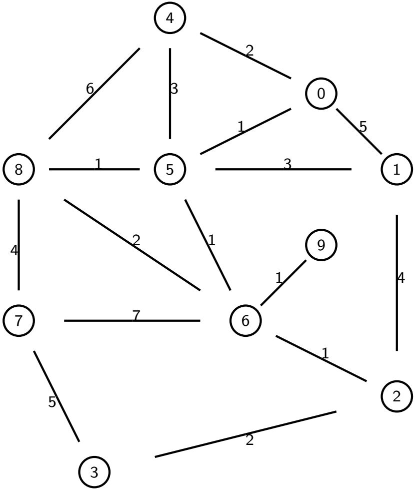
Bellman-Fordův algoritmus

text pro studenty učitelství na FP TUL

Martina Šimůnková

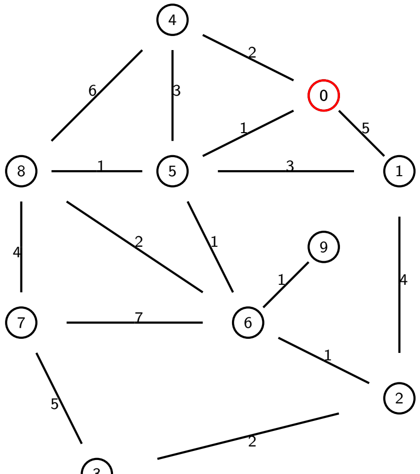
18. března 2025 (přeformulovala jsem poznámky k relaxaci)

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	∞	

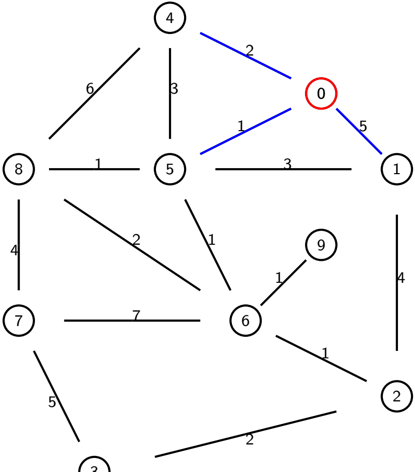
Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme vrchol $v = 1$ z fronty a zrelaxujeme ho (pojem relaxace vysvětlujeme v další prezentaci – znamená to přepočítat sousedům vzdálenosti a případně je vložit do fronty)
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme vrchol $v = 1$ z fronty a zrelaxujeme ho (pojem relaxace vysvětlujeme v další prezentaci – znamená to přepočítat sousedům vzdálenosti a případně je vložit do fronty)

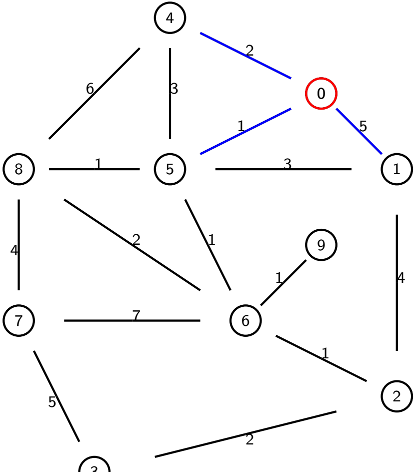
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme vrchol $v = 1$ z fronty a zrelaxujeme ho (pojem relaxace vysvětlujeme v další prezentaci – znamená to přepočítat sousedům vzdálenosti a případně je vložit do fronty)

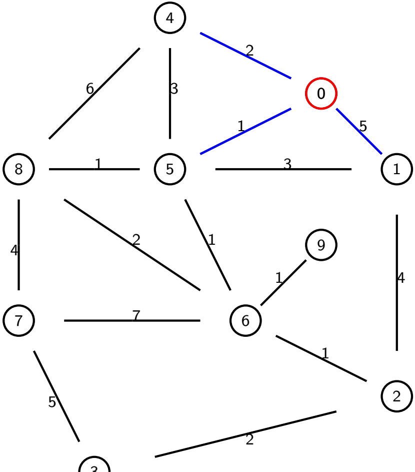
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme vrchol $v = 1$ z fronty a zrelaxujeme ho (pojem relaxace vysvětlujeme v další prezentaci – znamená to přepočítat sousedům vzdálenosti a případně je vložit do fronty)

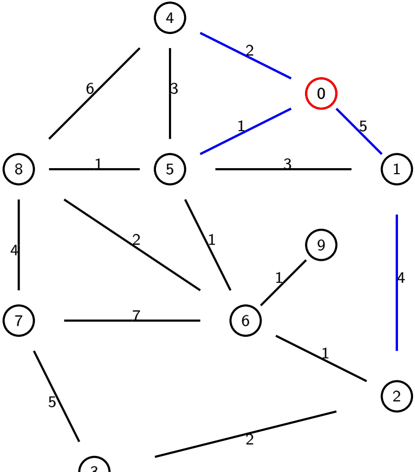
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme vrchol $v = 1$ z fronty a zrelaxujeme ho (pojem relaxace vysvětlujeme v další prezentaci – znamená to přepočítat sousedům vzdálenosti a případně je vložit do fronty)

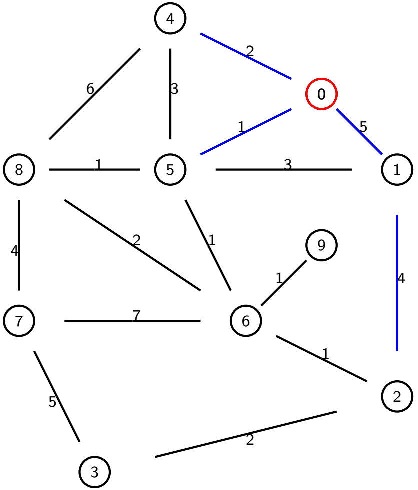
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

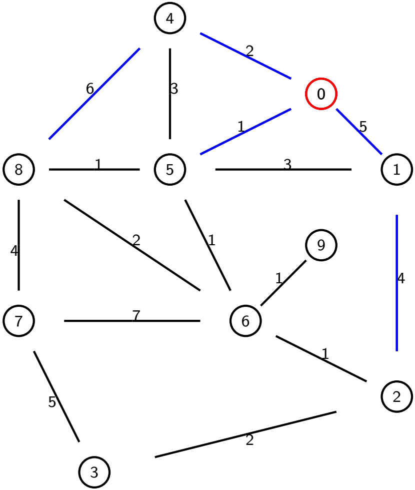
Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme vrchol $v = 4$ z fronty a zrelaxujeme ho.

Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

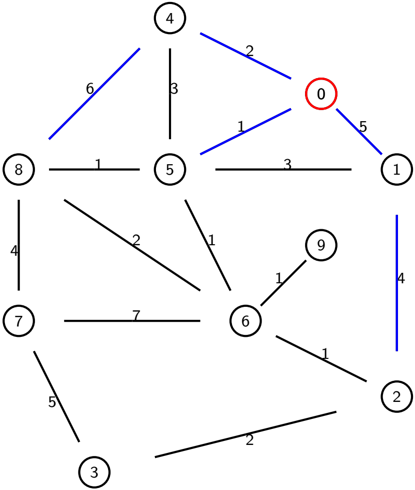
Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme vrchol $v = 4$ z fronty a zrelaxujeme ho.
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 5$ a zrelaxujeme ho.

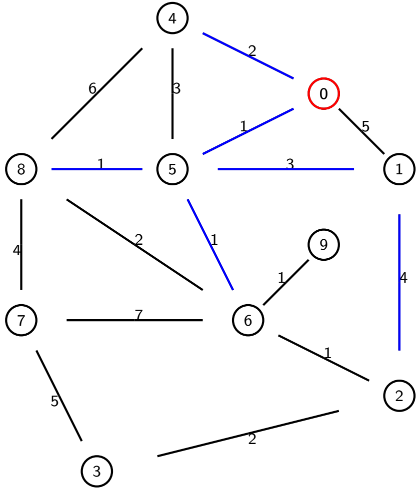
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 5$ a zrelaxujeme ho.

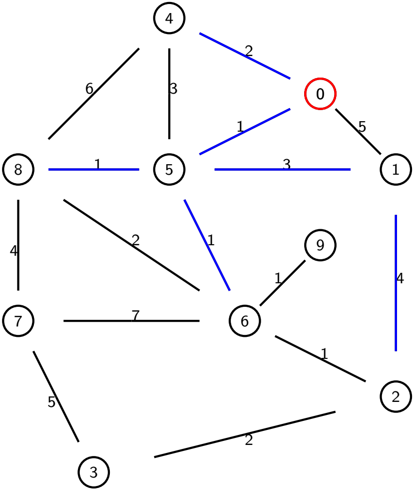
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 2$ a zrelaxujeme ho.

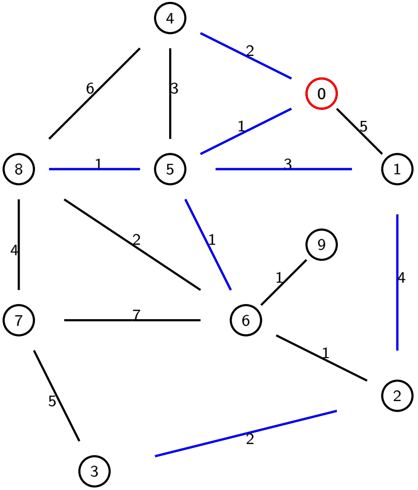
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 2$ a zrelaxujeme ho.

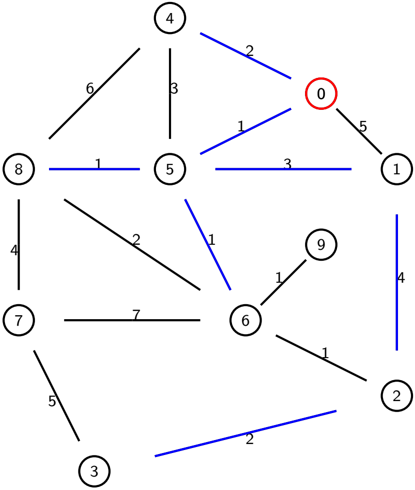
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	9	1
3	11	2
4	2	0
5	1	0
6	2	5
7	∞	
8	2	5
9	∞	

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 8$ a zrelaxujeme ho.

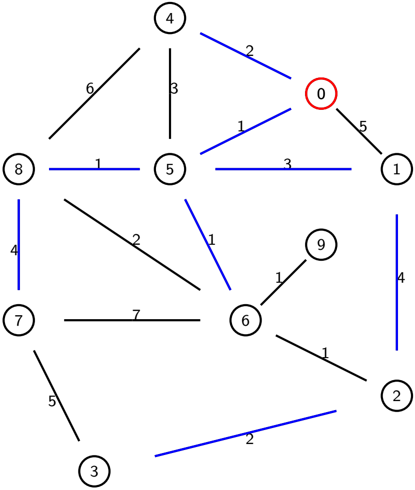
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	9	1
3	11	2
4	2	0
5	1	0
6	2	5
7	∞	
8	2	5
9	∞	

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 8$ a zrelaxujeme ho.

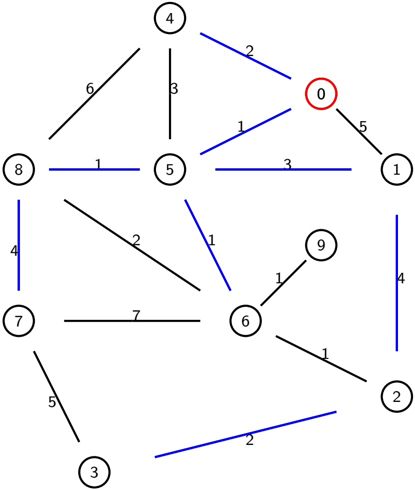
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	9	1
3	11	2
4	2	0
5	1	0
6	2	5
7	6	8
8	2	5
9	∞	

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 1$ a zrelaxujeme ho.

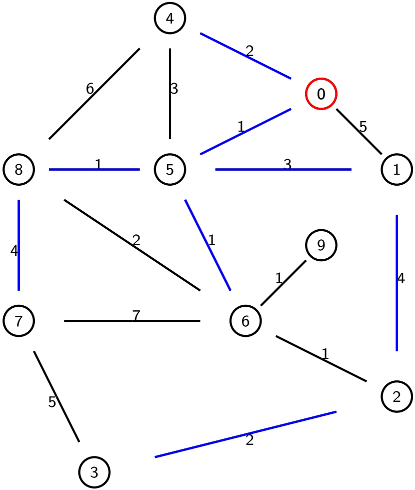
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	9	1
3	11	2
4	2	0
5	1	0
6	2	5
7	6	8
8	2	5
9	∞	

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 1$ a zrelaxujeme ho.

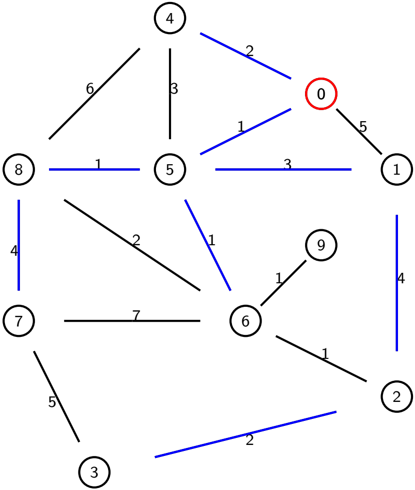
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	8	1
3	11	2
4	2	0
5	1	0
6	2	5
7	6	8
8	2	5
9	∞	

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 6$ a zrelaxujeme ho.

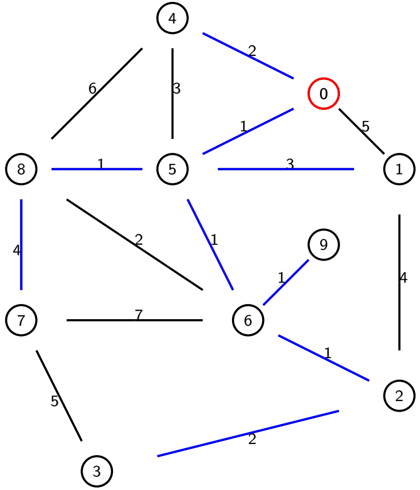
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	8	1
3	11	2
4	2	0
5	1	0
6	2	5
7	6	8
8	2	5
9	∞	

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 6$ a zrelaxujeme ho.

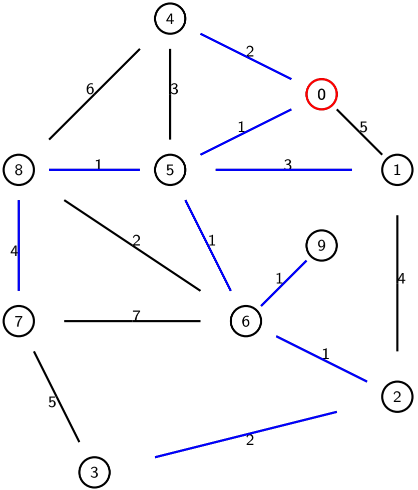
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	3	6
3	11	2
4	2	0
5	1	0
6	2	5
7	6	8
8	2	5
9	3	6

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 3$ není soused k relaxaci.

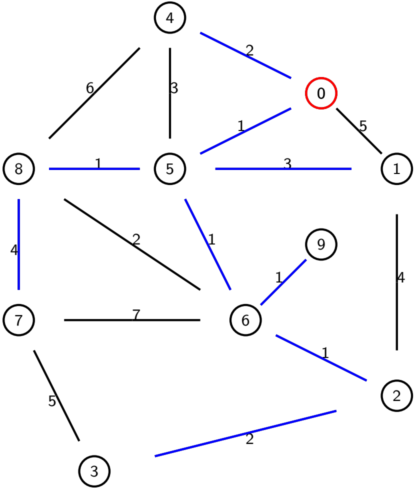
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	3	6
3	11	2
4	2	0
5	1	0
6	2	5
7	6	8
8	2	5
9	3	6

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 3$ není soused k relaxaci.

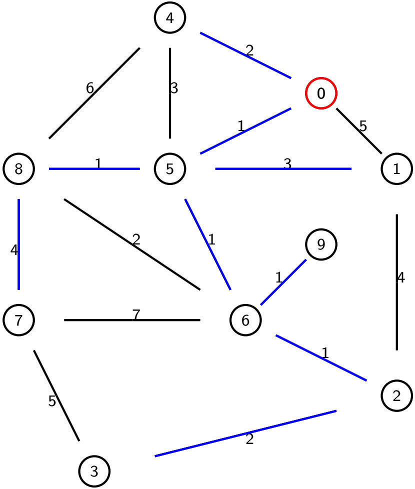
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	3	6
3	11	2
4	2	0
5	1	0
6	2	5
7	6	8
8	2	5
9	3	6

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 7$, není soused k relaxaci.

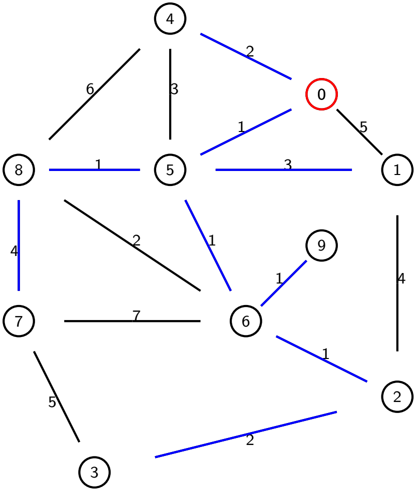
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	3	6
3	11	2
4	2	0
5	1	0
6	2	5
7	6	8
8	2	5
9	3	6

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 7$, není soused k relaxaci.

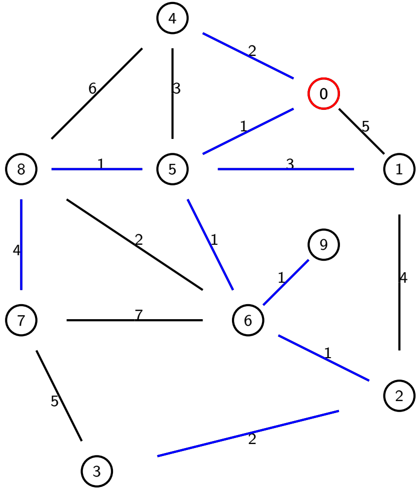
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	3	6
3	11	2
4	2	0
5	1	0
6	2	5
7	6	8
8	2	5
9	3	6

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 2$ a zrelaxujeme ho.

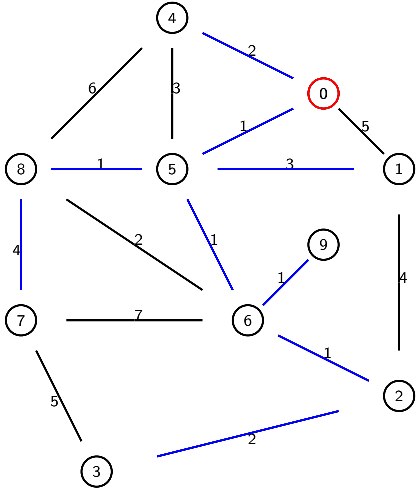
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



v	$h(v)$	$P(v)$
0	0	
1	4	5
2	3	6
3	11	2
4	2	0
5	1	0
6	2	5
7	6	8
8	2	5
9	3	6

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 2$ a zrelaxujeme ho.

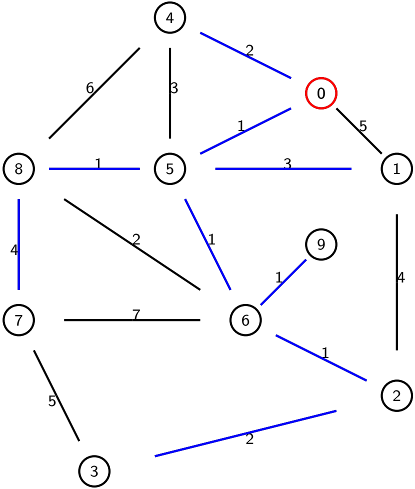
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 9$, není soused k relaxaci.

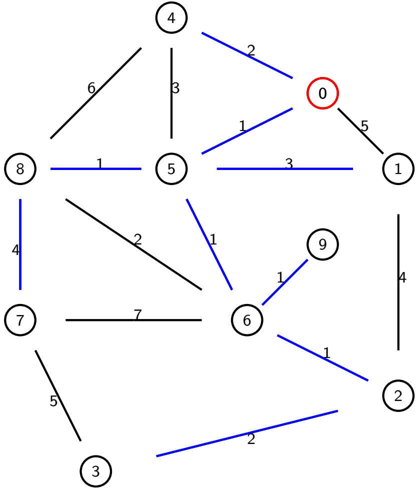
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 9$, není soused k relaxaci.

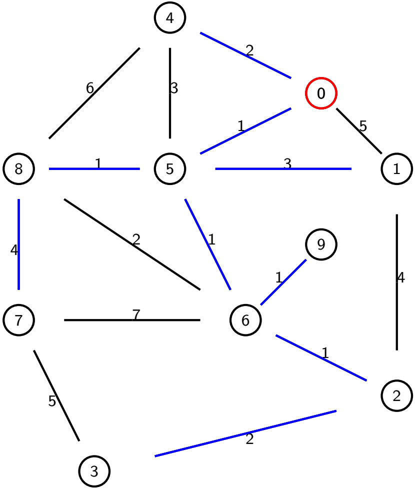
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 3$, není soused k relaxaci.

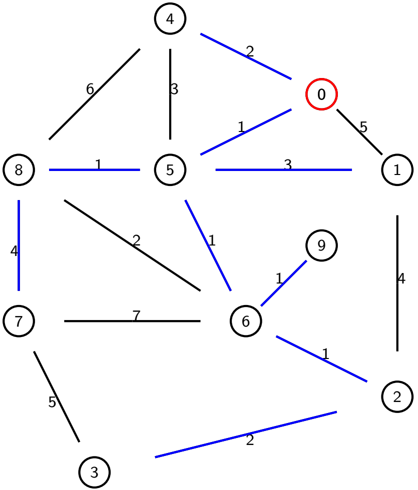
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Vyjmeme z fronty vrchol $v = 3$, není soused k relaxaci.

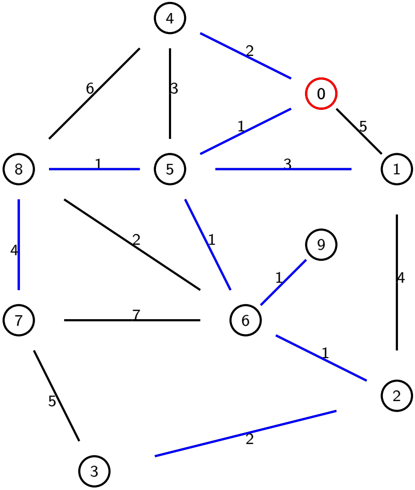
Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Vybereme počáteční vrchol $v_0 = 0$, sousedy vložíme do fronty, spočítáme jim vzdálenost od v_0 a nastavíme předchůdce $P(v)$ na nejkratší cestě. Fronta je prázdná, algoritmus skončil.

Fronta: 1 4 5 | 2 8 1 6 | 3 7 2 9 | 3



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

Na předchozím slajdu jsme demonstrovali, *jak* Bellman-Fordův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest. Zobrazíme si k tomu frontu, kterou jsme během algoritmu vytvářeli:

1 4 5 | 2 8 1 6 | 3 7 2 9 | 3

Zarážky |, které jsme do fronty vložili, vytvářejí pomyslné fáze algoritmu. V první fázi jsme do fronty vložili sousedy vrcholu v_0 a přiřadili jim délku cesty o jedné hraně. Ve druhé fázi jsme do fronty vložili vrcholy, do kterých vede cesta o dvou hranách a přiřadili jim délku nejkratší cesty o dvou hranách a vrcholům z první fáze zrelaxovali délku – máme pro ně délku nejkratší cesty o dvou a méně hranách. Ve třetí fázi jsme vrcholům spočítali délku nejkratší cesty o třech a méně hranách. Tak jsme pokračovali až do fáze čtvrté. V obecném případě cesty v grafu o n vrcholech obsahují nejvýše $n-1$ hran, zkáme tedy délky nejkratších cest nejpozději po $n-1$ fázích.

Na předchozím slajdu jsme demonstrovali, *jak* Bellman-Fordův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest. Zobrazíme si k tomu frontu, kterou jsme během algoritmu vytvářeli:

1 4 5 | 2 8 1 6 | 3 7 2 9 | 3

Zarážky |, které jsme do fronty vložili, vytvářejí pomyslné fáze algoritmu. V první fázi jsme do fronty vložili sousedy vrcholu v_0 a přiřadili jim délku cesty o jedné hraně. Ve druhé fázi jsme do fronty vložili vrcholy, do kterých vede cesta o dvou hranách a přiřadili jim délku nejkratší cesty o dvou hranách a vrcholům z první fáze zrelaxovali délku – máme pro ně délku nejkratší cesty o dvou a méně hranách. Ve třetí fázi jsme vrcholům spočítali délku nejkratší cesty o třech a méně hranách. Tak jsme pokračovali až do fáze čtvrté. V obecném případě cesty v grafu o n vrcholech obsahují nejvýše $n-1$ hran, zkáme tedy délky nejkratších cest nejpozději po $n-1$ fázích.

Na předchozím slajdu jsme demonstrovali, *jak* Bellman-Fordův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest. Zobrazíme si k tomu frontu, kterou jsme během algoritmu vytvářeli:

1 4 5 | 2 8 1 6 | 3 7 2 9 | 3

Zarážky |, které jsme do fronty vložili, vytvářejí pomyslné fáze algoritmu. V první fázi jsme do fronty vložili sousedy vrcholu v_0 a přiřadili jim délku cesty o jedné hraně. Ve druhé fázi jsme do fronty vložili vrcholy, do kterých vede cesta o dvou hranách a přiřadili jim délku nejkratší cesty o dvou hranách a vrcholům z první fáze zrelaxovali délku – máme pro ně délku nejkratší cesty o dvou a méně hranách. Ve třetí fázi jsme vrcholům spočítali délku nejkratší cesty o třech a méně hranách. Tak jsme pokračovali až do fáze čtvrté. V obecném případě cesty v grafu o n vrcholech obsahují nejvýše $n-1$ hran, zkáme tedy délky nejkratších cest nejpozději po $n-1$ fázích.

Na předchozím slajdu jsme demonstrovali, *jak* Bellman-Fordův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest. Zobrazíme si k tomu frontu, kterou jsme během algoritmu vytvářeli:

1 4 5 | 2 8 1 6 | 3 7 2 9 | 3

Zarážky |, které jsme do fronty vložili, vytvářejí pomyslné fáze algoritmu. V první fázi jsme do fronty vložili sousedy vrcholu v_0 a přiřadili jim délku cesty o jedné hraně. Ve druhé fázi jsme do fronty vložili vrcholy, do kterých vede cesta o dvou hranách a přiřadili jim délku nejkratší cesty o dvou hranách a vrcholům z první fáze zrelaxovali délku – máme pro ně délku nejkratší cesty o dvou a méně hranách. Ve třetí fázi jsme vrcholům spočítali délku nejkratší cesty o třech a méně hranách. Tak jsme pokračovali až do fáze čtvrté. V obecném případě cesty v grafu o n vrcholech obsahují nejvýše $n-1$ hran, zkáme tedy délky nejkratších cest nejpozději po $n-1$ fázích.

Na předchozím slajdu jsme demonstrovali, *jak* Bellman-Fordův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest. Zobrazíme si k tomu frontu, kterou jsme během algoritmu vytvářeli:

1 4 5 | 2 8 1 6 | 3 7 2 9 | 3

Zarážky |, které jsme do fronty vložili, vytvářejí pomyslné fáze algoritmu. V první fázi jsme do fronty vložili sousedy vrcholu v_0 a přiřadili jim délku cesty o jedné hraně. Ve druhé fázi jsme do fronty vložili vrcholy, do kterých vede cesta o dvou hranách a přiřadili jim délku nejkratší cesty o dvou hranách a vrcholům z první fáze zrelaxovali délku – máme pro ně délku nejkratší cesty o dvou a méně hranách. Ve třetí fázi jsme vrcholům spočítali délku nejkratší cesty o třech a méně hranách. Tak jsme pokračovali až do fáze čtvrté. V obecném případě cesty v grafu o n vrcholech obsahují nejvýše $n-1$ hran, zkáme tedy délky nejkratších cest nejpozději po $n-1$ fázích.

Na předchozím slajdu jsme demonstrovali, *jak* Bellman-Fordův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest. Zobrazíme si k tomu frontu, kterou jsme během algoritmu vytvářeli:

1 4 5 | 2 8 1 6 | 3 7 2 9 | 3

Zarážky |, které jsme do fronty vložili, vytvářejí pomyslné fáze algoritmu. V první fázi jsme do fronty vložili sousedy vrcholu v_0 a přiřadili jim délku cesty o jedné hraně. Ve druhé fázi jsme do fronty vložili vrcholy, do kterých vede cesta o dvou hranách a přiřadili jim délku nejkratší cesty o dvou hranách a vrcholům z první fáze zrelaxovali délku – máme pro ně délku nejkratší cesty o dvou a méně hranách. Ve třetí fázi jsme vrcholům spočítali délku nejkratší cesty o třech a méně hranách. Tak jsme pokračovali až do fáze čtvrté. V obecném případě cesty v grafu o n vrcholech obsahují nejvýše $n-1$ hran, zkáme tedy délky nejkratších cest nejpozději po $n-1$ fázích.

Na předchozím slajdu jsme demonstrovali, *jak* Bellman-Fordův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest. Zobrazíme si k tomu frontu, kterou jsme během algoritmu vytvářeli:

1 4 5 | 2 8 1 6 | 3 7 2 9 | 3

Zarážky |, které jsme do fronty vložili, vytvářejí pomyslné fáze algoritmu. V první fázi jsme do fronty vložili sousedy vrcholu v_0 a přiřadili jim délku cesty o jedné hraně. Ve druhé fázi jsme do fronty vložili vrcholy, do kterých vede cesta o dvou hranách a přiřadili jim délku nejkratší cesty o dvou hranách a vrcholům z první fáze zrelaxovali délku – máme pro ně délku nejkratší cesty o dvou a méně hranách. Ve třetí fázi jsme vrcholům spočítali délku nejkratší cesty o třech a méně hranách. Tak jsme pokračovali až do fáze čtvrté. V obecném případě cesty v grafu o n vrcholech obsahují nejvýše $n-1$ hran, zkáme tedy délky nejkratších cest nejpozději po $n-1$ fázích.

Na předchozím slajdu jsme demonstrovali, *jak* Bellman-Fordův algoritmus funguje.
Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest. Zobrazíme si k tomu frontu, kterou jsme během algoritmu vytvářeli:
1 4 5 | 2 8 1 6 | 3 7 2 9 | 3
Zarážky |, které jsme do fronty vložili, vytvářejí pomyslné fáze algoritmu. V první fázi jsme do fronty vložili sousedy vrcholu v_0 a přiřadili jim délku cesty o jedné hraně. Ve druhé fázi jsme do fronty vložili vrcholy, do kterých vede cesta o dvou hranách a přiřadili jim délku nejkratší cesty o dvou hranách a vrcholům z první fáze zrelaxovali délku – máme pro ně délku nejkratší cesty o dvou a méně hranách. Ve třetí fázi jsme vrcholům spočítali délku nejkratší cesty o třech a méně hranách. Tak jsme pokračovali až do fáze čtvrté. V obecném případě cesty v grafu o n vrcholech obsahují nejvýše $n-1$ hran, zkáme tedy délky nejkratších cest nejpozději po $n-1$ fázích.

Na předchozím slajdu jsme demonstrovali, *jak* Bellman-Fordův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest. Zobrazíme si k tomu frontu, kterou jsme během algoritmu vytvářeli:

1 4 5 | 2 8 1 6 | 3 7 2 9 | 3

Zarážky |, které jsme do fronty vložili, vytvářejí pomyslné fáze algoritmu. V první fázi jsme do fronty vložili sousedy vrcholu v_0 a přiřadili jim délku cesty o jedné hraně. Ve druhé fázi jsme do fronty vložili vrcholy, do kterých vede cesta o dvou hranách a přiřadili jim délku nejkratší cesty o dvou hranách a vrcholům z první fáze zrelaxovali délku – máme pro ně délku nejkratší cesty o dvou a méně hranách. Ve třetí fázi jsme vrcholům spočítali délku nejkratší cesty o třech a méně hranách. Tak jsme pokračovali až do fáze čtvrté. V obecném případě cesty v grafu o n vrcholech obsahují nejvýše $n-1$ hran, zkáme tedy délky nejkratších cest nejpozději po $n-1$ fázích.

Na předchozím slajdu jsme demonstrovali, *jak* Bellman-Fordův algoritmus funguje.

Vysvětlíme *proč* algoritmus funguje, tj. proč po jeho skončení získáme délky nejkratších cest. Zobrazíme si k tomu frontu, kterou jsme během algoritmu vytvářeli:

1 4 5 | 2 8 1 6 | 3 7 2 9 | 3

Zarážky |, které jsme do fronty vložili, vytvářejí pomyslné fáze algoritmu. V první fázi jsme do fronty vložili sousedy vrcholu v_0 a přiřadili jim délku cesty o jedné hraně. Ve druhé fázi jsme do fronty vložili vrcholy, do kterých vede cesta o dvou hranách a přiřadili jim délku nejkratší cesty o dvou hranách a vrcholům z první fáze zrelaxovali délku – máme pro ně délku nejkratší cesty o dvou a méně hranách. Ve třetí fázi jsme vrcholům spočítali délku nejkratší cesty o třech a méně hranách. Tak jsme pokračovali až do fáze čtvrté. V obecném případě cesty v grafu o n vrcholech obsahují nejvýše $n-1$ hran, zkáme tedy délky nejkratších cest nejpozději po $n-1$ fázích.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n-1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n+2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n-1)(n+2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m-1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n-1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n+2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n-1)(n+2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m-1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n - 1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n + 2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n - 1)(n + 2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m - 1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n - 1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n + 2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n - 1)(n + 2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m - 1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n - 1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n + 2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n - 1)(n + 2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m - 1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n-1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n+2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n-1)(n+2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m-1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n-1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n+2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n-1)(n+2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m-1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n-1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n+2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n-1)(n+2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m-1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n-1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n+2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n-1)(n+2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m-1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n-1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n+2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n-1)(n+2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m-1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n-1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n+2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n-1)(n+2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m-1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.

Spočítáme časovou složitost pro nejhorší případ. Budeme počítat operace v každé fázi a výsledek vynásobíme jejich počtem $n-1$. Je dobré si uvědomit, že vrcholy mohou být do fronty vkládány opakovaně, ale během jedné fáze je každý vrchol do fronty vložen nejvýše jednou. Odebrání vrcholu z fronty je jedna operace za kterou následuje relaxace jeho sousedů. Tady je dobré sčítat relaxace za celou fázi: Každá hrana se během jedné fáze podílí na nejvýše dvou relaxacích (vždy při odebrání svého koncového vrcholu z fronty). Při jedné fázi tedy proběhne nejvýše $2m$ relaxací. Jedna fáze má tedy $n+2m$ operací (odebrání až n vrcholů z fronty a až $2m$ relaxací). Operací tedy provedeme $(n-1)(n+2m)$ (nebo méně). Protože nás zajímá případ velkých vstupů, zanedbáme v prvním členu 1 proti n. Ve druhém členu můžeme zanedbat n proti $2m$, protože nedosažitelné vrcholy do fronty nevložíme (neexistuje pro ně žádná cesta) a dosažitelných vrcholů je nejvýše m (strom m hranách má $m-1$ vrcholů a pokud není stromem, má vrcholů méně). A protože nás multiplikativní (tj. násobící) konstanty při zkoumání časové složitosti nezajímají, vynecháme dvojku a uděláme závěr: asymptotická časová složitost Bellman-Fordova algoritmu je $O(nm)$.