All Pairs Shortest Paths

- **Input: weighted, directed graph $G = (V, E)$, with weight function $w : E \rightarrow$ R.**
- **The weight of path $p = < v_0, v_1, \ldots, v_k >$ is the sum of the weights of its constituent edges:**

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

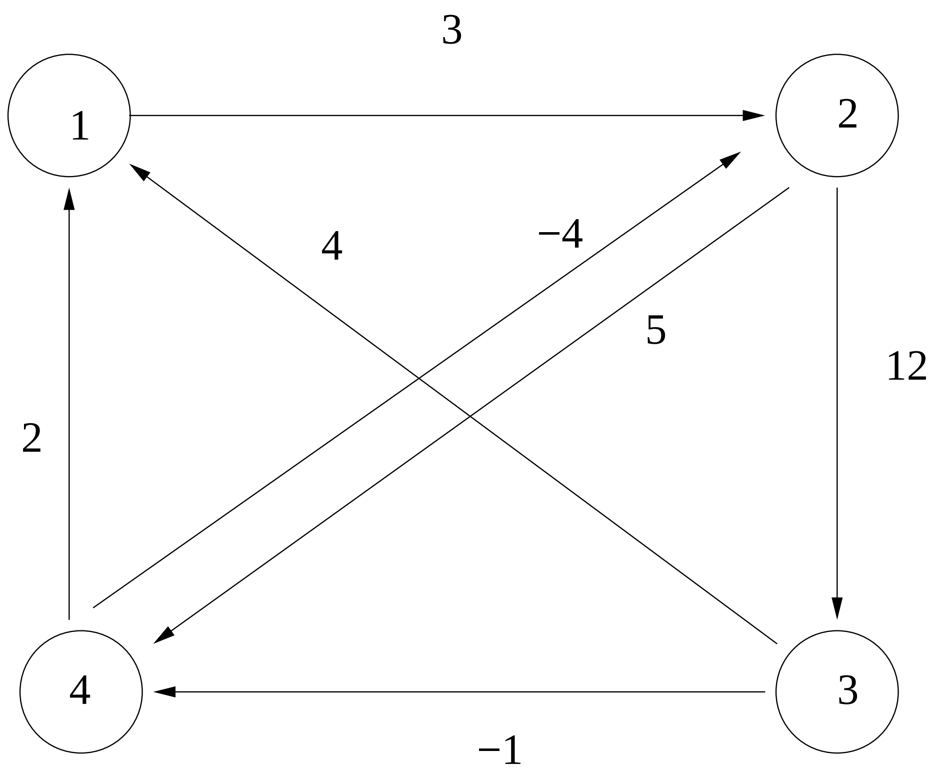
- **The shortest-path weight from u to v is**

$$\delta(u, v) = \begin{cases} \min\{w(p)\} & \textbf{if there is a path } p \textbf{ from } u \textbf{ to } v \ , \\ \infty & \textbf{otherwise} \ . \end{cases}$$

- **A shortest path from vertex u to vertex v is then defined as any path p with weight $w(p) = \delta(u, v)$.**

All Pairs Shortest Paths: **Compute $d(u, v)$ the shortest path distance from u to v for all pairs of vertices u and v.**

Example



Solution

$$\begin{pmatrix} 0 & 3 & 15 & 8 \\ 7 & 0 & 12 & 5 \\ 1 & 4 & 0 & -1 \\ 2 & -4 & 8 & 0 \end{pmatrix}$$

Approach 1

Run Single source shortest paths V times

- $O(V^2E)$ **for general graphs**
- $O(VE + V^2 \log V)$ **for graphs with non-negative edge weights**

Other approaches : Share information between the various computations

Floyd-Warshall, Dynamic Programming

- **Let $d_{ij}^{(k)}$ be the weight of a shortest path from vertex i to vertex j for which all intermediate vertices are in the set $\{1, 2, \ldots, k\}$.**
- **When $k = 0$, a path from vertex i to vertex j with no intermediate vertex numbered higher than 0 has no intermediate vertices at all, hence $d_{ij}^{(0)} = w_{ij}$.**

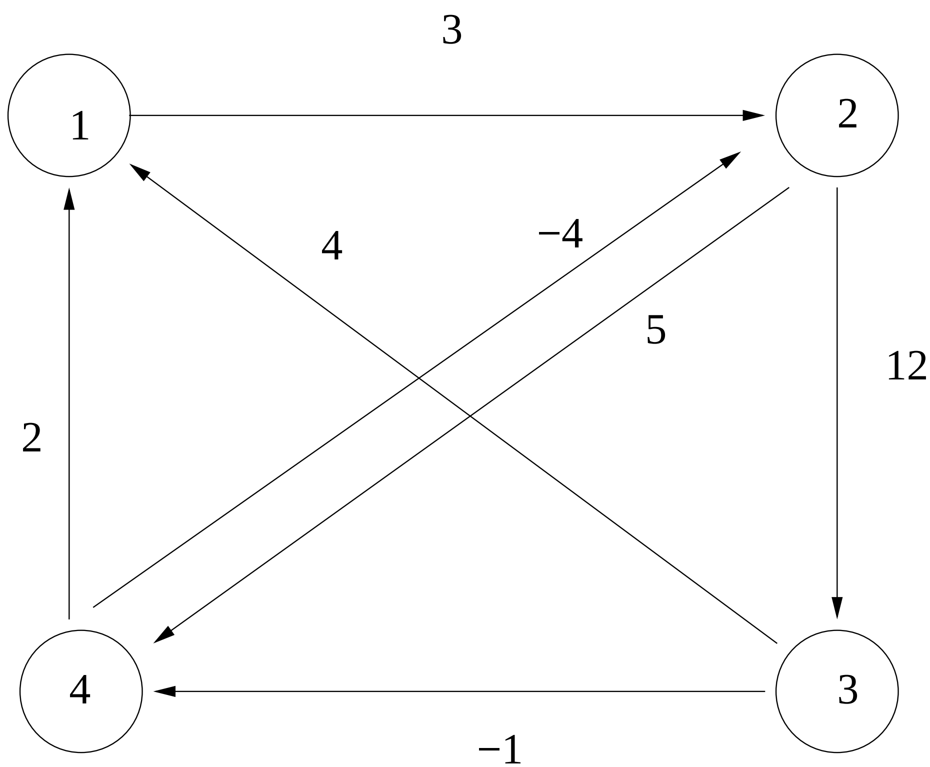
$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \textbf{if } k = 0 \ , \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \textbf{if } k \geq 1 \ . \end{cases} \tag{1}$$

Floyd-Warshall(W)

$n \leftarrow rows[W]$
$D^{(0)} \leftarrow W$
for $k \leftarrow 1$ **to** n
  do for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
      do $d_{ij}^{(k)} \leftarrow \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
return $D^{(n)}$

Running time $O(V^3)$

Example



$$D^0 = \begin{pmatrix} 0 & 3 & 0 & 0 \\ 0 & 0 & 12 & 5 \\ 4 & 0 & 0 & -1 \\ 2 & -4 & 0 & 0 \end{pmatrix} D^1 = \begin{pmatrix} 0 & 3 & 0 & 0 \\ 0 & 0 & 12 & 5 \\ 4 & 7 & 0 & -1 \\ 2 & -4 & 0 & 0 \end{pmatrix} D^2 = \begin{pmatrix} 0 & 3 & 15 & 8 \\ 0 & 0 & 12 & 5 \\ 4 & 7 & 0 & -1 \\ 2 & -4 & 8 & 0 \end{pmatrix}$$

$$D^3 = \begin{pmatrix} 0 & 3 & 15 & 8 \\ 16 & 0 & 12 & 5 \\ 4 & 7 & 0 & -1 \\ 2 & -4 & 8 & 0 \end{pmatrix} D^4 = \begin{pmatrix} 0 & 3 & 15 & 8 \\ 7 & 0 & 12 & 5 \\ 1 & -5 & 0 & -1 \\ 2 & -4 & 8 & 0 \end{pmatrix}$$

Another Algorithm

RESET ALL DEFINITIONS OF D.

- **Let** w_{ij} **be the length of edge** ij
- **Let** $w_{ii} = 0$
- **Let** d_{ij}^m **be the shortest path from** i **to** j **using** m **or fewer edges**

$$d_{ij}^1 = w_{ij}$$

$$d_{ij}^m = \min\{d_{ij}^{m-1}, \min_{1 \leq k \leq n, k \neq j} d_{ik}^{m-1} + w_{kj}\}$$

Combining these two, we get

$$d_{ij}^m = \min_{1 \leq k \leq n}\{d_{ik}^{m-1} + w_{kj}\}$$

This would give an $O(V^4)$ **algorithm**

Using matrix multiplication analogy

Note the similarity of

$$d_{ij}^m = \min_{1 \le k \le n} \{d_{ik}^{m-1} + w_{kj}\}$$

with matrix multiplication:

$$c_{ij} = \mathbf{sum}_{1 \le k \le n} \{a_{ik} \cdot k_{kj}\}$$

Make the following substitutions (which have the right algebraic properties:

$$\begin{aligned} \mathbf{sum} &\to \min \\ a_{ij} &\to d_{ik}^{m-1} \\ \cdot &\to + \\ b_{kj} &\to w_{ij} \\ c &\to d^m \end{aligned}$$

Using this matrix multiplication terminology, we have

$$\begin{array}{lll} D^1 & & = W \\ D^2 & = D^1 \cdot W & = W^2 \\ D^3 & = D^2 \cdot W & = W^3 \\ \dots & \dots & \dots \\ D^m & = D^{m-1} W & = W^m \end{array}$$

But we can execute W^m be repeated squaring and get $O(V^3 \log V)$ time.