

# All-Pairs Shortest Paths Problem

## The Floyd-Warshall Algorithm

Given a weighted digraph  $G = (V, E)$  with weight function  $w: E \rightarrow R$ , ( $R$ , is the set of real numbers) determine the length of the shortest path (i.e., distance between all pairs of vertices in  $G$ ). Here we assume that there are no cycles with zero or negative cost.

# Decomposition

- **Definition:** The vertices  $v_2, v_2, \dots, v_{l-1}$  are called the **intermediate vertices** of the path  $p = \{v_1, v_2, v_2, \dots, v_{l-1}, v_l\}$ .
- Let  $c^{(k)}_{ij}$  be the length of the shortest path from  $i$  to  $j$  such that all intermediate vertices on the path (in any) are in set  $\{1, 2, \dots, k\}$ .
- $c^{(0)}_{ij}$  is set to be  $w_{ij}$ , i.e., no intermediate vertex.
- Let  $D^{(k)}$  be the  $n \times n$  matrix  $[c^{(k)}_{ij}]$

# Decomposition

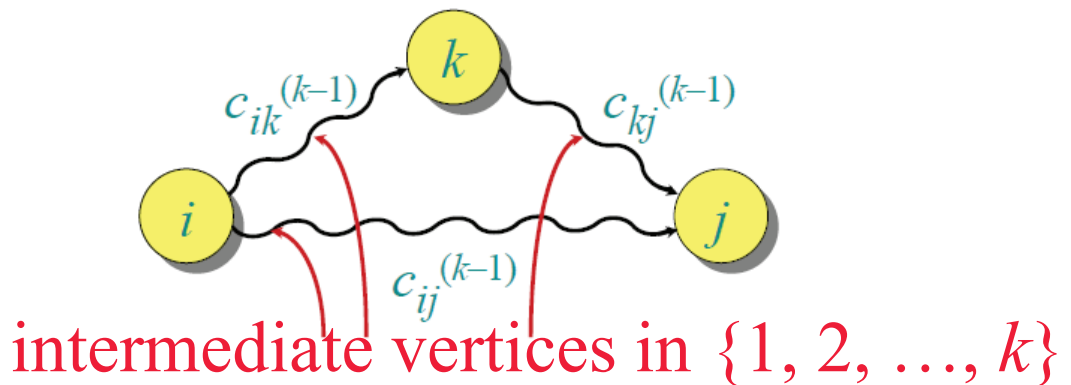
- $c^{(k)}_{ij}$  is the distance from  $i$  to  $j$ .
- our aim is to compute  $D^{(n)}$ .
- Subproblems:
  - compute  $D^{(k)}$  for  $k=0, 1, \dots, n$ .

# Structure of shortest paths

- **Observation 1:** A shortest path does not contain the same vertex twice.
- **Observation 2:** For a shortest path from  $i$  to  $j$  such that any intermediate vertices on the path are chosen from the set  $\{1, 2, \dots, k\}$  there are two possibilities:
  - $k$  is not a vertex on the path, the shortest such path has length  $c^{(k-1)}_{ij}$
  - $k$  is a vertex on the path, the shortest such path has length  $c^{(k-1)}_{ik} + c^{(k-1)}_{kj}$ .

# Floyd-Warshall recurrence

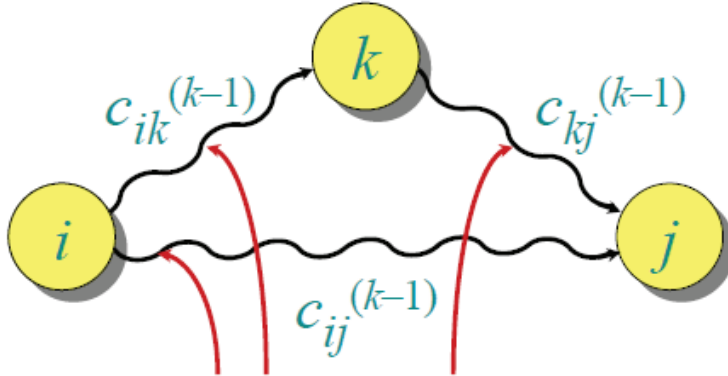
- Consider a shortest path from  $i$  to  $j$  containing the vertex  $k$ . It consists of a subpath from  $i$  to  $k$  and  $k$  to  $j$ .
- Each subpath can only contain intermediate vertices in  $\{1, 2, \dots, k-1\}$  and must be as short as possible, namely they have lengths  $c_{ik}^{(k-1)}$  and  $c_{kj}^{(k-1)}$ .
- Hence the path has length  $c_{ik}^{(k-1)} + c_{kj}^{(k-1)}$ .



# Floyd-Warshall recurrence

- Combining the two cases we get

$$c_{ij}^{(k)} = \min_k \{c_{ij}^{(k-1)}, c_{ik}^{(k-1)} + c_{kj}^{(k-1)}\}$$



intermediate vertices in  $\{1, 2, \dots, k\}$

# Bottom-up Computation

- Bottom:  $D^{(0)}$   
=  $[w_{ij}]$  the weight matrix.
- Compute  $D^{(k)}$  from  $D^{(k-1)}$  using
$$c^{(k)}_{ij} = \min \{ c^{(k-1)}_{ij}, c^{(k-1)}_{ik} + c^{(k-1)}_{kj} \} \text{ for } k = 1, 2, \dots, n$$

# Pseudocode for Floyd-Warshall

Floyd-Warshall( $W$ )

$n \leftarrow \text{rows}[W]$

$D^{(0)} \leftarrow W$

for  $k \leftarrow 1$  to  $n$

    do for  $i \leftarrow 1$  to  $n$

        do for  $j \leftarrow 1$  to  $n$

            do  $c^{(k)}_{ij} \leftarrow \min\{c^{(k-1)}_{ij}, c^{(k-1)}_{ik} + c^{(k-1)}_{kj}\}$

return  $D^{(n)}$



# Extracting the Shortest Paths

- Construct a predecessor matrix  $\Pi$  from the  $D$  matrix.
- Given the predecessor matrix  $\Pi$ , **print-all-pairs-shortest-path** procedure can use to print vertices on a given shortest path.
- The predecessor pointers  $\pi_{ij}$  can be used to extract the final path. The idea is as follows.
- Whenever we discover that the shortest path from  $i$  to  $j$  passes through an intermediate vertex  $k$ , we set  $\pi_{ij} = k$
- If the shortest path does not pass through any intermediate vertex, then  $\pi_{ij} = \text{NIL}$ .

# Pseudocode for Floyd-Warshall (Extracting the Shortest Paths)

Floyd-Warshall( $W$ )

$n \leftarrow \text{rows}[W]$

$C^{(0)} \leftarrow W = [w_{ij}]$

$\Pi^{(0)} \leftarrow [\pi^{(0)}_{ij}]$

for  $k \leftarrow 1$  to  $n$

  do for  $i \leftarrow 1$  to  $n$

    do for  $j \leftarrow 1$  to  $n$

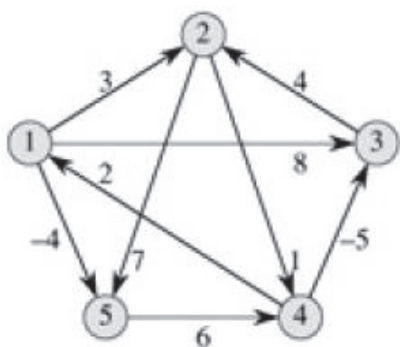
      do  $c^{(k)}_{ij} \leftarrow \min \{c^{(k-1)}_{ij}, c^{(k-1)}_{ik} + c^{(k-1)}_{kj}\}$

$\pi_{ij} \leftarrow k$

return  $\Pi^{(n)}$

$\pi^{(0)}_{ij} = \text{NIL}$  if  $i=j$  or  $w_{ij} = \infty$   
 $= i$  if  $i \neq j$  and  $w_{ij} < \infty$

# Example



Floyd-Warshall( $W$ )

$n \leftarrow \text{rows}[W]$

$D^{(0)} \leftarrow W$

for  $k \leftarrow 1$  to  $n$

  for  $i \leftarrow 1$  to  $n$

    for  $j \leftarrow 1$  to  $n$

$c_{ij}^{(k)} \leftarrow \min\{c_{ij}^{(k-1)}, c_{ik}^{(k-1)} + c_{kj}^{(k-1)}\}$

$\pi_{ij} \leftarrow k$

return  $D^{(n)}$

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \quad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \quad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

# Run time comparison with Floyd-Warshall Algorithm

- $O(V^2E)$  Bellman-Ford (sparse graph)
- $O(V^4)$  Bellman-Ford (dense graph)
- $O(VE \lg V)$  Dijkstra (binary-heap) (sparse graph)
- $O(V^3 \lg V)$  Dijkstra (binary-heap) (dense graph)
- $O(V^2 \lg V + VE)$  Dijkstra (Fibonacci-heap) (sparse graph)
- $O(V^2 \lg V + V^3)$  Dijkstra (Fibonacci-heap) (dense graph)
  - ve weight edges are **not** allowed in case of Dijkstra's algorithm
- **Dynamic Programming based algorithms:**
  - $O(V^4)$  matrix multiplication methods
  - $O(V^3 \lg V)$  repeated squaring
  - $O(V^3)$  Floyd-Warshall Algorithm

## Transitive closure of a directed graph

- Given a digraph  $G=(V,E)$  we may wish to find whether there is a path in  $G$  from  $i$  to  $j$  for all vertex pairs  $i,j$ .
- the transitive closure of  $G$  is defined as the graph  $G^*=(V,E^*)$   $E^*=\{(i,j): \text{there is a path from } i \text{ to } j \text{ in } G\}$

Compute  $t_{ij} = \begin{cases} 1 & \text{if there exists a path from } i \text{ to } j, \\ 0 & \text{otherwise.} \end{cases}$

## Transitive closure of a directed graph

Compute  $t_{ij} = \begin{cases} 1 & \text{if there exists a path from } i \text{ to } j, \\ 0 & \text{otherwise.} \end{cases}$

**IDEA:** Use Floyd-Warshall, but with  $(\vee, \wedge)$  instead of  $(\min, +)$ :

$$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}).$$

Time =  $\Theta(n^3)$ .