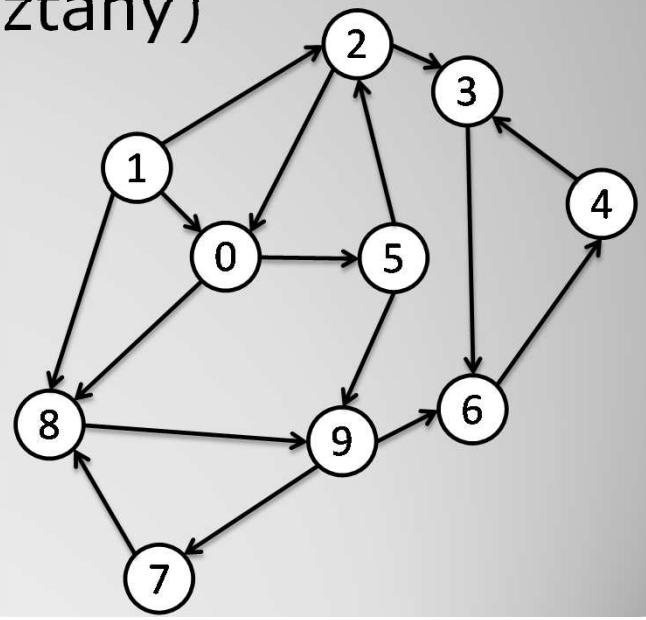
KIV/ZEP - 2011

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



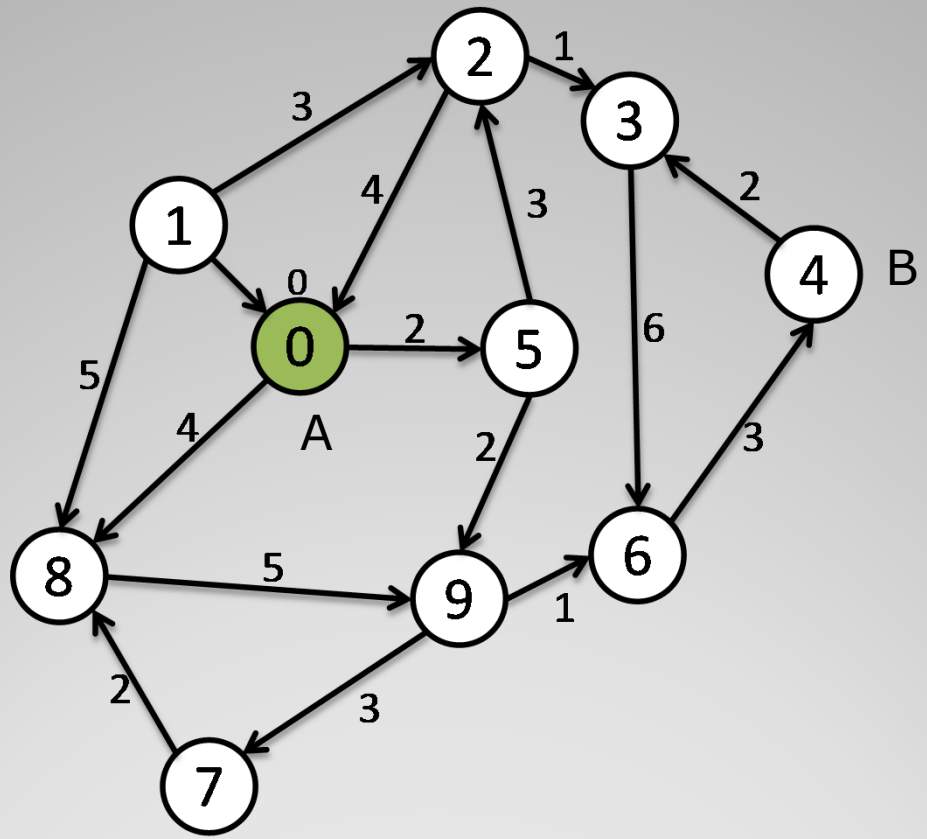
Hledání cesty v grafech

- 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

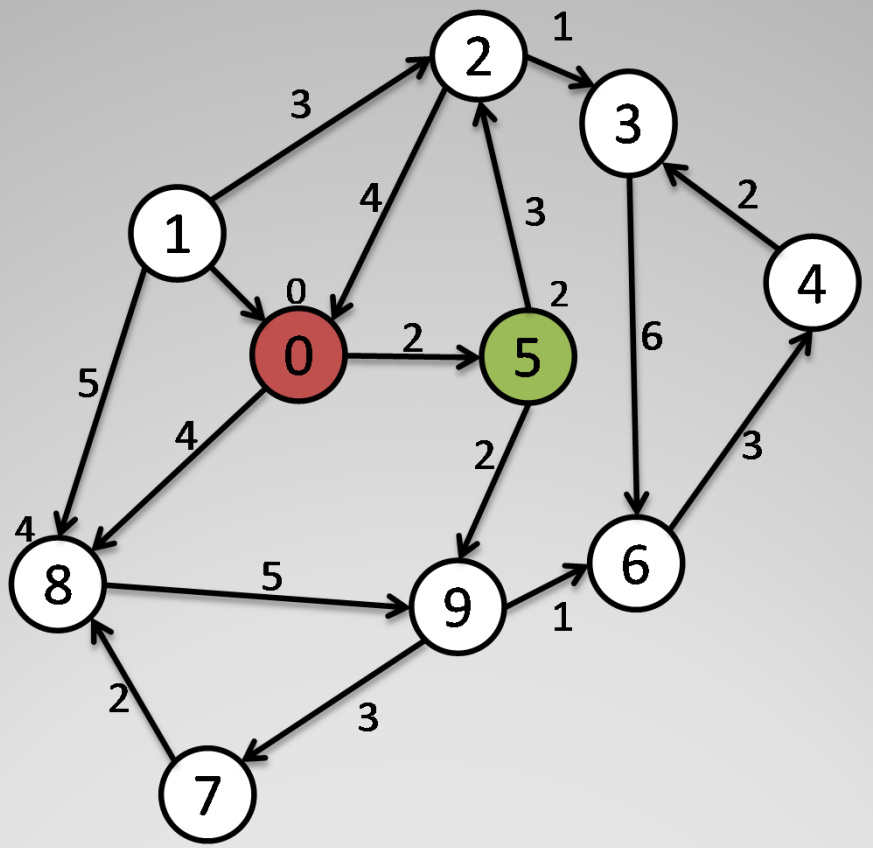
Dijkstra



PQ:

0					
0					

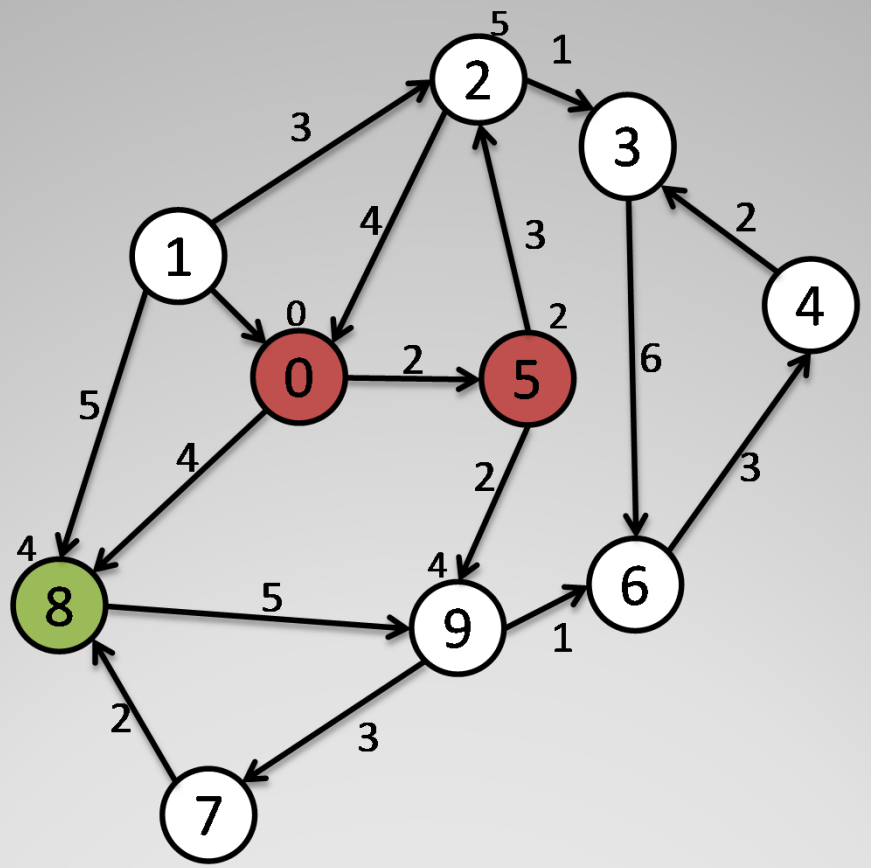
Dijkstra



PQ:

5	8				
2	4				

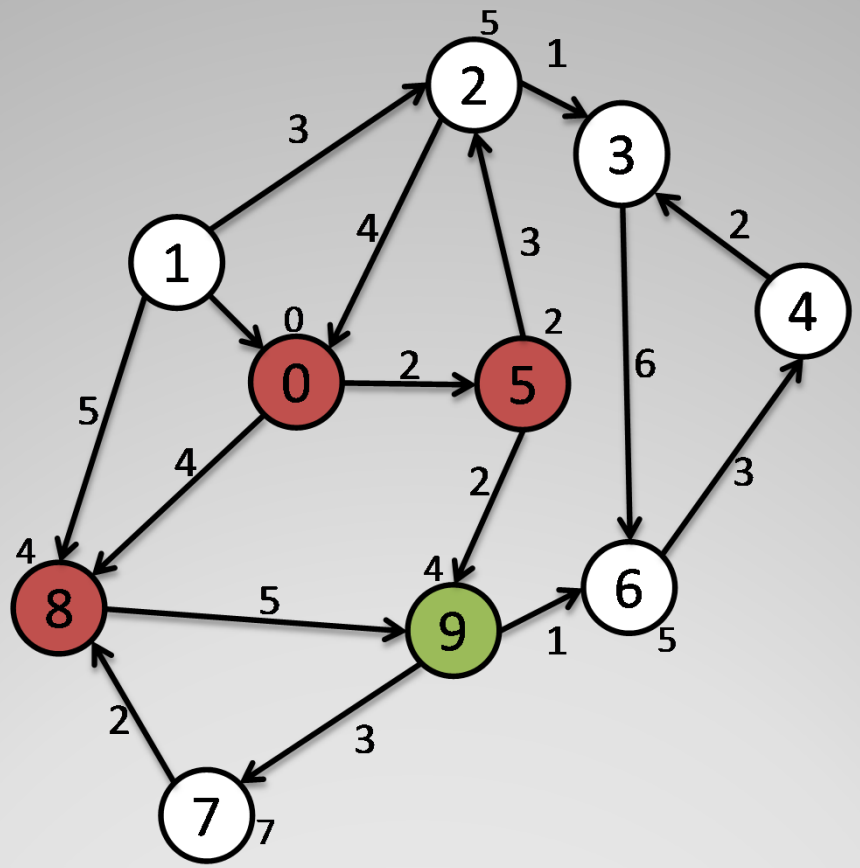
Dijkstra



PQ:

8	9	2			
4	4	5			

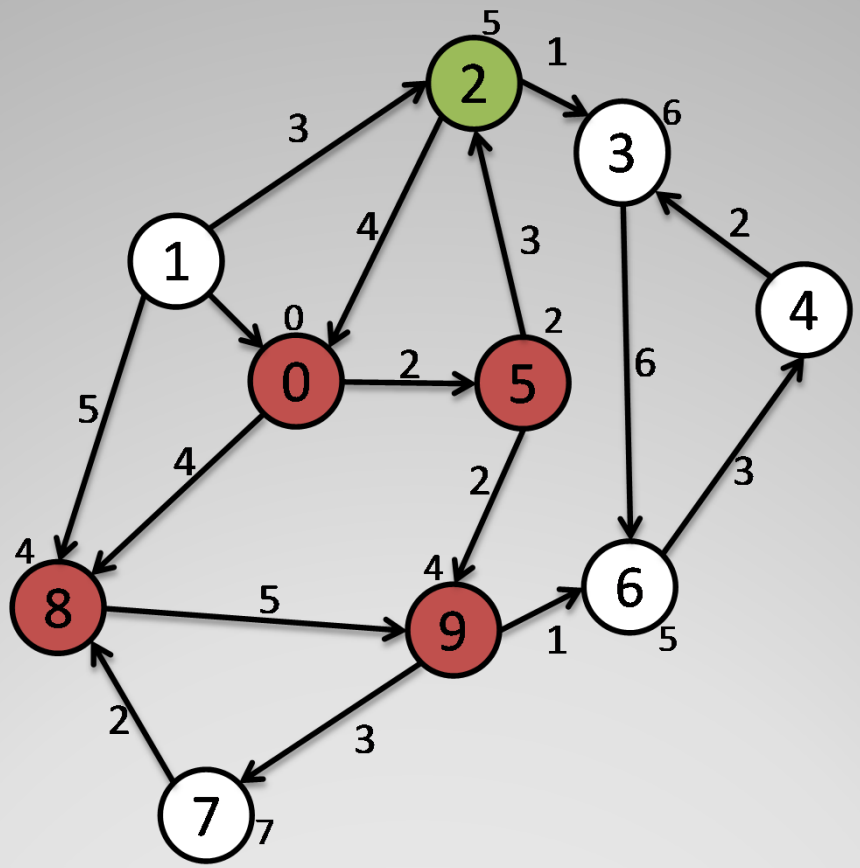
Dijkstra



PQ:

9	2	6	7		
4	5	5	7		

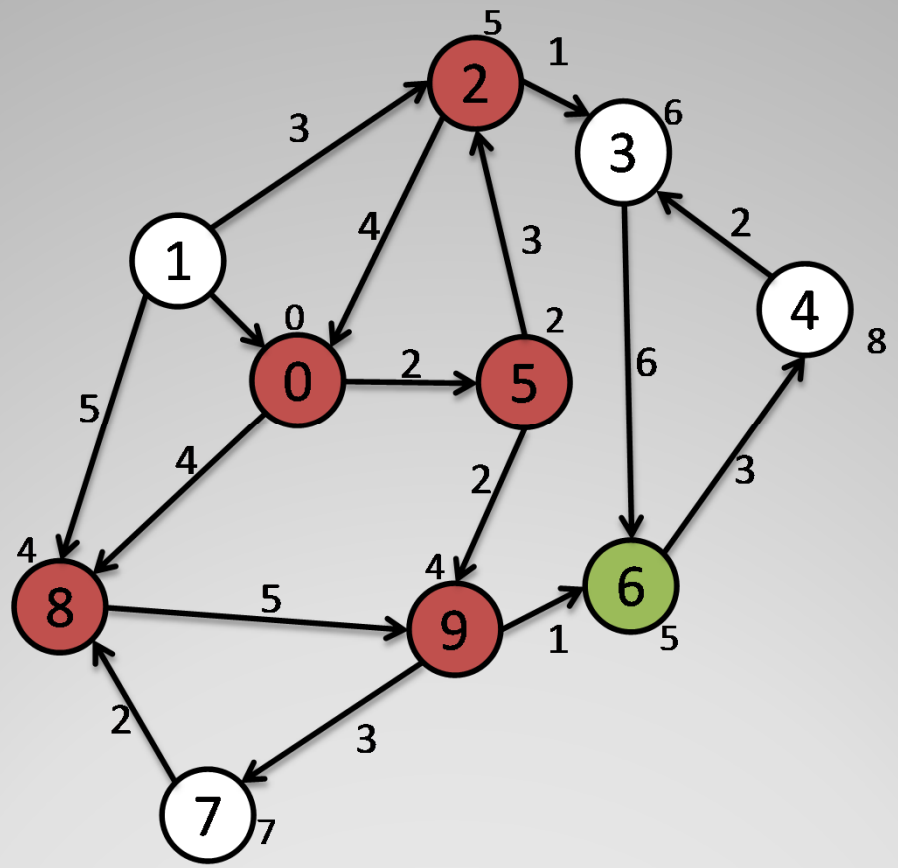
Dijkstra



PQ:

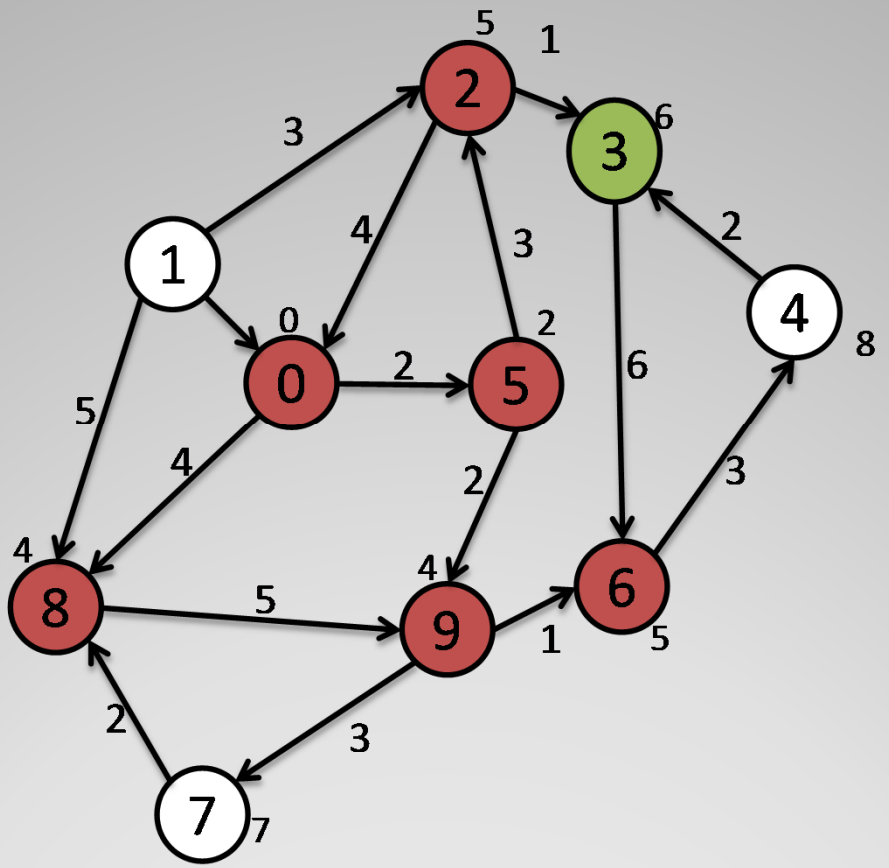
2	6	3	7		
5	5	6	7		

Dijkstra



PQ:

6	3	7	4		
5	6	7	8		

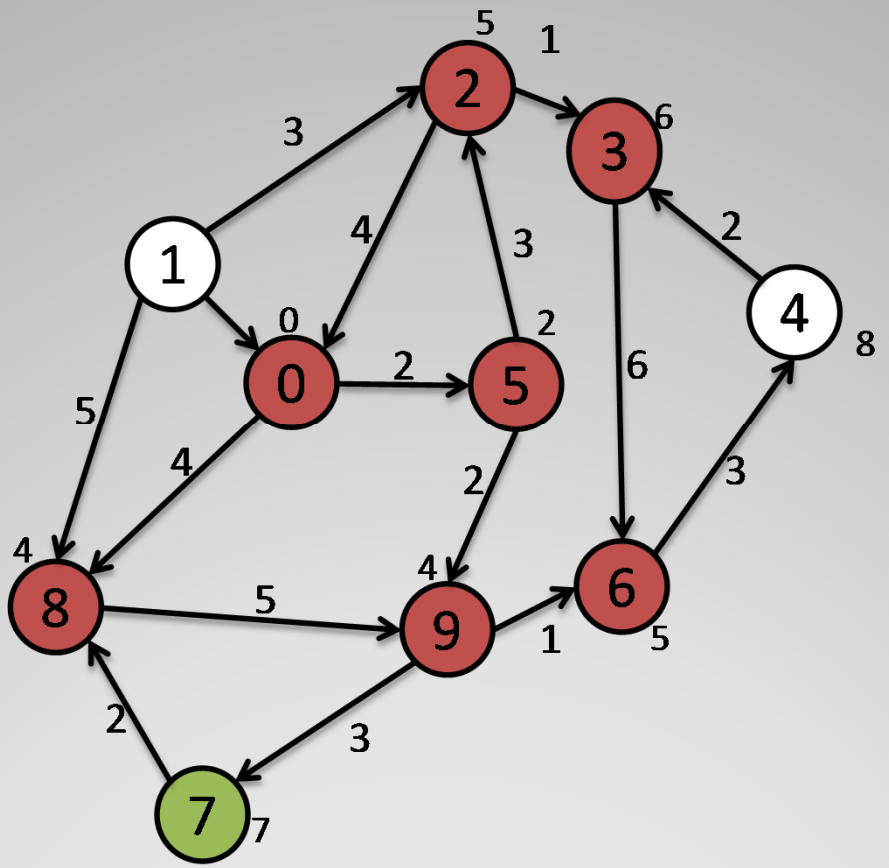


PQ:

3	7	4			
6	7	8			

Dijkstra

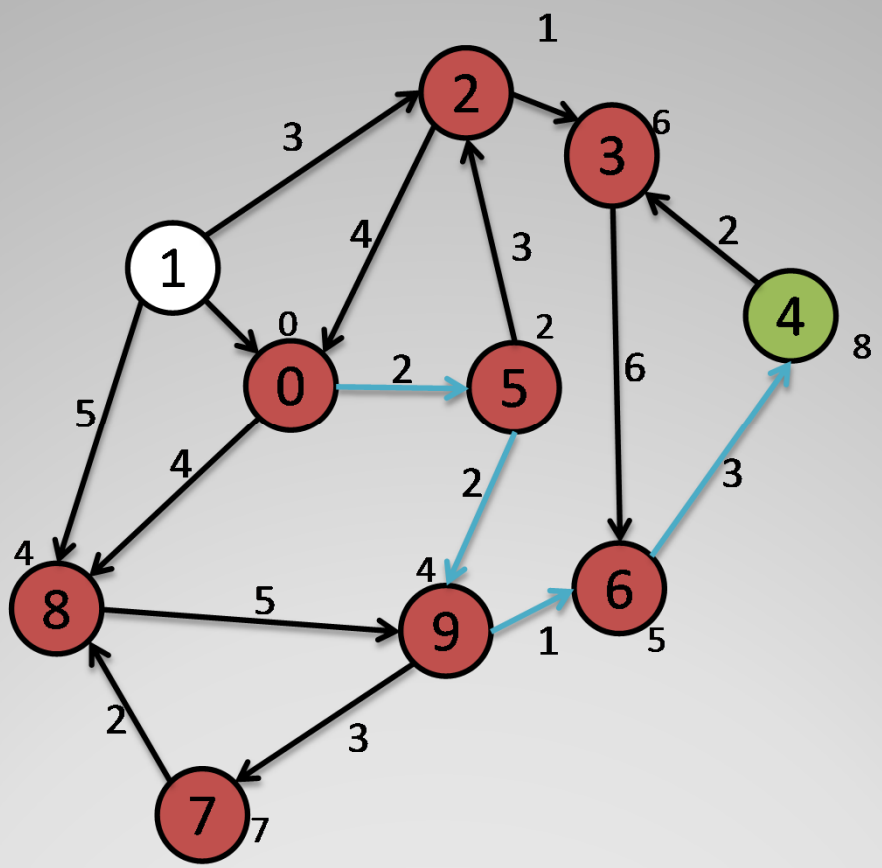
Dijkstra



PQ:

7	4				
7	8				

Dijkstra



PQ:

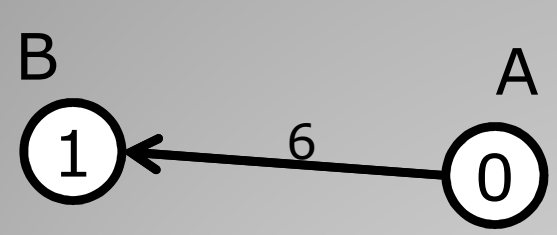
4					
8					

Dijkstra

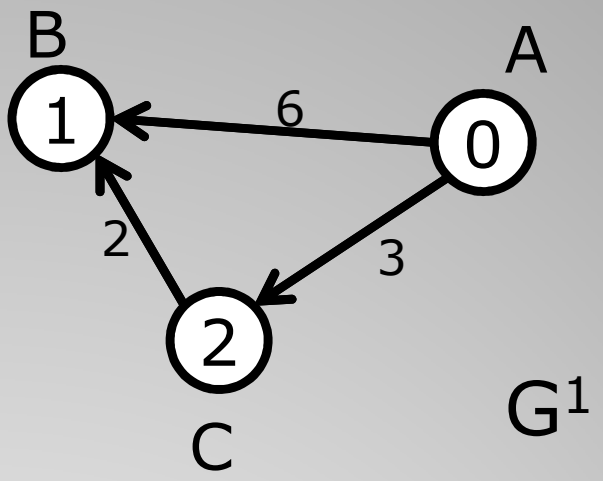
- Složitost:
  - počet uzlů grafu = N
  - počet hran grafu = H (často ~ N^2)
  - vložení nového uzlu do PQ = log(N)
  - vložení celkem = N·log(N)
  - aktualizace ceny uzlu v PQ = log(N)
  - aktualizace celkem = H·log(N)
  - → O((N + H)·log(N)) ~ O(N^2·log(N))

Floyd-Warshall

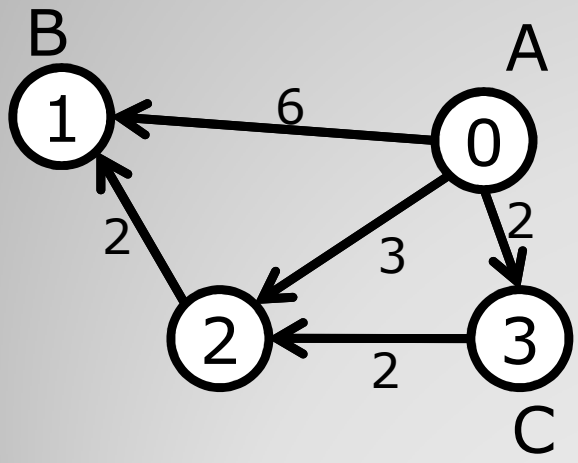
- úloha: najít cestu mezi všemi uzly
  - Dijkstra: $O(N \cdot (N + H) \cdot \log(N))$, tj. $O(N^3 \cdot \log(N))$ v nejhorším případě
  - lépe: Floyd-Warshal
- myšlenka F-W algoritmu
  - hledá se nejkratší cesta z uzlu A do uzlu B v podgrafech obsahujících 0 – N-2 dalších uzlů
  - na nejkratší cestě 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], j);
  }
}
```

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;
       }
     }
   }
}
```

Probrání úloh