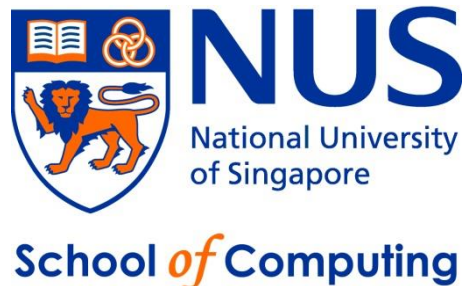


CS2010 – Data Structures and Algorithms II

Lecture 12 – Four Lines Wonder Finding Shortest Paths between All Pairs of Points

chongket@comp.nus.edu.sg



Outline

- Review: The **Single-Source** Shortest Paths Problem
- Introducing: The **All-Pairs** Shortest Paths Problem
 - With One motivating example
- Floyd Warshall's **Dynamic Programming** algorithm
 - The short code 😊 + Basic Idea (discussed in class)
 - The DP formulation (not discussed in class)
- Cool Floyd Warshall's variants

The SSSP problem is about...

1. Finding the shortest path between **a pair** of vertices in the graph (source to destination)
2. Finding the shortest paths between **any pair** of vertices
3. Finding the shortest paths between one vertex to the other vertices in the graph

The four lines wonder

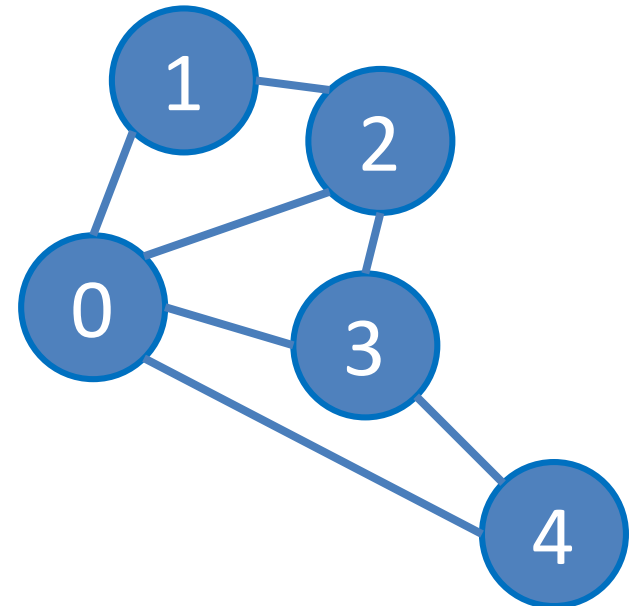
ALL-PAIRS SHORTEST PATHS

Motivating Problem

Diameter of a Graph

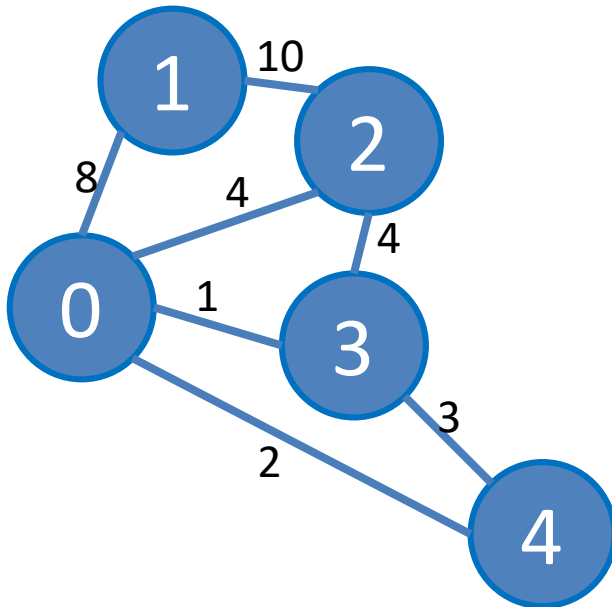
The diameter of a graph is defined as the **greatest shortest path distance** between any pair of vertices

- For example, the diameter of this graph is **2**
 - The paths with length equal to diameter are:
 - 1-0-3 (or the reverse path)
 - 1-2-3 (or the reverse path)
 - 1-0-4 (or the reverse path)
 - 2-0-4 (or the reverse path)
 - 2-3-4 (or the reverse path)



What is the diameter of this graph?

1. 8, path = _____
2. 10, path = _____
3. 12, path = _____
4. I do not know ☹...



All-Pairs Shortest Paths (APSP)

Simple problem definition:

Find the shortest paths between any pair of vertices in the given directed weighted graph

APSP Solutions with SSSP Algorithms

Several solutions from what we have known earlier:

- On unweighted graph
 - Call BFS V times, once from each vertex
 - Time complexity: $O(V * (V+E)) = O(V^3)$ if $E = O(V^2)$
- On weighted graph, for simplicity, non (-ve) weighted graph
 - Call Bellman Ford's V times, once from each vertex
 - Time complexity: $O(V * VE) = O(V^4)$ if $E = O(V^2)$
 - Call Dijkstra's V times, once from each vertex
 - Time complexity: $O(V * (V+E) * \log V) = O(V^3 \log V)$ if $E = O(V^2)$

APSP Solution: Floyd Warshall's

Floyd Warshall's uses an **2D Matrix** for SP cost: $D[|V|][|V|]$

- At start, $D[i][i] = 0$, $D[i][j]$ = the weight of **edge(i, j)** if there is an edge $i \rightarrow j$, otherwise it is ∞
- After Floyd Warshall's stop, it contains the weight of **shortestpath(i, j)**

```
for (int k = 0; k < V; k++) // remember, k first
    for (int i = 0; i < V; i++) // before i
        for (int j = 0; j < V; j++) // then j
            D[i][j] = Math.min(D[i][j], D[i][k] + D[k][j]);
```

PS6 B



If $(D[i][k] + D[k][j] < D[i][j])$

$D[i][j] = D[i][k] + D[k][j]$

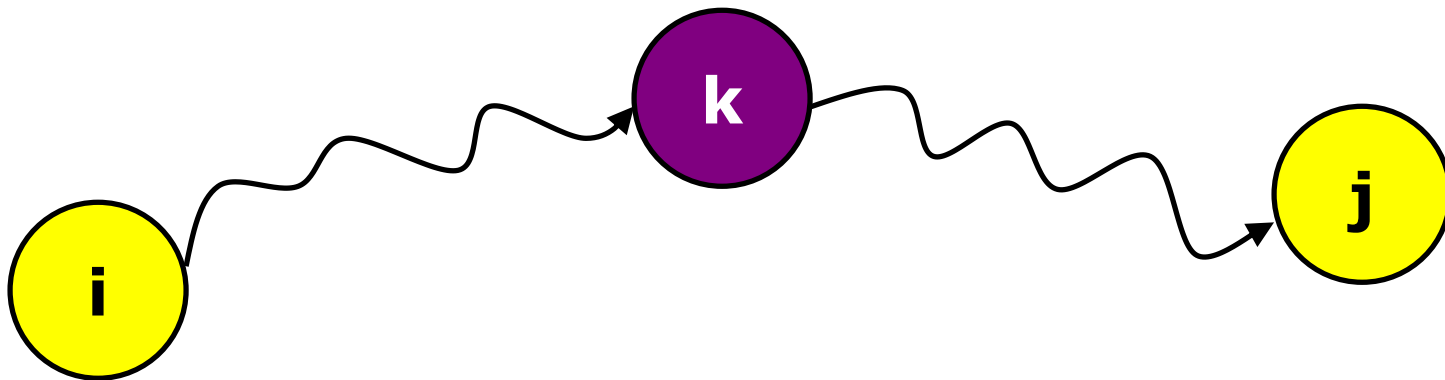
Relaxation of $D[i][j]$

It runs in $O(V^3)$ since we have three nested loops!

- PS: Apparently, if we only given a short amount of time, we can only solve the APSP problem for small graphs, as none of the APSP solution in this and last slides runs better than $O(V^3)$

Floyd Warshall's – Basic Idea

- Assume that the vertices are labeled as $[0 .. V - 1]$.
- Now let $\mathbf{sp(i, j, k)}$ denotes the shortest path between vertex \mathbf{i} and vertex \mathbf{j} with the restriction that the vertices on the shortest path (excluding \mathbf{i} and \mathbf{j}) can only consist of vertices from $[0 .. \mathbf{k}]$
 - How Robert Floyd and Stephen Warshall managed to arrive at this formulation is *beyond this lecture...*
- Initially $\mathbf{k} = -1$ (or to say, we only use direct edges only)
 - Then, iteratively add \mathbf{k} by one until $\mathbf{k} = V - 1$



Floyd Warshall's – DP

Recursive Solution / Optimal Sub-structure

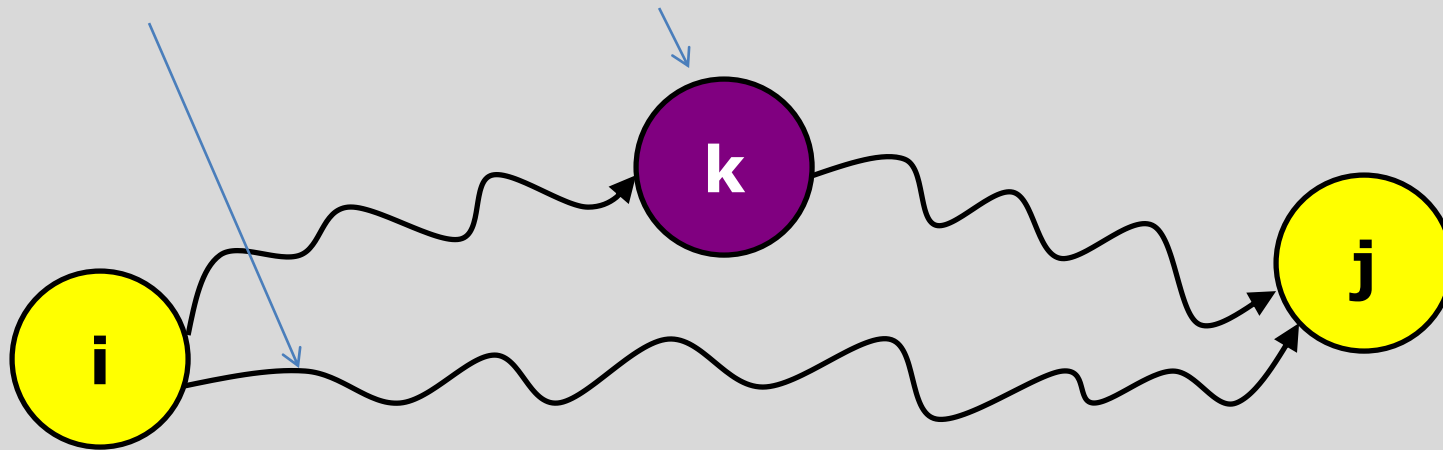
$D_{i,j}^{-1}$: Shortest distance from i to j involving only direct edge from i to j

$D_{i,j}^k$: Shortest distance from i to j involving only $[0..k]$ as intermediate vertices

$$D_{i,j}^k = \begin{cases} w_{i,j} & \text{for } k = -1 \\ \min(D_{i,j}^{k-1}, D_{i,k}^{k-1} + D_{k,j}^{k-1}) & \text{for } k \geq 0 \end{cases}$$

Not using vertex k

Using vertex k



Recurrence seems to imply need for a 3D array since 3 parameters i,j,k . How this can be reduced to 2D array using parameters i,j only is beyond this lecture

Usefulness of APSP:

Preprocessing Step (for lots of queries)

This is another problem solving technique

- Preprocess the data once (can be a costly operation)
- All future queries (of which there is a lot) can be (much) faster by working on the processed data

Example with the APSP problem:

- Once we have pre-processed the APSP information with $O(V^3)$ Floyd Warshall's algorithm...
 - Future queries that ask “*what is the shortest path cost between vertex i and j* ” can now be answered in $O(1)$...

VARIANTS OF FLOYD WARSHALL'S

Variant 1 – Print the Actual SP (1)

We have learned to use array/Vector p (predecessor/parent) to record the BFS/DFS/SP Spanning Tree

- But now, we are dealing with **all-pairs** of paths :O

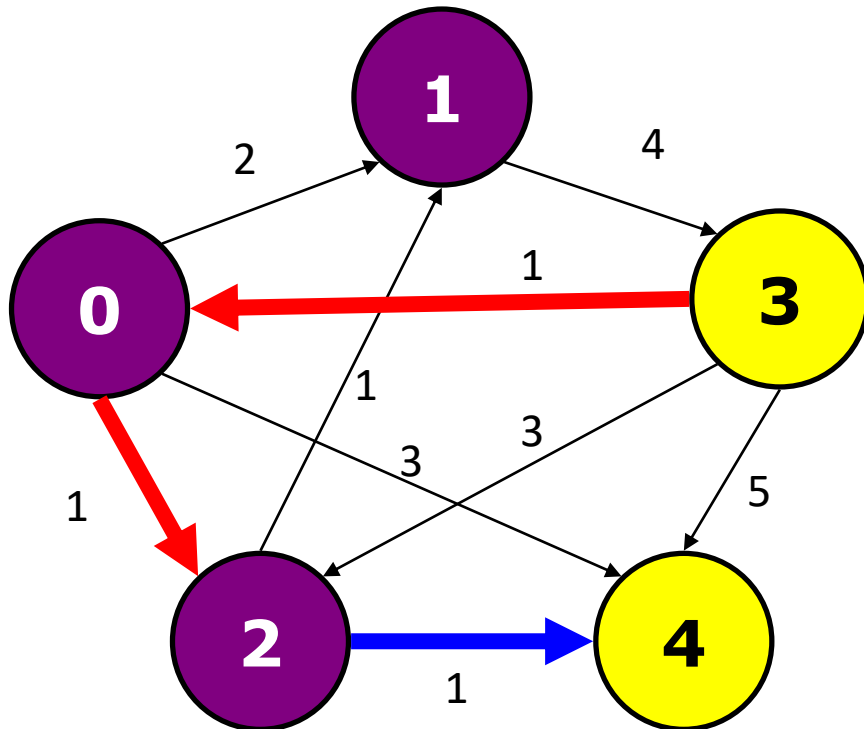
Solution: Use predecessor **matrix** p

- let p be a 2D predecessor matrix, where $p[i][j]$ is the predecessor of j on a shortest path from i to j , i.e. $i \rightarrow \dots \rightarrow p[i][j] \rightarrow j$
- Initially, $p[i][j] = i$ for all pairs of i and j
- If $D[i][k] + D[k][j] < D[i][j]$, then $D[i][j] = D[i][k] + D[k][j]$ and $p[i][j] = p[k][j] \leftarrow$ this will be the predecessor of j in the shortest path

Variant 1 – Print the Actual SP (2)

The two matrices, **D** and **p**

- The shortest path from 3 \leadsto 4
– 3 \rightarrow 0 \rightarrow 2 \rightarrow 4



D	0	1	2	3	4
0	0	2	1	6	2
1	5	0	6	4	7
2	6	1	0	5	1
3	1	3	2	0	3
4	∞	∞	∞	∞	0

p	0	1	2	3	4
0	0	0	0	1	2
1	3	1	0	1	2
2	3	2	2	1	2
3	3	0	0	3	2
4	4	4	4	4	4

Variant 2 – Transitive Closure (1)

Floyd Warshall's algorithm was initially invented for solving the **transitive closure problem**

- Given a graph, determine if vertex i is connected to vertex j either directly (via an edge) or indirectly (via a path)

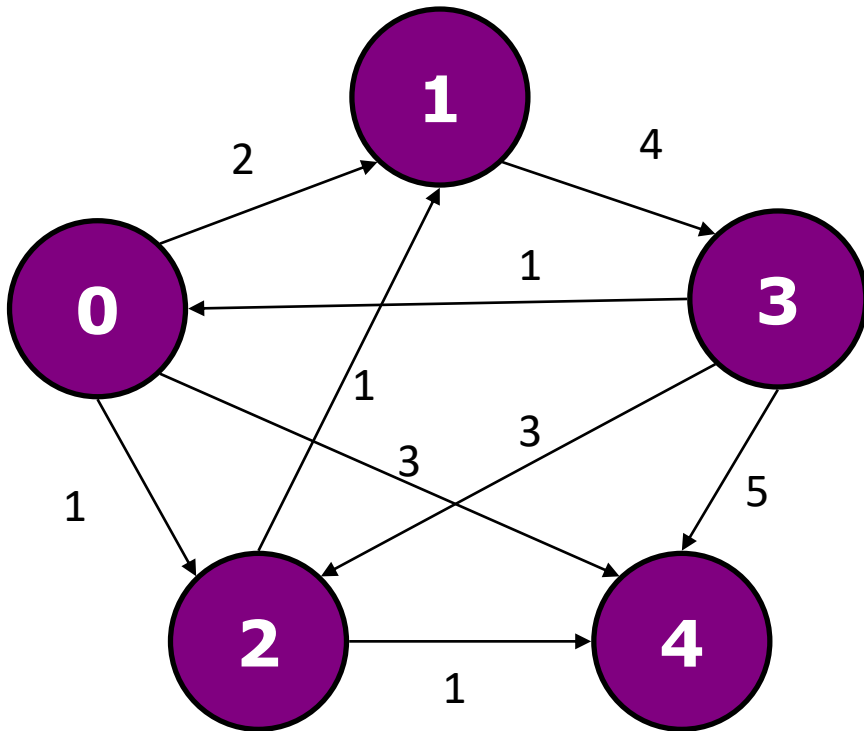
Solution: Modify the matrix D to contain only 0/1

- Modification of Floyd Warshall's algorithm:

```
// Initially:  $D[i][i] = 0$   
//  $D[i][j] = 1$  if edge( $i, j$ ) exist; 0 otherwise  
// the three nested loops as per normal  
 $D[i][j] = D[i][j] \mid (D[i][k] \ \& \ D[k][j]);$  // faster
```


Variant 2 – Transitive Closure (2)

The matrix **D**,
before and after



D _{init}	0	1	2	3	4
0	0	1	1	0	1
1	0	0	0	1	0
2	0	1	0	0	1
3	1	0	1	0	1
4	0	0	0	0	0

D _{final}	0	1	2	3	4
0	1	1	1	1	1
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	0	0	0	0	0

Variant 3 – Minimax/Maximin (1)

The minimax problem is a problem of finding the minimum of maximum edge weight along all possible paths from vertex i to vertex j (maximin is the reverse)

- For a single path from i to j , we pick the maximum edge weight along this path
- Then, for all possible paths from i to j , we pick the one with the minimum max-edge-weight
- $D[i][j]$ will store this minimum max-edge-weight

PS4 A+B

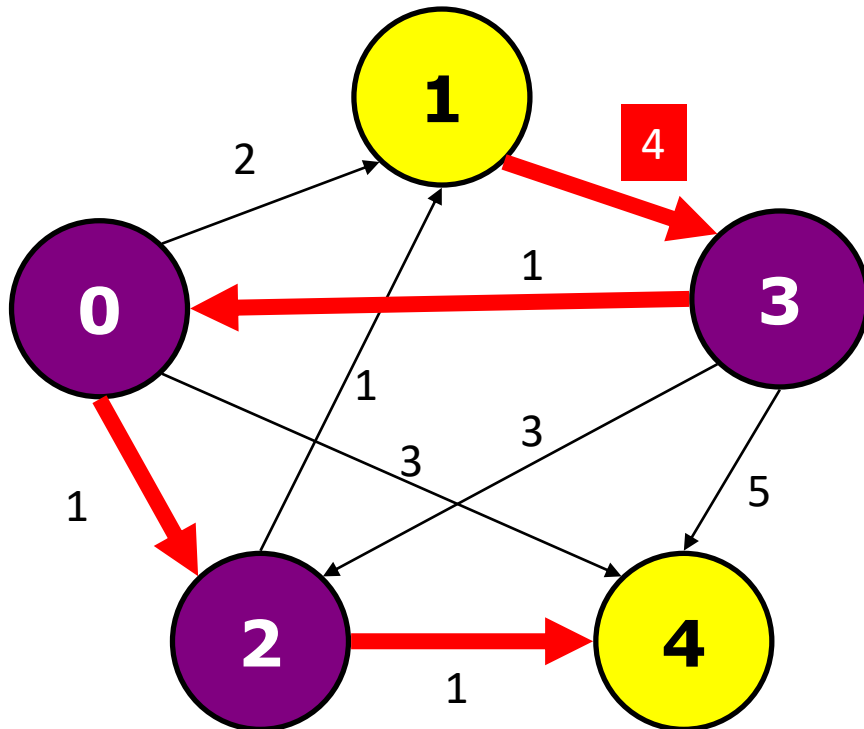
Solution: Again, a modification of Floyd Warshall's

```
// Initially:  $D[i][i] = 0$   
//  $D[i][j]$  = weight of edge( $i, j$ ) exist; INF otherwise  
// the three nested loops as per normal  
 $D[i][j] = \text{Math.min}(D[i][j], \text{Math.max}(D[i][k], D[k][j]));$ 
```

Variant 3 – Minimax/Maximin (2)

The minimax path from 1 to 4 is 4, via edge (1, 3)

- $1 \rightarrow 3 \rightarrow 0 \rightarrow 2 \rightarrow 4$

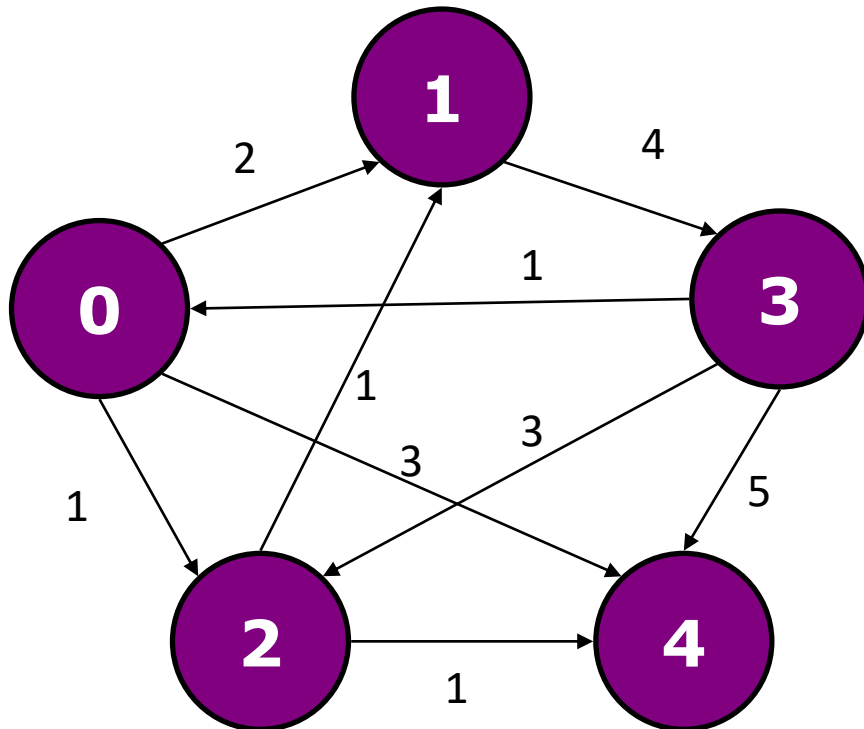


D _{init}	0	1	2	3	4
0	0	2	1	∞	3
1	∞	0	∞	4	∞
2	∞	1	0	∞	1
3	1	∞	3	0	5
4	∞	∞	∞	∞	0

D _{final}	0	1	2	3	4
0	0	1	1	4	1
1	4	0	4	4	4
2	4	1	0	4	1
3	1	1	1	0	1
4	∞	∞	∞	∞	0

Variant 4 – Detecting +ve/-ve Cycle

1. Set the main diagonal of D to ∞
2. Run Floyd Warshall's
3. Recheck the main diagonal
 - I. $< \infty$ but $\geq 0 \rightarrow$ positive cycle
 - II. $< 0 \rightarrow$ negative cycle



D _{init}	0	1	2	3	4
0	∞	2	1	∞	3
1	∞	∞	∞	4	∞
2	∞	1	∞	∞	1
3	1	∞	3	∞	5
4	∞	∞	∞	∞	∞

D _{final}	0	1	2	3	4
0	7	2	1	6	2
1	5	7	6	4	7
2	6	1	7	5	1
3	1	3	2	7	3
4	∞	∞	∞	∞	∞

Java Implementations

See `FloydWarshallDemo.java` for more details

- These four variants are listed inside that demo code

Summary

In this lecture, we have seen:

- Introduction to the APSP problem (with 1 motivating example)
- Introduction to the Floyd Warshall's DP algorithm
 - But many just view this algorithm as “another graph algorithm”
- Introduction to 4 variants of Floyd Warshall's