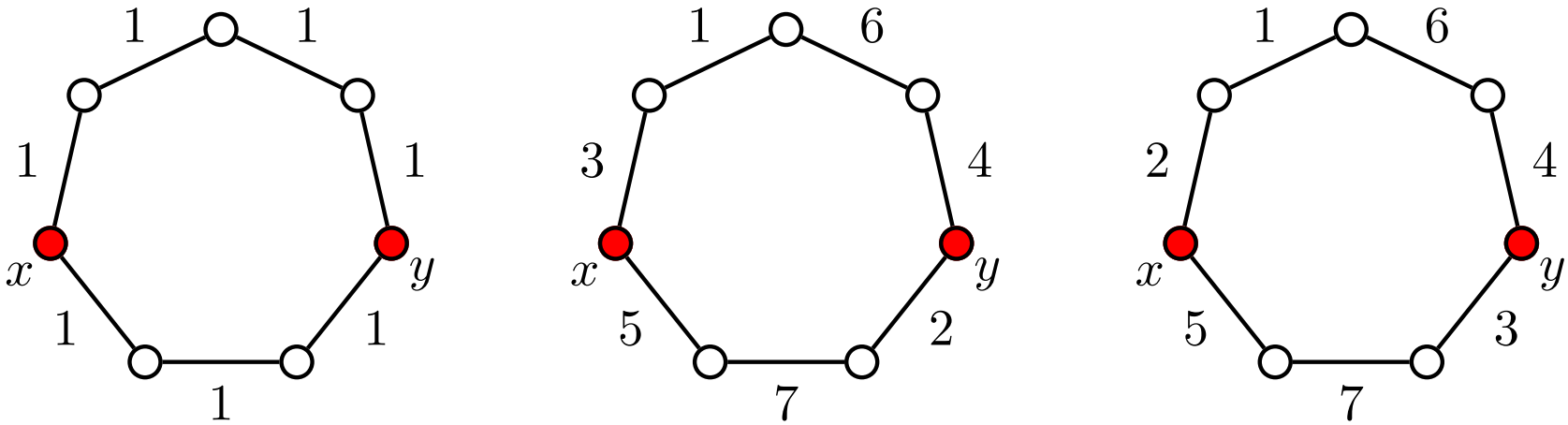
nebo

„Jak dlouho trvá cesta z vrcholu u do vrcholu v?“

Vzdálenost neurčuje pouhý *počet hran* (počet různých silnic) ale důležitá je jejich *délka*. Uvědomte si, že informace o délce v grafu (zatím) obsažena není.

Zavedeme obecně pojem ohodnocení hran. Význam ohodnocení je rozmanitý: délka, tloušťka, kapacita, barva, ...

Při ohodnocování obvykle vystačíme s přirozenými čísly (vhodná volba měřítka).



Různé vzdálenosti mezi vrcholy u a v v grafu C_7.

V levém grafu

je vzdálenost mezi vrcholy x a y rovna $3 =$ počet hran nejkratší cesty (sledu).

V prostředím grafu

je vzdálenost mezi vrcholy x a y rovna $14 = 3 + 1 + 6 + 4 = 5 + 7 + 2$.

V pravém grafu

je vzdálenost mezi vrcholy x a y rovna $13 = 2 + 1 + 6 + 4$.

Vzdálenost v grafu

Pro neohodnocené grafy můžeme předpokládat, že délka každé hrany je 1.

Délkou sledu v (neohodnoceném) grafu rozumíme počet hran v posloupnosti vrcholů a hran

$$v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n,$$

kde každá hrana e_i má koncové vrcholy v_{i-1} a v_i.

Definice

Vzdálenost $\mathrm{dist}_G(u, v)$ mezi vrcholy u a v grafu G je dána délkou nejkratšího sledu mezi u a v v G. Pokud sled mezi u, v neexistuje, položíme vzdálenost $\mathrm{dist}_G(u, v) = \infty$.

Všimněte si, že

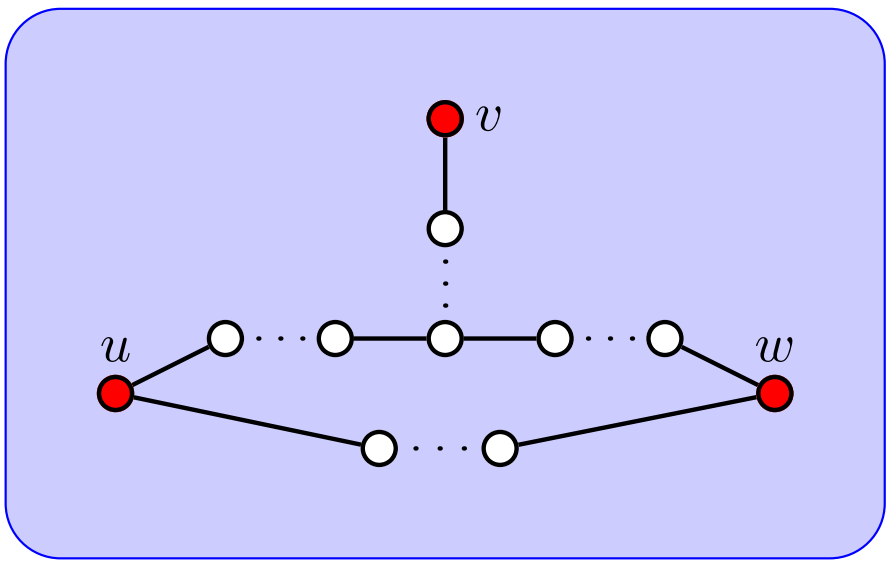
- nejkratší sled (obsahuje nejméně hran) je vždy cestou
- v neorientovaných grafech platí $\mathrm{dist}_G(u, v) = \mathrm{dist}_G(v, u)$
- $\mathrm{dist}_G(u, u) = 0$
- je-li $\mathrm{dist}_G(u, v) = 1$, tak hrana $uv \in E(G)$

Lemma

Vzdálenost mezi vrcholy v grafu G splňuje tzv. *trojúhelníkovou nerovnost*:

$$\forall u, v, w \in V(G)\colon\ \mathrm{dist}_G(u, w) \leq \mathrm{dist}_G(u, v) + \mathrm{dist}_G(v, w).$$

Důkaz Nerovnost ihned plyne ze zřejmého pozorování, že na sled délky $\mathrm{dist}_G(u, v)$ mezi vrcholy u, v lze navázat sled délky $\mathrm{dist}_G(v, w)$ mezi v, w, čímž dostaneme sled délky $\mathrm{dist}_G(u, v) + \mathrm{dist}_G(v, w)$ mezi u, w. Proto nemůže být $\mathrm{dist}_G(u, w) > \mathrm{dist}_G(u, v) + \mathrm{dist}_G(v, w)$. Může však existovat kratší sled mezi u, w, proto $\mathrm{dist}_G(u, w) \leq \mathrm{dist}_G(u, v) + \mathrm{dist}_G(v, w)$. □



Sledy mezi u a v, mezi v a w; kratší sled mezi u a w.

Určení vzdáleností (výpočet metriky)

Při určování vzdálenosti nemá smysl zkoumat všechny možné cesty.

Příklad

Určete počet všech cest mezi dvěma vrcholy u, v v kompletním grafu K_n.

1. Jestliže $u = v$, tak existuje jediná triviální cesta z u do v.
2. Pro $u \neq v$ existuje celkem $V(n-2,k) = \frac{(n-2)!}{(n-2-k)!}$ různých takových cest, kde $0 \leq k \leq n-2$ je počet interních vrcholů cesty.

Celkem je počet hledaných cest z u do v roven $\sum_{k=0}^{n-2} \frac{(n-2)!}{(n-2-k)!}$.

Řádově $O((n-2)!)$ různých cest ... příliš mnoho.

Pro $n = 10$ je v grafu K_{10} 109 601 různých (u,v)-cest.
Pro $n = 15$ je v grafu K_{15} už 16 926 797 486 různých (u,v)-cest.
A pro $n = 20$ je v grafu K_{20} už $1.74 \cdot 10^{16}$ různých (u,v)-cest.
V Ostravě je více než 670 zastávek...

Existuje jednoduchá modifikace algoritmu pro prohledávání grafu s postupem do šířky (úschovna je fronta F).
Určíme vzdálenosti z pevně zvoleného vrcholu do všech ostatních vrcholů.

Každému nově objevenému vrcholu w přiřadíme vzdálenost o jedničku větší, než vzdálenost právě zpracovávaného vrcholu v.
Vzdálenosti uložíme do jednorozměrné pole `vzdal[]`.

Algoritmus: Vzdálenosti z daného vrcholu

```
// na vstupu je graf G
vstup < graf G;
stav(všechny vrcholy G) = iniciační;
fronta F = libovolný vrchol u grafu G;
stav(u) = nalezený;
vzdal(u) = 0;                              // vzdálenost u
```

Algoritmus: Vzdálenosti z daného vrcholu (pokračování)

```
// zpracování vybrané komponenty G
while (F je neprázdná) {
   vyber vrchol v a odeber jej z fronty F: F = F - v;
   for (hrany e vycházející z v) {    // pro všechny hrany
      w = druhý vrchol hrany e = vw;  // známe sousedy?
      if (stav(w) == iniciační) {
         stav(w) = nalezený;
         přidej vrchol w do fronty: F = F + w;
         vzdal[w] = vzdal[v]+1;       // vzdálenost w
      }
   }
   stav(v) = zpracovaný;
}
// vrcholy z dalších komponent jsou nedosažitelné!
while (v grafu G je nezpracovaný vrchol w) {
   vzdal[w] = MAX_INT;                // nekonečno
   stav(w) = zpracovaný;
}
```

Všimněte si:

- Počet kroků závisí na počtu vrcholů a hran daného grafu.
  Složitost algoritmu je $O(n + m)$, kde n udává počet vrcholů a m je počet hran daného grafu.
- Pokud při nalezení každého vrcholu uložíme informaci o *předchozím* vrcholu na nejkratší cestě, budeme ji umět zrekonstruovat.
  Za řádek `vzdal[w] = vzdal[v]+1;`
  přidáme řádek `pre[w] = v;`
  - poslední vrchol takové cesty je vrchol w,
  - předposlední vrchol cesty je vrchol $pre[w]$,
  - předpředposlední vrchol cesty je vrchol $pre[pre[w]]$,
  - ...
  - první (tj. výchozí) vrchol cesty je vrchol $pre[\ldots pre[pre[w]]] = u$.

Předpokládali jsme, že bližší vrcholy budou zpracovány dříve než vzdálenější vrcholy.
Toto pozorování dokážeme a použijeme v důkazu správnosti algoritmu.

Lemma

Nechť u, v, w jsou takové vrcholy souvislého grafu G, že $\mathrm{dist}_G(u, v) < \mathrm{dist}_G(u, w)$. Pak při procházení grafu G postupem *do šířky* z vrcholu u je vrchol v nalezen dříve než vrchol w.

Důkaz Postupujeme indukcí vzhledem ke vzdálenosti $\mathrm{dist}_G(u, v)$:
Základ indukce: Pro $\mathrm{dist}_G(u, v) = 0$, tj. $u = v$ je tvrzení zřejmé: vrchol u jako počátek prohledávání byl nalezen první.
Indukční krok: Mějme $\mathrm{dist}_G(u, v) = d > 0$ a označme v' předposlední vrchol na nejkratší cestě z vrcholu u do vrcholu v; je $\mathrm{dist}_G(u, v') = d - 1$.
Označme w' předposlední vrchol na nejkratší cestě z vrcholu u do vrcholu w; je tedy $\mathrm{dist}_G(u, v') < \mathrm{dist}_G(u, w')$.
Podle indukčního předpokladu bude vrchol v' bude při prohledávání do šířky nalezen dříve než vrchol w'. To znamená, že v' se dostal do fronty dříve než w', a proto sousedé vrcholu v' (mezi nimi i vrchol v) budou při prohledávání nalezeni dříve než sousedé vrcholu w'. □

Důsledek

Základní algoritmus procházení grafu do šířky lze použít pro výpočet vzdáleností z vrcholu u do všech ostatních vrcholů.

Důkaz ve skriptech.

Otázky

Proč nelze použít stejný algoritmus s postupem do hloubky?
Která část algoritmu by nefungovala správně?

Sestavení metriky

Metrikou grafu rozumíme soubor všech vzdáleností mezi každou dvojicí vrcholů v grafu. Předpokládáme, že metrika splňuje „přirozené“ vlastnosti.

Formálně: množina vrcholů (souvislého grafu) spolu s funkcí vzdálenosti tvoří metrický prostor. V nesouvislém grafu máme navíc ∞.

Definice

Metrika ρ na množině A je takové zobrazení $\rho : A \times A \to \mathbb{R}$, že $\forall x, y \in A$ platí

1. $\rho(x, y) \geq 0$ přičemž $\rho(x, y) = 0$ jen pro $x = y$,
2. $\rho(x, y) = \rho(y, x)$,
3. $\rho(x, y) + \rho(y, z) \geq \rho(x, z)$.

Neformálně: Metrika grafu G je matice (dvourozměrné pole) `d[][]`, kde prvek `d[i][j]` udává vzdálenost mezi vrcholy i a j (vrcholy $0, 1, \ldots, |V(G)| - 1$).

Pro sestavení metriky bychom mohli (opakovaně pro každý výchozí vrchol) použít algoritmus pro určení vzdálenosti z daného vrcholu.
Existuje však jednodušší (ne rychlejší) algoritmus:

Metoda: Sestavení metriky skládáním cest

Označíme vrcholy grafu $0, 1, 2, \ldots, N-1$.

- Položíme `d[i][j]` rovno 1 (případně délce hrany ij), nebo ∞ pokud hrana ij v grafu není.
- Po každé iteraci $t \geq 0$ bude `d[i][j]` udávat délku nejkratší cesty mezi i, j, která jde pouze přes vrcholy z množiny $\{0, 1, 2, \ldots, t\}$.
- V průběhu každé iterace t upravíme vzdálenost pro každou dvojici vrcholů i, j. Rozlišíme dvě možnosti:
  1) buď najdeme kratší cestu přes nově povolený vrchol t; nahradíme hodnotu `d[i][j]` menší vzdáleností `d[i][t] + d[t][j]`,
  2) nebo přidání vrcholu t do seznamu „povolených“ vrcholů nepomůže najít cestu kratší než byla `d[i][j]` v předchozím kroku; potom hodnotu `d[i][j]` ponecháme beze změny.

Floydův algoritmus: Sestavení metriky skládáním cest

```
vstup: matice sousednosti G[][] grafu na N vrcholech, kde
       G[i][j]=1 pro hranu mezi i,j a
       G[i][j]=0 jinak;

// inicializace (hodnotu MAX_INT/2 považujeme za "nekonečno")
for (i=0; i<N; i++)
   for (j=0; j<N; j++)
      d[i][j] = (i==j ? 0 : (G[i][j] ? 1 : MAX_INT/2));

// cyklus pro všechny vrcholy t, index z intervalu [0,N-1]
for (t=0; t<N; t++)
   // probereme všechny dvojice vrcholů
   for (i=0; i<N; i++)
      for (j=0; j<N; j++)
         // vede přes vrchol t kratší cesta?
         d[i][j] = min(d[i][j], d[i][t]+d[t][j]);
```

V počítači implementujeme ∞ třeba jako `MAX_INT/2`.

Výhody:

- jednoduchá implementace
- najde vzdálenost mezi každými dvěma vrcholy

Nevýhody:

- i když nás zajímá jen vzdálenost dvou vrcholů, *musí* najít vzdálenost mezi každými dvěma vrcholy
- složitost řádově $O(n^3)$, kde n je počet vrcholů daného grafu
- nenajdeme nejkratší cesty, jen vzdálenosti!
  (z výsledku nelze zrekonstruovat)

Vzdálenost v ohodnocených grafech

Přiřadíme hranám čísla: délka, tloušťka, kapacita, barva, ...

Definice

Ohodnocení grafu G je funkce $w : E(G) \to \mathbb{R}$, která každé hraně přiřadí reálné číslo $w(e)$, kterému říkáme váha hrany. Ohodnocený graf je graf G spolu s ohodnocením hran reálnými čísly.
Kladně ohodnocený (vážený) graf G má takové ohodnocení w, že pro každou hranu $e \in E(G)$ je její váha $w(e)$ kladná.

Váhy hran – anglicky „weight“.

V praktických aplikacích:

- ohodnocení je většinou nezáporné,
- obor hodnot jsou při vhodné volbě jednotek (měřítka) přirozená čísla.

Vážený graf je speciálním případem ohodnoceného grafu.

Nyní zavedeme vzdálenost v ohodnocených grafech.

Definice

Mějme ohodnocený graf G s ohodnocením w.
Délkou ohodnoceného sledu $S = v_0, e_1, v_1, e_2, v_2, \ldots, e_n, v_n$ v G rozumíme součet vah všech jeho hran

$$d_G^w(S) = w(e_1) + w(e_2) + \cdots + w(e_n)$$

(váha každé hrany se počítá tolikrát, kolikrát se ve sledu S objevuje).
Vzdáleností mezi dvěma vrcholy u, v ve váženém (kladně ohodnoceném) grafu (G, w) rozumíme číslo

$$\mathrm{dist}_G^w(u, v) = \min\{d_G^w(S), \text{kde } S \text{ je } \textbf{cesta} \text{ s koncovými vrcholy } u,\ v\}.$$

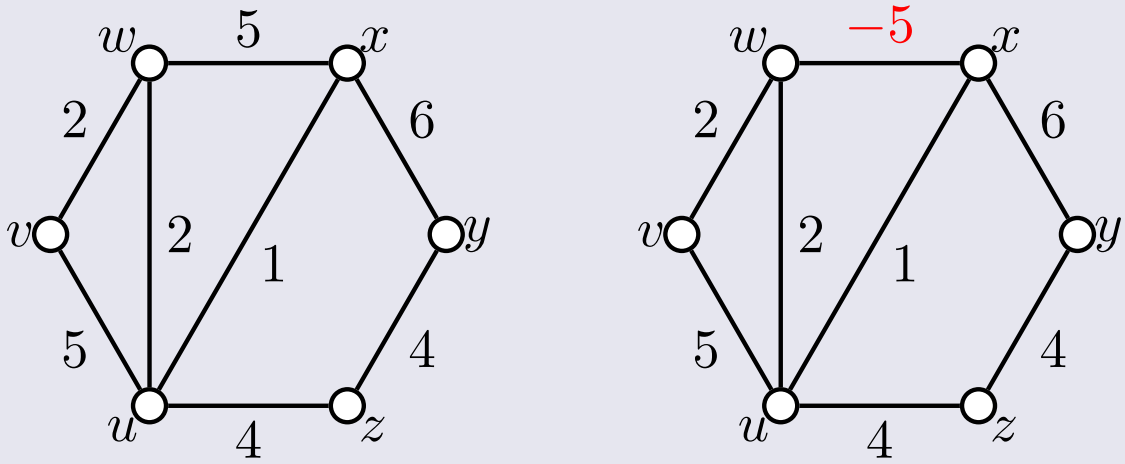
Jestliže vrcholy u a v jsou nedosažitelné, klademe $\mathrm{dist}_G^w(u, v) = \infty$.

Lemma

Vážená vzdálenost v (kladně) váženém grafu G splňuje *trojúhelníkovou nerovnost* $\quad \forall u, v, w \in V(G)\colon\ \mathrm{dist}_G^w(u, w) \leq \mathrm{dist}_G^w(u, v) + \mathrm{dist}_G^w(v, w)$.

Proč uvažujeme nezáporná ohodnocení?

Příklad



Dvě různá ohodnocení grafu G.

Otázky

- Jaká je v levém obrázku vzdálenost vrcholů v a y? 13? 12? 11? 10?
- V grafu nedovolíme záporné délky, neboť by nemusela existovat minimální vzdálenost.
- Jaká je v pravém obrázku vzdálenost vrcholů v a y? 1?, 0?, -1?, 10? $-n$?

Nejkratší cesta v ohodnoceném grafu

Pro hledání nejkratší (ohodnocené) cesty mezi dvěma vrcholy váženého (kladně ohodnoceného) grafu se používá Dijkstrův algoritmus.

- je složitější než Algoritmus s využitím skládání cest
- je *výrazně rychlejší*; najde vzdálenosti z jednoho vrcholu do ostatních (místo mezi všemi dvojicemi vrcholů)

Právě Dijkstrův algoritmus (resp. jeho modifikace) se užívá při vyhledávání vlakových a autobusových spojení.

Dijkstrův algoritmus

- Je variantou procházení grafu do šířky – pro každý nalezený vrchol v máme proměnnou `vzdal[]` udávající *vzdálenost od výchozího vrcholu* u (délku nejkratšího sledu u, v).
- Z úschovny nalezených vrcholů vždy vybíráme ten vrchol v *s nejmenší vzdáleností* od u (kratší cesta do v neexistuje).
- Na konci zpracování dostaneme pole, která udává *vzdálenost* z počátečního vrcholu do všech ostatních vrcholů.

Dijkstrův algoritmus (iniciace)

Najde nejkratší cestu mezi vrcholy u a v (kladně) váženého grafu G, daného seznamem sousedů vrcholů.

```
vstup: graf na N vrcholech daný seznamem sousedů sous[][]
       a w[][], kde sous[i][0], ..., sous[i][st[i]-1]} jsou
       sousedé vrcholu i stupně st[i] a hrana z i do
       sous[i][k] má délku w[i][k] > 0;
vstup: u, v (hledáme cestu z u do v);
// stav[i] udává stav vrcholu i:
//    0 ... iniciační
//    1 ... zpracovaný
// vzdal[i] udává zatím nalezenou vzdálenost do vrcholu i
// pre[i] udává, ze kterého vrcholu jsme do i přišli

// inicializace
for (i=0; i<=N; i++)   // MAX_INT dáme navíc i do vzdal[N]!
   { vzdal[i] = MAX_INT; stav[i] = iniciační; }
vzdal[u] = 0;
```

Dijkstrův algoritmus (pokračování)

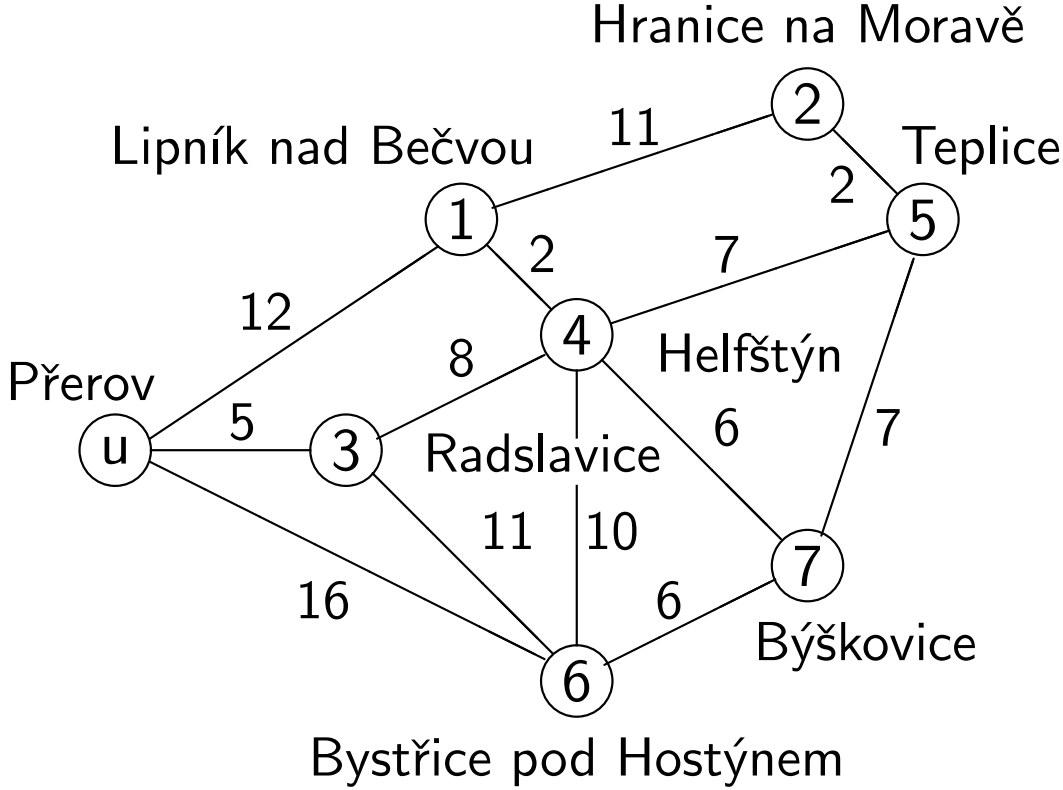
```
while (stav[v] == iniciační) {
   for (i=0, j=N; i<N; i++)      // vzdal[N] = MAX_INT
      if (stav[i] == iniciační && vzdal[i] < vzdal[j])
         j = i;
   // našli jsme nejbližší nezpracovaný vrchol j
   // zpracujeme jej
   if (vzdal[j] == MAX_INT) return NENI_CESTA;
   stav[j] = zpracovaný;
   for (k=0; k<st[j]; k++)
      if (vzdal[j]+w[j][k] < vzdal[sous[j][k]]) {
         vzdal[sous[j][k]] = vzdal[j]+w[j][k];
         pre[sous[j][k]] = j;
      }
      // pole pre[] obsahuje informaci o tom,
      // odkud jsme se do každého vrcholu dostali
}
výstup: Cesta délky vzdal[v], uložená pozpátku v poli pre[];
```

Poznámky k Dijkstrovu algoritmu

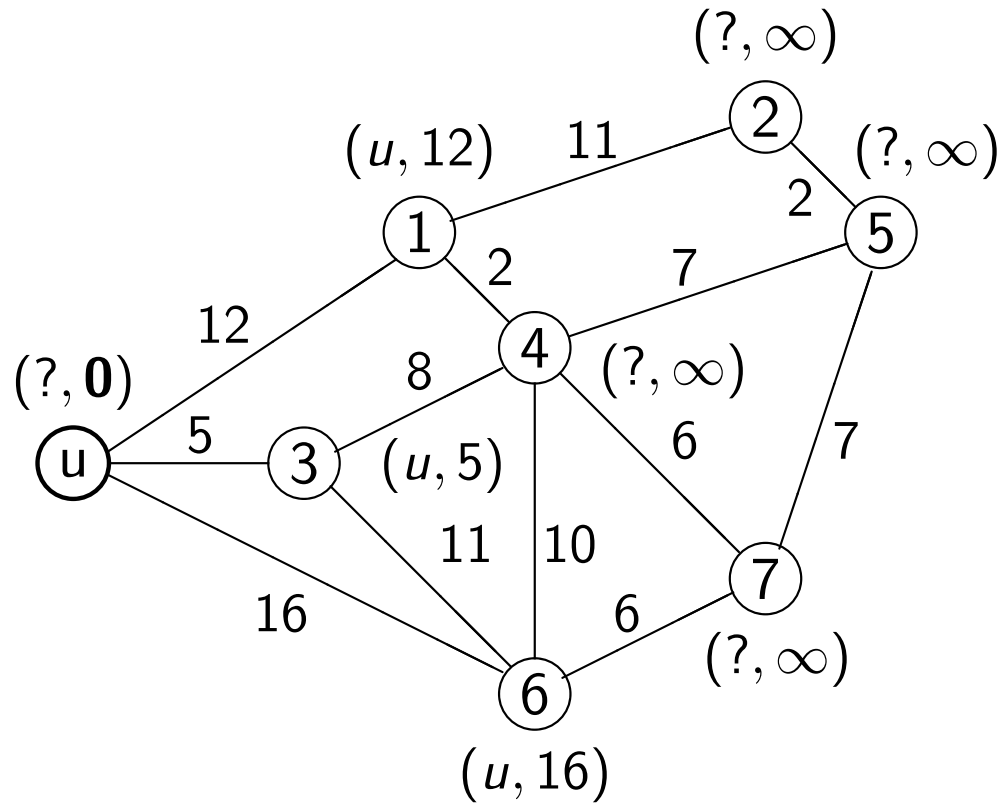
- Poběží-li cyklus nikoli do splnění podmínky `stav[v] == iniciační`, ale pro všechny vrcholy, najde algoritmus vzdálenosti z počátečního *u* vrcholu do všech ostatních vrcholů a uloží je do pole `vzdal[i]`.
- Celkový počet kroků potřebný v Dijkstrově Algoritmu k nalezení nejkratší cesty z `u` do `v` je zhruba $O(N^2)$, kde N je počet vrcholů grafu.
- Při vhodné implementaci úschovny nezpracovaných vrcholů (např. haldou s nalezenou vzdáleností jako klíčem) lze dosáhnout na řídkých grafech i rychlejšího běhu algoritmu $O(m)$ – čas zhruba úměrný počtu hran grafu.
- Algoritmus najde nejkratší cesty i v *orientovaném grafu*.
- Můžeme jej modifikovat i pro hledání *nejširší cesty*.

Následuje příklad...

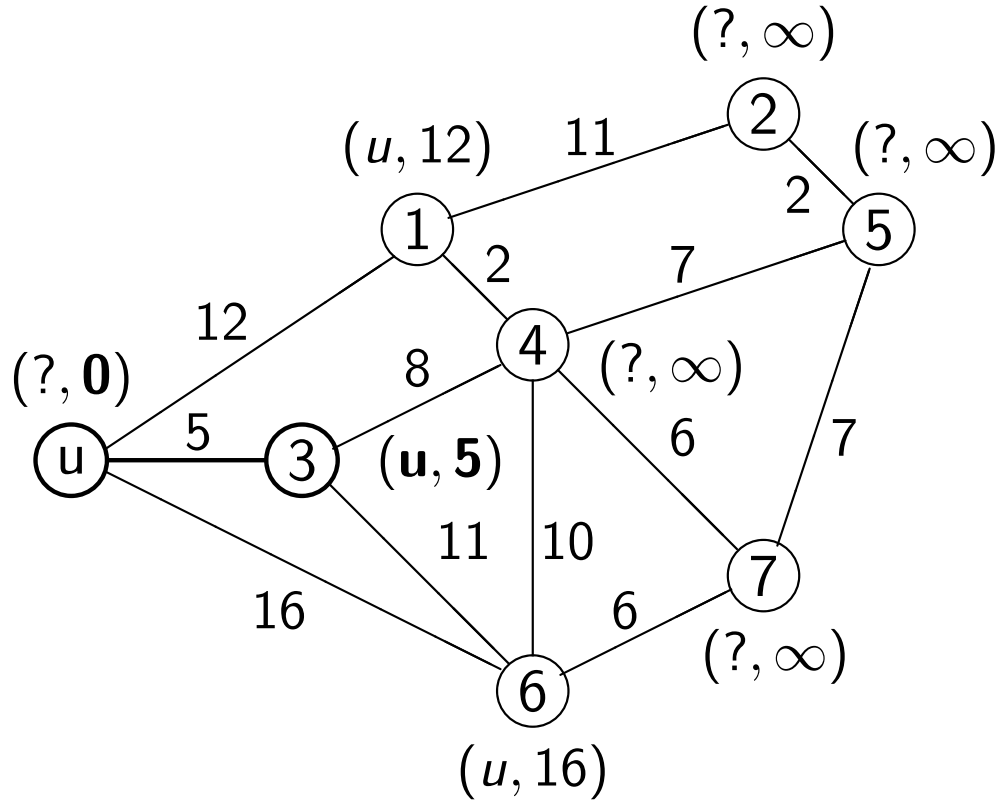
Vezměme například fragment mapy východně od Přerova. Budeme chtít najít nejkratší cesty z Přerova do všech ostatních míst.



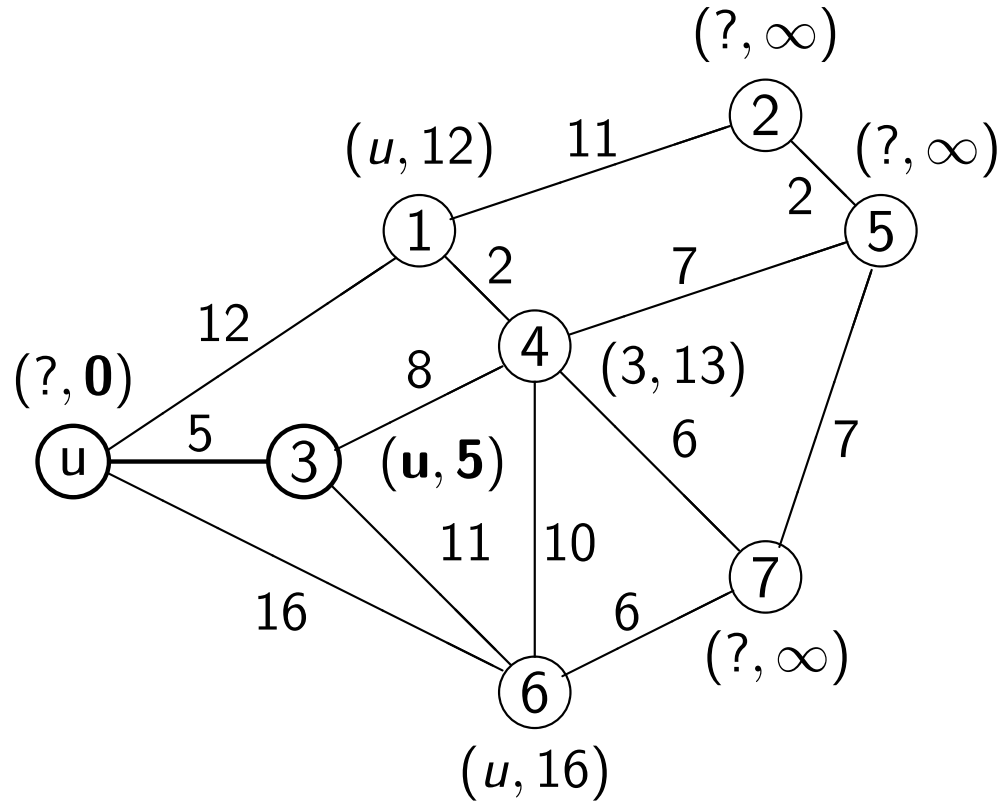
Toto je skica uvedené mapky. Jsou na ní vyznačeny přibližné vzdálenosti jednotlivých míst v kilometrech. Tato skica je zároveň i *grafovou interpretací* dané úlohy. Vrcholy reprezentují města a silnice mezi městy jsou hrany mezi odpovídajícími vrcholy. Vrchol i budeme značit $(pre[i], vzdal[i])$.



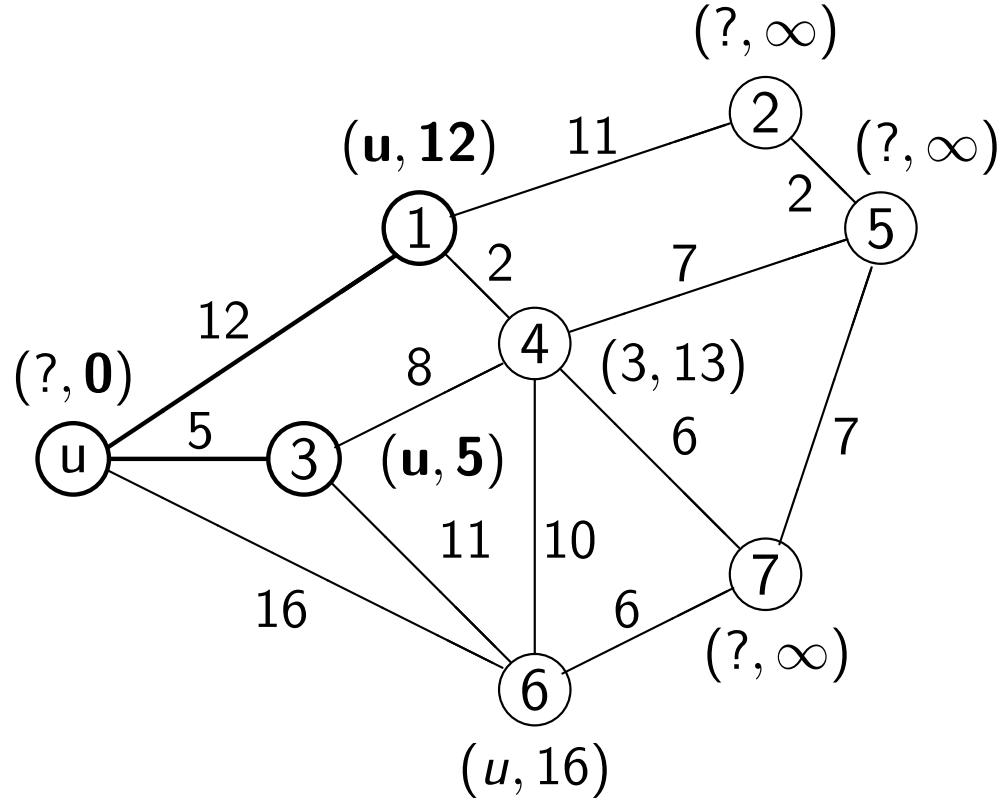
Můžeme přistoupit k inicializaci Dijkstrova algoritmu. Každý vrchol dostane stav 0 (iniciační). Pouze výchozí vrchol u dostane vzdálenost 0, tj. značku $(0, 0)$. Ostatní vrcholy dostanou značku $(?, \infty)$.
V prvním kroku všechny vrcholy j, které jsou sousední s vrcholem u, dostanou značku $(s, w[s][j])$.



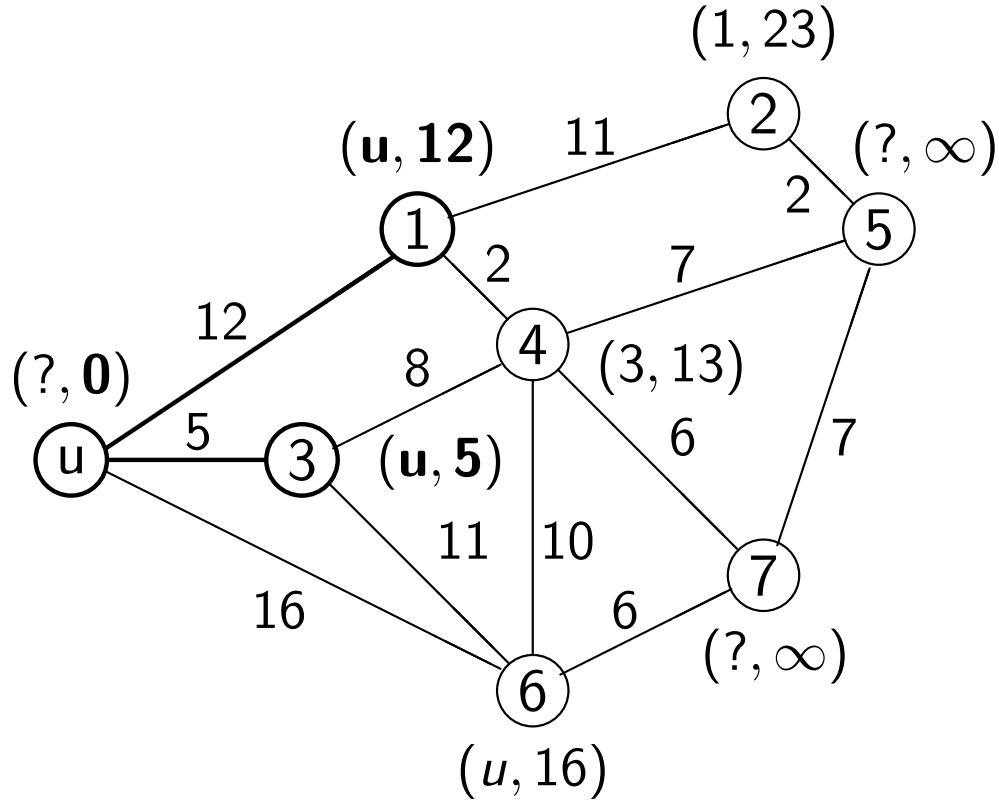
Dále najdeme takový vrchol j, který má od vrcholu u nejmenší vzdálenost. Tímto vrcholem je vrchol 3.



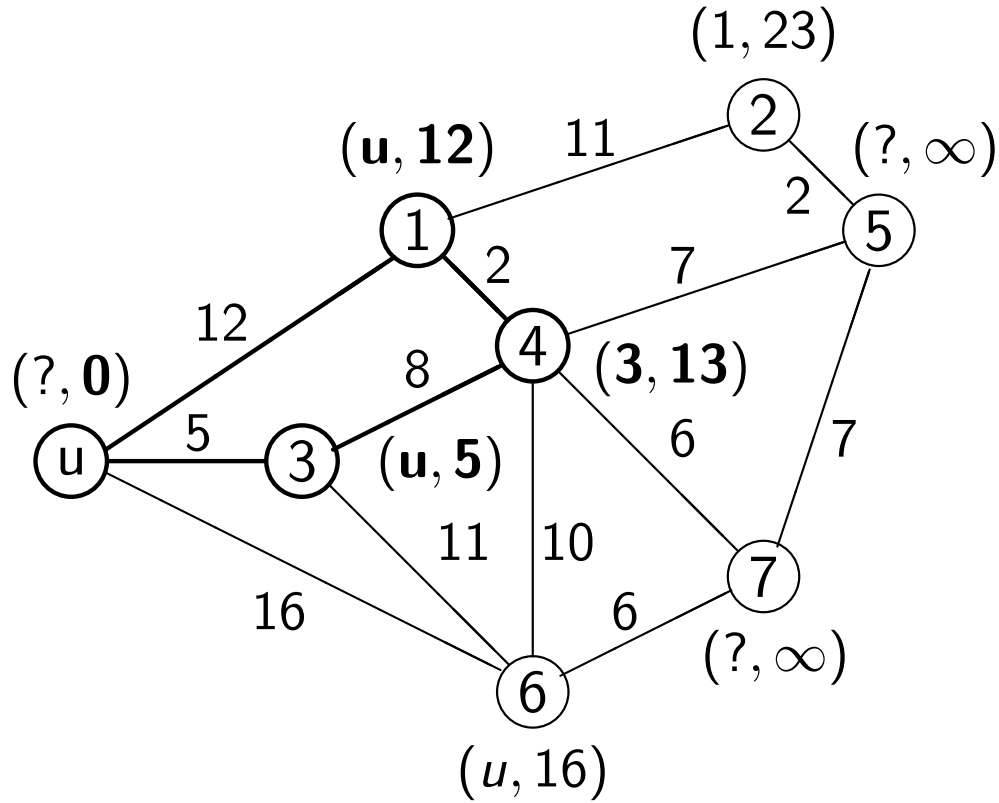
V dalším kroku budeme měnit značky všech sousedů vrcholu 3 (nejbližší nezpracovaný vrchol).
Změníme značku vrcholu 4. Nová značka vrcholu 4 bude $(3, 13)$.
Značku vrcholu 6 měnit nebudeme. Vrchol u je rovněž sousední s vrcholem 3, protože je zpracovaný, jeho značku už měnit nebudeme.



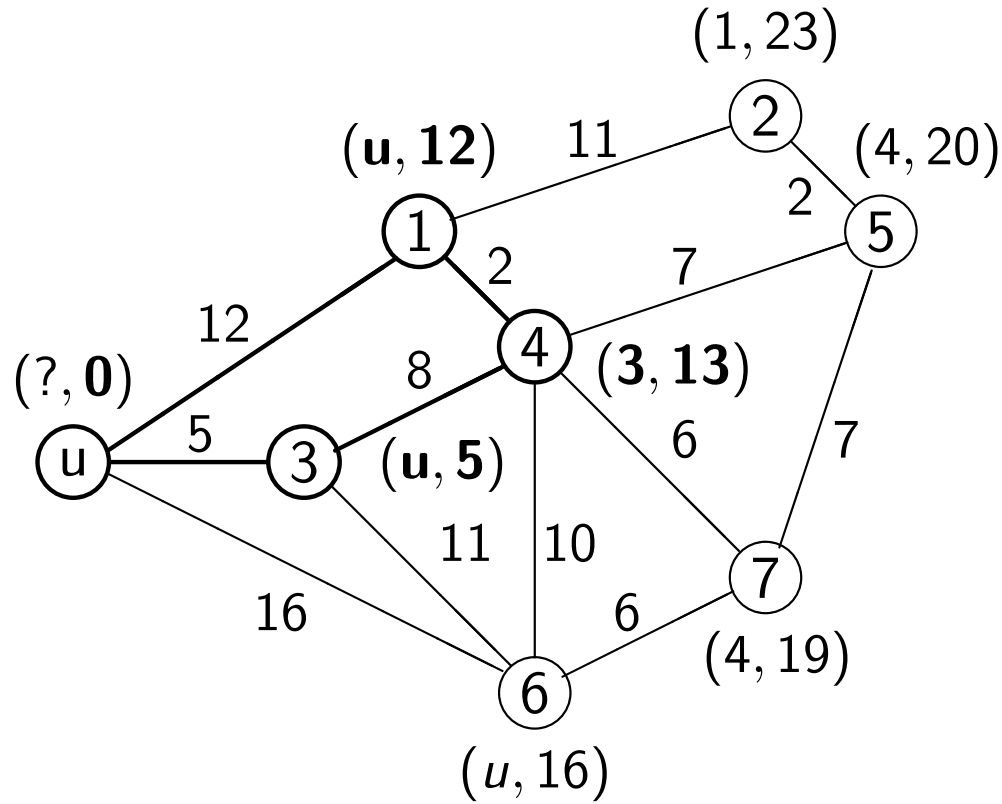
Budeme hledat takový vrchol j, který má nejmenší vzdálenost od u. Takovým vrcholem je vrchol 1 ($vzdal[1] = 12$).



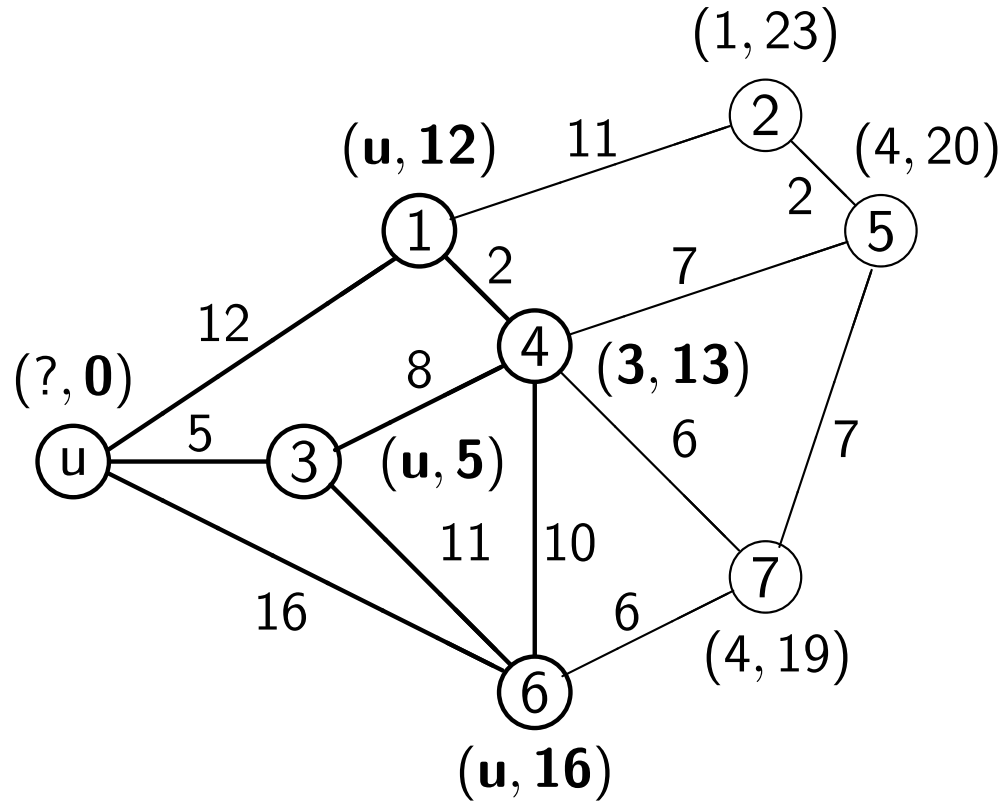
Vrchol 2 dostane proto značku $(1, 23)$, protože $\infty > \mathit{vzdal}[1] + w[1][2]$. Avšak značku vrcholu 4 měnit nebudeme.



Vrchol 4 je nejblíže vrcholu u a proto jej zpracujeme.



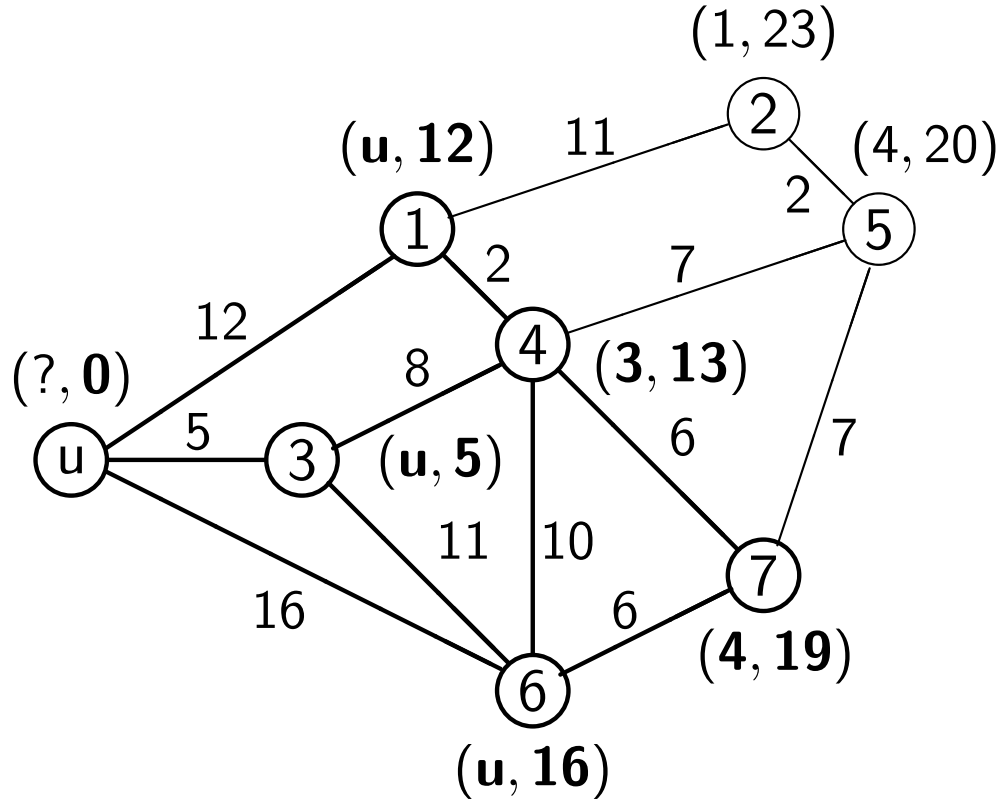
Podíváme se na sousedy vrcholu 4. Jedná se o vrcholy 5, 6 a 7. Protože $vzdal[5] > vzdal[4] + w[4][5]$ $(\infty > 13 + 7)$, bude nová značka vrcholu 5 $(4, 20)$. Značku vrcholu 6 měnit nebudeme. Avšak vrchol 7 dostane novou značku $(4, 13 + 6)$.



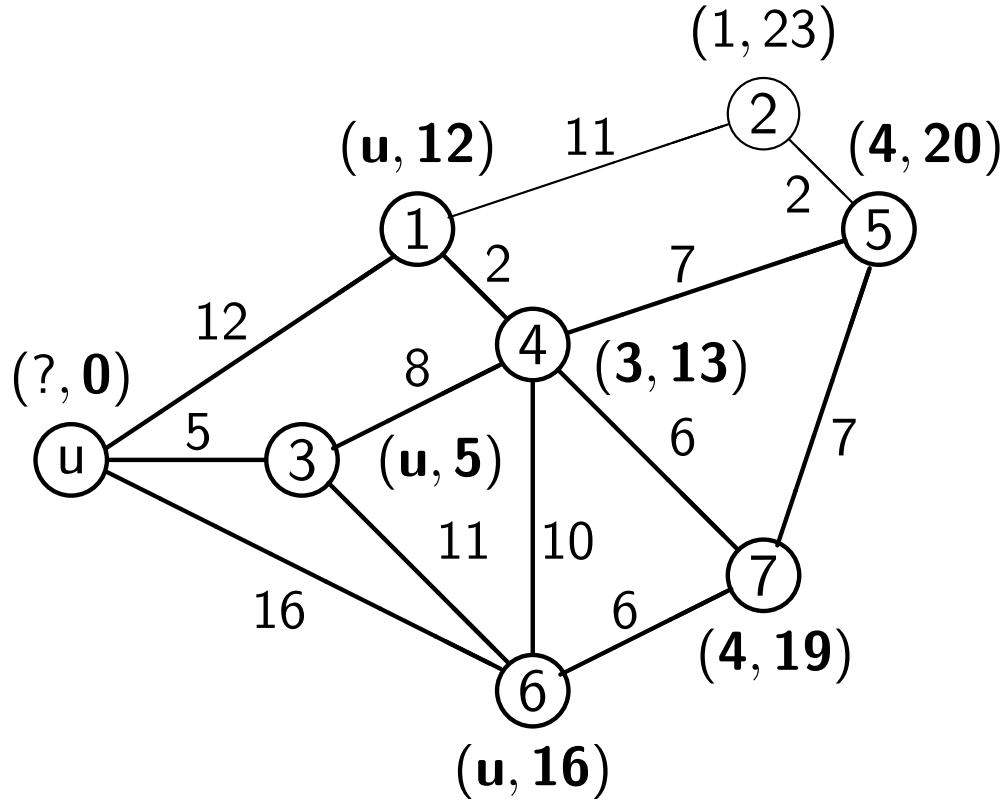
Nejblíže vrcholu u je vrchol 6. Ostatní vrcholy 2, 5 a 7 mají větší hodnotu $vzdal[i]$.

Nebudeme měnit značku žádného vrcholu, není co vylepšovat!

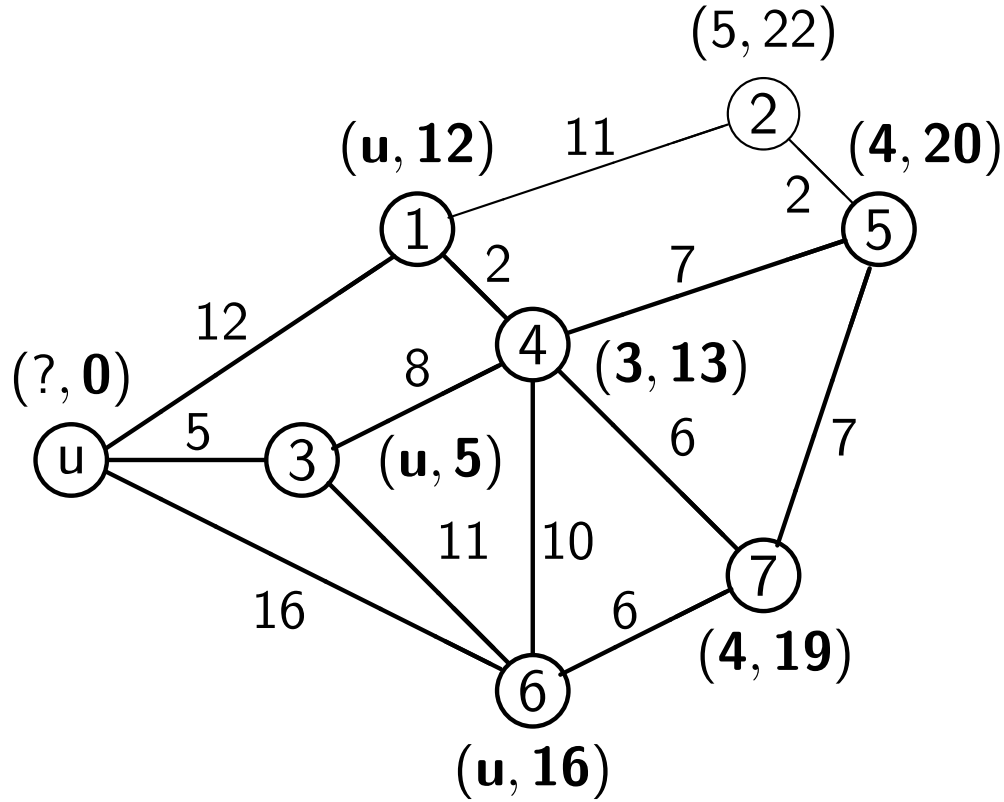
Poznámka: Pokud by existovalo více vrcholů se stejnou vzdáleností od u vybereme libovolný z nich.



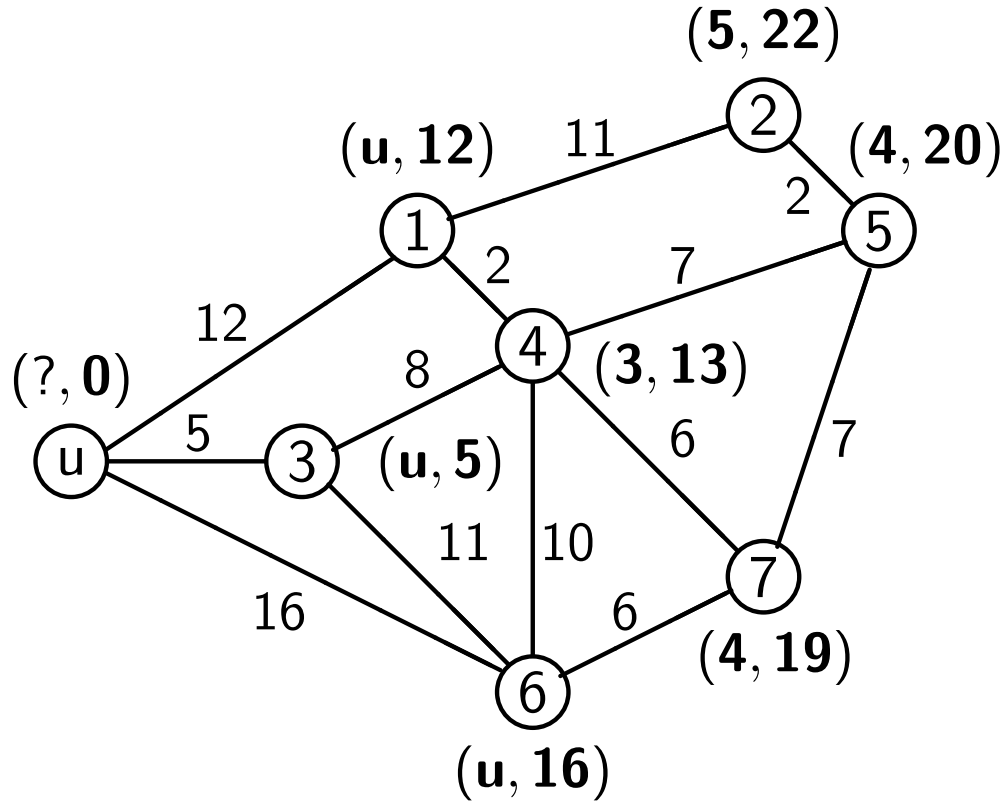
Nejblíže vrcholu u je vrchol 7 ($vzdal[7] = 19$). Opět nebudeme měnit značku žádného vrcholu.



Nejblíže vrcholu u je vrchol 5, protože $vzdal[5] < vzdal[2]$ $(20 < 23)$.
žádnou značku neměníme.



Posledním vrcholem, který není ve stavu 1 (zpracovaný), je vrchol 2. Skutečně platí $vzdal[2] > vzdal[5] + w[5][2]$ $(23 > 22)$ a nová značka vrcholu 2 bude $(5, 22)$.



Nejbližším (jediným) vrcholem k vrcholu u je vrchol 2. Změníme jeho stav a algoritmus končí.
Získali jsme vzdálenosti všech vrcholů od výchozího vrcholu u.

Důkaz správnosti Dijkstrova Algoritmu

Věta

Mějme (kladně) vážený graf G a v něm dva pevně zvolené vrcholy u a v. Dijkstrův Algoritmus najde nejkratší cestu z vrcholu u do vrcholu v.

Důkaz

Označme S množinu vrcholů, které jsou již zpracované.
Klíčové pozorování je, že v každé iteraci obsahuje pole `vzdal[]` takové vzdálenosti z u do ostatních vrcholů, které vedou pouze přes vrcholy v S. Uvědomte si, že tyto vzdálenosti jsou však nejkratší i v celém G.

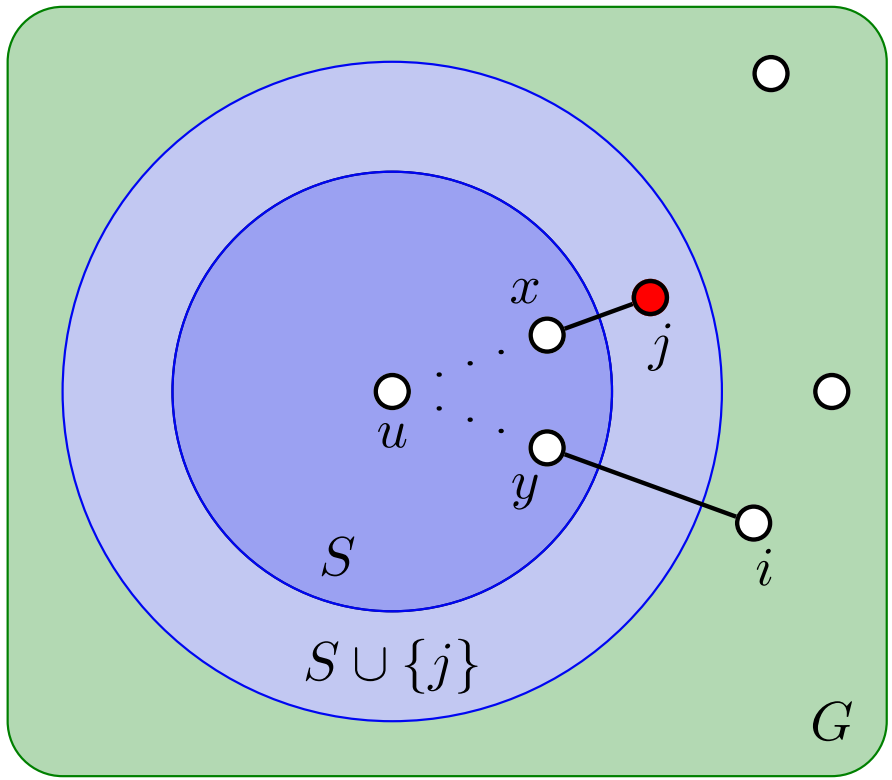
Postupujeme indukcí vzhledem k počtu iterací:

Základ indukce: V první iteraci Dijkstrova Algoritmu je v iniciačním stavu jediný vrchol u. Zpracujeme jej a upravíme vzdálenosti všem jeho sousedům.

Uvedené pozorování je splněno triviálně, neboť na konci první iterace je $S = \{u\}$ a získané vzdálenosti do všech vrcholů jsou nejmenší, jestliže bereme v úvahu pouze cesty přes vrcholy množiny S.

Důkaz (pokračování)

Indukční krok: V každé další iteraci vybereme ten vrchol j, který má ze všech nezpracovaných vrcholů nejmenší nalezenou vzdálenost od u. Současně je cesta do vrcholu j nejkratší možná v celém grafu (žádná kratší cesta už do j nevede), neboť každá „oklika“ přes nezpracované vrcholy i mimo S musí být delší (díky výběru vrcholu j).



Využíváme nezápornosti ohodnocení, přes i musí být cesty delší než přes j. Podle principu matematické indukce je důkaz hotov. □

Příště

Kapitola Stromy

- motivace
- základní vlastnosti stromů
- kořenové stromy
- isomorfismus stromů
- kostry grafů