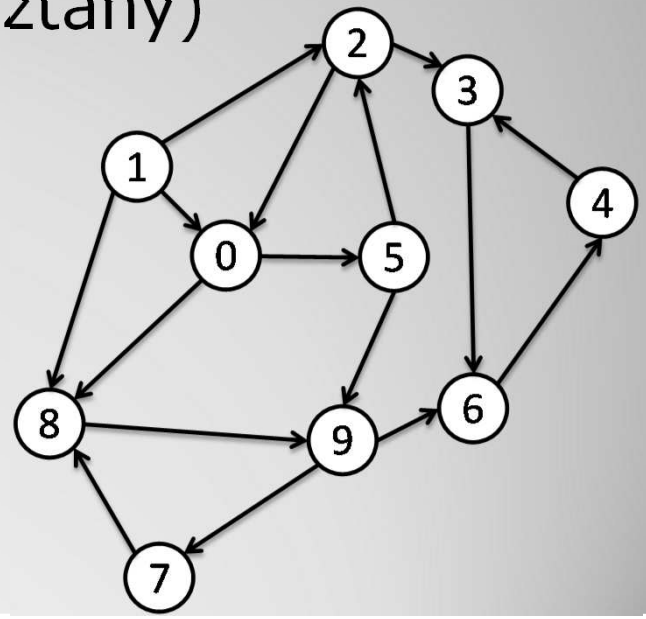
KIV/ZEP (PRJ2) - 2009

Grafy na 1001 způsobů

Grafy

- orientovaný vs. neorientovaný
- ohodnocený vs. neohodnocený
- základem mnoha algoritmů
  - plánování cesty pro postavičku
  - plánování cesty pro síťový paket
  - analýza obrazu (vyjadřuje vztahy)
  - speciálně: FEM
- reprezentace
  - matice
  - seznam

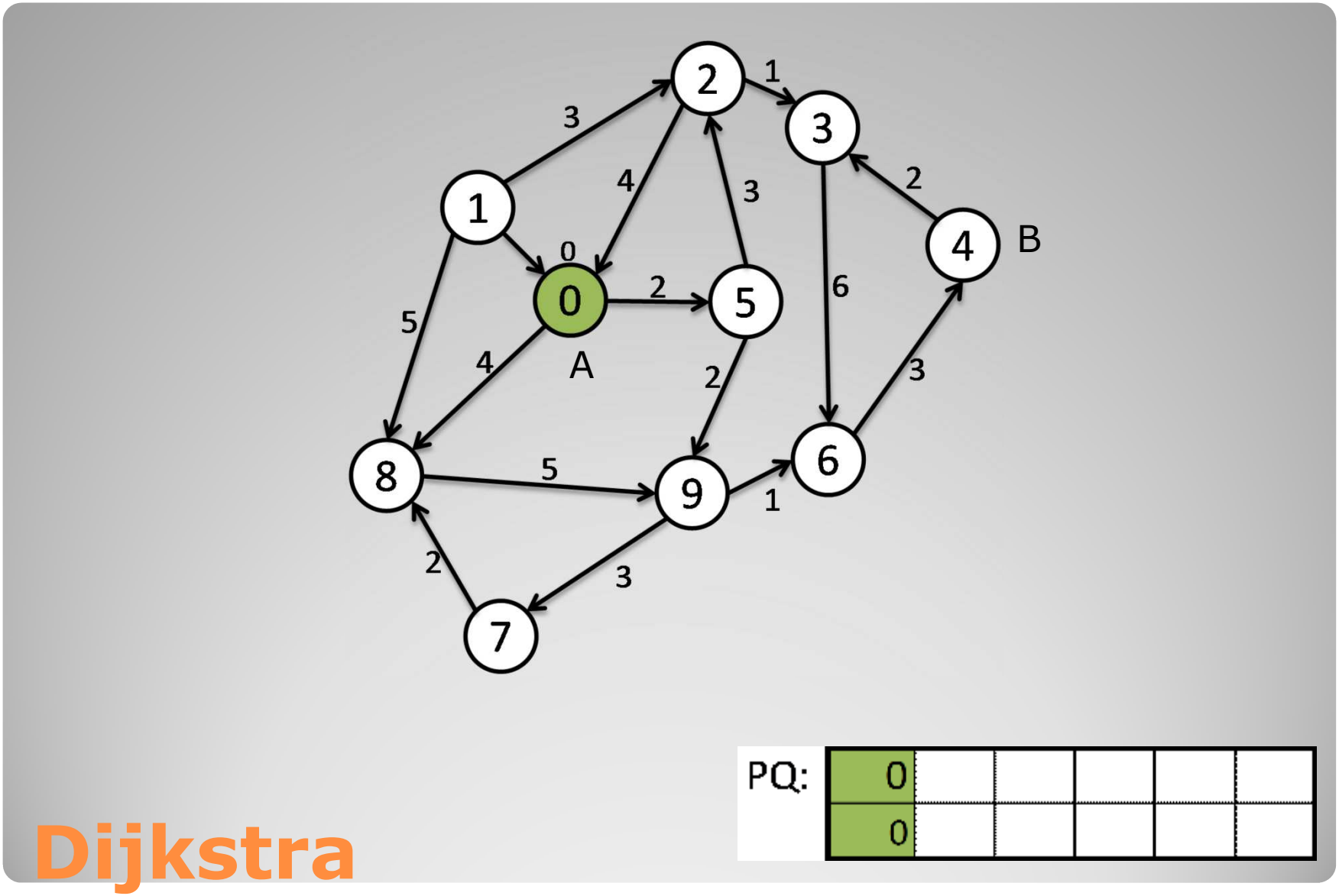


Hledání cesty v grafech

- nejkratší cesta (resp. nejméně cenná)
  - často se připouští cesta blízká nejkratší
- ohodnocení hran se může měnit
  - typicky strategické PC hry
- různé úlohy
  - najít cestu z uzlu A do B
  - najít cestu z uzlu A do ostatních
  - najít cestu z libovolného uzlu do libovolného
- základem je BFS

Dijkstra

- úloha: najít cestu z uzlu A do B
- ohodnocení nesmí být negativní
- uchovává cenu pro každý uzel
  - počátek: A = 0, všechny ostatní = INF
- BFS
  - fronta prioritní podle ceny uzlu
  - aktualizace ceny každého dosud nenavštíveného uzlu

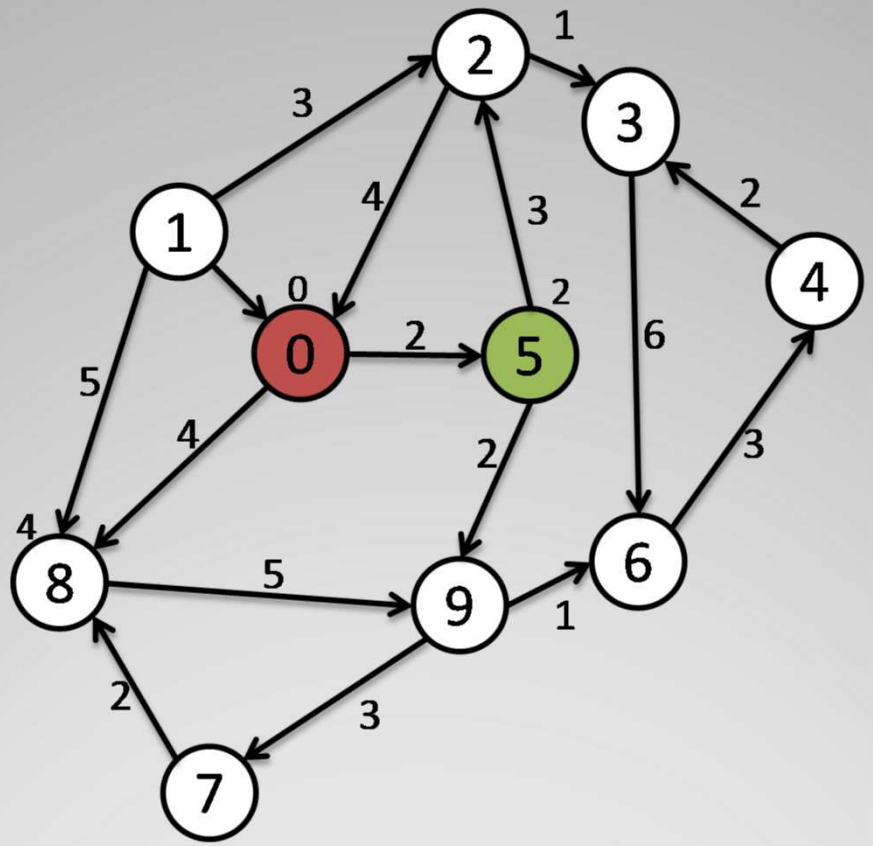


PQ:

0					
0					

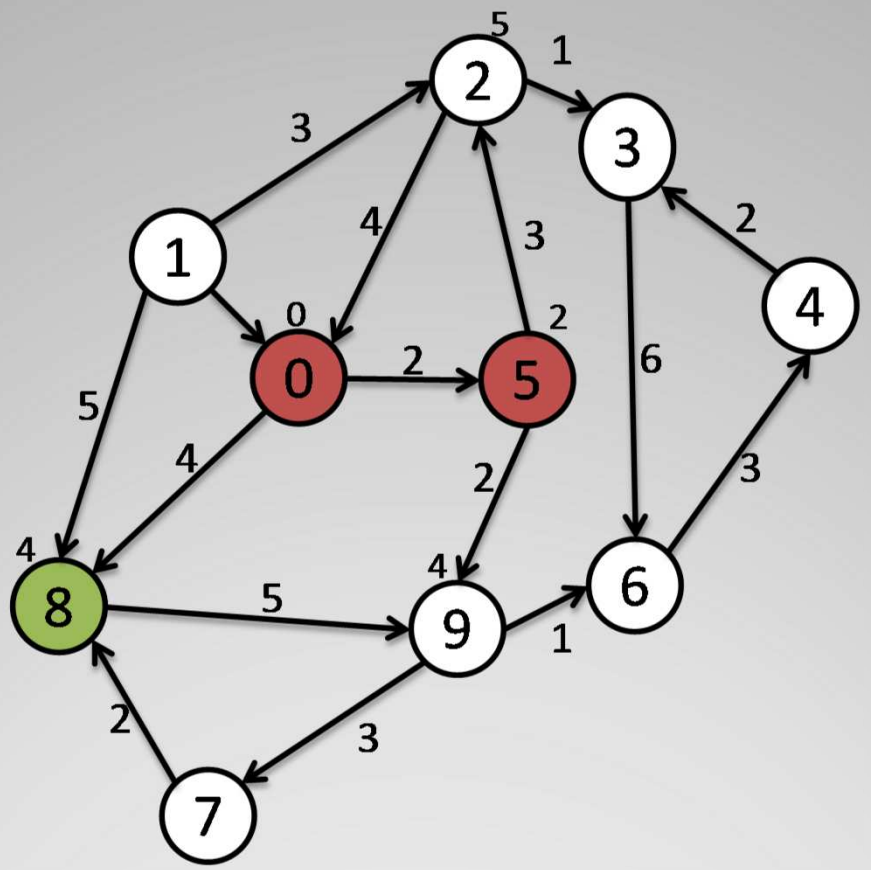
Dijkstra

Dijkstra

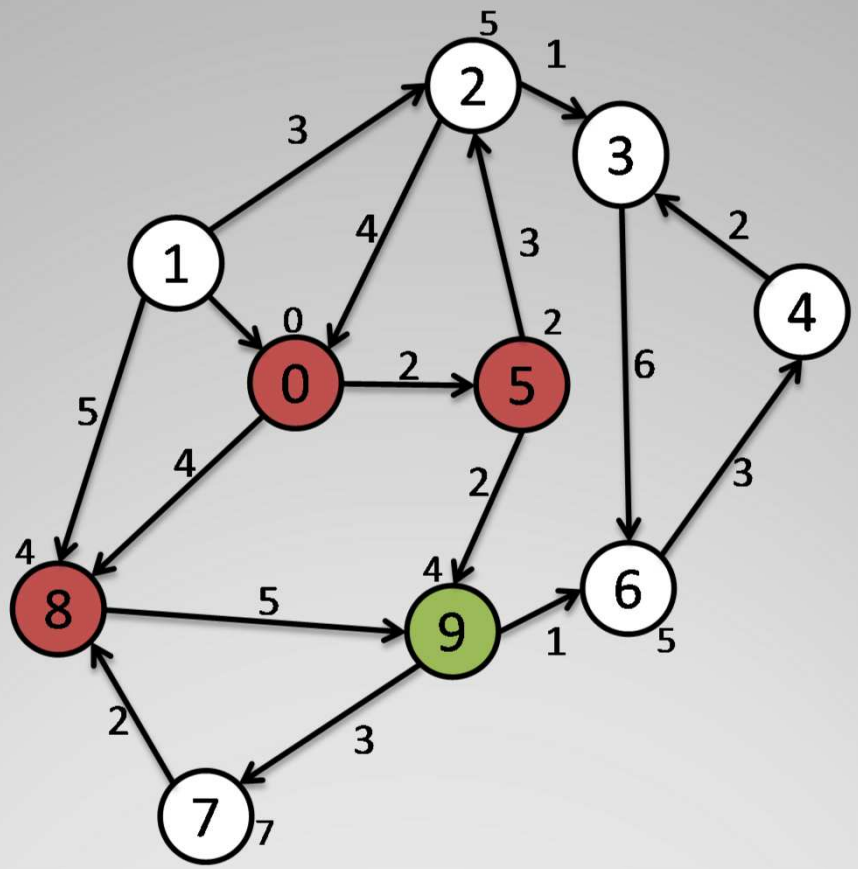


PQ:						
	5	8				
	2	4				

Dijkstra

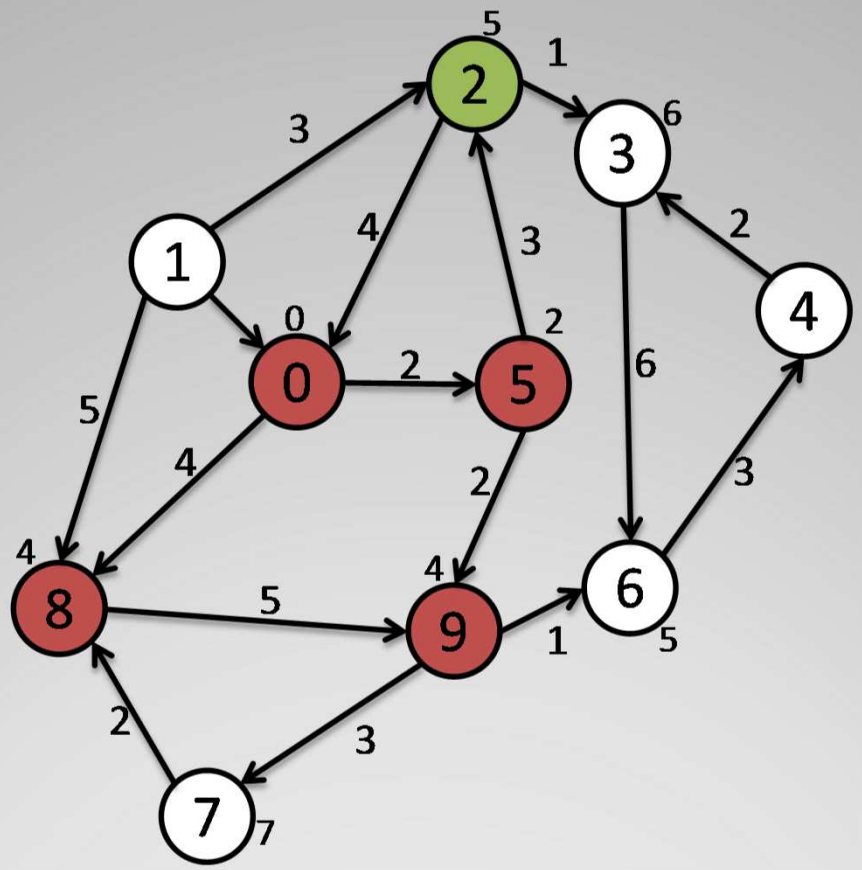


PQ:	8	9	2			
	4	4	5			



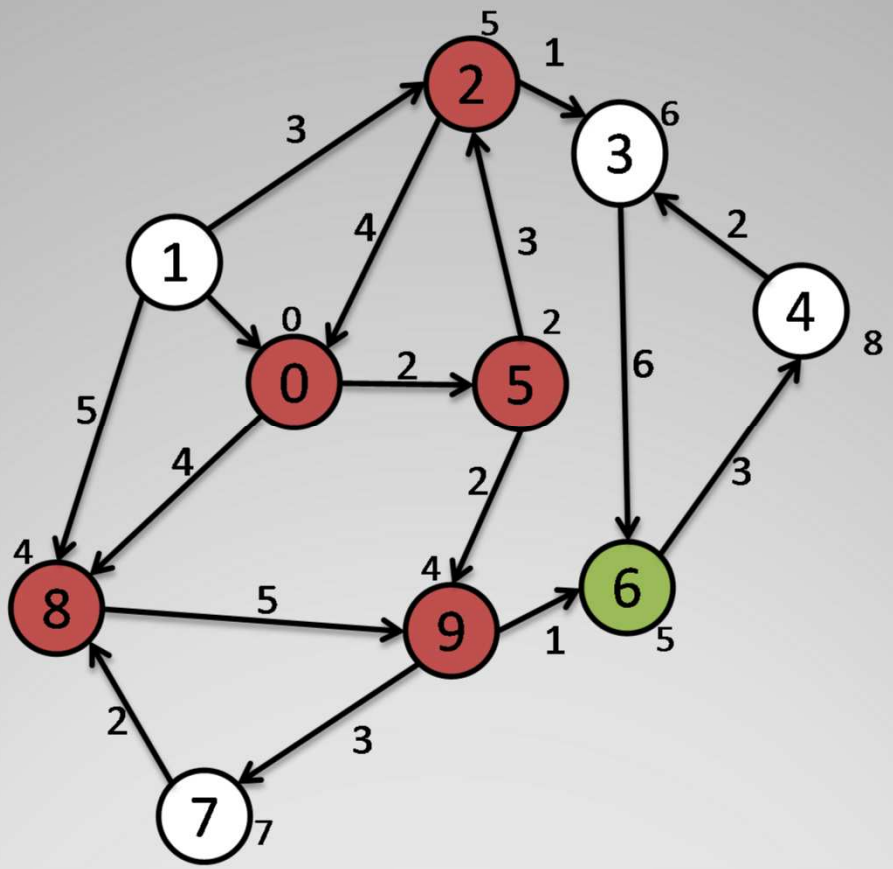
PQ:	9	2	6	7		
	4	5	5	7		

Dijkstra



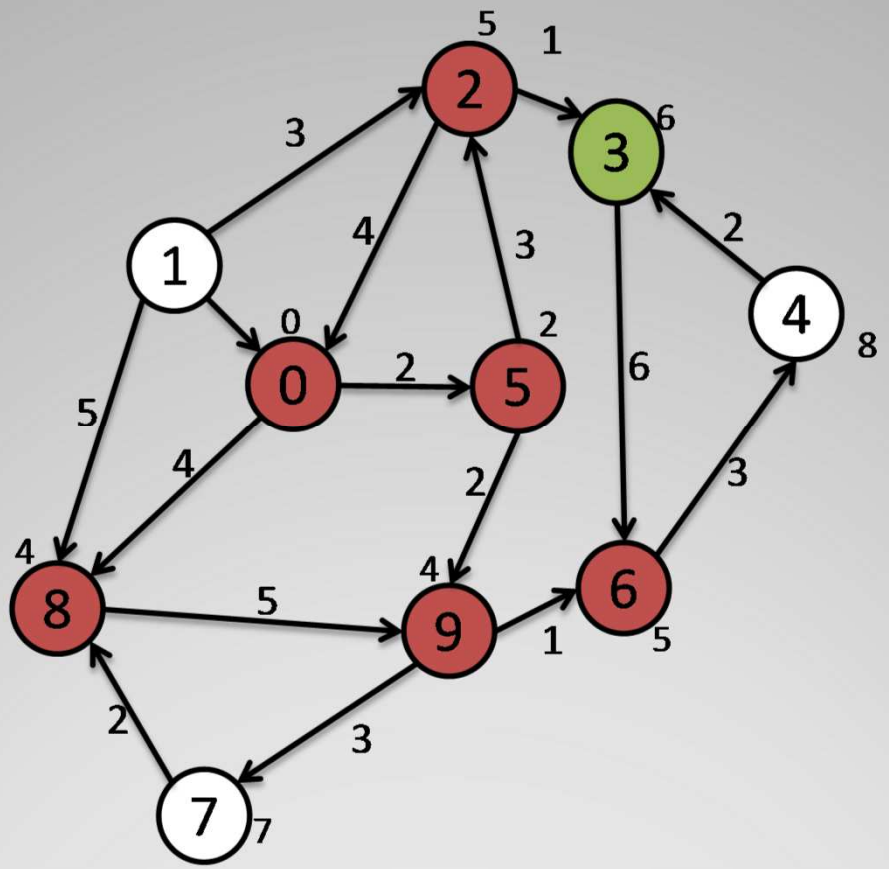
Dijkstra

PQ:	2	6	3	7		
	5	5	6	7		



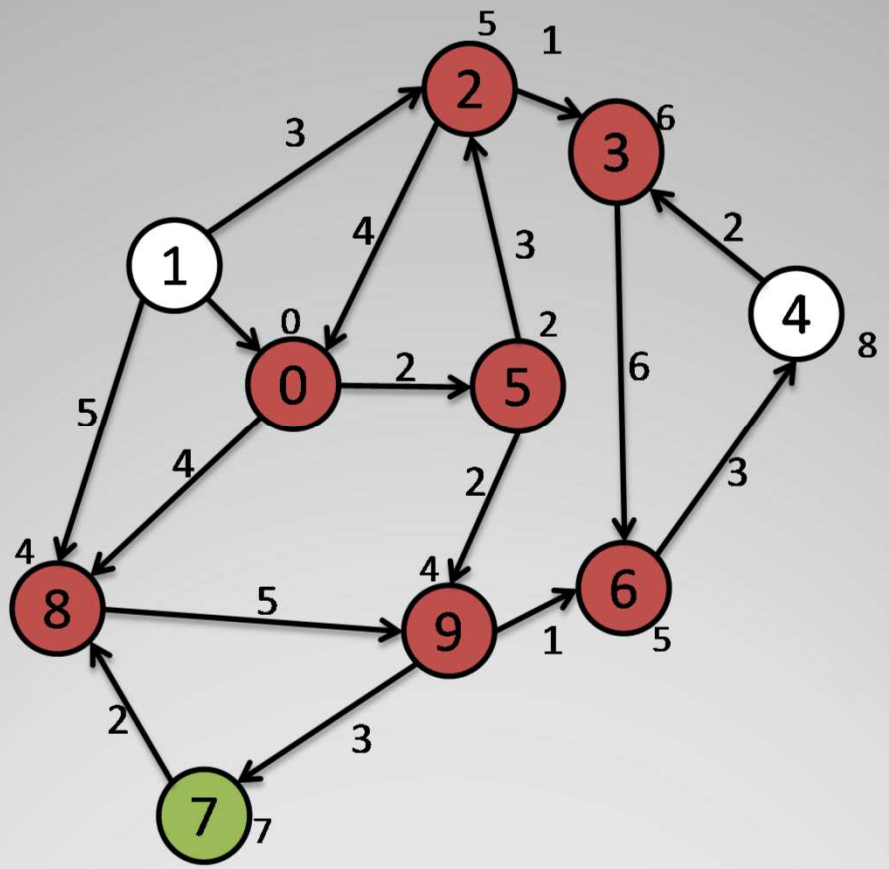
PQ:						
	6	3	7	4		
	5	6	7	8		

Dijkstra



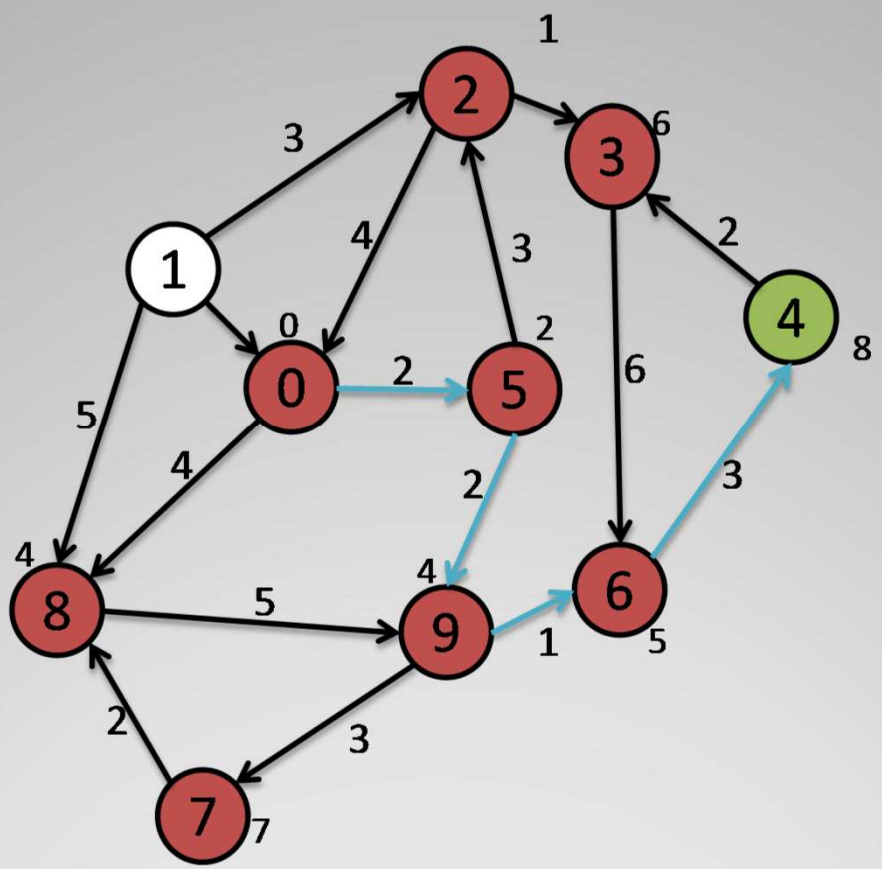
PQ:						
	3	7	4			
	6	7	8			

Dijkstra



PQ:						
	7	4				
	7	8				

Dijkstra



PQ:

4					
8					

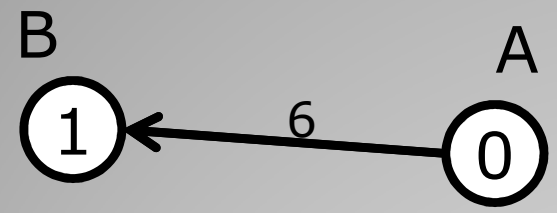
Dijkstra

Dijkstra

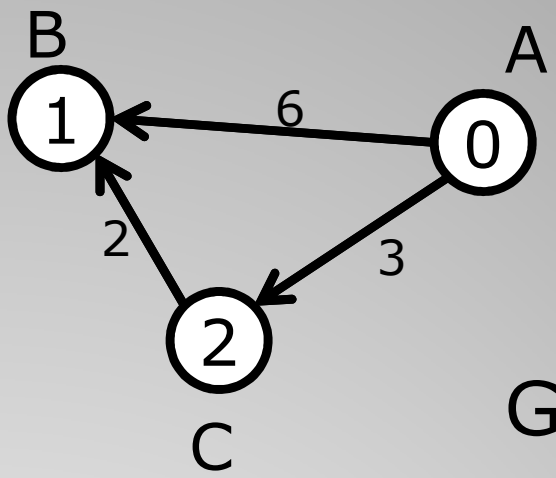
- Složitost:
  - počet uzlů grafu = V
  - počet hran grafu = E (často ~ V^2)
  - vložení nového uzlu do PQ = log(V)
  - vložení celkem = V·log(V)
  - aktualizace ceny uzlu v PQ = log(V)
  - aktualizace celkem = E·log(V)
  - → O((V + E)·log(V))

Floyd-Warshall

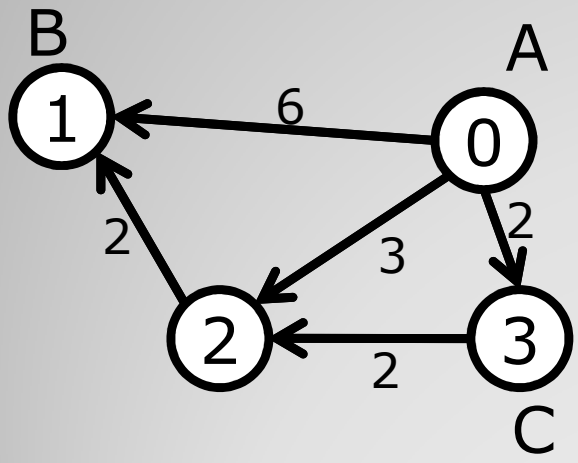
- úloha: najít cestu mezi všemi uzly
  - Dijkstra: $O(N \cdot (N + H) \cdot \log(N))$, tj. $O(N^3 \cdot \log(N))$ v nejhorším případě
  - lépe: Floyd-Warshal
- myšlenka F-W algoritmu
  - hledá se nejkratší cesta z uzlu A do uzlu B v podgrafech obsahujících 0 – N-2 dalších uzlů
  - na nejkratší cestě se každý uzel nalézá max. jednou
  - nejkratší cesta v podgrafu o *k* dalších uzlech buď
    - vede přes *k*-tý uzel C, a pak její cena odpovídá součtu ceny nejkratší cesty z A do C a ceny nejkratší cesty z C do B v podgrafu o *k*-1 dalších uzlů
    - nevede přes *k*-tý uzel C, a pak je totožná s nejkratší cestou v podgrafu o k-1 dalších uzlů



G^0



G^1



G^2

Floyd-Warshall

Floyd-Warshall

- d_{ij}^{k} – cena nejkratší cesty z *i*-tého uzlu do *j*-tého v *k*-tém podgrafu
- d_{ij}^{0} = ohodnocení hrany i-j (resp. ∞)
- $d_{ij}^{k} = \min(d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1})$
- d_{ij}^{N} = cena nejkratší cesty z *i*-tého uzlu do *j*-tého v celém grafu
- lze řešit rekurzí, ale není optimální
  - spoustu věcí počítáno duplicitně
- $\rightarrow$ začneme od k=0 a budeme mezivýsledky ukládat

```
//nechť uzly grafu číslovány 0..N-1
//inicializace
for (int i = 0; i < N; i++) {
  for (int j = 0; j < N; j++) {
     d[0, i, j] = ohodnoceni(i,j);
     pred[i,j] = NULL;
  }
}
```

Floyd-Warshall

```
//výpočet jednotlivých podgrafů
for (int k = 0; k < N; k++) {
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      double c = d[k,i,k] + d[k,k,j]; //cesta přes k
      if (c < d[k,i,j]) {
            d[k+1,i,j] = c; pred[i,j] = k;
      } else d[k+1,i,j] = d[k,i,j];
    }
  }
}
```

Floyd-Warshall

- cestu lze extrahovat z pred[i,j] rekurentně

```
void path(i,j) {
  if (pred[i,j] == NULL)
        pridej_hranu(i,j);     //hrana na ceste
  else {     //cesta prochazi uzlem pred[i,j]
        path(i, pred[i,j]);
        path(pred[i,j], k);
  }
}
```

Floyd-Warshall

Floyd-Warshall

- složitost časová: $O(N^3)$
- složitost paměťová: $O(N^3)$
  - jak ji snížit?

Floyd-Warshall

- zajímají nás cesty v plném grafu → není třeba ukládat ceny podgrafů, tj. lze $O(N^2)$

```
//výpočet jednotlivých podgrafů
for (int k = 0; k < N; k++) {
   for (int i = 0; i < N; i++) {
     for (int j = 0; j < N; j++) {
       double c = d[i,k] + d[k,j]; //cesta přes k
       if (c < d[i,j]) {
                d[i,j] = c; pred[i,j] = k;
       }
     }
   }
}
```

Porovnávání grafů

- přesné vs. nepřesné
- se stejným počtem vrcholů vs. s různým počtem vrcholů
- využití
  - rozpoznávání (včetně 3D – Reebův graf),
  - analýza tvarů
  - plánování
  - deformace na základě skeletonu

- pro každý uzel grafu G1 dáno, jak moc je podobný jednotlivým uzlům grafu G2
  - heuristika podobnosti
    - záleží na konkrétní úloze
    - např. stupeň uzlu, velikost podstromů, vzdálenost v Euklidovském prostoru apod.
- chceme přiřadit každému uzlu grafu G1 právě jeden uzel z G2 tak, aby celková podobnost byla maximální

Hungarian marriage (Munkres)

- Příklad: 4 dělníci a 4 úlohy k vykonání
  - každý dělník potřebuje jiný čas
  - jak přidělit úlohy tak, aby celkový čas provedení všech úloh byl minimální?
  - brute-force: vygenerovat všechny možnosti a porovnat je → O(n!)

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	14	5	8	5
Dělník 2	2	12	6	5
Dělník 3	7	8	3	9
Dělník 4	2	4	6	10

Hungarian marriage (Munkres)

- Od každého řádku odečti jeho minimum
  - přidání konstanty nezmění výsledné přiřazení

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	14	5	8	5
Dělník 2	2	12	6	5
Dělník 3	7	8	3	9
Dělník 4	2	4	6	10

Hungarian marriage (Munkres)

- Od každého řádku odečti jeho minimum
  - přidání konstanty nezmění výsledné přiřazení

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	9	0	3	0
Dělník 2	0	10	4	3
Dělník 3	4	5	0	6
Dělník 4	0	2	4	8

Hungarian marriage (Munkres)

- Nalezni největší množinu nezávislých nul
  - dvě nuly jsou nezávislé, pokud leží v různých řádcích a různých sloupcích
  - odpovídá minimálnímu počtu čar potřebných k pokrytí všech nul v tabulce

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	9	0	3	0
Dělník 2	0	10	4	3
Dělník 3	4	5	0	6
Dělník 4	0	2	4	8

Hungarian marriage (Munkres)

- Jak nalézt čáry?
  - označ * každou nulu, pro kterou v tabulce není 0* na tomtéž řádku ani sloupci
  - přikryj čárou každý sloupec s 0*

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	9	0*	3	0
Dělník 2	0*	10	4	3
Dělník 3	4	5	0*	6
Dělník 4	0	2	4	8

Hungarian marriage (Munkres)

- Je-li počet čar = počet úloh, jsme hotovi
  - optimální přiřazení je na pozicích s 0*
- Je-li nějaká nezakrytá 0 v řádku s 0*
  - označ ji 0‘, odkryj sloupec s 0* a přikryj řádek

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	9	0*	3	0’
Dělník 2	0*	10	4	3
Dělník 3	4	5	0*	6
Dělník 4	0	2	4	8

Hungarian marriage (Munkres)

- Pokud není hotovo
  - nalezni minimum v nezakrytých buňkách
  - odečti minimum od všech nezakrytých a přičti ho ke všem buňkám zakrytým více čarami

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	9	0	3	0
Dělník 2	0	10	4	3
Dělník 3	4	5	0	6
Dělník 4	0	2	4	8

Hungarian marriage (Munkres)

- Opakuj celý postup

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	11	0*	5	0
Dělník 2	0*	8	4	1
Dělník 3	4	3	0*	4
Dělník 4	0	0	4	6

Hungarian marriage (Munkres)

- Existuje nezakrytá nula a v řádku není 0*
  - označ ji 0‘ a toto je člen a_1 posloupnosti střídajících se 0’ a 0*
  - nechť a_i = 0‘, a_{i+1} je 0* ve sloupci s a_i, a a_{i+2} je 0’ v řádce s a_{i+1}

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	11	0* (a_2)	5	0’ (a_3)
Dělník 2	0*	8	4	1
Dělník 3	4	3	0*	4
Dělník 4	0	0’ (a_1)	4	6

Hungarian marriage (Munkres)

- Změň 0* v posloupnosti na 0 a 0’ na 0*, změň všechny 0’ v tabulce na 0, zruš všechny horizontální čáry a pokračuj

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	11	0	5	0*
Dělník 2	0*	8	4	1
Dělník 3	4	3	0*	4
Dělník 4	0	0*	4	6

Hungarian marriage (Munkres)

- A jsme hotovi
  - cena přiřazení: 2 + 4 + 3 + 5 = 14
  - složitost $O(N^3)$

	Úloha 1	**Úloha 2**	**Úloha 3**	**Úloha 4**
Dělník 1	11	0	5	0*
Dělník 2	0*	8	4	1
Dělník 3	4	3	0*	4
Dělník 4	0	0*	4	6

Hungarian marriage (Munkres)