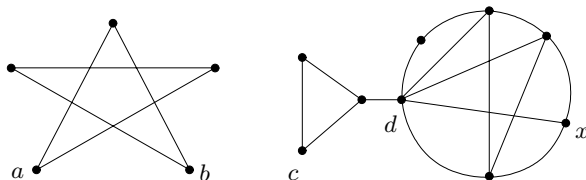


3 Vzdálenost a metrika v grafech

V minulé lekci jsme mluvili o souvislosti grafu, tj. o možnosti procházení z jednoho vrcholu do jiného. Někdy je prostá informace o souvislosti dostačující, ale většinou bychom rádi věděli i jak je z **jednoho vrcholu do druhého daleko** ...



V jednodušším případě se při zjišťování grafové vzdálenosti díváme jen na minimální počet prošlých hran z vrcholu do vrcholu. V obecném případě však při určování vzdálenosti bereme do úvahy **délky jednotlivých hran** podél cesty (tyto délky musí být nezáporné!). □

Stručný přehled lekce

- Vzdálenost v grafech a její vlastnosti.
- Výpočet metriky grafu (Floyd–Warshall).
- Dijkstrův algoritmus pro nejkratší (ohodnocenou) cestu v grafu.

3.1 Vzdálenost v grafu

Vzpomeňme si, že sledem délky n v grafu G rozumíme posloupnost vrcholů a hran $v_0, e_1, v_1, e_2, v_2, \dots, e_n, v_n$, ve které hrana e_i má koncové vrcholy v_{i-1}, v_i .

Definice 3.1. **Vzdálenost** $d_G(u, v)$ dvou vrcholů u, v v grafu G

je dána délkou nejkratšího sledu mezi u a v v G .

Pokud sled mezi u, v neexistuje, je vzdálenost $d_G(u, v) = \infty$. \square

Neformálně řečeno, vzdálenost mezi u, v je rovna *nejmenšímu počtu hran*, které musíme projít, pokud se chceme dostat z u do v . Speciálně $d_G(u, u) = 0$.

Uvědomme si, že nejkratší sled je vždy cestou (vrcholy se neopakují) – Věta 2.2.

Fakt: V neorientovaném grafu je vzdálenost symetrická, tj. $d_G(u, v) = d_G(v, u)$. \square

Lema 3.2. *Vzdálenost v grafech 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 snadno plyne ze zřejmého pozorování, že na sled délky $d_G(u, v)$ mezi u, v lze navázat sled délky $d_G(v, w)$ mezi v, w , čímž vznikne sled délky $d_G(u, v) + d_G(v, w)$ mezi u, w . Skutečná vzdálenost mezi u, w pak už může být jen menší. \square

Zjištění vzdálenosti

Věta 3.3. *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 podle vzdálenosti $d_G(u, v)$: Pro $d_G(u, v) = 0$, tj. $u = v$ je tvrzení jasné – vrchol u jako počátek prohledávání byl nalezen první. Proto nechť $d_G(u, v) = d > 0$ a označme v' souseda vrcholu v bližšího k u , tedy $d_G(u, v') = d - 1$. Obdobně uvažme libovolného souseda w' vrcholu w . Pak

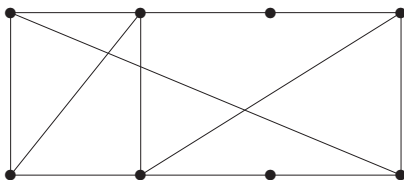
$$d_G(u, w') \geq d_G(u, w) - 1 > d_G(u, v) - 1 = d_G(u, v'),$$

a tudíž vrchol v' byl nalezen v prohledávání do šířky **dříve než** vrchol w' podle indukčního předpokladu. To znamená, že v' se dostal do fronty úschovny dříve než w' . Proto sousedé v' (mezi nimi v) jsou při pokračujícím prohledávání **také nalezeni dříve než** sousedé w' (mezi nimi w). □

Důsledek 3.4. *Algoritmus procházení grafu do šířky lze použít pro výpočet grafové vzdálenosti z daného vrcholu u . □*

Důsledek 3.4 funguje jen pro “vzdálenost” s jednotkovou délkou všech hran. My si dále ukážeme obecnější Dijkstrův algoritmus, který obdobným postupem počítá nejkratší vzdálenost při libovolně kladně ohodnocených délkách hran.

Další pojmy a fakta



Definice. Mějme graf G . Definujeme (vzhledem k G) následující pojmy a značení:

- **Excentricita** vrcholu $exc(v)$ je nejdelší vzdálenost z v do jiného vrcholu grafu; $exc(v) = \max_{x \in V(G)} d_G(v, x)$. \square
- **Průměr** $diam(G)$ grafu G je největší excentricita jeho vrcholů, naopak **poloměr** $rad(G)$ grafu G je nejmenší excentricita jeho vrcholů. \square
- **Centrem** grafu je množina vrcholů $U \subseteq V(G)$ takových, jejichž excentricita je rovna poloměru $rad(G)$. \square
- **Steinerova vzdálenost** mezi vrcholy libovolné podmnožiny $W \subseteq V(G)$ je rovna minimálnímu počtu hran souvislého podgrafu v G obsahujícího všechny vrcholy W .

3.2 Výpočet metriky

Definice: Metrikou grafu myslíme soubor vzdáleností mezi všemi dvojicemi vrcholů grafu. Jinak řečeno, *metrikou grafu* G je matice (dvourozměrné pole) $d[,]$, ve kterém prvek $d[i, j]$ udává vzdálenost mezi vrcholy i a j . □

Metoda 3.5. Dynamický výpočet metriky skládáním cest

- Na počátku nechť $d[i, j]$ udává 1 (případně **délku hrany** $\{i, j\}$), nebo ∞ pokud hrana mezi i, j není. □
- Po každém kroku $t \geq 0$ nechť $d[i, j]$ udává délku nejkratší cesty mezi i, j , která jde pouze přes vnitřní vrcholy z množiny $\{0, 1, 2, \dots, t - 1\}$. □
- Při přechodu z t na následující krok $t + 1$ upravujeme vzdálenost pro každou dvojici vrcholů – jsou vždy vždy pouhé dvě možnosti:
 - Buď je cesta délky $d[i, j]$ z předchozího kroku stále nejlepší (tj. nově povolený vrchol t nám nepomůže),
 - **nebo** cestu vylepšíme spojením přes nově povolený vrchol t , čímž získáme menší vzdálenost $d[i, t] + d[t, j]$.

Věta 3.6. *Metoda 3.5 v poli $d[i, j]$ správně vypočte vzdálenost mezi vrcholy i, j .*

Poznámka: V implementaci pro symbol ∞ použijeme velkou konstantu, třeba $\text{MAX_INT}/2$.

Algoritmus 3.7. Výpočet metriky grafu; Floyd–Warshall

input: *matice sousednosti* $G[] []$ *grafu na* N *vrcholech (číslovaných* $0 \dots N-1$),
kde $G[i, j]=1$ *pro hranu mezi* i, j *a* $G[i, j]=0$ *jinak;*

```
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));
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]);
}
return 'Matice vzdáleností d[] []'; □
```

Algoritmus 3.7 je implementačně velmi jednoduchý a provede zhruba N^3 kroků pro výpočet celé metriky.

Jeho jedinou (ale velkou) nevýhodou je, že vzdálenosti mezi všemi dvojicemi vrcholů je třeba *počítat najednou*. V praktických situacích však obvykle požadujeme zjištění vzdálenosti mezi jednou dvojicí vrcholů, a pak celý zbytek výpočtu je k ničemu...

3.3 Vážená (ohodnocená) vzdálenost

Definice 3.8. Vážený graf je graf G spolu s ohodnocením w hran reálnými čísly $w : E(G) \rightarrow \mathbf{R}$.

Kladně vážený graf G, w je takový, že $w(e) > 0$ pro všechny hrany e . \square

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

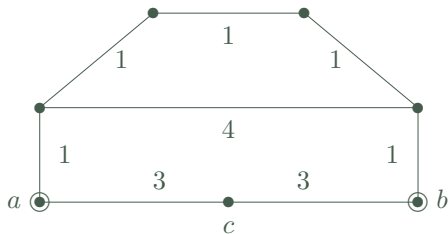
$$d_G^w(S) = w(e_1) + w(e_2) + \dots + w(e_n).$$

Váženou vzdáleností v G, w mezi dvěma vrcholy u, v pak myslíme

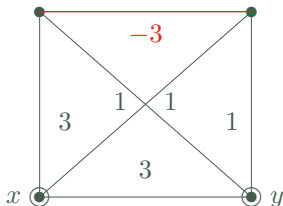
$$d_G^w(u, v) = \min\{d_G^w(S) : S \text{ je sled s konci } u, v\} . \square$$

Obdobně Oddílu 3.1 snadno dokážeme:

Lema 3.9. *Vážená vzdálenost v kladně vážených grafech také splňuje trojúhelníkovou nerovnost.*



Vzdálenost mezi vrcholy a, c je 3, stejně tak mezi b, c . Co ale mezi a, c ? Je jejich vzdálenost 6? Kdepak, vzdálenost a, b je 5, cesta vede po „horních“ vrcholech.



A jaká je v našem druhém grafu vzdálenost mezi x, y ? Je to 3 nebo 1?
 Ne, ta vzdálenost je $-\infty$.

To je nakonec dobrý důvod, proč **zakázat záporné hrany**.

3.4 Hledání nejkratší cesty

Pro nalezení nejkratší (vážené) cesty mezi dvěma vrcholy kladně váženého grafu se používá tradiční *Dijkstrův algoritmus* či jeho vhodná vylepšení.

Takové algoritmy se například používají při vyhledávání vlakových spojení. Pravděpodobně se i vy někdy dostanete do situace, kdy budete nejkratší cestu hledat, proto si popsaný algoritmus včetně jeho vylepšení A^* zapamatujte.

Poznámka: Dijkstrův algoritmus je sice poněkud složitější než Algoritmus 3.7, ale na druhou stranu je *výrazně rychlejší*, pokud nás zajímá jen nejkratší vzdálenost z jednoho vrcholu místo všech dvojic vrcholů. □

Dijkstrův algoritmus

- Je variantou procházení grafu (skoro jako do šířky), kdy pro každý nalezený vrchol ještě máme *proměnnou udávající vzdálenost* – délku nejkratšího sledu (od počátku), kterým jsme se do tohoto vrcholu zatím dostali. □
- Z úschovny nalezených vrcholů vždy vybíráme **vrchol s nejmenší vzdáleností** (mezi uschovanými vrcholy) – do takového vrcholu se už lépe dostat nemůžeme, protože všechny jiné cesty by byly dle výběru delší. □
- Na konci zpracování tyto proměnné vzdálenosti udávají správně **nejkratší vzdálenosti** z počátečního vrcholu do ostatních.

Algoritmus 3.10. Dijkstrův pro nejkratší cestu v grafu

Tento algoritmus nalezne nejkratší cestu mezi vrcholy u a v kladně váženého grafu G .

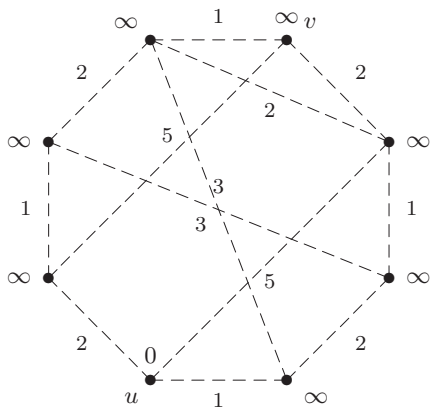
input: graf na N vrcholech daný seznamem sousedů $sous[] []$ a $del[] []$,
kde $sous[i][0], \dots, sous[i][st[i]-1]$ jsou sousedé vrcholu i stupně $st[i]$
a hrana z i do $sous[i][k]$ má délku $del[i][k] > 0$;
input: u, v , kde hledáme cestu z u do v ; □

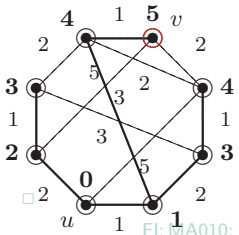
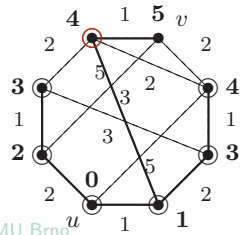
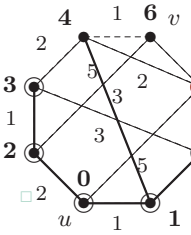
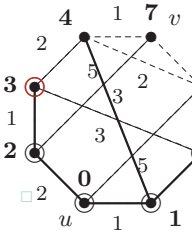
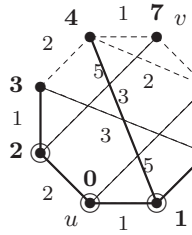
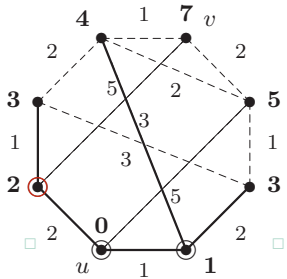
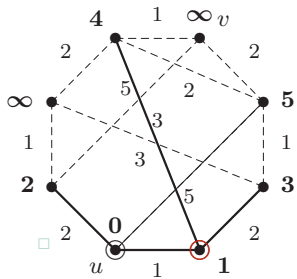
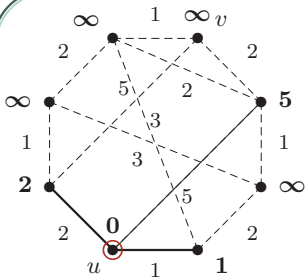
```
// stav[i] udává zpracovanost vrcholu, vzdal[i] zatím nalezenou vzdálenost
for (i=0; i<=N; i++) { vzdal[i] = MAX_INT; stav[i] = 0; }
vzdal[u] = 0; □
while (stav[v]==0) {
    for (i=0, j=N; i<N; i++)
        if (stav[i]==0 && vzdal[i]<vzdal[j]) j = i;
    // zde jsme našli nejbližší nezpracovaný vrchol j, ten teď zpracujeme
    if (vzdal[j]==MAX_INT) return 'Není cesta';
    stav[j] = 1; □
    for (k=0; k<st[j]; k++)
        if (vzdal[j]+del[j][k]<vzdal[sous[j][k]]) {
            prich[sous[j][k]] = j;
            vzdal[sous[j][k]] = vzdal[j]+del[j][k];
        }
}
return 'Cesta délky vzdal[v], uložená pozpátku v poli prich[]';
```

Poznámka: Uvědomme si, že pokud necháme tento algoritmus proběhnout až do zpracování všech vrcholů, získáme ve `vzda1[i]` nejkratší vzdálenosti z počátečního vrcholu do všech ostatních vrcholů.

Všimněme si dále, že Algoritmus 3.10 počítá nejkratší cestu i v **orientovaném grafu**.

Příklad 3.11. Ukázka běhu Dijkstrova Algoritmu 3.10 pro nalezení nejkratší cesty mezi vrcholy u, v v následujícím grafu.





Fakt: Celkový počet kroků potřebný v Algoritmu 3.10 k nalezení nejkratší cesty z u do v je zhruba N^2 , kde N je počet vrcholů grafu. □

Na druhou stranu, při lepší implementaci úschovny nezpracovaných vrcholů (třeba *haldou* s nalezenou vzdáleností jako klíčem) lze dosáhnout i mnohem rychlejšího běhu tohoto algoritmu na řídkých grafech – času **téměř úměrného počtu hran grafu**. □

Věta 3.12. *V každé iteraci Algoritmu 3.10 (počínaje stavem po prvním průchodu cyklem `while()`) proměnná `vzda1[i]` udává nejkratší vzdálenost z vrcholu u do vrcholu i při cestě pouze po vnitřních vrcholech x , jejichž `stav[x]==1`. □*

Důkaz: Stručně matematickou indukcí:

- V prvním kroku algoritmu je jako vrchol ke zpracování vybrán první $j=u$ a potom jsou jeho sousedům upraveny vzdálenosti od u podle délek hran z u . □
- V každém dalším kroku je vybrán jako vrchol j ke zpracování ten, který má ze všech nezpracovaných vrcholů nejkratší nalezenou vzdálenost od počátku u . To ale znamená, že žádná kratší cesta z u do j nevede, neboť každá „oklika“ přes jiné nezpracované vrcholy musí být delší dle výběru j a indukčního předpokladu. (V tomto bodě potřebujeme **nezápornost ohodnocení** `del[][]`.) □

Naopak každá nová nejkratší cesta z u do nezpracovaného vrcholu i procházející přes j musí mít j coby předposlední vrchol, tj. poslední hranu ij , a proto je upravená hodnota `vzda1[i]` správná i po přidání j mezi zpracované vrcholy. □

Ještě lépe než Dijkstrův algoritmus se chová vylepšený algoritmus A^* , který použitím vhodného **potenciálu** „směřuje“ celé prohledávání grafu ke správnému cíli a je skvěle použitelný ve všech situacích, kdy pojem „**směr k cíli**“ má význam. To je například při navigování v mapě.

Algoritmus A^*

- Je reimplementací Dijkstrova algoritmu s „vhodně“ **upravenými délkami** hran. \square
- Necht' „potenciál“ $p_v(x)$ udává libovolný **dolní odhad** vzdálenosti z vrcholu x do cíle v . Například při navigaci v mapě může $p_v(x)$ udávat přímou (Euklidovskou) vzdálenost z bodu x do bodu v . \square
- Každá (orientovaná!) hrana xy grafu G, w dostane nové délkové ohodnocení $w'(xy) = w(xy) + p_v(y) - p_v(x)$. Potenciál p_v je **přípustný**, pokud všechna upravená ohodnocení jsou nezáporná, neboli $w(xy) \geq p_v(x) - p_v(y)$.
Potenciál přímé vzdál. z x do cíle v je vždy přípustný podle trojúh. nerovnosti. \square
- Upravená délka lib. sledu S z u do v pak je $d_G^{w'}(S) = d_G^w(S) + p_v(v) - p_v(u)$, což je konstantní rozdíl oproti původní délce S . Takže S je optimální pro původní délkové ohodnocení w , právě když je optimální pro nové w' .

Dijkstrův algoritmus pro w' upravené potenciálnem přímé vzdálenosti do v pak bude „silně preferovat“ hrany vedoucí ve směru k cíli v .