Chapter 26: All-Pairs Shortest Path

A trivial solution is to use SSSP algorithms for APSP

With Dijkstra's algorithm (no negative weights!) the running time would become $O(V(V \lg V + E)) = O(V^2 \lg V + VE)$

With the Bellman-Ford algorithm the running time would become $O(V(VE)) = O(V^2 E)$

Three approaches for improvement:

algorithm	cost
matrix multiplication	$\Theta(V^3 \lg V)$
Floyd-Warshall	$\Theta(V^3)$
Johnson	$O(V^2 \lg V + VE)$

Matrix Multiplication

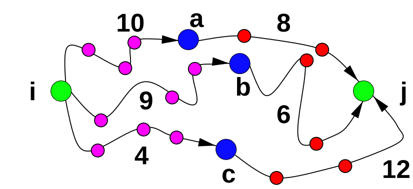
Define the $V \times V$ matrix $D^{(m)} = (d_{ij}^{(m)})$ by:

$d_{ij}^{(m)}$ = the length of the shortest path from i to j with $\leq m$. Then

$$d_{ij}^{(1)} = \begin{cases} 0 & \text{if } i = j, \\ w_{ij} & \text{if } i \neq j, (i,j) \in E, \\ \infty & \text{otherwise,} \end{cases}$$

and for all i, j, p, q,

$$d_{ij}^{(p+q)} = \min_{1 \leq k \leq n} (d_{ik}^{(p)} + d_{kj}^{(q)}).$$

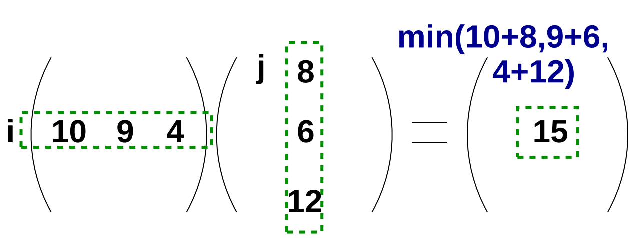


$D^{(V-1)}$ is the matrix $(\delta(i,j))$.

Computing $D^{(p+q)}$ from $D^{(p)}$ and $D^{(q)}$ using matrix multiplication

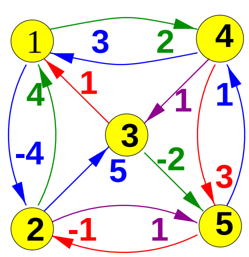
$$D^{(p+q)} = D^{(p)} \cdot D^{(q)}$$

where $(\min, +)$ is used as the computational basis instead of $(+, \times)$



The complexity is $O(V^3 \lg V)$

Q1. *How can you check the existence of negative weight cycles?*



$D(1)$:

$$\begin{pmatrix} 0 & -4 & \infty & 2 & \infty \\ 4 & 0 & 5 & \infty & 1 \\ 1 & \infty & 0 & \infty & -2 \\ 3 & \infty & 1 & 0 & 3 \\ \infty & -1 & \infty & 1 & 0 \end{pmatrix}$$

$D(2)$:

$$\begin{pmatrix} 0 & -4 & 1 & 2 & -3 \\ 4 & 0 & 5 & 2 & 1 \\ 1 & -3 & 0 & -1 & -2 \\ 3 & -1 & 1 & 0 & -1 \\ 3 & -1 & 2 & 1 & 0 \end{pmatrix}$$

$D(4), D(8)$:

$$\begin{pmatrix} 0 & -4 & 1 & -2 & -3 \\ 4 & 0 & 3 & 2 & 1 \\ 1 & -3 & 0 & -1 & -2 \\ 3 & -2 & 1 & 0 & -1 \\ 3 & -1 & 2 & 1 & 0 \end{pmatrix}$$

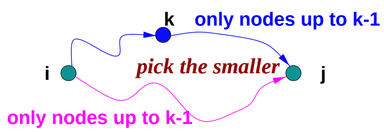
Method 2: Floyd-Warshall

Define the $V \times V$ matrix $W^{(m)} = (f_{ij}^{(m)})$ by:

$f_{ij}^{(m)}$ is the shortest path length from i to j with only nodes $\leq m$ in between

Define $f_{ij}^{(0)} = w_{ij}$. Then for every i, j and every $k \geq 1$,

$$f_{ij}^{(k)} = \min(f_{ij}^{(k-1)}, f_{ik}^{(k-1)} + f_{kj}^{(k-1)}).$$



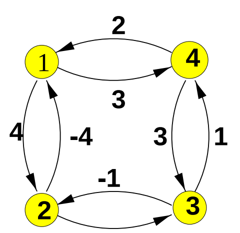
$W^{(V)}$ is the matrix $(\delta(i, j))$.

Compute W^k from W^{k-1} for $k = 1, \ldots, V$

Q2. *How many steps are needed for computing an entry?*

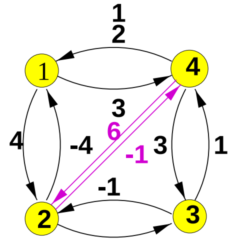
Q3. *How many entries are evaluated in total?*

Q4. *So, what is the total cost?*



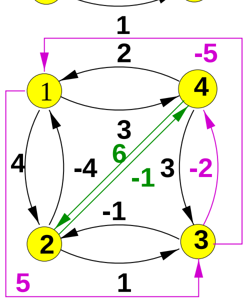
$F(0)$:

$$\begin{pmatrix} 0 & 4 & \infty & 3 \\ -4 & 0 & 1 & \infty \\ \infty & -1 & 0 & 1 \\ 2 & \infty & 3 & 0 \end{pmatrix}$$



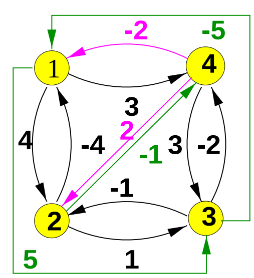
$F(1)$:

$$\begin{pmatrix} 0 & 4 & \infty & 3 \\ -4 & 0 & 1 & -1^* \\ \infty & -1 & 0 & 1 \\ 2 & 6^* & 3 & 0 \end{pmatrix}$$



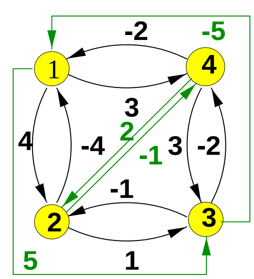
$F(2)$:

$$\begin{pmatrix} 0 & 4 & 5^* & 3 \\ -4 & 0 & 1 & -1 \\ -5^* & -1 & 0 & -2^* \\ 2 & 6 & 3 & 0 \end{pmatrix}$$



$F(3)$:

$$\begin{pmatrix} 0 & 4 & 5 & 3 \\ -4 & 0 & 1 & -1 \\ -5 & -1 & 0 & -2 \\ -2^* & 2^* & 3 & 0 \end{pmatrix}$$



$F(4)$:

$$\begin{pmatrix} 0 & 4 & 5 & 3 \\ -4 & 0 & 1 & -1 \\ -5 & -1 & 0 & -2 \\ -2 & 2 & 3 & 0 \end{pmatrix}$$

Johnson's Algorithm

Define **a new weight function** $\widehat{w}$ so that

- the shortest paths are preserved and
- $\widehat{\ }(u, v) \geq 0$ for all u, v

Then use Dijkstra's algorithm to compute the shortest path

Theorem A Let h be any mapping of V to $\mathbf{R}$. Define $\widehat{w}(u, v) = w(u, v) + h(u) - h(v)$ and $\widehat{\delta}(u, v) =$ the shortest path with respect to $\widehat{w}$. If $\widehat{\delta}(u, v)$ is defined for all u, v, then

$$\delta(u, v) = \widehat{\delta}(u, v) + h(v) - h(u);$$

i.e., the new weight function preserves the shortest paths.

Proof For any path $p = [v_1, \ldots, v_k]$ the path length of p under $\widehat{w}$ is

$$\sum_{i=1}^{k-1} \Big(w(v_{i+1}, v_i) + h(v_i) - h(v_{i+1}) \Big).$$

This is equal to

$$\left(\sum_{i=1}^{k-1} \right) + \sum_{i=1}^{k-1} \Big(h(v_i) - h(v_{i+1}) \Big).$$

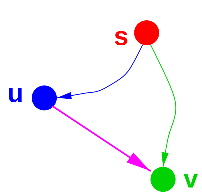
The right hand-side is $h(v_1) - h(v_k)$. So for every u and v, $\widehat{\delta}(u, v) = \delta(u, v) + h(u) - h(v)$.∎

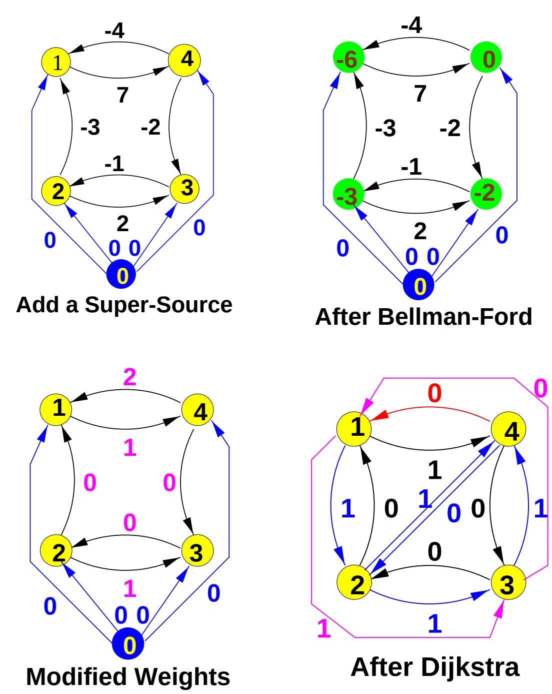
1. Add a **new node** s with no incoming edges and with a 0-weight outgoing edge to every other node
2. Use *the Bellman-Ford* algorithm to compute $h(u) = \delta(s, u)$ for all u
3. Let $\widehat{w}(u, v) = w(u, v) + h(u) - h(v)$ and use *Dijkstra's method* to compute $\widehat{\delta}(u, v)$
4. Output for each u, v, $\delta(u, v)$ as $\widehat{\delta}(u, v) + h(v) - h(u)$

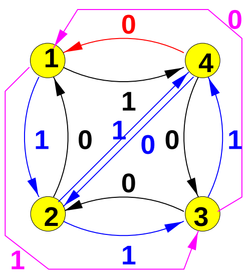
The use of Dijkstra's method is possible because for every u, v,

$$\delta(s, v) \leq w(u, v) + \delta(s, u)$$

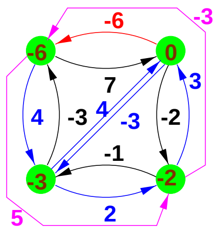
$$\widehat{w}(u, v) = w(u, v) + h(u) - h(v) \geq 0$$







After Dijkstra



Back to Original Weights