Přehled přednášky

Kapitola 8. Vzdálenost a metrika

- motivace
- vzdálenost v grafu
- výpočet metriky
- vážená (ohodnocená) vzdálenost
- hledání nejkratší cesty

Opakování

Budeme používat následující pojmy:

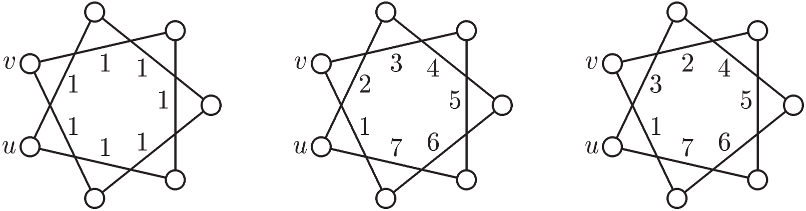
- stupeň vrcholu
- cesta
- délka cesty
- podgraf

Motivace

V mnoha praktických aplikacích má smysl v grafu „měřit“ vzdálenosti.
V grafu interpretujícím silniční síť je velmi přirozené ptát se

> „Jak daleko je z vrcholu u do vrcholu v?“

Vzdálenost při tom nebude pouhý *počet hran* (počet různých silnic) ale důležitá bude jejich *délka*.
Zavedeme obecně pojem *ohodnocení* hran. Význam ohodnocení je rozmanitý: délka, tloušťka, kapacita, barva, ...



Obrázek: *Různé vzdálenosti mezi vrcholy u a v v grafu C_7.*

V levém grafu

je nejmenší vzdálenost mezi vrcholy u a v rovna 3 = počet hran nejkratší cesty (sledu)

V prostředím grafu

je nejmenší vzdálenost mezi vrcholy u a v rovna $13 = 2 + 4 + 6 + 1$

V pravém grafu

je nejmenší vzdálenost mezi vrcholy u a v rovna $14 = 3 + 4 + 6 + 1 = 2 + 5 + 7$

8.1. Vzdálenost v grafu

Nejprve pro neohodnocené grafy, tj. kdy každá hrana má délku 1.

Délkou sledu 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 8.1. *Vzdálenost* $d_G(u, v)$
mezi vrcholy u a v grafu G je dána délkou nejkratšího sledu mezi u a v v G. Pokud sled mezi u, v neexistuje, je položíme vzdálenost $d_G(u, v) = \infty$.

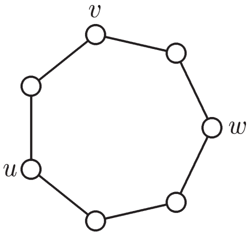
Všimněte si, že

- nejkratší sled (nejméně hran) je vždy cestou
- v neorientovaných grafech je $d_G(u, v) = d_G(v, u)$
- $d_G(u, u) = 0$
- je-li $d_G(u, v) = 1$, je $uv \in E(G)$

Lemma 8.2. Vzdálenost v grafu G splňuje *trojúhelníkovou nerovnost*:

$$\forall u, v, w \in V(G) : \; d_G(u, v) + d_G(v, w) \geq d_G(u, w).$$

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



Obrázek: *Dva sledy* u, v *a* v, w *délky* 2; *sled mezi* u, w *délky* 3.

Věta 8.3. Nechť u, v, w jsou vrcholy souvislého grafu G takové, že $d_G(u,v) < d_G(u,w)$. Pak při algoritmu procházení grafu G *do šířky* z vrcholu u je vrchol v nalezen dříve než vrchol w.

Důkaz Postupujeme indukcí vzhledem ke vzdálenosti $d_G(u,v)$:
Pro $d_G(u,v) = 0$, tj. $u = v$ je tvrzení zřejmé – vrchol u jako počátek prohledávání byl nalezen první.
Proto mějme $d_G(u,v) = d > 0$ a označme v' souseda vrcholu v bližšího na sledu u, v k u, je $d_G(u,v') = d - 1$. Stejně označme w' souseda vrcholu w bližšího na sledu u, w k u, je tedy $d_G(u,w') > d_G(u,v')$.
To znamená, že vrchol v' bude v prohledávání do šířky nalezen dříve než vrchol w' podle indukčního předpokladu. To také znamená, že v' se dostal do fronty úschovny dříve než w', a proto sousedé vrcholu v' (mezi nimi i vrchol v) budou při prohledávání nalezeni dříve než sousedé vrcholu w'. □

Důsledek 8.4. Základní algoritmus procházení grafu do šířky lze použít pro výpočet vzdáleností z vrcholu u do všech ostatních vrcholů.

8.2. Výpočet metriky

Metrikou grafu rozumíme soubor všech vzdáleností mezi každou dvojicí vrcholů v grafu. Předpokládáme, že metrika splňuje „přirozené“ vlastnosti.
Metrika grafu G je matice (dvourozměrné pole) `d[][]`, kde prvek `d[i][j]` udává vzdálenost mezi vrcholy i a j. (vrcholy 0, 1, ..., $|V(G)| - 1$).

Metoda 8.5. Výpočet 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 každém kroku $t \geq 0$ bude `d[i][j]` udávat délku nejkratší cesty mezi i, j, která jde pouze přes vrcholy z množiny $\{0, 1, 2, \ldots, t-1\}$.
- V průběhu $(t+1)$ kroku upravíme vzdálenost pro každou dvojici vrcholů, jsou dvě možnosti:
  - povolení vrcholu $t+1$ nepomůže najít cestu kratší než byla `d[i][j]` v předchozím kroku,
  - najdeme kratší cestu přes nově povolený vrchol $t+1$; nahradíme `d[i][j]` menší vzdáleností `d[i][t] + d[t][j]`.

Algoritmus 8.6.

```
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
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));
//                          hodnota MAX_INT/2 je "nekonečno"

// cyklus pro všechny vrcholy
for (t=0; t<N; t++)
     for (i=0; i<N; i++)
          for (j=0; j<N; j++)
               d[i][j] = min(d[i][j], d[i][t]+d[t][j]);
```

Výhody:

- jednoduchá implementace
- najde metriku mezi každými dvěma vrcholy

Nevýhody:

- složitost řádově N^3
- i když nás zajímá jen vzdálenost dvou vrcholů, *musí* najít vzdálenost mezi každými dvěma vrcholy

8.3. Vážená (ohodnocená) vzdálenost

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

Definice 8.7. *Vážený graf*
je graf G s ohodnocením hran reálnými čísly $w : E(G) \to \mathbf{R}$.
Kladně vážený (ohodnocený) graf G
má takové ohodnocení w, že $w(e) > 0$ pro $\forall e \in E(G)$.

Definice Mějme (kladně) vážený graf G s ohodnocením w. *Délkou váženého sledu* $S = v_0, e_1, v_1, e_2, v_2, \dots, e_n, v_n$ v G rozumíme součet

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

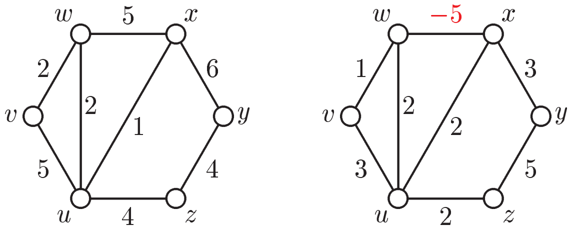
Váženou vzdáleností ve váženém G, w mezi dvěma vrcholy u, v rozumíme

$$d_G^w(u,v) = \min\{d_G^w(S) : S \text{ je sled s koncovými vrcholy } u, v\}.$$

Lemma 8.8. Vážená vzdálenost v kladně vážených grafech splňuje *trojúhelníkovou nerovnost*

$$\forall u, v, w \in V(G) :\ d_G^w(u,v) + d_G^w(v,w) \geq d_G^w(u,w).$$

Příklad:



Obrázek: *Dvě různá ohodnocení grafu* G.

Otázky:

Jaká je v levém obrázku vzdálenost vrcholů v a y? 13? 12? 11? 10?
A jaká vzdálenost vrcholů w a z?

V grafu nedovolíme záporné délky, neboť by nemusela existovat minimální vzdálenost.

Otázka:

Jaká je v pravém obrázku vzdálenost vrcholů v a w? 1?, 0?, -1?, 10? $-n$?

8.4. Hledání nejkratší cesty

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

- je složitější než Algoritmus 8.6.
- je *výrazně rychlejší*; najde nejkratší vzdálenosti z jednoho vrcholu místo mezi všemi dvojicemi vrcholů

Právě Dijkstrův algoritmus se užívá při vyhledávání vlakových a autobusových spojení.

Dijkstrův algoritmus

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

Algoritmus 8.9. Dijkstrův algoritmus

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 wh[][], 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 wh[i][k] > 0;
vstup: u,v, kde 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
// pri[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;
```

```
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]+wh[j][k] < vzdal[sous[j][k]]) {
               pri[sous[j][k]] = j;
               vzdal[sous[j][k]] = vzdal[j]+wh[j][k];
          }
          // pole pri[] obsahuje informaci o tom,
          // odkud jsme se do kterého vrcholu dostali
}
výstup: 'Cesta délky vzdal[v], uložená pozpátku v poli pri[]';
```

Poznámky k Dijkstrovu algoritmu

- Poběží-li cyklus nikoli do splnění podmínky `stav[v] == iniciační`, ale pro všechny vrcholy, najde algoritmus nejkratší vzdálenosti z počátečního u vrcholu do všech ostatních vrcholů a uloží je do pole `vzdal[i]`.

- Algoritmus najde nejkratší cesty i v *orientovaném grafu*.

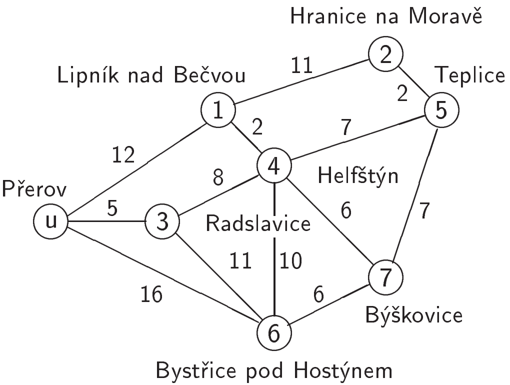
- Můžeme jej modifikovat i pro hledání *nejširší cesty*.

- Celkový počet kroků potřebný v Dijkstrově Algoritmu k nalezení nejkratší cesty z `u` do `v` je zhruba N^2, kde N je počet vrcholů grafu.

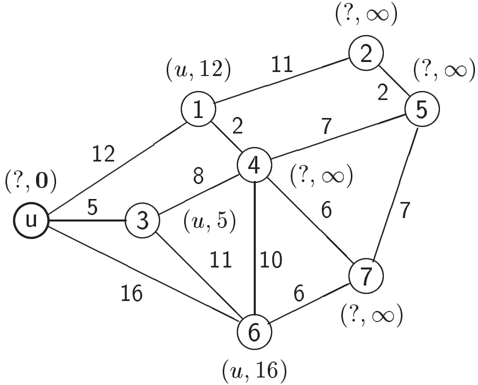
- Při vhodné implementaci úschovny nezpracovaných vrcholů (např. haldou s nalezenou vzdáleností jako klíčem) lze dosáhnout i rychlejšího běhu tohoto algoritmu na řídkých grafech – čas zhruba úměrný počtu hran grafu.

Následuje příklad. . .

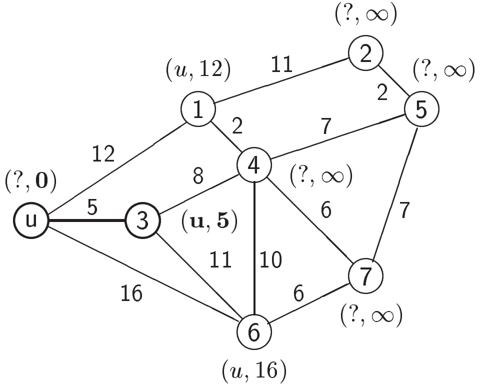
Vezměme například fragment mapy východně od Přerova. Budeme chtít najít nejkratší cesty z Přerova do všech ostatních míst.



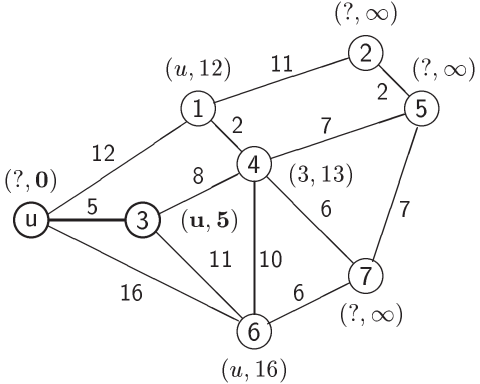
Toto je skica uvedené mapky. Jsou na ní vyznačeny přibližné vzdálenosti jednotlivých míst v kilometrech. Tato skica je zároveň i *grafovou interpretací* dané úlohy. Vrcholy reprezentují města a silnice mezi městy jsou hrany mezi odpovídajícími vrcholy. Vrchol i budeme značit $(pri[i], vzdal[i])$.



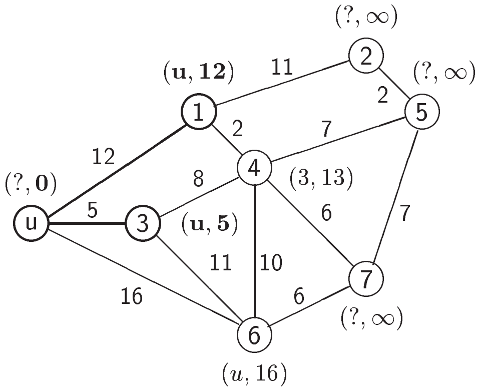
Můžeme přistoupit k inicializeci Dijkstrova algoritmu. Každý vrchol dostane stav 0 (iniciační). Pouze výchozí vrchol s dostane vzdálenost 0, tj. značku (0,0) Všechny takové vrcholy j, které jsou incidentní s vrcholem s, dostanou značku $(s, wh[s][j])$. Ostatní vrcholy dostanou značku (s, ∞).



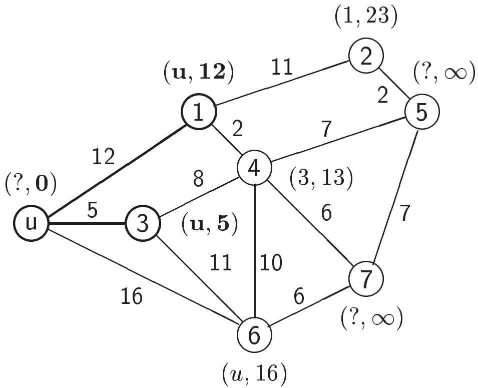
V prvním kroku najdeme takový vrchol k, který má od vrcholu s nejmenší vzdálenost. Tímto vrcholem je vrchol 3.



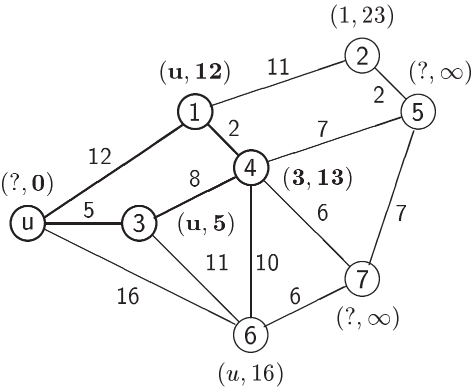
V dalším kroku budeme měnit značky všech sousedů vrcholu 3 (nejbližší nezpracovaný vrchol).
Změníme značku vrcholu 4. Nová značka vrcholu 4 bude $(3, 13)$.
Značku vrcholu 6 měnit nebudeme. Vrchol s je rovněž incidentní s vrcholem 3, ale jeho značku také měnit nebudeme.



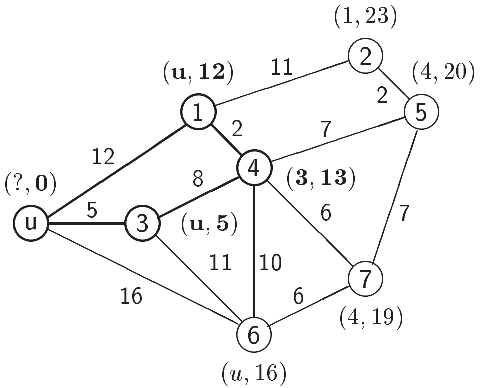
Budeme hledat takový vrchol k, který má nejmenší vzdálenost od s.
Takovým vrcholem je vrchol 1 ($vzdal[1] = 12$).



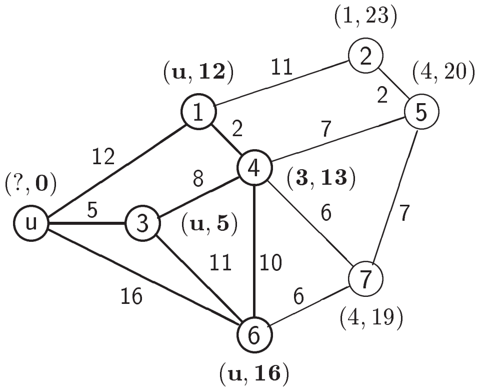
Vrchol 2 dostane proto značku $(1, 23)$, protože $\infty > vzdal[1] + wh[1][2]$. Avšak značku vrcholu 4 měnit nebudeme. Nebudeme měnit ani značku vrcholu s.



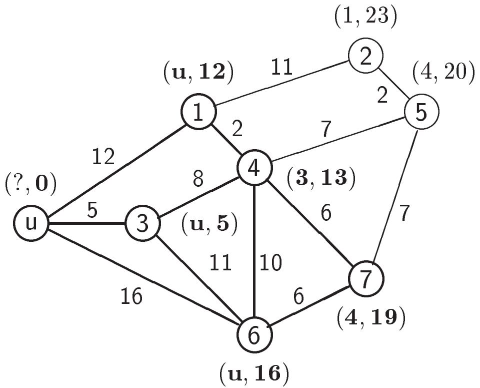
Vrchol 4 je nejbíže vrhchlu s a proto jej zpracujeme.



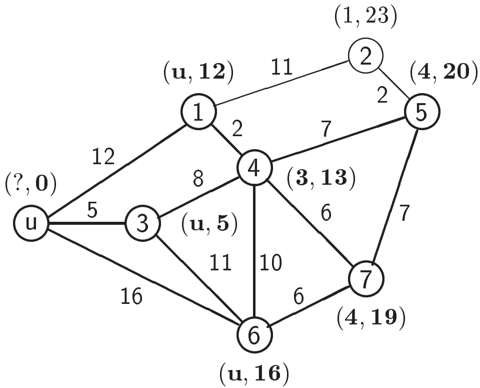
Podíváme se na sousedy vrcholu 4. Jedná se o vrcholy 5, 6 a 7. Protože $vzdal[5] > vzdal[4] + wh[4][5]$ $(\infty > 13 + 7)$, bude nová značka vrcholu 5 $(4, 20)$. Značku vrcholu 6 měnit nebudeme. Avšak vrchol 7 dostane novou značku: $(4, 13 + 6)$.



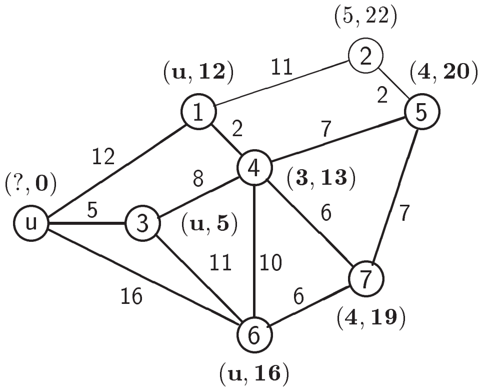
Nejblíže s je vrchol 6. Ostatní vrcholy 2, 5 a 7 mají větší hodnotu $vzdal[i]$. Nebudeme měnit značku žádného vrcholu, není co vylepšovat!



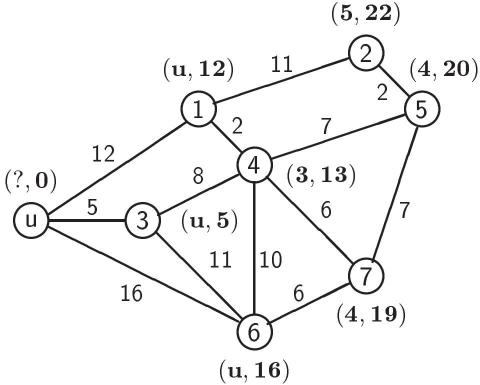
Nejblíže s je nyní vrchol 7 ($vzdal[7] = 19$). Opět nebudeme měnit značku žádného vrcholu.



Nejblíže s je vrchol 5, protože $vzdal[5] < vzdal[2]$ $(20 < 23)$. Změníme jeho značku.



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



Nejbližším (jediným) vrcholem k vrcholu s je vrchol 2. Změníme jeho stav a algoritmus končí.

Získali jsme vzdálenosti všech vrcholů od výchozího vrcholu s.

Důkaz správnosti Dijkstrova Algoritmu

Klíčovým faktem je, že v každém kroku (počínaje stavem po prvním průchodu cyklem `while()`) proměnná `vzdal[i]` udává nejkratší vzdálenost z vrcholu `u` do vrcholu `i` při cestě po vnitřních (zpracovaných) vrcholech x, jejichž `stav[x]==1`:

- V prvním kroku algoritmu vybereme první `j=u`; potom jeho sousedům upravíme vzdálenosti od `u` podle délek hran z `u`.
- V každém dalším kroku vybereme ten vrchol `j`, který má ze všech nezpracovaných vrcholů nejkratší nalezenou vzdálenost od `u`. To znamená, že žádná kratší cesta už do `j` nevede, neboť každá oklika přes jiné nezpracované vrcholy musí být delší dle výběru vrcholu `j`. (Využíváme nezápornost ohodnocení `wh[][]`.) Naopak můžeme podle délek hran vycházejících z `j` „vylepšit" cesty do jeho sousedů.

Zbývá sestavit nejkratší cestu z `u` do `v`. Je pozpátku uložená v poli `pri[]`: předposlední vrchol před `v` je `pri[v]`, před předposlední `pri[pri[v]]`, atd. . .

Příště

Kapitola 9. Stromy a les

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