All-Pairs Shortest Paths

Heejin Park
Division of Computer Science and Engineering
Hanyang University

Contents

- Using SSSP (single source shortest path) algorithms
- Floyd-Warshall algorithm
- Transitive closure of a directed graph

Using SSSP algorithms

- We can solve an all-pairs shortest-paths problem by running a ***single-source shortest-paths algorithm |V| times***, once for each vertex as the source.

- **Nonnegative-weight** edges
  - Dijkstra's algorithm
    - The linear-array implementation
      - $O(V \cdot V^2) = O(V^3)$.
    - The binary min-heap implementation
      - $O(V \cdot (V \lg V + E \lg V)) = O(V^2 \lg V + V E \lg V)$

비교x

Using SSSP algorithms

- ***Negative-weight edges***
  - Bellman-Ford algorithm
    - $O(V \cdot VE) = O(V^2E)$
    - $O(V^4)$ on a dense graph

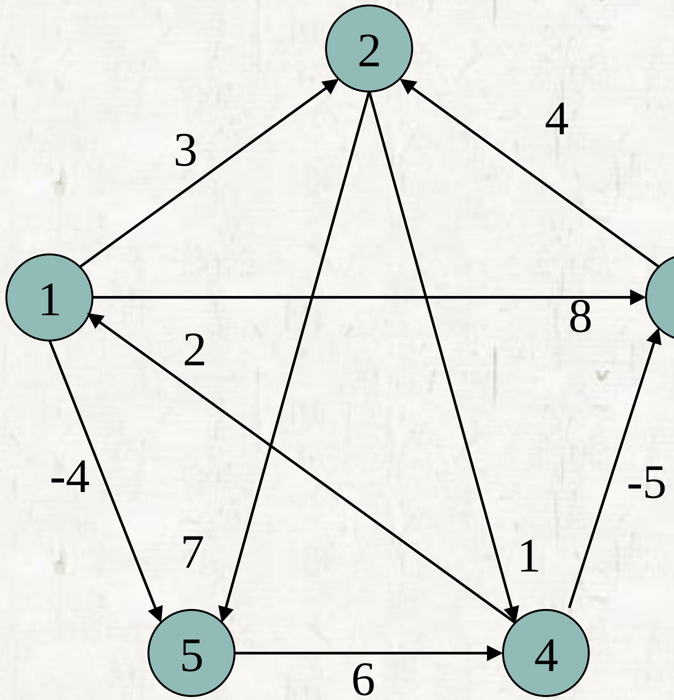
- DAG

Contents

- *Using SSSP (single source shortest path) algorithms*
- Floyd-Warshall algorithm
  - $\Theta(V^3)$-time
- Transitive closure of a directed graph

Definition

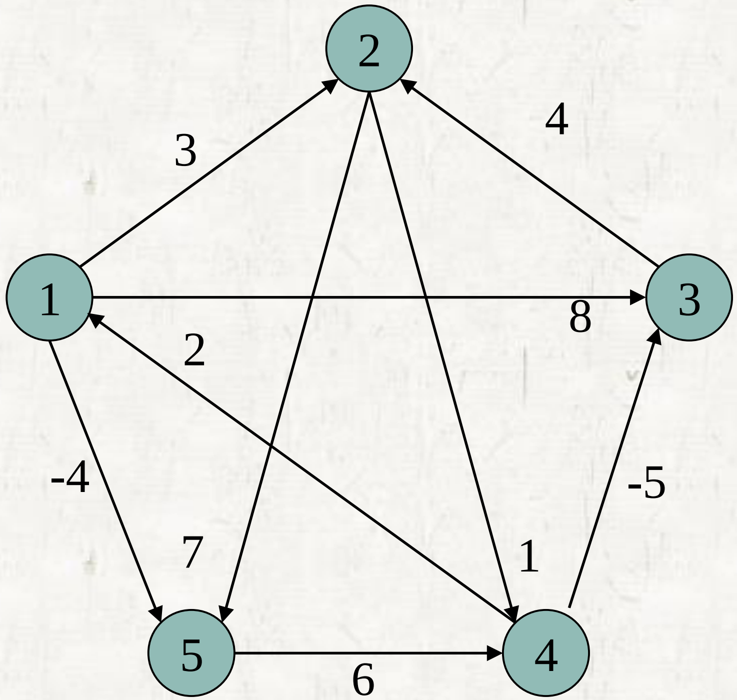
- Adjacency Matrix W
  - $w_{ij} = w(i,j)$



$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Definition

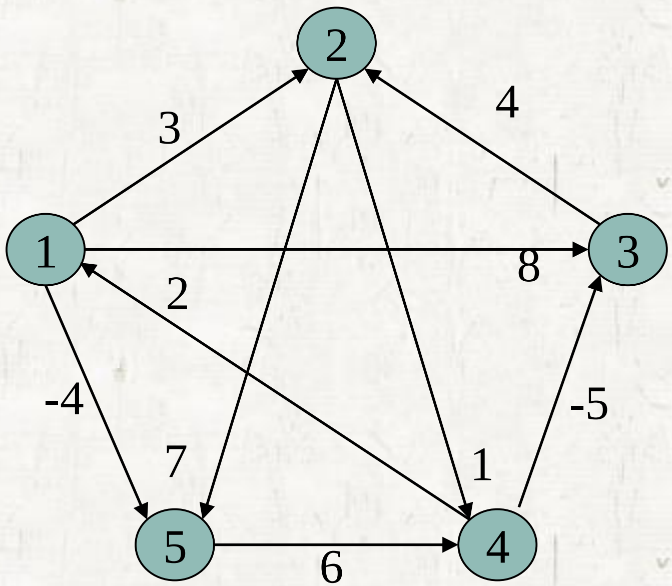
- Shortest Distance Matrix D
  - $d_{ij} = \delta(i,j)$



$$\begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Definition

- Predecessor Matrix **Π**
  - π_{ij} = NIL if either $i = j$ or there is not path from i to j.
  - π_{ij} is the predecessor of j on some shortest path from i to j.



$$\begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Definition

- The following procedure prints a shortest path from i to j due to the optimal substructure of the shortest-paths problem.

PRINT-ALL-PAIRS-SHORTEST-PATH(Π, *i*, *j*)

if $i == j$
    print i
elseif π_{ij} == NIL
    print “no path from” i “to” j “exists”
else PRINT-ALL-PAIRS-SHORTEST-PATH(Π, i, π_{ij})
    print j

Floyd-Warshall algorithm

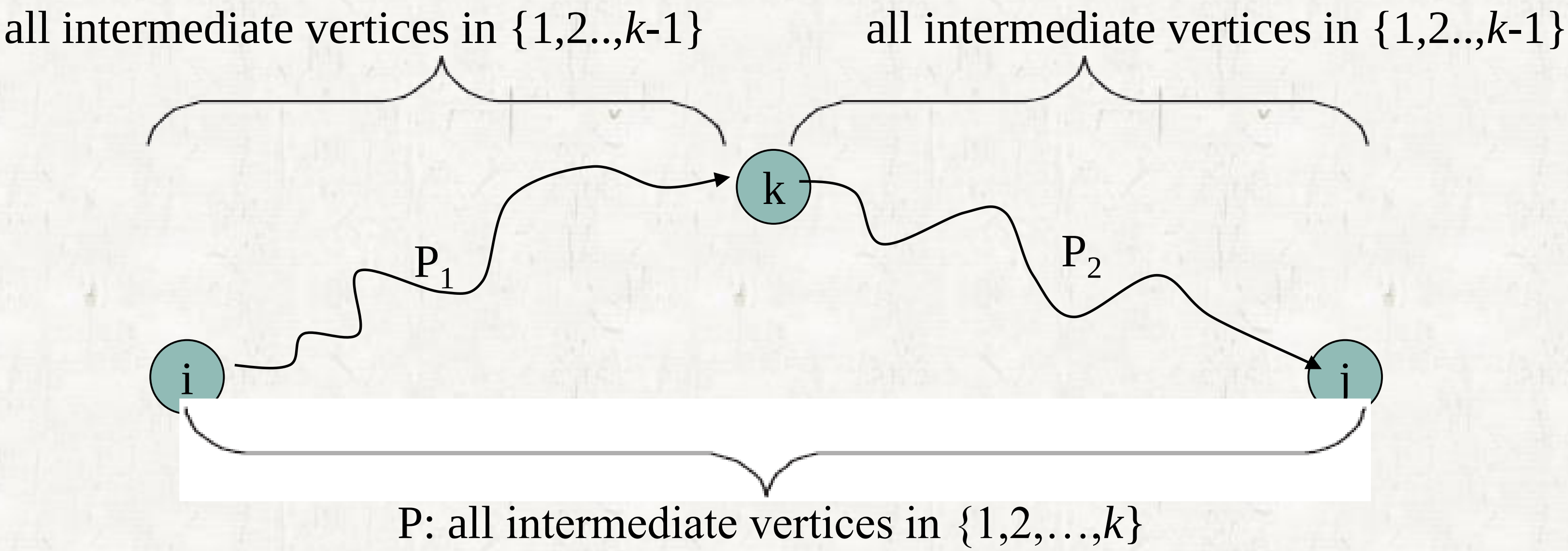
- **Intermediate Vertex**
  - An intermediate vertex of a simple path $p = \langle v_1, v_2, \cdots, v_l \rangle$ is any vertex of p between v_1 and v_l.

Floyd-Warshall algorithm

- **The structure of a shortest path**
  - Floyd-Warshall algorithm is based on the observation of the intermediate vertices, which costs $\Theta(V^3)$ time.
  - Let $V = \{1, 2, \cdots, n\}$.
  - For any pair of vertices $i, j \in V$, consider all paths from i to j whose intermediate vertices are all drawn from $\{1, 2, \cdots, k\}$, and let p be a minimum weight path from among them.

Floyd-Warshall algorithm

- If k is not an intermediate vertex of path p, then all intermediate vertices of p are in $\{1, 2, \cdots, k-1\}$.
- If k is an intermediate vertex of path p, then we break p down into $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$.



Floyd-Warshall algorithm

- A recursive solution to the all-pairs shortest-paths problem
  - 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 $\{1, 2, \cdots, k\}$.
  - We have the following recurrence:

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0, \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1. \end{cases} \qquad (25.5)$$

  - Because for any path, all intermediate vertices are in the set $\{1, 2, \cdots, n\}$, the matrix $D^{(n)} = d_{ij}^{(n)}$ gives the final answer: $d_{ij}^{(n)} = \partial(i, j)$ for all $i, j \in V$.

Floyd-Warshall algorithm

FLOYD-WARSHALL(W)

$n = W.rows$

$D^{(0)} = W$

for $k = 1$ **to** n

  let $D^{(k)} = (d_{ij}^{(k)})$ be a new n x n matrix

  for $i = 1$ **to** n

    for $j = 1$ to n

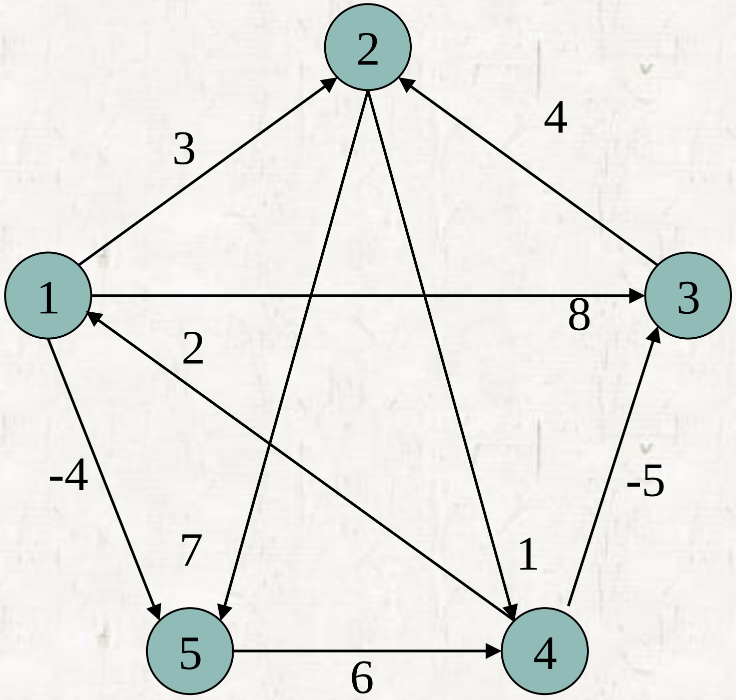
      $d_{ij}^{(k)} = \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$

return $D^{(n)}$

k x O

- costs $\Theta(n^3)$ time.

Floyd-Warshall algorithm



$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Floyd-Warshall algorithm

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

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

$$\Pi^{(0)} = \begin{pmatrix} \mathrm{NIL} & 1 & 1 & \mathrm{NIL} & 1 \\ \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 2 & 2 \\ \mathrm{NIL} & 3 & \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} \\ 4 & \mathrm{NIL} & 4 & \mathrm{NIL} & \mathrm{NIL} \\ \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 5 & \mathrm{NIL} \end{pmatrix}$$

$$\Pi^{(1)} = \begin{pmatrix} \mathrm{NIL} & 1 & 1 & \mathrm{NIL} & 1 \\ \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 2 & 2 \\ \mathrm{NIL} & 3 & \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} \\ 4 & 1 & 4 & \mathrm{NIL} & 1 \\ \mathrm{NIL} & \mathrm{NIL} & \mathrm{NIL} & 5 & \mathrm{NIL} \end{pmatrix}$$

Floyd-Warshall algorithm

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

$$\Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & \underline{1} & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix} \qquad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & \underline{3} & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

Floyd-Warshall algorithm

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

$$\Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix} \qquad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

Floyd-Warshall algorithm

- Constructing A Shortest Path
  - Let Π_{ij}^{k} be the predecessor of vertex j on a shortest path from vertex i with all intermediate vertices in $\{1, 2, \cdots, k\}$.

$$\Pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j \text{ or } w_{ij} = \infty, \\ i & \text{if } i \neq j \text{ and } w_{ij} < \infty. \end{cases}$$

$$\Pi_{ij}^{(k)} = \begin{cases} \Pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}, \\ \Pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}. \end{cases}$$

Floyd-Warshall algorithm

- Transitive Closure of Graph
  - Given a directed graph $G = (V, E)$ with vertex set $V = \{1, 2, \cdots, n\}$.
  - The transitive closure of G is defined as the graph $G* = (V, E*)$, where $E* = \{(i, j)$: there is a path from vertex i to vertex j in G$\}$.