Dynamic Programming

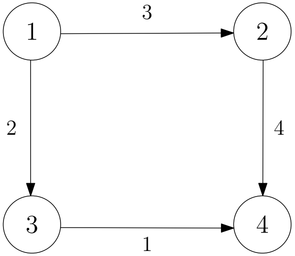
- All Pairs Shortest Paths Problem
- APSP: Dynamic Programming Formulation
- Floyd Warshall Algorithm

Imdad ullah Khan

APSP Problem

Input: A weighted graph $G = (V, E, w)$
Output: Shortest paths from every vertex $u \in V$ to every other $v \in V$

The APSP problem can be represented by a $n \times n$ matrix $D = [d_{ij}]$, where $d_{ij} = d(u_i, u_j)$ for $i, j = 1, \ldots, n$, and n is the number of vertices in V.



	1	2	3	4
1	0	3	2	3
2	∞	0	∞	4
3	∞	∞	0	1
4	∞	∞	∞	0

The goal is to compute the matrix D efficiently.

APSP: Naive Approach

On $G = (V, E, w)$, run a SSSP algorithm from each vertex as source

For non-negative weights, use Dijkstra's algorithm with a priority queue

- $O(n(m + n \log n)) = O(n^2 \log n + nm)$ if G is given as adjacency list
- $O(n^3 \log n)$ if G is given as adjacency matrix

If negative weights are allowed, Bellman-Ford's algorithm

- $O(n^2 m)$ if G is given as adjacency list
- $O(n^4)$ if G is given as adjacency matrix

APSP: Dynamic Programming Formulation

Dynamic programming idea: For any pair of vertices (i, j), consider all possible intermediate vertices k that lie on a shortest path from i to j

Fix some ordering on vertices

Let $d_{ij}^{(k)}$ denote the length of a shortest path from i to j that only uses vertices $\{1, 2, \ldots, k\}$ as intermediate vertices

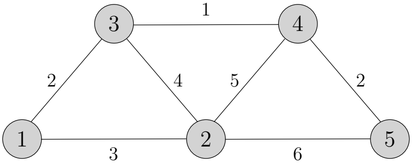
Then we have the following recurrence relation:

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

The base case is $d_{ij}^{(0)} = w_{ij}$, where w_{ij} is the weight of the edge (i, j), or ∞ if there is no such edge

The final solution is $d_{ij}^{(n)}$, where n is the number of vertices in the graph

APSP: Dynamic Programming Formulation



The values of $d_{15}^{(k)}$ for different values of k

k	1	2	3	4	5
$d_{15}^{(k)}$	∞	9	5	5	5

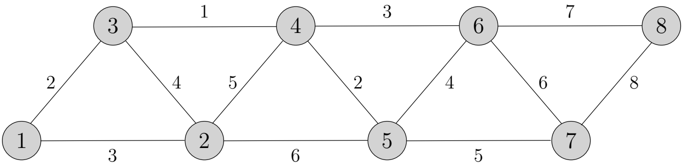
- The shortest path from 1 to 5 is 5, which is achieved when $k = 3$, $k = 4$ or $k = 5$.
- The corresponding shortest path is $(1, 3, 4, 5)$.
- Note that $d_{15}^{(k)}$ does not change when $k > 4$, since vertex 5 is not used as an intermediate vertex.

APSP: Floyd-Warshall Algorithm

Algorithm 1 Floyd-Warshall Algorithm $G = (V, E, w)$

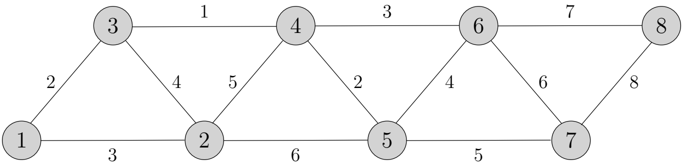
$n \leftarrow$ number of vertices in G
$D^{(0)} \leftarrow$ matrix of edge weights of G
for $k = 1$ to n **do**
  $D^{(k)} \leftarrow$ new $n \times n$ matrix
  for $i = 1$ to n **do**
    for $j = 1$ to n **do**
      $D_{ij}^{(k)} \leftarrow \min\{D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)}\}$
return $D^{(n)}$

APSP: Floyd-Warshall Algorithm



$D^{(0)}$	1	2	3	4	5	6	7	8
1	0	3	2	∞	∞	∞	∞	9
2	3	0	4	5	6	∞	∞	∞
3	2	4	0	1	∞	∞	∞	∞
4	∞	5	1	0	2	3	∞	∞
5	∞	6	∞	2	0	4	5	∞
6	∞	∞	∞	3	4	0	6	7
7	∞	∞	∞	∞	5	6	0	8
8	9	∞	∞	∞	∞	7	8	0

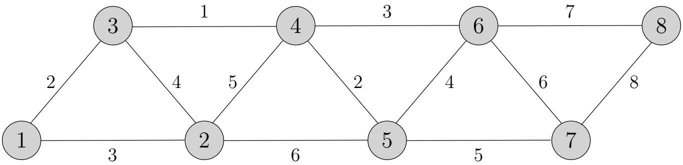
APSP: Floyd-Warshall Algorithm



$D^{(1)}$	1	2	3	4	5	6	7	8
1	0	3	2	∞	∞	∞	∞	9
2	3	0	4	5	6	∞	∞	12
3	2	4	0	1	∞	∞	∞	11
4	∞	5	1	0	2	3	∞	∞
5	∞	6	∞	2	0	4	5	∞
6	∞	∞	∞	3	4	0	6	7
7	∞	∞	∞	∞	5	6	0	8
8	9	12	11	∞	∞	7	8	0

Table: matrix $D^{(1)}$ with vertex 1 as intermediary. updated cells are red

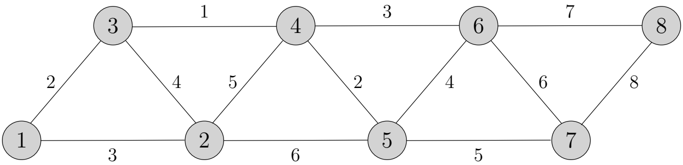
APSP: Floyd-Warshall Algorithm



$D^{(2)}$	1	2	3	4	5	6	7	8
1	0	3	2	8	9	∞	∞	9
2	3	0	4	5	6	∞	∞	12
3	2	4	0	1	10	∞	∞	11
4	8	5	1	0	2	3	∞	17
5	9	6	10	2	0	4	5	18
6	∞	∞	∞	3	4	0	6	7
7	∞	∞	∞	∞	5	6	0	8
8	9	12	11	17	18	7	8	0

Table: matrix $D^{(2)}$ with vertex 2 as intermediary. updated cells are red

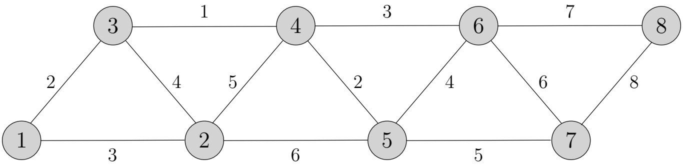
APSP: Floyd-Warshall Algorithm



$D^{(3)}$	1	2	3	4	5	6	7	8
1	0	3	2	3	9	∞	∞	9
2	3	0	4	5	6	∞	∞	12
3	2	4	0	1	10	∞	∞	11
4	3	5	1	0	2	3	∞	12
5	9	6	10	2	0	4	5	18
6	∞	∞	∞	3	4	0	6	7
7	∞	∞	∞	∞	5	6	0	8
8	9	12	11	12	18	7	8	0

Table: matrix $D^{(3)}$ with vertex 3 as intermediary. updated cells are red

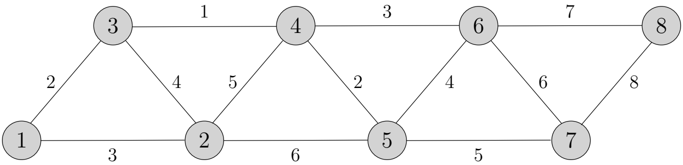
APSP: Floyd-Warshall Algorithm



$D^{(4)}$	1	2	3	4	5	6	7	8
1	0	3	2	3	5	6	∞	9
2	3	0	4	5	6	8	∞	12
3	2	4	0	1	3	4	∞	11
4	3	5	1	0	2	3	∞	12
5	5	6	3	2	0	4	5	14
6	6	8	4	3	4	0	6	7
7	∞	∞	∞	∞	5	6	0	8
8	9	12	11	12	14	7	8	0

Table: matrix $D^{(4)}$ with vertex 4 as intermediary. updated cells are red

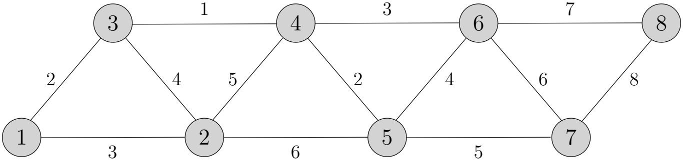
APSP: Floyd-Warshall Algorithm



$D^{(5)}$	1	2	3	4	5	6	7	8
1	0	3	2	3	5	6	10	9
2	3	0	4	5	6	8	11	12
3	2	4	0	1	3	4	8	11
4	3	5	1	0	2	3	7	12
5	5	6	3	2	0	4	5	14
6	6	8	4	3	4	0	6	7
7	10	11	8	7	5	6	0	8
8	9	12	11	12	14	7	8	0

Table: matrix $D^{(5)}$ with vertex 5 as intermediary. updated cells are red

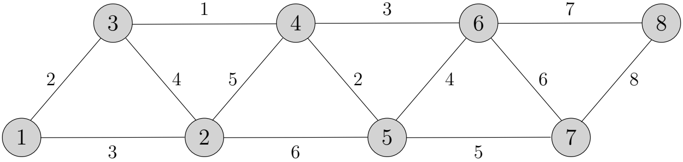
APSP: Floyd-Warshall Algorithm



$D^{(6)}$	1	2	3	4	5	6	7	8
1	0	3	2	3	5	6	10	9
2	3	0	4	5	6	8	11	12
3	2	4	0	1	3	4	8	11
4	3	5	1	0	2	3	7	10
5	5	6	3	2	0	4	5	11
6	6	8	4	3	4	0	6	7
7	10	11	8	7	5	6	0	8
8	9	12	11	10	11	7	8	0

Table: matrix $D^{(6)}$ with vertex 6 as intermediary. updated cells are red

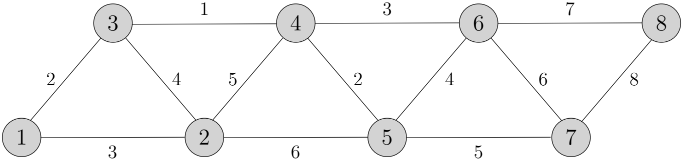
APSP: Floyd-Warshall Algorithm



$D^{(7)}$	1	2	3	4	5	6	7	8
1	0	3	2	3	5	6	10	9
2	3	0	4	5	6	8	11	12
3	2	4	0	1	3	4	8	11
4	3	5	1	0	2	3	7	10
5	5	6	3	2	0	4	5	11
6	6	8	4	3	4	0	6	7
7	10	11	8	7	5	6	0	8
8	9	12	11	10	11	7	8	0

Table: matrix $D^{(7)}$ with vertex 7 as intermediary. updated cells are red

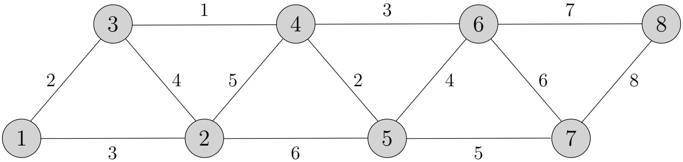
APSP: Floyd-Warshall Algorithm



$D^{(8)}$	1	2	3	4	5	6	7	8
1	0	3	2	3	5	6	10	9
2	3	0	4	5	6	8	11	12
3	2	4	0	1	3	4	8	11
4	3	5	1	0	2	3	7	10
5	5	6	3	2	0	4	5	11
6	6	8	4	3	4	0	6	7
7	10	11	8	7	5	6	0	8
8	9	12	11	10	11	7	8	0

Table: matrix $D^{(8)}$ with vertex 8 as intermediary. updated cells are red

APSP: Floyd-Warshall Algorithm



$D^{(8)}$	1	2	3	4	5	6	7	8
1	0	3	2	3	5	6	10	9
2	3	0	4	5	6	8	11	12
3	2	4	0	1	3	4	8	11
4	3	5	1	0	2	3	7	10
5	5	6	3	2	0	4	5	11
6	6	8	4	3	4	0	6	7
7	10	11	8	7	5	6	0	8
8	9	12	11	10	11	7	8	0

Table: matrix $D^{(8)}$ with vertex 8 as intermediary. updated cells are red

- $D^{(8)}$ gives the shortest path between all pairs
- distance from 1 to 7 is 10, achieved by path $(1, 2, 5, 7)$
- distance from 3 to 6 is 4, achieved by path $(3, 4, 6)$
- distance from 8 to 1 is 9, achieved by path $(8, 1)$