All-Pair-Shortest-Paths (APSP)

I/P: Given a weighted directed graph $G = (V, E)$ with a weight fun $w : E \to \mathbb{R}$.

Q: find, for every Pair of vertices $u, v \in V$, a shortest weighted Path from u to v

Q How Single Source Shortest Path Problem is related (or can be used) APSP?

Q How many invocations of a SSSP subroutine are needed to solve the apsp? [n: # of vertices]

A We can solve APSP problem by running a SSSP algorithm n times, once for each vertex as the source.

→ if all edge weights are non-negative we can use Dijkstra's algorithm, n times.

Running time $= O(nm \log n)$ [using heap implementation]

if G is sparse ie, $m = \Theta(n)$ then $O(n^2 \log n)$

if G is dense ie, $m = \Theta(n^2)$ then $\Theta(n^3 \log n)$

↳ we need at $\Omega(n^3)$ time find apsp

→ if the graph has negative edge weights we can use Bellman-Ford algorithm once from each vertex.

Running time $= O(n^2 m)$

$O(n^3)$ if $m = \Theta(n)$

$O(n^4)$ if $m = \Theta(n^2)$

Q: Can we do better?

A YES [Floyd-Warshall algorithm]

↓

- $O(n^3)$ algorithm even on graphs with negative edge lengths
- uses Dynamic Programming

Basic Notation

→ We use adjacency matrix representation of the graph.

- We assume that the vertices are numbered $1, 2, \ldots n$.

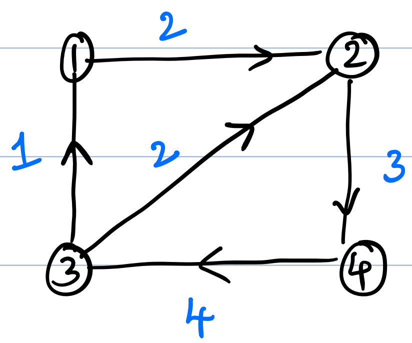
W - weight matrix - $n \times n$
$= (w_{ij})$

$$w_{ij} = \begin{cases} 0 & \text{if } i = j \\ \text{weight of the edge} & \text{if } i \neq j \ (i,j) \in E \\ \infty & \text{if } i \neq j, (i,j) \notin E \end{cases}$$

We allow negative-weighted edges but we assume, graph has no negative weighted cycles.

The output of the APSP algorithm is an $n \times n$ matrix $D = (d_{ij})$, where entry d_{ij} contains the weight of a shortest path from vertex i to vertex j. ie, $d_{ij} = \delta(i,j)$

Example:



$$W = \begin{bmatrix} 0 & 2 & \infty & \infty \\ \infty & 0 & \infty & 3 \\ 1 & 2 & 0 & \infty \\ \infty & \infty & 4 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 2 & 9 & 5 \\ 8 & 0 & 7 & 3 \\ 1 & 2 & 0 & 5 \\ 5 & 6 & 4 & 0 \end{bmatrix}$$

Structure of a Shortest Path:

Q: What is the Maximum length of any shortest Path in a graph with n-vertices.

A: $n-1$.

Consider a Path P from i to j

P contains at most m-edges

if $i \neq j$

$$i \overset{P'}{\rightsquigarrow} k \rightarrow j$$

Where P' contains at most $m-1$ edges

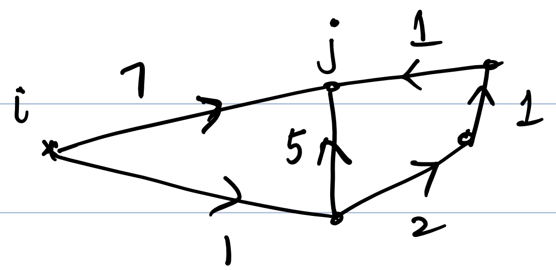
& it is a shortest Path from i to k

So $\delta(i,j) = \delta(i,k) + \omega_{kj}$

A Recursive solution:

Let $l_{ij}^{(m)} \to$ minimum weight of any path from i to j that contains at most m edges

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i=j \\ \infty & \text{if } i \neq j \end{cases}$$



$m \geq 1$

$$l_{ij}^{(m)} = \min\left(l_{ij}^{(m-1)}, \min_{1 \leq k \leq n} \{ l_{ik}^{m-1} + w_{kj} \} \right)$$

$$= \min_{1 \leq k \leq n} \{ l_{ik}^{m-1} + w_{kj} \} \quad (w_{jj} = 0 \ \forall j)$$

Remark: $\delta(i,j) = l_{ij}^{(n-1)} = l_{ij}^{(n)} = \cdots$

because $\delta(i,j) \leq n-1$

Computing Shortest Path weights bottom up.

```
EXTEND-SHORTEST-PATHS(L, W)
n = L.rows
let L' = (l'_ij) be a new n × n matrix
for i = 1 to n
    for j = 1 to n
        l'_ij = ∞
        for k = 1 to n
            l'_ij = min(l'_ij, l_ik + w_kj)
return L'
```

$W = w_{ij}$

$L^{(1)}, L^{(2)}, \ldots L^{(n-1)}$

$L^{(m)} = \left(l_{ij}^{(m)}\right)$

The final matrix $L^{(n-1)}$ contains the actual shortest-path weights.

$L^{(1)} = W$

The above algorithm, Given two matrices $L^{(m-1)}$ & W, Returns $L^{(m)}$.

Running time $= \Theta(n^3)$

$$L^{(1)} = L^{(0)} \cdot W = W$$
$$L^{(2)} = L^{(1)} \cdot W = W^2$$
$$L^{(3)} = \qquad = W^3$$

$$L^{(n-1)} = L^{(n-2)} \cdot W = W^{n-1}.$$

```
SLOW-ALL-PAIRS-SHORTEST-PATHS(W)
n = W.rows
L^(1) = W
for m = 2 to n - 1
    let L^(m) be a new n × n matrix
    L^(m) = EXTEND-SHORTEST-PATHS(L^(m-1), W)
return L^(n-1)
```

$\Big\}\ \Theta(n^4)$

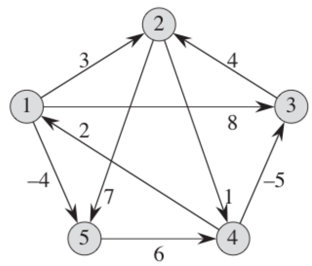
Improving the Running time:

$$L^{(1)} = W$$
$$L^{(2)} = W^2$$
$$L^{(4)} = W^4$$
$$L^{2^{\lceil \log(n-1) \rceil}} = W^{2^{(\log \lceil n-1 \rceil)}}$$

```
FASTER-ALL-PAIRS-SHORTEST-PATHS(W)
n = W.rows
L^(1) = W
m = 1
while m < n - 1
    let L^(2m) be a new n × n matrix
    L^(2m) = EXTEND-SHORTEST-PATHS(L^(m), L^(m))
    m = 2m
return L^(m)
```

$\Big\}\ \Theta(n^3 \log n)$

Example :-



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

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

The final iteration computes $L^{(n-1)}$ by actually computing $L^{(2m)}$ for some $n-1 \leq 2m \leq 2n-2$

But $L^{(2m)} = L^{(n-1)}$

Next, we study a different DP formulation to solve APSP, known as Floyd-Warshall algorithm runs in $\Theta(n^3)$ time.

Optimal Substructure

Main Idea: $V(G) = \{1, 2, 3, \ldots n\}$

Let $V^{(k)} = \{1, 2, \ldots k\}$

The Floyd-Warshall algorithm

- Uses Dynamic Programming
- Running time $= \Theta(n^3)$, n = # of vertices
- No negative cycles, negative weight edges may present.

The Floyd-Warshall algorithm

- Uses Dynamic Programming
- Running time $= \Theta(n^3)$, $n = \#$ of vertices
- No negative cycles

Structure of the Shortest Path:

An intermediate vertex of a simple path $P = \langle v_1, v_2, \ldots v_l \rangle$ is any vertex P other than v_1 & v_l.

Algorithm relies on the following observation

Recall $V = \{1, 2, \ldots n\}$

Consider a subset $\{1, 2, \ldots k\}$ of vertices, for some k

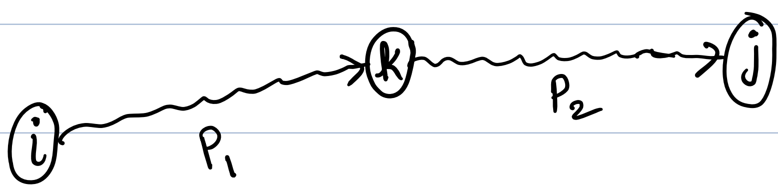
For any pair of vertices $i, j \in V$, consider all paths from i to j whose intermediate vertices are all drawn from $\{1, 2, \ldots k\}$

Let P be a minimum weight path among them

Q: What is the relationship b/w P and shortest paths from i to j with all intermediate vertices are from $\{1, 2, \ldots k-1\}$

1. If k is not an intermediate vertex of path p, then all intermediate vertices of path p are in the set $\{1, 2, \ldots, k-1\}$. Thus, a shortest path from vertex i to vertex j with all intermediate vertices in the set $\{1, 2, \ldots, k-1\}$ is also a shortest path from i to j with all intermediate vertices in the set $\{1, 2, \ldots, k\}$.

2. If k is an intermediate vertex of path p, then we decompose p into $i \overset{p_1}{\leadsto} k \overset{p_2}{\leadsto} j$, as Figure 25.3 illustrates. By Lemma 24.1, p_1 is a shortest path from i to k with all intermediate vertices in the set $\{1, 2, \ldots, k\}$. In fact, we can make a slightly stronger statement. Because vertex k is not an intermediate vertex of path p_1, all intermediate vertices of p_1 are in the set $\{1, 2, \ldots, k-1\}$. Therefore, p_1 is a shortest path from i to k with all intermediate vertices in the set $\{1, 2, \ldots, k-1\}$. Similarly, p_2 is a shortest path from vertex k to vertex j with all intermediate vertices in the set $\{1, 2, \ldots, k-1\}$.



P: all intermediate vertices in $\{1, 2, \ldots k\}$

P_1, P_2: all " " " $\{1, 2, \ldots k-1\}$

A Recursive Solution to APSP:

$d_{ij}^{(k)}$ = weight of a shortest path from i to j for which all intermediate vertices are in the set $\{1, 2, \ldots k\}$

Base Case:

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

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

For any path, all intermediate vertices are in the set $\{1, 2, \ldots n\}$,

$$
d_{ij}^{(n)} = \delta(i, j) \quad \text{for all } i, j \in V \quad \text{— Final answer}
$$

Computing the shortest-path weights bottom up :

```
FLOYD-WARSHALL(W)
n = W.rows
D^(0) = W
for k = 1 to n
    let D^(k) = (d_ij^(k)) be a new n × n matrix
    for i = 1 to n
        for j = 1 to n
            d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
return D^(n)
```

$$
d_{ij}^{(k)} = \min\left(\underbrace{d_{ij}^{(k-1)}}_{\text{Case 1}}, \underbrace{d_{ik}^{(k-1)} + d_{kj}^{(k-1)}}_{\text{Case 2}}\right)
$$

Running time $= \Theta(n^3)$

Q1 How to check if input graph G has a negative cycle?

Ans: $d_{ii}^{(n)} < 0$ for at least one $i \in V(G)$ at the end of the algorithm.

Q2 How to construct a shortest i-j path?

Ans (See next page)

Idea is to compute max label of an internal node on a shortest i-j path.

Constructing a Shortest Path:

We can compute the Predecessor matrix Π while the algorithm computes the matrices $D^{(k)}$

$\Pi^{(0)}, \Pi^{(1)}, \ldots \Pi^{(n)}$,

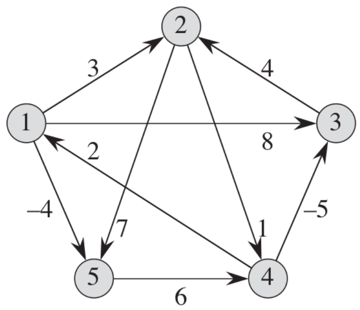
where $\Pi_{ij}^{(k)}$ as the Predecessor of vertex j on a shortest ij Path with all intermediate vertices in the set $\{1, 2, \ldots k\}$

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

$i \rightsquigarrow k \rightsquigarrow j \qquad k \neq j \qquad \{1, 2, \ldots k\}$

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

Example:



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

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

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

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

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

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

How to

Exercise 1 : Construct shortest Paths using the matrix $\Pi^{(k)}$?

Exercise 2: How can we use the O/P of FW algorithm to detect the Presence of a negative-weight cycle?

HINT: There is a negative weight cycle if and only if $d_{ii}^{(n)} < 0$ for some i

[Write the Proof]