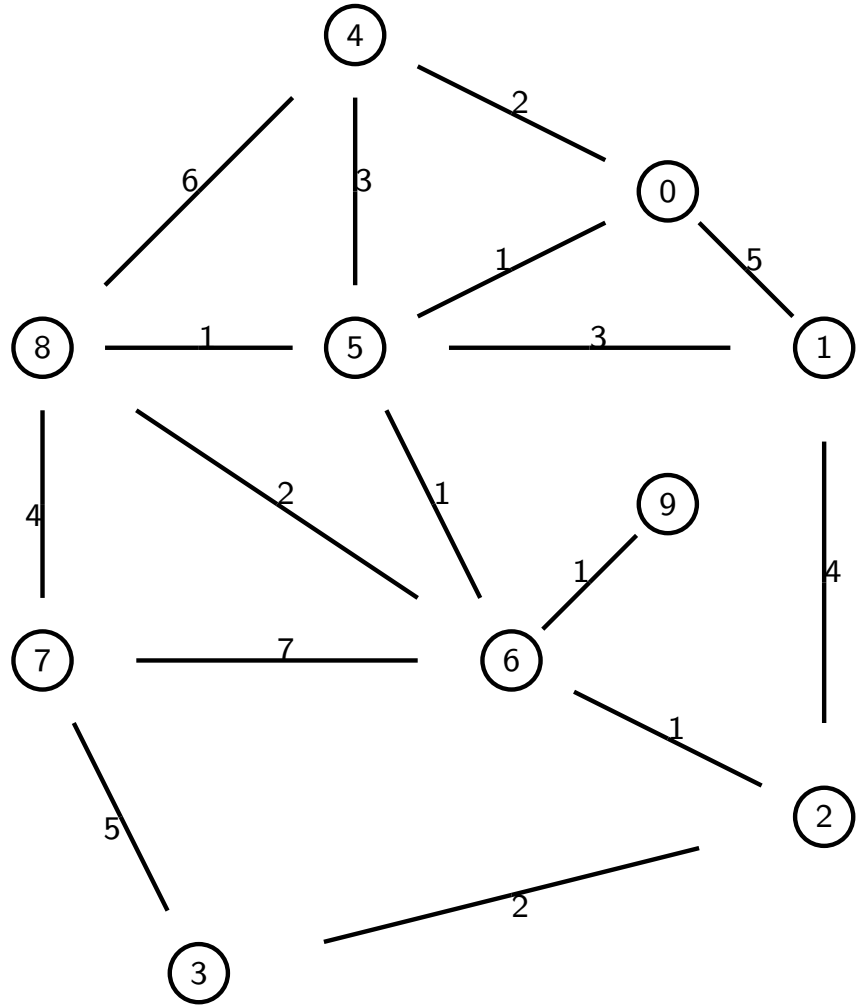
Dijkstrův algoritmus

text pro studenty učitelství na FP TUL

Martina Šimůnková

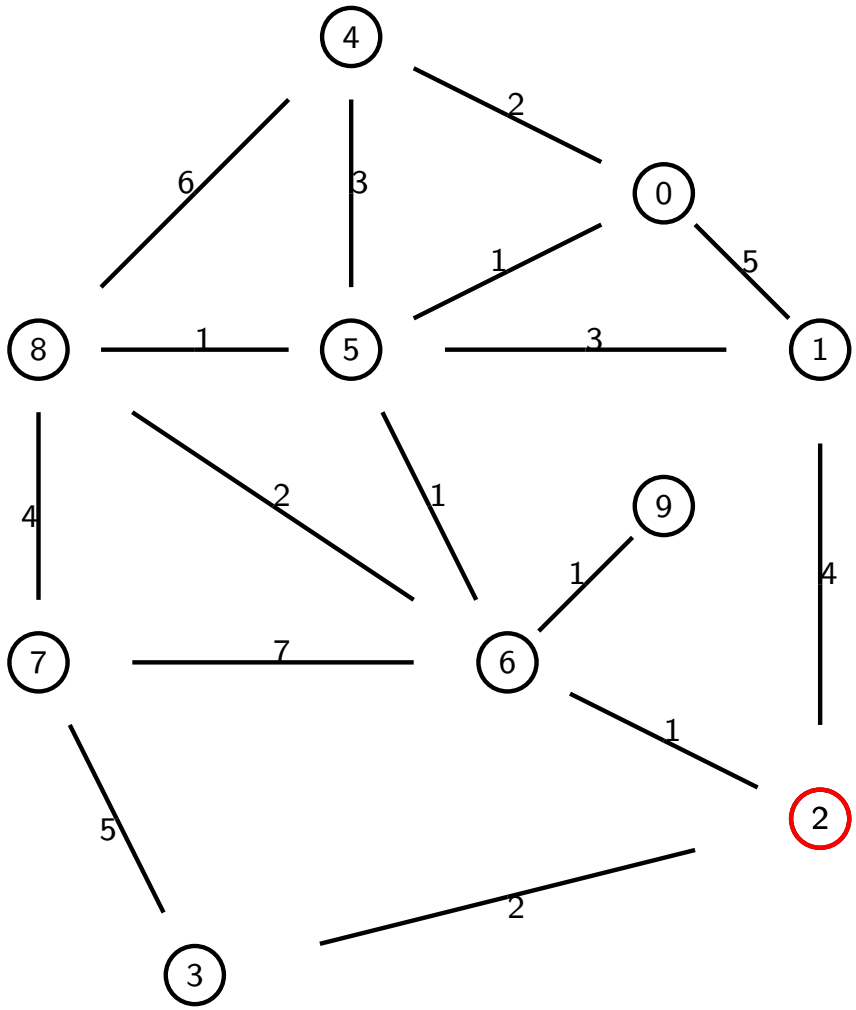
23. května 2023

V grafu vybereme počáteční vrchol v_0 a pro všechny vrcholy v grafu budeme hledat délku nejkratší cesty z v_0 do v. V tabulce zaznamenáme délku $h(v)$ dosud nalezené cesty a předposlední vrchol $P(v)$ na této cestě. Pomocí předchůdců $P(v)$ pak dokážeme zrekonstruovat celou cestu.



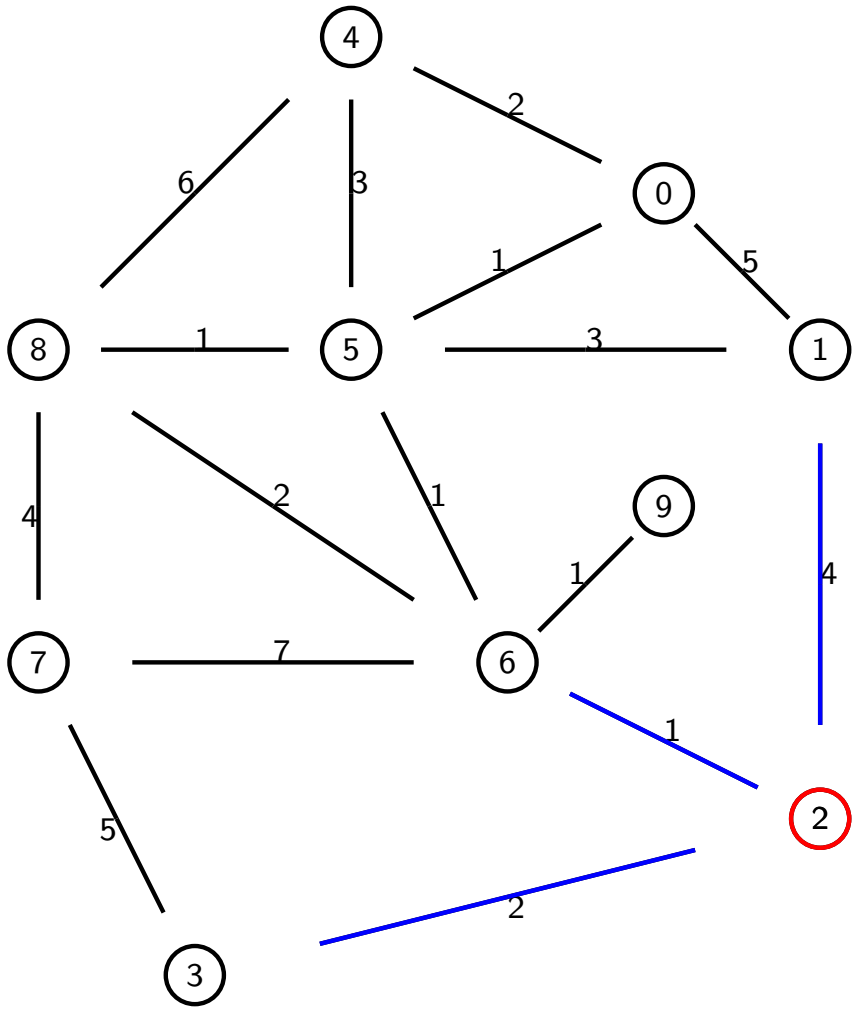
v	$h(v)$	$P(v)$
0	∞	
1	∞	
2	∞	
3	∞	
4	∞	
5	∞	
6	∞	
7	∞	
8	∞	
9	∞	

Vybereme počáteční vrchol $v_0 = 2$. Jeho sousedům spočítáme vzdálenost a vybereme nejbližší vrchol $v = 6$. Jeho sousedům spočítáme vzdálenost od v_0 a vybereme nejbližší. Spočítáme vzdálenosti (u vrcholu 7 jsme našli kratší, *relaxujeme*), vybereme, není co relaxovat, tak vybereme a vybereme.



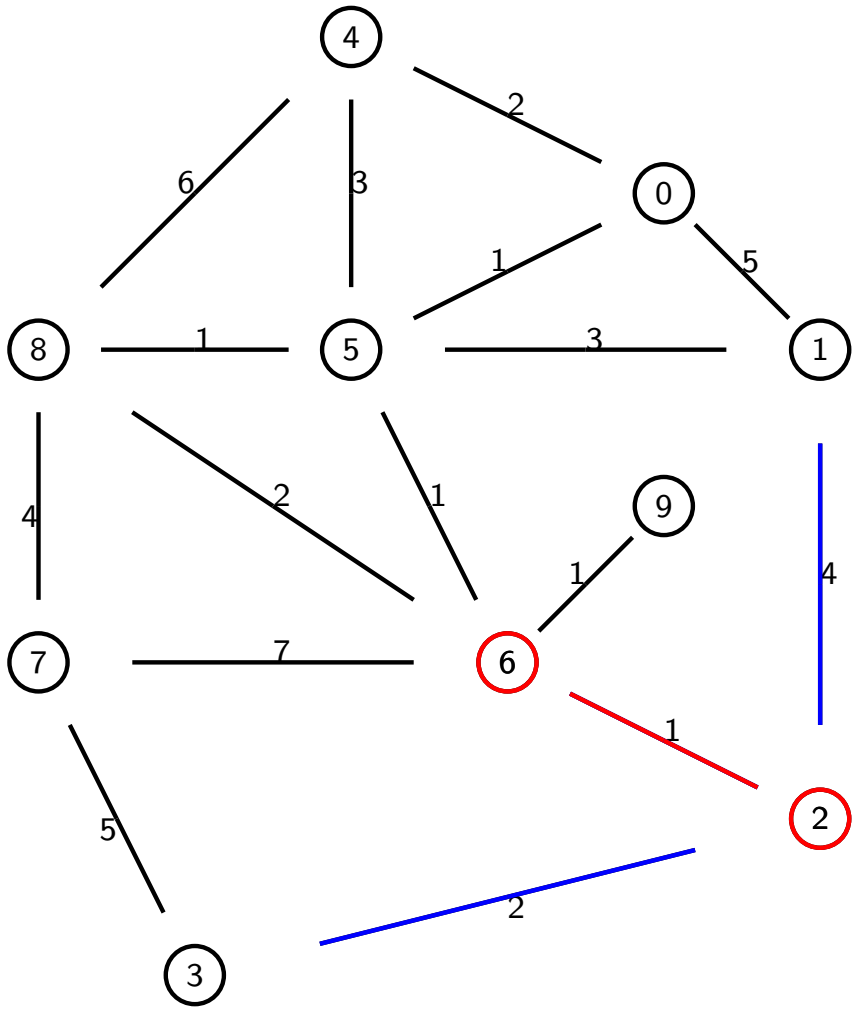
v	$h(v)$	$P(v)$
0	∞	
1	∞	
2	0	
3	∞	
4	∞	
5	∞	
6	∞	
7	∞	
8	∞	
9	∞	

Vybereme počáteční vrchol $v_0 = 2$. Jeho sousedům spočítáme vzdálenost a vybereme nejbližší vrchol $v = 6$. Jeho sousedům spočítáme vzdálenost od v_0 a vybereme nejbližší. Spočítáme vzdálenosti (u vrcholu 7 jsme našli kratší, *relaxujeme*), vybereme, není co relaxovat, tak vybereme a vybereme.



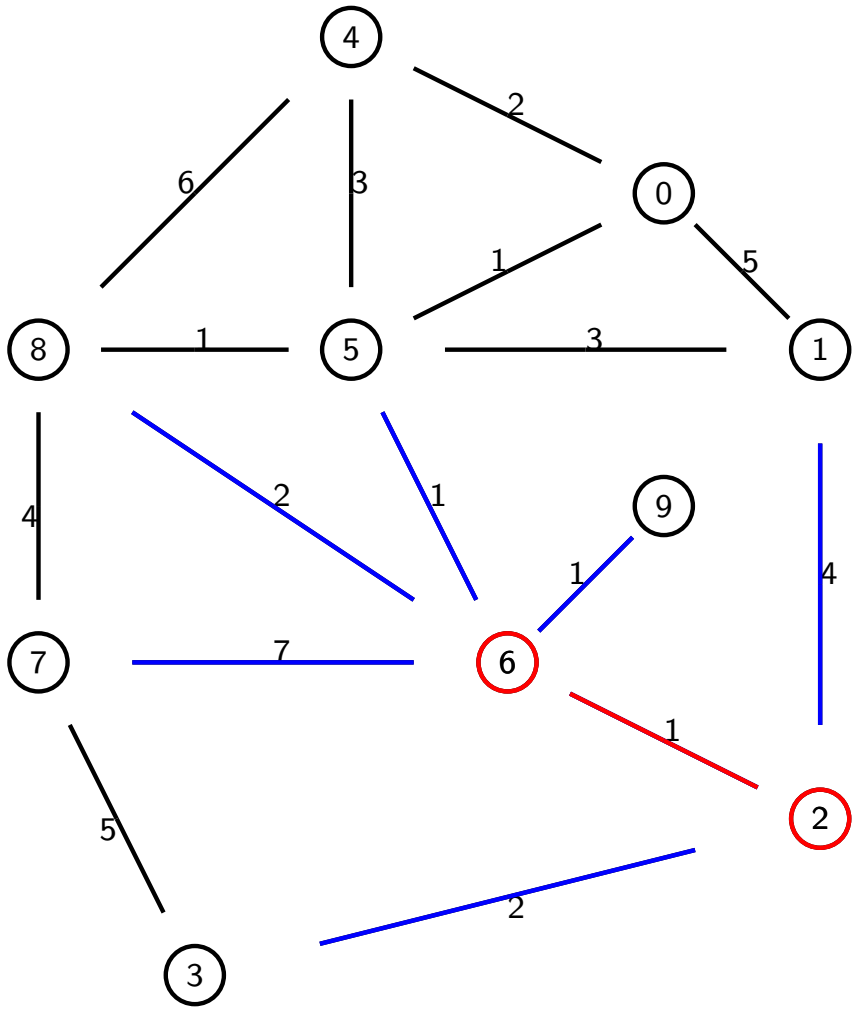
v	$h(v)$	$P(v)$
0	∞	
1	4	2
2	0	
3	2	2
4	∞	
5	∞	
6	1	2
7	∞	
8	∞	
9	∞	

Vybereme počáteční vrchol $v_0 = 2$. Jeho sousedům spočítáme vzdálenost a vybereme nejbližší vrchol $v = 6$. Jeho sousedům spočítáme vzdálenost od v_0 a vybereme nejbližší. Spočítáme vzdálenosti (u vrcholu 7 jsme našli kratší, *relaxujeme*), vybereme, není co relaxovat, tak vybereme a vybereme.



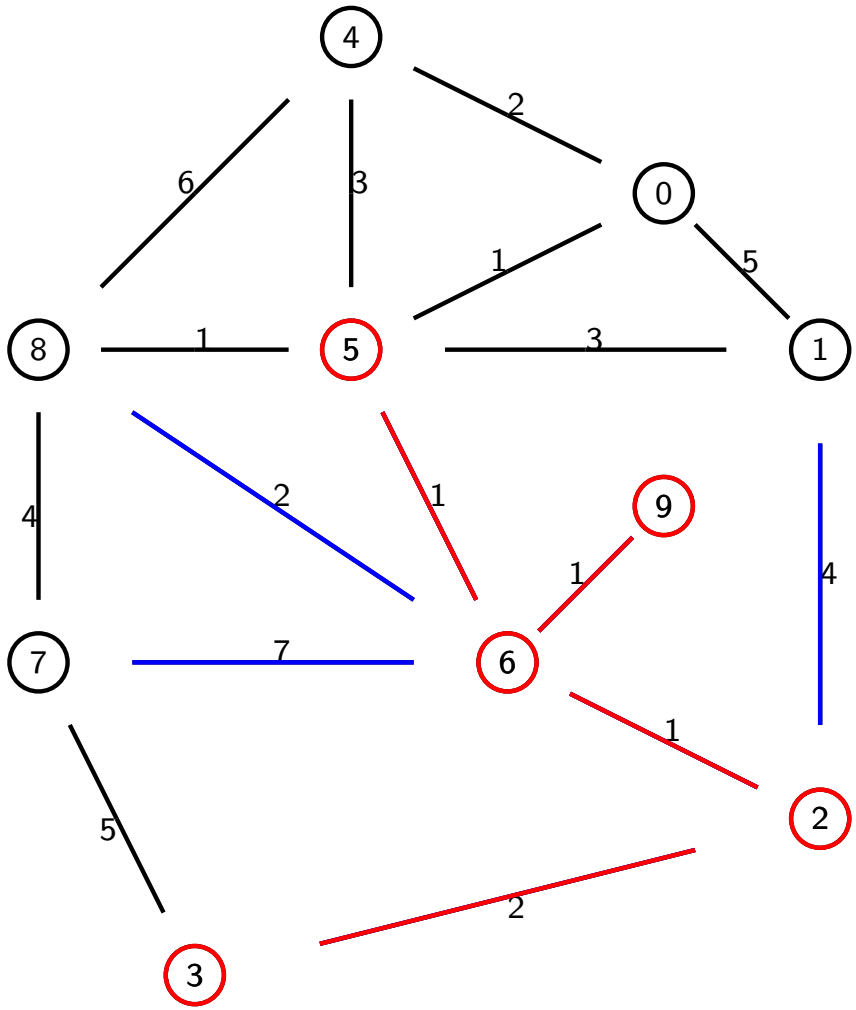
v	$h(v)$	$P(v)$
0	∞	
1	4	2
2	0	
3	2	2
4	∞	
5	∞	
6	1	2
7	∞	
8	∞	
9	∞	

Vybereme počáteční vrchol $v_0 = 2$. Jeho sousedům spočítáme vzdálenost a vybereme nejbližší vrchol $v = 6$. Jeho sousedům spočítáme vzdálenost od v_0 a vybereme nejbližší. Spočítáme vzdálenosti (u vrcholu 7 jsme našli kratší, *relaxujeme*), vybereme, není co relaxovat, tak vybereme a vybereme.



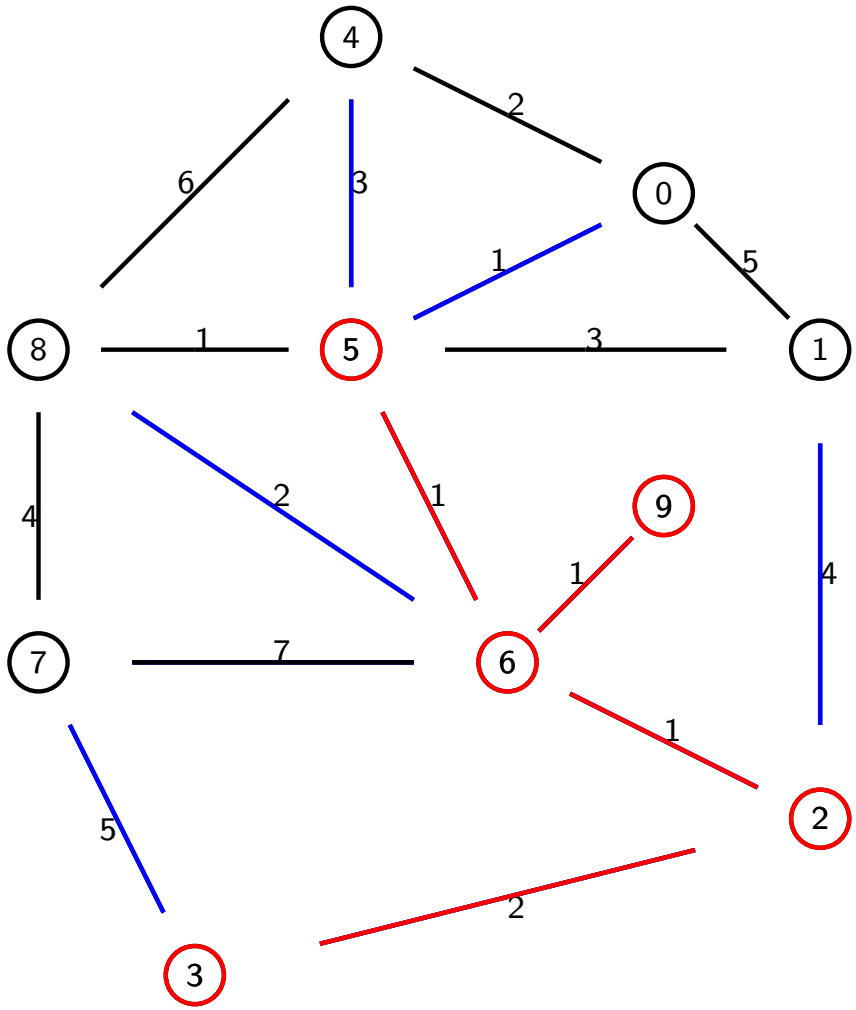
v	$h(v)$	$P(v)$
0	∞	
1	4	2
2	0	
3	2	2
4	∞	
5	2	6
6	1	2
7	8	6
8	3	6
9	2	6

Vybereme počáteční vrchol $v_0 = 2$. Jeho sousedům spočítáme vzdálenost a vybereme nejbližší vrchol $v = 6$. Jeho sousedům spočítáme vzdálenost od v_0 a vybereme nejbližší. Spočítáme vzdálenosti (u vrcholu 7 jsme našli kratší, *relaxujeme*), vybereme, není co relaxovat, tak vybereme a vybereme.



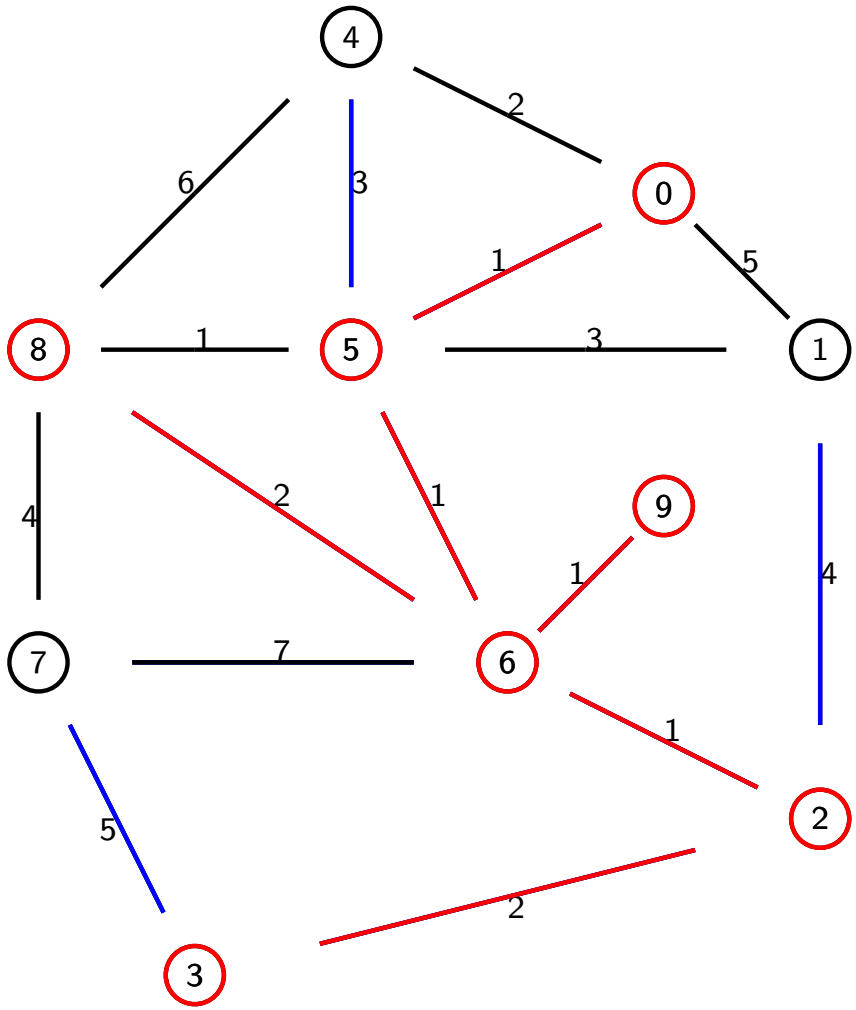
v	$h(v)$	$P(v)$
0	∞	
1	4	2
2	0	
3	2	2
4	∞	
5	2	6
6	1	2
7	8	6
8	3	6
9	2	6

Vybereme počáteční vrchol $v_0 = 2$. Jeho sousedům spočítáme vzdálenost a vybereme nejbližší vrchol $v = 6$. Jeho sousedům spočítáme vzdálenost od v_0 a vybereme nejbližší. Spočítáme vzdálenosti (u vrcholu 7 jsme našli kratší, *relaxujeme*), vybereme, není co relaxovat, tak vybereme a vybereme.



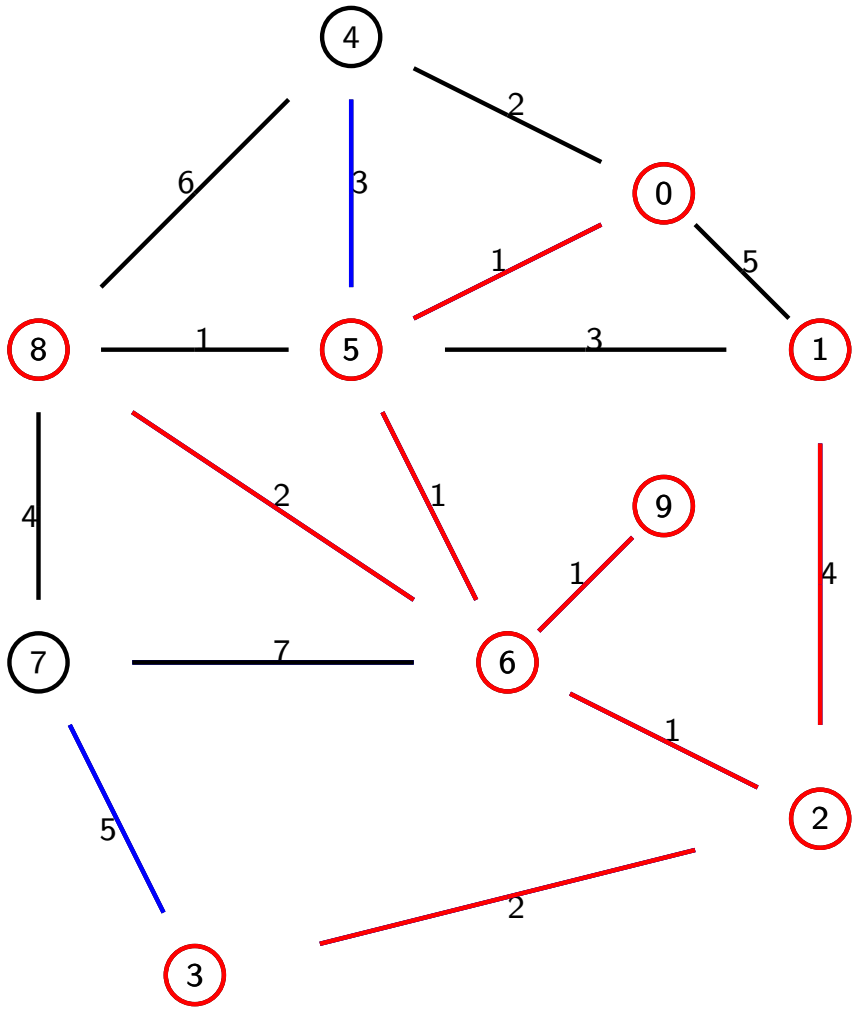
v	$h(v)$	$P(v)$
0	3	5
1	4	2
2	0	
3	2	2
4	5	5
5	2	6
6	1	2
7	7	3
8	3	6
9	2	6

Vybereme počáteční vrchol $v_0 = 2$. Jeho sousedům spočítáme vzdálenost a vybereme nejbližší vrchol $v = 6$. Jeho sousedům spočítáme vzdálenost od v_0 a vybereme nejbližší. Spočítáme vzdálenosti (u vrcholu 7 jsme našli kratší, *relaxujeme*), vybereme, není co relaxovat, tak vybereme a vybereme.



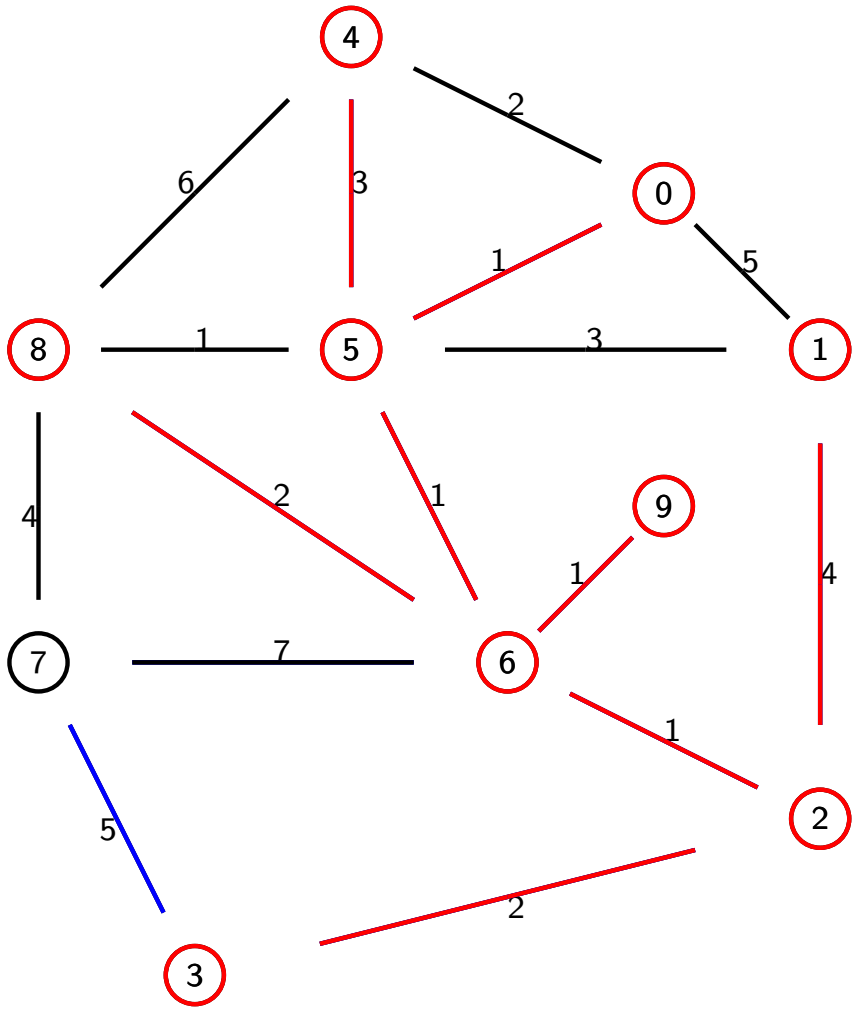
v	$h(v)$	$P(v)$
0	3	5
1	4	2
2	0	
3	2	2
4	5	5
5	2	6
6	1	2
7	7	3
8	3	6
9	2	6

Vybereme počáteční vrchol $v_0 = 2$. Jeho sousedům spočítáme vzdálenost a vybereme nejbližší vrchol $v = 6$. Jeho sousedům spočítáme vzdálenost od v_0 a vybereme nejbližší. Spočítáme vzdálenosti (u vrcholu 7 jsme našli kratší, *relaxujeme*), vybereme, není co relaxovat, tak vybereme a vybereme.



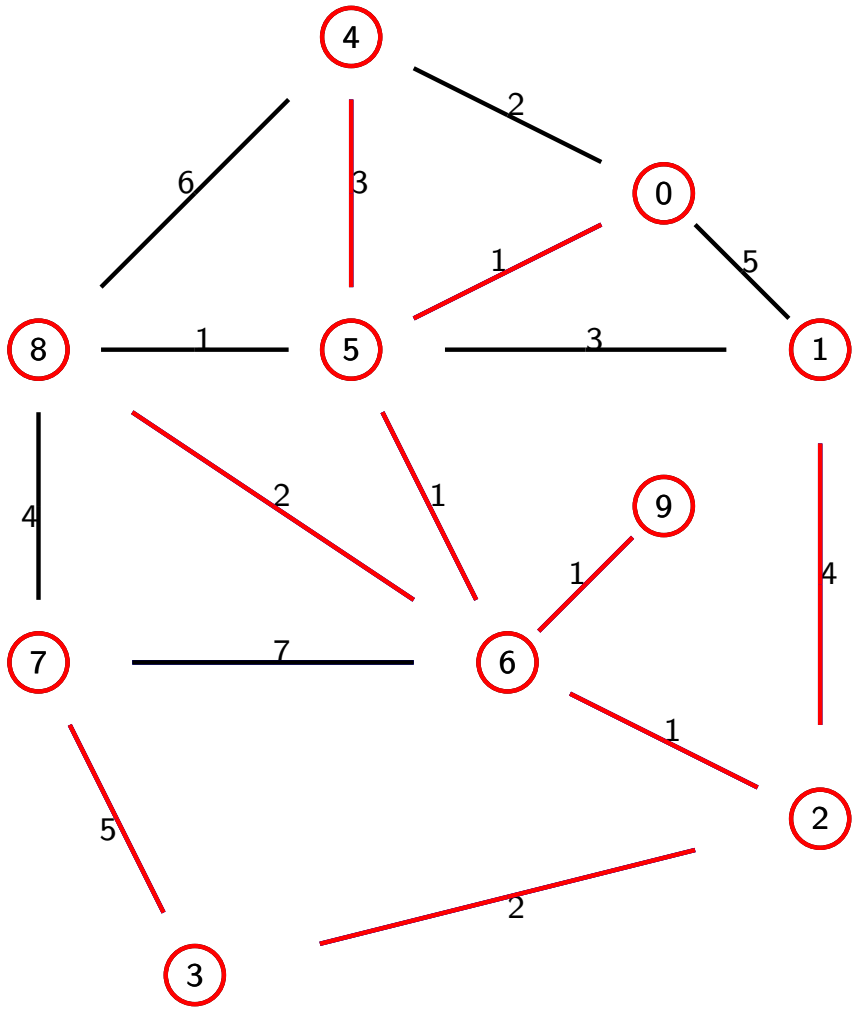
v	$h(v)$	$P(v)$
0	3	5
1	4	2
2	0	
3	2	2
4	5	5
5	2	6
6	1	2
7	7	3
8	3	6
9	2	6

Vybereme počáteční vrchol $v_0 = 2$. Jeho sousedům spočítáme vzdálenost a vybereme nejbližší vrchol $v = 6$. Jeho sousedům spočítáme vzdálenost od v_0 a vybereme nejbližší. Spočítáme vzdálenosti (u vrcholu 7 jsme našli kratší, *relaxujeme*), vybereme, není co relaxovat, tak vybereme a vybereme.



v	$h(v)$	$P(v)$
0	3	5
1	4	2
2	0	
3	2	2
4	5	5
5	2	6
6	1	2
7	7	3
8	3	6
9	2	6

Vybereme počáteční vrchol $v_0 = 2$. Jeho sousedům spočítáme vzdálenost a vybereme nejbližší vrchol $v = 6$. Jeho sousedům spočítáme vzdálenost od v_0 a vybereme nejbližší. Spočítáme vzdálenosti (u vrcholu 7 jsme našli kratší, *relaxujeme*), vybereme, není co relaxovat, tak vybereme a vybereme.



v	$h(v)$	$P(v)$
0	3	5
1	4	2
2	0	
3	2	2
4	5	5
5	2	6
6	1	2
7	7	3
8	3	6
9	2	6

Na předchozím slajdu jsme demonstrovali, *jak* Dijkstrův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest.

Vysvětlení – důkaz – budeme provádět indukcí. Podstatné při důkazu je, že ohodnocení hran je nezáporné.

Na předchozím slajdu jsme demonstrovali, *jak* Dijkstrův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest.

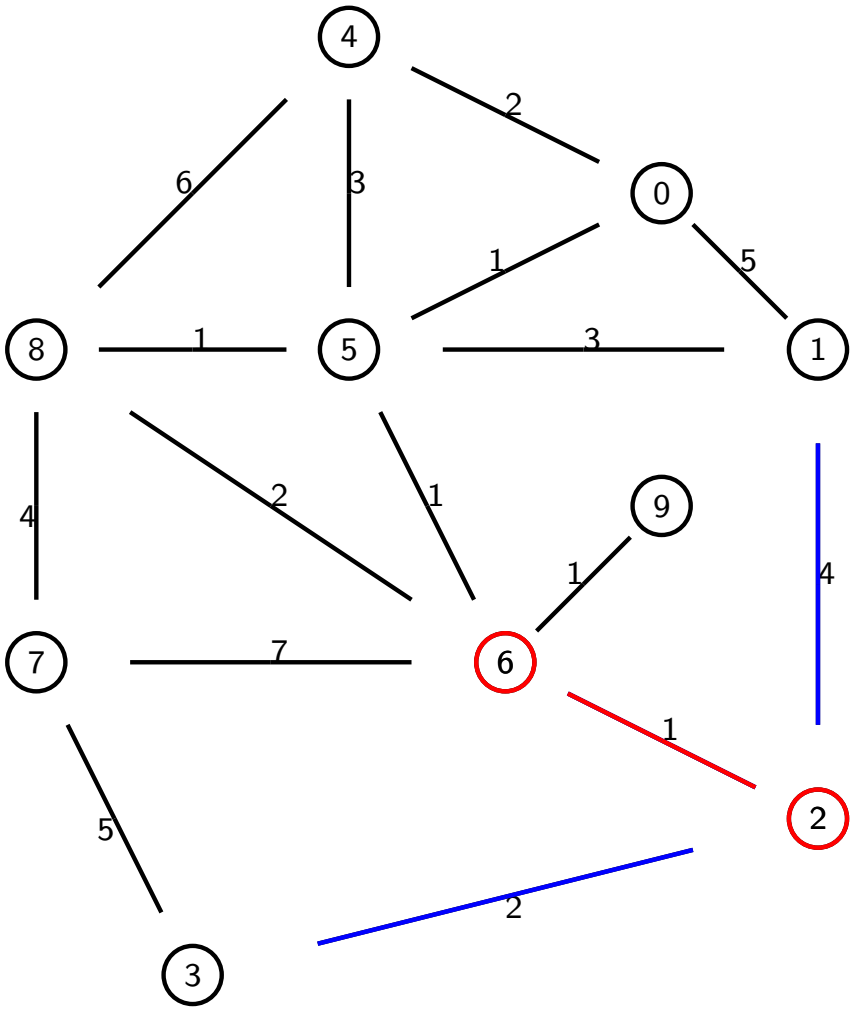
Vysvětlení – důkaz – budeme provádět indukcí. Podstatné při důkazu je, že ohodnocení hran je nezáporné.

Na předchozím slajdu jsme demonstrovali, *jak* Dijkstrův algoritmus funguje.

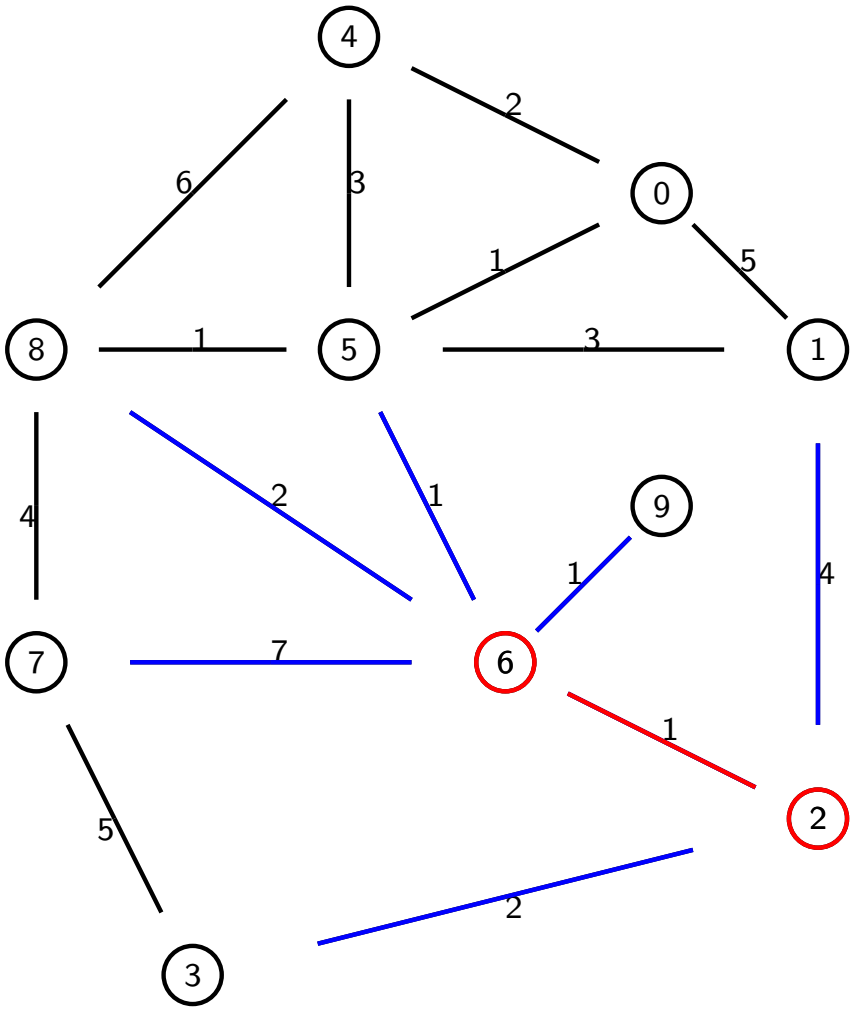
Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest.

Vysvětlení – důkaz – budeme provádět indukcí. Podstatné při důkazu je, že ohodnocení hran je nezáporné.

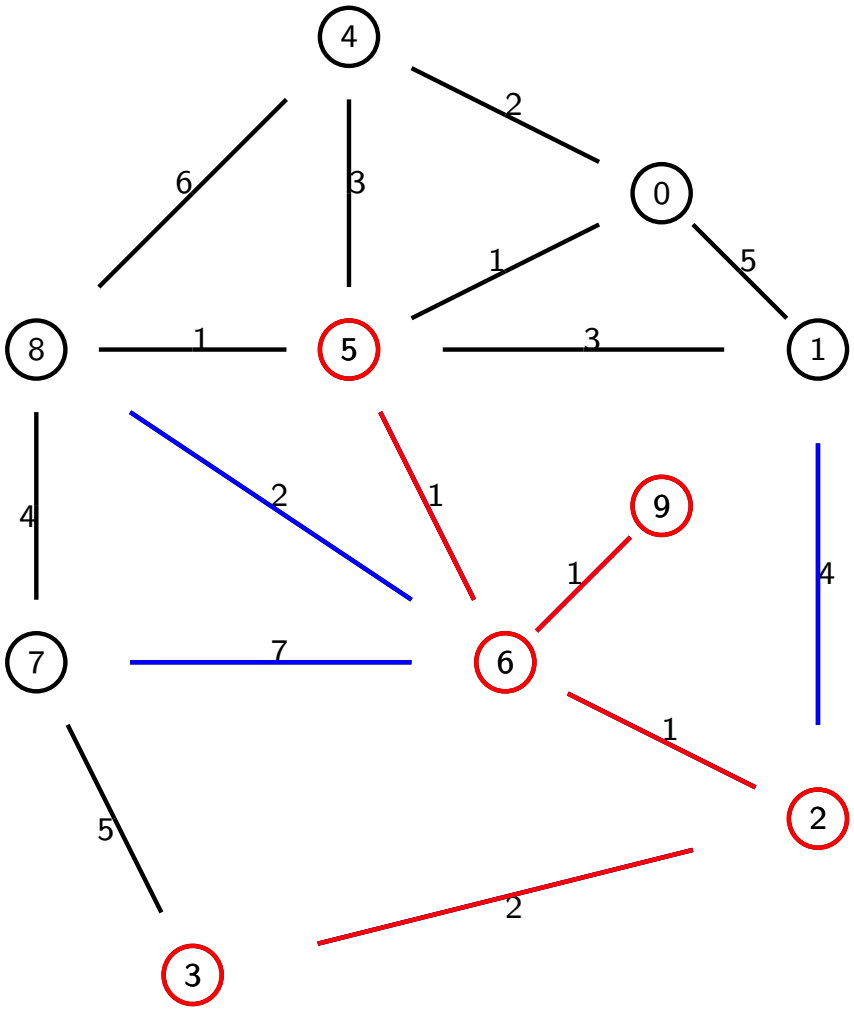
Vrchol na konci nejkratší hrany má nejkratší cestu po této jedné hraně, protože jakákoliv jiná cesta by měla delší cestu už po první hraně. Dále je dobré si všimnout, že nalezené nejkratší cesty tvoří postupně rostoucí strom a k tomuto stromu vždy připojíme vrchol nejbližší k počátečnímu.



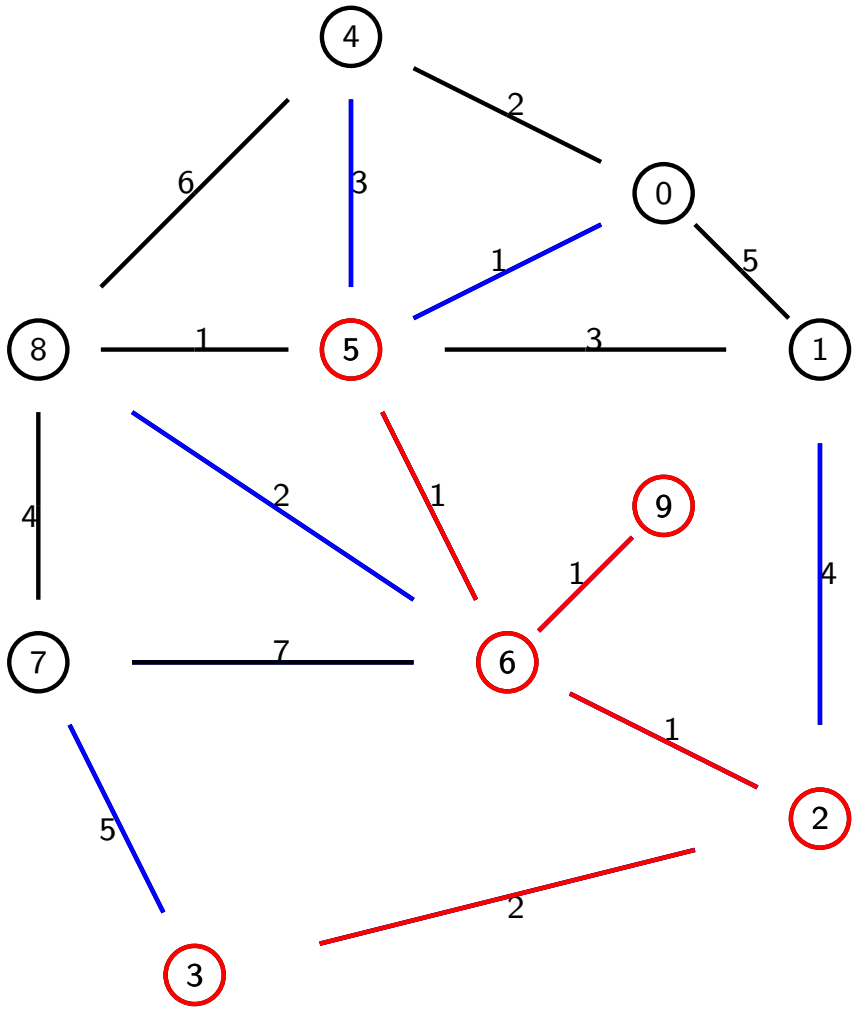
Vrchol na konci nejkratší hrany má nejkratší cestu po této jedné hraně, protože jakákoliv jiná cesta by měla delší cestu už po první hraně. Dále je dobré si všimnout, že nalezené nejkratší cesty tvoří postupně rostoucí strom a k tomuto stromu vždy připojíme vrchol nejbližší k počátečnímu.



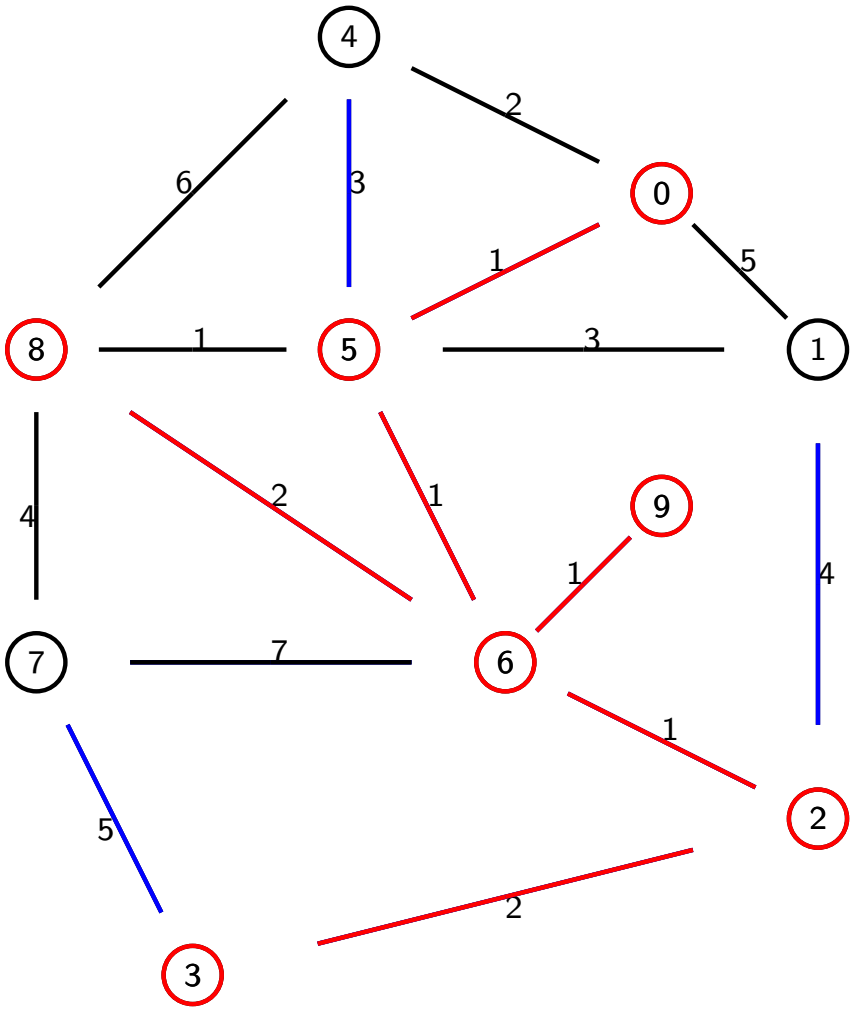
Vrchol na konci nejkratší hrany má nejkratší cestu po této jedné hraně, protože jakákoliv jiná cesta by měla delší cestu už po první hraně. Dále je dobré si všimnout, že nalezené nejkratší cesty tvoří postupně rostoucí strom a k tomuto stromu vždy připojíme vrchol nejbližší k počátečnímu.



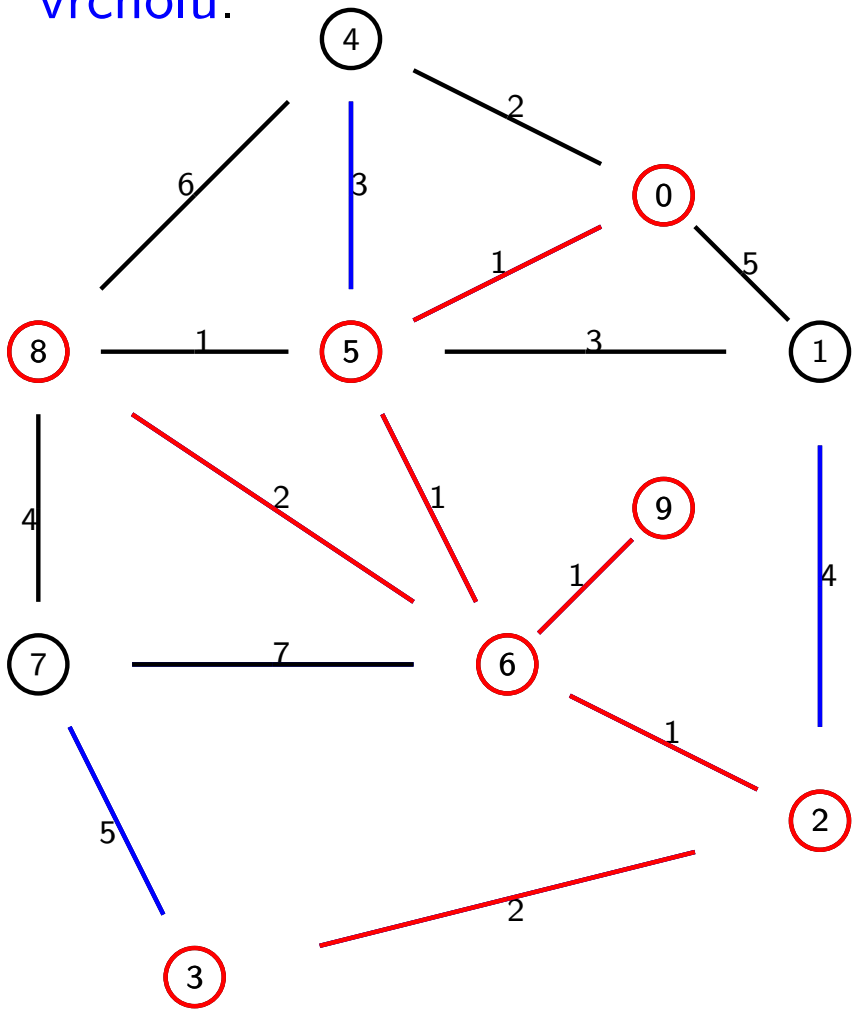
Vrchol na konci nejkratší hrany má nejkratší cestu po této jedné hraně, protože jakákoliv jiná cesta by měla delší cestu už po první hraně. Dále je dobré si všimnout, že nalezené nejkratší cesty tvoří postupně rostoucí strom a k tomuto stromu vždy připojíme vrchol nejbližší k počátečnímu.



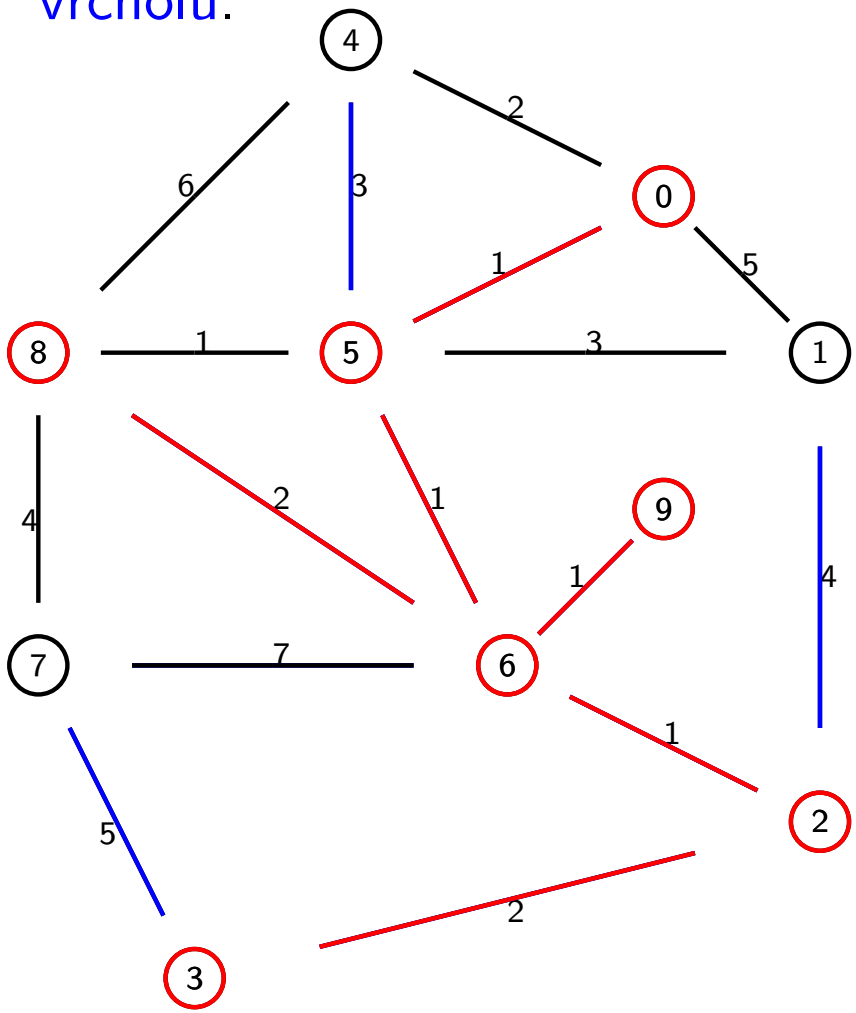
Vrchol na konci nejkratší hrany má nejkratší cestu po této jedné hraně, protože jakákoliv jiná cesta by měla delší cestu už po první hraně. Dále je dobré si všimnout, že nalezené nejkratší cesty tvoří postupně rostoucí strom a k tomuto stromu vždy připojíme vrchol nejbližší k počátečnímu.



Podstata důkazu indukcí spočívá v tom, že předpokládáme, že dosud nalezený strom obsahuje pouze nejkratší cesty a chceme dokázat, že tato vlastnost zůstane zachovaná i po připojení dalšího vrcholu.

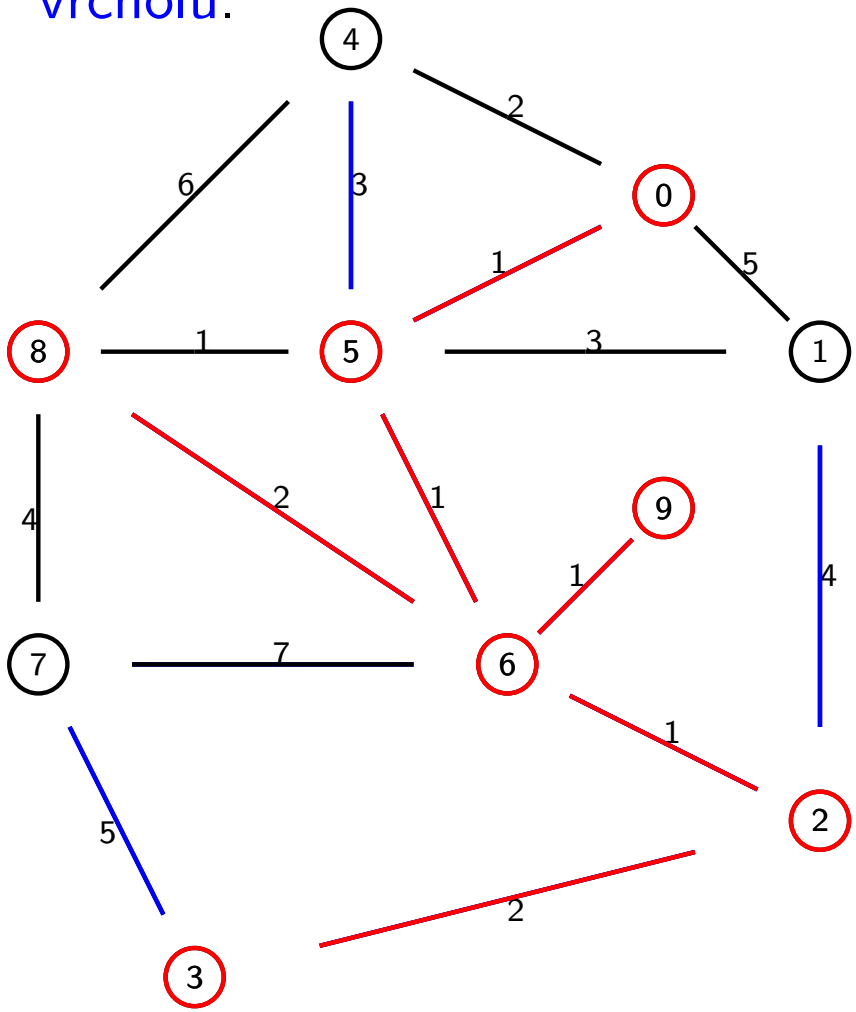


Podstata důkazu indukcí spočívá v tom, že předpokládáme, že dosud nalezený strom obsahuje pouze nejkratší cesty a chceme dokázat, že tato vlastnost zůstane zachovaná i po připojení dalšího vrcholu.



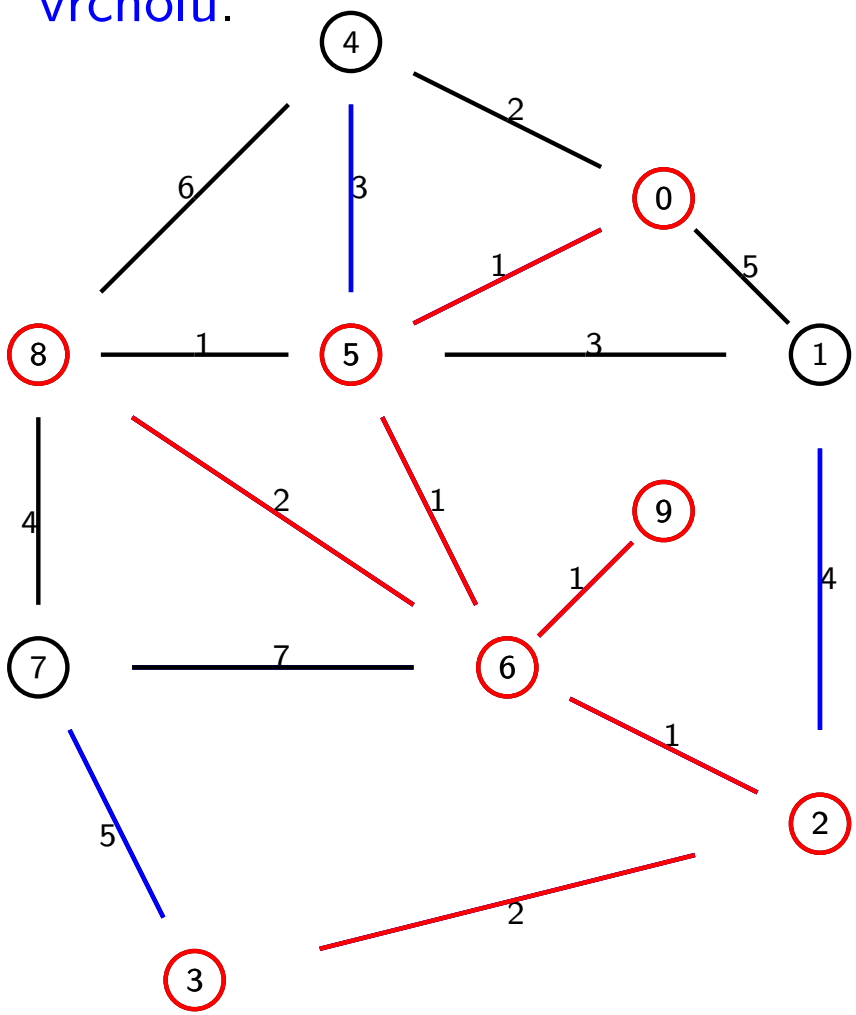
To je pravda, protože nejkratší cesta k novému vrcholu určitě existuje (cest do tohoto vrcholu je v grafu konečně mnoho a mezi nimi je jedna nebo více nejkratších, pokud jich je více, stačí nám nalézt jednu z nich) a je vidět, že první vrchol na této cestě, který není vrcholem dosud nalezeného stromu už leží dále od počátečního vrcholu než námi připojovaný vrchol. Proto cesta do našeho vrcholu která opustí strom jinde než těsně před ním, nemůže být kratší.

Podstata důkazu indukcí spočívá v tom, že předpokládáme, že dosud nalezený strom obsahuje pouze nejkratší cesty a chceme dokázat, že tato vlastnost zůstane zachovaná i po připojení dalšího vrcholu.



To je pravda, protože nejkratší cesta k novému vrcholu určitě existuje (cest do tohoto vrcholu je v grafu konečně mnoho a mezi nimi je jedna nebo více nejkratších, pokud jich je více, stačí nám nalézt jednu z nich) a je vidět, že první vrchol na této cestě, který není vrcholem dosud nalezeného stromu už leží dále od počátečního vrcholu než námi připojovaný vrchol. Proto cesta do našeho vrcholu která opustí strom jinde než těsně před ním, nemůže být kratší.

Podstata důkazu indukcí spočívá v tom, že předpokládáme, že dosud nalezený strom obsahuje pouze nejkratší cesty a chceme dokázat, že tato vlastnost zůstane zachovaná i po připojení dalšího vrcholu.



To je pravda, protože nejkratší cesta k novému vrcholu určitě existuje (cest do tohoto vrcholu je v grafu konečně mnoho a mezi nimi je jedna nebo více nejkratších, pokud jich je více, stačí nám nalézt jednu z nich) a je vidět, že první vrchol na této cestě, který není vrcholem dosud nalezeného stromu už leží dále od počátečního vrcholu než námi připojovaný vrchol. Proto cesta do našeho vrcholu která opustí strom jinde než těsně před ním, nemůže být kratší.

Spočítáme časovou složitost Dijkstrova algoritmu pro nejhorší případ. Během algoritmu každý vrchol jednou *otevřeme* (když mu poprvé spočítáme vzdálenost) a jednou *uzavřeme* (když má ze všech otevřených vrcholů nejmenší vzdálenost). Dále budeme některé vrcholy *relaxovat*, přitom každá hrana přispěje k nejvýše jedné relaxaci, počet relaxací je tedy nejvýše m. Časová složitost bude záviset na datové struktuře použité na nalezení nejbližšího vrcholu. Označíme N_i počet operací na *vložení vrcholu* do datové struktuty, N_x na *odebrání vrcholu* z datové struktuty, N_d na *relaxaci vrcholu*. Počet operací pak bude $nN_i + nN_x + mN_d$. Pro binární haldu je pro nejhorší případ $N_i = N_x = N_d = \log_2 n$. Celkový počet operací tedy je $(2n + m) \log_2 n$. Protože počet dosažitelných vrcholů je nejvýše m a konstanta 2 nás při zkoumání časové složitosti nezajímá, dostáváme výsledek: *asymptotická časová složitost Dijkstrova algoritmu při použití binární haldy je* $O(m \log_2 n)$. Při použití listu je $N_i = N_d = 1$, $N_x = n$. Počet operací tedy je $n + n^2 + m$. Zde zanedbáme n ve srovnání s n^2 a protože počet hran je $m \leq n(n-1)/2$, zanedbáme též m a dostaneme závěr: *asymptotická časová složitost Dijkstrova algoritmu při použití listu je* $O(n^2)$.

Spočítáme časovou složitost Dijkstrova algoritmu pro nejhorší případ. Během algoritmu každý vrchol jednou *otevřeme* (když mu poprvé spočítáme vzdálenost) a jednou *uzavřeme* (když má ze všech otevřených vrcholů nejmenší vzdálenost). Dále budeme některé vrcholy *relaxovat*, přitom každá hrana přispěje k nejvýše jedné relaxaci, počet relaxací je tedy nejvýše m. Časová složitost bude záviset na datové struktuře použité na nalezení nejbližšího vrcholu. Označíme N_i počet operací na *vložení vrcholu* do datové struktuty, N_x na *odebrání vrcholu* z datové struktuty, N_d na *relaxaci vrcholu*. Počet operací pak bude $nN_i + nN_x + mN_d$. Pro binární haldu je pro nejhorší případ $N_i = N_x = N_d = \log_2 n$. Celkový počet operací tedy je $(2n + m) \log_2 n$. Protože počet dosažitelných vrcholů je nejvýše m a konstanta 2 nás při zkoumání časové složitosti nezajímá, dostáváme výsledek: *asymptotická časová složitost Dijkstrova algoritmu při použití binární haldy je* $O(m \log_2 n)$. Při použití listu je $N_i = N_d = 1$, $N_x = n$. Počet operací tedy je $n + n^2 + m$. Zde zanedbáme n ve srovnání s n^2 a protože počet hran je $m \leq n(n-1)/2$, zanedbáme též m a dostaneme závěr: *asymptotická časová složitost Dijkstrova algoritmu při použití listu je* $O(n^2)$.

Spočítáme časovou složitost Dijkstrova algoritmu pro nejhorší případ. Během algoritmu každý vrchol jednou *otevřeme* (když mu poprvé spočítáme vzdálenost) a jednou *uzavřeme* (když má ze všech otevřených vrcholů nejmenší vzdálenost). Dále budeme některé vrcholy *relaxovat*, přitom každá hrana přispěje k nejvýše jedné relaxaci, počet relaxací je tedy nejvýše m. Časová složitost bude záviset na datové struktuře použité na nalezení nejbližšího vrcholu. Označíme N_i počet operací na *vložení vrcholu* do datové struktuty, N_x na *odebrání vrcholu* z datové struktuty, N_d na *relaxaci vrcholu*. Počet operací pak bude $nN_i + nN_x + mN_d$. Pro binární haldu je pro nejhorší případ $N_i = N_x = N_d = \log_2 n$. Celkový počet operací tedy je $(2n + m) \log_2 n$. Protože počet dosažitelných vrcholů je nejvýše m a konstanta 2 nás při zkoumání časové složitosti nezajímá, dostáváme výsledek: *asymptotická časová složitost Dijkstrova algoritmu při použití binární haldy je* $O(m \log_2 n)$. Při použití listu je $N_i = N_d = 1$, $N_x = n$. Počet operací tedy je $n + n^2 + m$. Zde zanedbáme n ve srovnání s n^2 a protože počet hran je $m \leq n(n-1)/2$, zanedbáme též m a dostaneme závěr: *asymptotická časová složitost Dijkstrova algoritmu při použití listu je* $O(n^2)$.

Spočítáme časovou složitost Dijkstrova algoritmu pro nejhorší případ. Během algoritmu každý vrchol jednou *otevřeme* (když mu poprvé spočítáme vzdálenost) a jednou *uzavřeme* (když má ze všech otevřených vrcholů nejmenší vzdálenost). Dále budeme některé vrcholy *relaxovat*, přitom každá hrana přispěje k nejvýše jedné relaxaci, počet relaxací je tedy nejvýše m. Časová složitost bude záviset na datové struktuře použité na nalezení nejbližšího vrcholu. Označíme N_i počet operací na *vložení vrcholu* do datové struktuty, N_x na *odebrání vrcholu* z datové struktuty, N_d na *relaxaci vrcholu*. Počet operací pak bude $nN_i + nN_x + mN_d$. Pro binární haldu je pro nejhorší případ $N_i = N_x = N_d = \log_2 n$. Celkový počet operací tedy je $(2n + m) \log_2 n$. Protože počet dosažitelných vrcholů je nejvýše m a konstanta 2 nás při zkoumání časové složitosti nezajímá, dostáváme výsledek: *asymptotická časová složitost Dijkstrova algoritmu při použití binární haldy je* $O(m \log_2 n)$. Při použití listu je $N_i = N_d = 1$, $N_x = n$. Počet operací tedy je $n + n^2 + m$. Zde zanedbáme n ve srovnání s n^2 a protože počet hran je $m \leq n(n-1)/2$, zanedbáme též m a dostaneme závěr: *asymptotická časová složitost Dijkstrova algoritmu při použití listu je* $O(n^2)$.

Spočítáme časovou složitost Dijkstrova algoritmu pro nejhorší případ. Během algoritmu každý vrchol jednou *otevřeme* (když mu poprvé spočítáme vzdálenost) a jednou *uzavřeme* (když má ze všech otevřených vrcholů nejmenší vzdálenost). Dále budeme některé vrcholy *relaxovat*, přitom každá hrana přispěje k nejvýše jedné relaxaci, počet relaxací je tedy nejvýše m. Časová složitost bude záviset na datové struktuře použité na nalezení nejbližšího vrcholu. Označíme N_i počet operací na *vložení vrcholu* do datové struktuty, N_x na *odebrání vrcholu* z datové struktuty, N_d na *relaxaci vrcholu*. Počet operací pak bude $nN_i + nN_x + mN_d$. Pro binární haldu je pro nejhorší případ $N_i = N_x = N_d = \log_2 n$. Celkový počet operací tedy je $(2n + m) \log_2 n$. Protože počet dosažitelných vrcholů je nejvýše m a konstanta 2 nás při zkoumání časové složitosti nezajímá, dostáváme výsledek: *asymptotická časová složitost Dijkstrova algoritmu při použití binární haldy je* $O(m \log_2 n)$. Při použití listu je $N_i = N_d = 1$, $N_x = n$. Počet operací tedy je $n + n^2 + m$. Zde zanedbáme n ve srovnání s n^2 a protože počet hran je $m \leq n(n-1)/2$, zanedbáme též m a dostaneme závěr: *asymptotická časová složitost Dijkstrova algoritmu při použití listu je* $O(n^2)$.

Spočítáme časovou složitost Dijkstrova algoritmu pro nejhorší případ. Během algoritmu každý vrchol jednou *otevřeme* (když mu poprvé spočítáme vzdálenost) a jednou *uzavřeme* (když má ze všech otevřených vrcholů nejmenší vzdálenost). Dále budeme některé vrcholy *relaxovat*, přitom každá hrana přispěje k nejvýše jedné relaxaci, počet relaxací je tedy nejvýše m. Časová složitost bude záviset na datové struktuře použité na nalezení nejbližšího vrcholu. Označíme N_i počet operací na *vložení vrcholu* do datové struktuty, N_x na *odebrání vrcholu* z datové struktuty, N_d na *relaxaci vrcholu*. Počet operací pak bude $nN_i + nN_x + mN_d$. Pro binární haldu je pro nejhorší případ $N_i = N_x = N_d = \log_2 n$. Celkový počet operací tedy je $(2n + m) \log_2 n$. Protože počet dosažitelných vrcholů je nejvýše m a konstanta 2 nás při zkoumání časové složitosti nezajímá, dostáváme výsledek: *asymptotická časová složitost Dijkstrova algoritmu při použití binární haldy je* $O(m \log_2 n)$. Při použití listu je $N_i = N_d = 1$, $N_x = n$. Počet operací tedy je $n + n^2 + m$. Zde zanedbáme n ve srovnání s n^2 a protože počet hran je $m \leq n(n-1)/2$, zanedbáme též m a dostaneme závěr: *asymptotická časová složitost Dijkstrova algoritmu při použití listu je* $O(n^2)$.

Spočítáme časovou složitost Dijkstrova algoritmu pro nejhorší případ. Během algoritmu každý vrchol jednou *otevřeme* (když mu poprvé spočítáme vzdálenost) a jednou *uzavřeme* (když má ze všech otevřených vrcholů nejmenší vzdálenost). Dále budeme některé vrcholy *relaxovat*, přitom každá hrana přispěje k nejvýše jedné relaxaci, počet relaxací je tedy nejvýše m. Časová složitost bude záviset na datové struktuře použité na nalezení nejbližšího vrcholu. Označíme N_i počet operací na *vložení vrcholu* do datové struktuty, N_x na *odebrání vrcholu* z datové struktuty, N_d na *relaxaci vrcholu*. Počet operací pak bude $nN_i + nN_x + mN_d$. Pro binární haldu je pro nejhorší případ $N_i = N_x = N_d = \log_2 n$. Celkový počet operací tedy je $(2n + m) \log_2 n$. Protože počet dosažitelných vrcholů je nejvýše m a konstanta 2 nás při zkoumání časové složitosti nezajímá, dostáváme výsledek: *asymptotická časová složitost Dijkstrova algoritmu při použití binární haldy je* $O(m \log_2 n)$. Při použití listu je $N_i = N_d = 1$, $N_x = n$. Počet operací tedy je $n + n^2 + m$. Zde zanedbáme n ve srovnání s n^2 a protože počet hran je $m \leq n(n-1)/2$, zanedbáme též m a dostaneme závěr: *asymptotická časová složitost Dijkstrova algoritmu při použití listu je* $O(n^2)$.

Spočítáme časovou složitost Dijkstrova algoritmu pro nejhorší případ. Během algoritmu každý vrchol jednou *otevřeme* (když mu poprvé spočítáme vzdálenost) a jednou *uzavřeme* (když má ze všech otevřených vrcholů nejmenší vzdálenost). Dále budeme některé vrcholy *relaxovat*, přitom každá hrana přispěje k nejvýše jedné relaxaci, počet relaxací je tedy nejvýše m. Časová složitost bude záviset na datové struktuře použité na nalezení nejbližšího vrcholu. Označíme N_i počet operací na *vložení vrcholu* do datové struktuty, N_x na *odebrání vrcholu* z datové struktuty, N_d na *relaxaci vrcholu*. Počet operací pak bude $nN_i + nN_x + mN_d$. Pro binární haldu je pro nejhorší případ $N_i = N_x = N_d = \log_2 n$. Celkový počet operací tedy je $(2n + m) \log_2 n$. Protože počet dosažitelných vrcholů je nejvýše m a konstanta 2 nás při zkoumání časové složitosti nezajímá, dostáváme výsledek: *asymptotická časová složitost Dijkstrova algoritmu při použití binární haldy je* $O(m \log_2 n)$. Při použití listu je $N_i = N_d = 1$, $N_x = n$. Počet operací tedy je $n + n^2 + m$. Zde zanedbáme n ve srovnání s n^2 a protože počet hran je $m \leq n(n-1)/2$, zanedbáme též m a dostaneme závěr: *asymptotická časová složitost Dijkstrova algoritmu při použití listu je* $O(n^2)$.