Design and Analysis of Algorithm
Module 4.3_ Transitive closure

21CS42

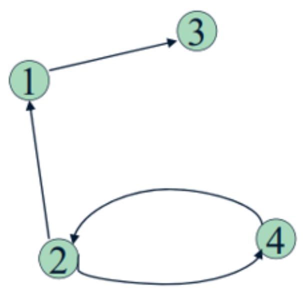
Dr. Avneet Dhingra

CMR Institute of Technology
Bangalore

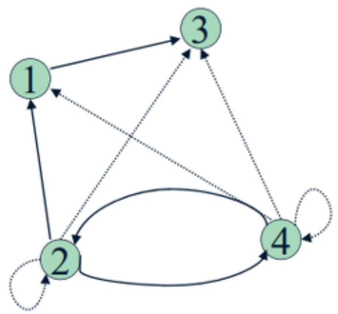
Reference: Design and Analysis of Algorithms, S Sridhar

Warshall's Algorithm: Transitive Closure

- Computes the transitive closure of a relation
- (Alternatively: all paths in a directed graph/ REACHABILITY MATRIX)
- Example of transitive closure:



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



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

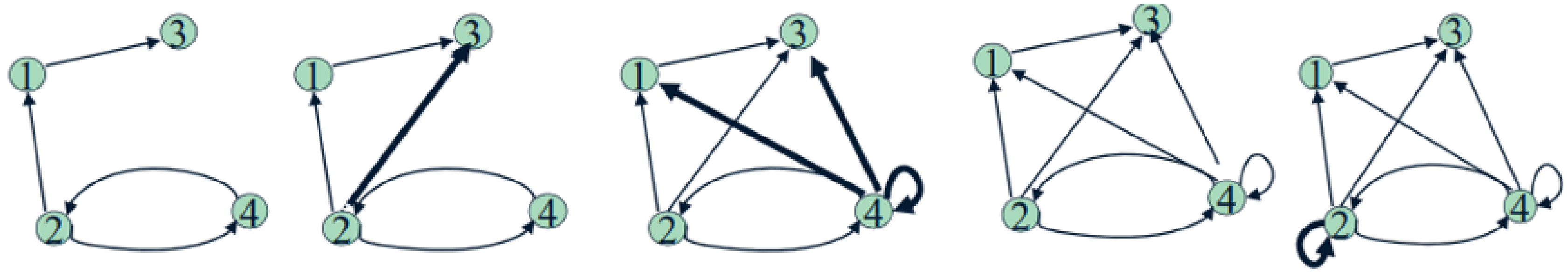
Warshall’s Algorithm using Dynamic Programming

- To find the existence of path between all the pair of vertices in a given weighted connected graph
- Applicable to both directed and undirected weighted graph
- Warshall’s Algorithm is to determine Transitive Closure of a Directed graph or all paths in a directed graph using adjacency matrix
- Generate Transitive Closure of a digraph with the help of DFS or BFS
- Applications:
  - Data flow and control flow dependencies
  - Redundancy checking,
  - Inheritance testing in object oriented software

Warshall’s Algorithm

Main idea:

- A path exists between two vertices i and j, if and only if
  - There is a n edge from i to j; or
  - There is a path from i to j going through vertex 1 and/or 2;
  - There is a path from I to j going through vertex 1,2, and/or 3; or
  - …
  - There is a path from I to j going through any of the other vertices

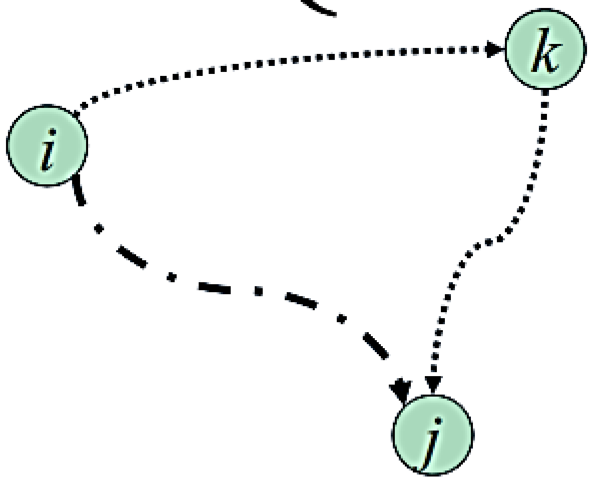


R_0	R_1	R_2	R_3	R_4
$\begin{matrix}0&0&1&0\\1&0&0&1\\0&0&0&0\\0&1&0&0\end{matrix}$	$\begin{matrix}0&0&1&0\\1&0&\mathbf{1}&1\\0&0&0&0\\0&1&0&0\end{matrix}$	$\begin{matrix}0&0&1&0\\1&0&1&1\\0&0&0&0\\\mathbf{1}&1&\mathbf{1}&\mathbf{1}\end{matrix}$	$\begin{matrix}0&0&1&0\\1&0&1&1\\0&0&0&0\\1&1&1&1\end{matrix}$	$\begin{matrix}0&0&1&0\\1&\mathbf{1}&1&1\\0&0&0&0\\1&1&1&1\end{matrix}$

Warshall's Algorithm

- **On the k^{th} iteration, the algorithm determine if a path exists between two vertices i, j using just vertices among 1,…,k allowed as intermediate**

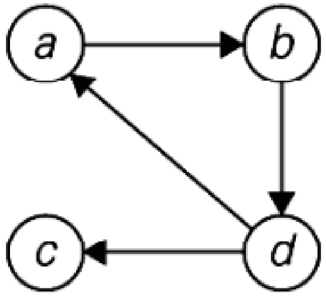
$$R^{(k)}[i,j] = \begin{cases} R^{(k-1)}[i,j] & \text{(path using just 1 ,…,} k\text{-1)} \\ \text{or} & \\ (R^{(k-1)}[i,k] \text{ and } R^{(k-1)}[k,j]) & \text{(path from } i \text{ to } k \text{ and from } k \text{ to } i \text{ using just 1 ,…,} k\text{-1)} \end{cases}$$



k^{th} iteration

Warshall’s Algorithm: Transitive Closure
REACHABILITY MATRIX

Transitive Closure of a directed graph with n vertices can be defined as n x n Boolean matrix $T = \{t_{ij}\}$ in which element in the i^{th} row and j^{th} column is 1 if there exists a non trivial path from i^{th} vertex to j^{th} vertex, otherwise $t_{ij} = 0$.



(a)

$$A = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

(b)

$$T = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

(c)

(a) Digraph. (b) Its adjacency matrix. (c) Its transitive closure.

Warshall's Algorithm (matrix generation)

Recurrence relating elements $R^{(k)}$ to elements of $R^{(k-1)}$ is:

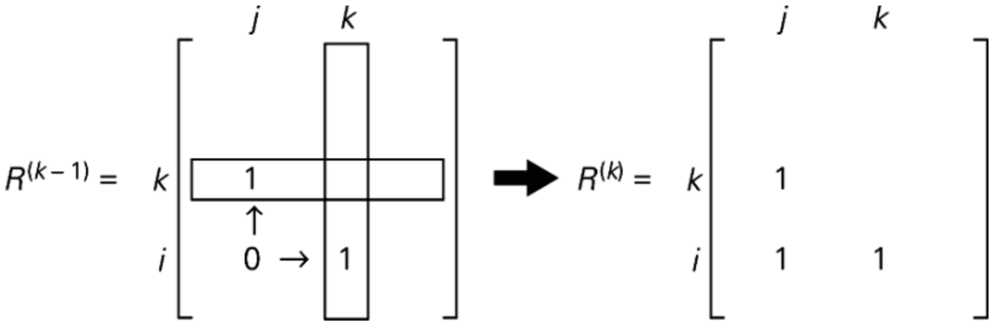
$$R^{(k)}[i,j] = R^{(k-1)}[i,j] \text{ or } (R^{(k-1)}[i,k] \text{ and } R^{(k-1)}[k,j])$$

It implies the following rules for generating $R^{(k)}$ from $R^{(k-1)}$:

Rule 1 If an element in row i and column j is 1 in $R^{(k-1)}$, it remains 1 in $R^{(k)}$

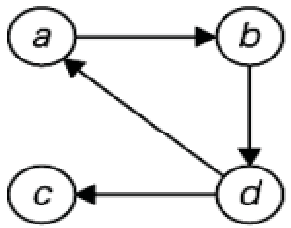
Rule 2 If an element in row i and column j is 0 in $R^{(k-1)}$, it has to be changed to 1 in $R^{(k)}$ if and only if the element in its row i and column k and the element in its column j and row k are both 1's in $R^{(k-1)}$

Warshall's Algorithm: Transitive Closure



Rule for changing zeros in Warshall's algorithm

To check whether there is an existence of path between every pair of vertices



$$R^{(0)} = \begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

Ones reflect the existence of paths with no intermediate vertices ($R^{(0)}$ is just the adjacency matrix); boxed row and column are used for getting $R^{(1)}$.

Step1: Create an adjacency matrix

$$R^{(1)} = \begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & \mathbf{1} & 1 & 0 \end{bmatrix} \end{array}$$

Ones reflect the existence of paths with intermediate vertices numbered not higher than 1, i.e., just vertex *a* (note a new path from *d* to *b*); boxed row and column are used for getting $R^{(2)}$.

Step2: Consider path through vertex ***a***

$$R^{(2)} = \begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & 1 & 0 & \mathbf{1} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & \mathbf{1} \end{bmatrix} \end{array}$$

Ones reflect the existence of paths with intermediate vertices numbered not higher than 2, i.e., *a* and *b* (note two new paths); boxed row and column are used for getting $R^{(3)}$.

Step2: Consider path through vertex ***b***

$$R^{(3)} = \begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Ones reflect the existence of paths with intermediate vertices numbered not higher than 3, i.e., *a*, *b*, and *c* (no new paths); boxed row and column are used for getting $R^{(4)}$.

Step3: Consider path through vertex ***c***

$$R^{(4)} = \begin{array}{cc} & \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} \mathbf{1} & 1 & \mathbf{1} & 1 \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Ones reflect the existence of paths with intermediate vertices numbered not higher than 4, i.e., *a*, *b*, *c*, and *d* (note five new paths).

Step4: Consider path through vertex ***d***

Warshall's Algorithm (pseudocode and analysis)

ALGORITHM *Warshall*$(A[1..n, 1..n])$

//Implements Warshall's algorithm for computing the transitive closure
//Input: The adjacency matrix A of a digraph with n vertices
//Output: The transitive closure of the digraph

$R^{(0)} \leftarrow A$
for $k \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow 1$ **to** n **do**
      $R^{(k)}[i, j] \leftarrow R^{(k-1)}[i, j]$ **or** $(R^{(k-1)}[i, k]$ **and** $R^{(k-1)}[k, j])$
return $R^{(n)}$

Time efficiency: $\Theta(n^3)$

Space efficiency: Matrices can be written over their predecessors

- Warshall Algorithm is used to find whether there is a path from each node to every other node in the given graph

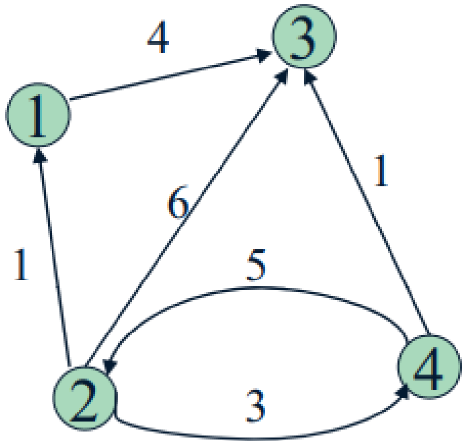
- Floyd’s (Warshall) algorithm is used to find the shortest distance from each node to every other node.

Floyd's Algorithm: All pairs shortest paths

Problem: **In a weighted (di)graph, find shortest paths between every pair of vertices**

Same idea: construct solution through series of matrices $D^{(0)}, \ldots, D^{(n)}$ using increasing subsets of the vertices allowed as intermediate

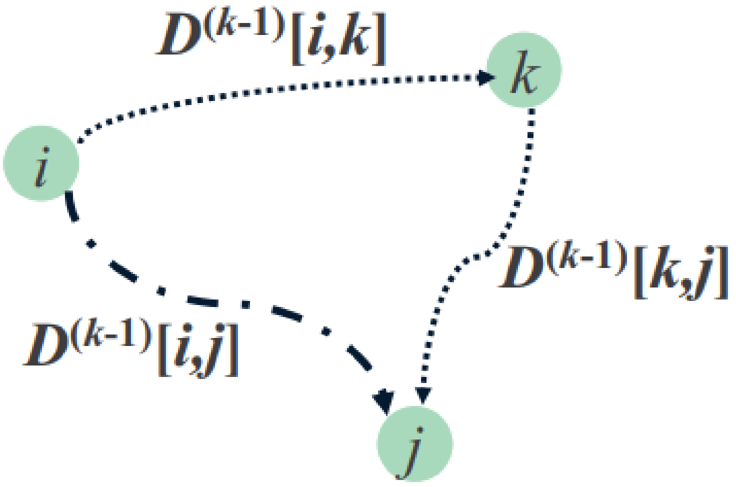
- **Example:**

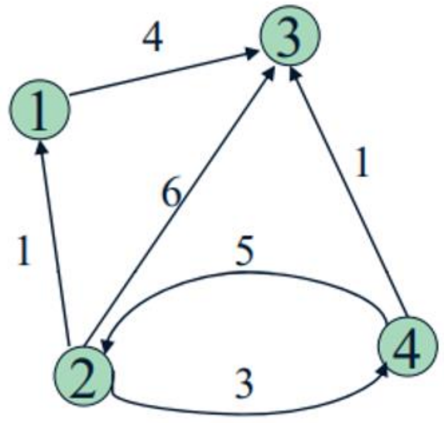


Floyd's Algorithm: Matrix Generation

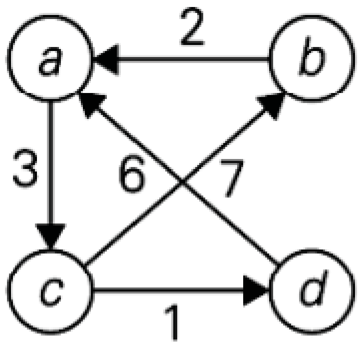
On the *k*-th iteration, the algorithm determines shortest paths between every pair of vertices *i, j* that use only vertices among 1,…,*k* as intermediate

$$D^{(k)}[i,j] = \min\{D^{(k-1)}[i,j],\ D^{(k-1)}[i,k] + D^{(k-1)}[k,j]\}$$





Example:



(a)

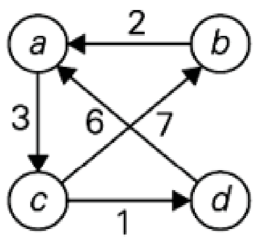
$$W = \begin{array}{c|cccc|} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \infty & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \infty & 0 \end{array}$$

(b)

$$D = \begin{array}{c|cccc|} & a & b & c & d \\ \hline a & 0 & 10 & 3 & 4 \\ b & 2 & 0 & 5 & 6 \\ c & 7 & 7 & 0 & 1 \\ d & 6 & 16 & 9 & 0 \end{array}$$

(c)

(a) Digraph. (b) Its weight matrix. (c) Its distance matrix.



$$D^{(0)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \infty & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \infty & 0 \end{array}$$

Lengths of the shortest paths with no intermediate vertices ($D^{(0)}$ is simply the weight matrix).

Step1: Generate the cost adjacency matrix

$$D^{(1)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \mathbf{5} & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \mathbf{9} & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 1, i.e. just *a* (note two new shortest paths from *b* to *c* and from *d* to *c*).

Step2: Consider shortest path through ***a***

b,c = min{ (b->c), (b->a + a->c)}
= min{ inf, 2+3} =5

$$D^{(2)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & 5 & \infty \\ c & \mathbf{9} & 7 & 0 & 1 \\ d & 6 & \infty & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 2, i.e. a and *b* (note a new shortest path from *c* to *a*).

Step3: Consider shortest path through ***b***

c,a = min{ c->a, c->b + b->a}
= min { inf, 7 + 2} = 9

$$D^{(3)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \mathbf{10} & 3 & \mathbf{4} \\ b & 2 & 0 & 5 & \mathbf{6} \\ c & 9 & 7 & 0 & 1 \\ d & 6 & \mathbf{16} & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 3, i.e. a, *b*, and *c* (note four new shortest paths from *a* to *b*, from a to *d*, from *b* to *d*, and from *d* to *b*).

Step4: Consider shortest path through ***c***

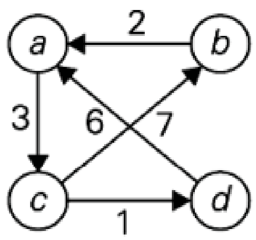
a,d = min{ a->d, a->c + c->d}
= min { inf, 3 + 1} = 4

$$D^{(4)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 10 & 3 & 4 \\ b & 2 & 0 & 5 & 6 \\ c & \mathbf{7} & 7 & 0 & 1 \\ d & 6 & 16 & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 4, i.e. a, *b*, *c*, and *d* (note a new shortest path from *c* to *a*).

Step5: Consider shortest path through ***d***

Number of iterations = number of vertices



$$D^{(0)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \infty & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \infty & 0 \end{array}$$

Lengths of the shortest paths with no intermediate vertices ($D^{(0)}$ is simply the weight matrix).

$$D^{(1)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \mathbf{5} & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \mathbf{9} & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 1, i.e. just *a* (note two new shortest paths from *b* to *c* and from *d* to *c*).

$$D^{(2)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & 5 & \infty \\ c & \mathbf{9} & 7 & 0 & 1 \\ d & 6 & \infty & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 2, i.e. a and *b* (note a new shortest path from *c* to *a*).

$$D^{(3)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \mathbf{10} & 3 & \mathbf{4} \\ b & 2 & 0 & 5 & \mathbf{6} \\ c & 9 & 7 & 0 & 1 \\ d & 6 & \mathbf{16} & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 3, i.e. a, *b*, and *c* (note four new shortest paths from *a* to *b*, from a to *d*, from *b* to *d*, and from *d* to *b*).

$$D^{(4)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 10 & 3 & 4 \\ b & 2 & 0 & 5 & 6 \\ c & \mathbf{7} & 7 & 0 & 1 \\ d & 6 & 16 & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 4, i.e. a, *b*, *c*, and *d* (note a new shortest path from *c* to *a*).

Path Matrix

$P^{(1)}$

	a	b	c	d
a				
b			1	
c				
d			1	

$P^{(2)}$

	a	b	c	d
a				
b			1	
c	2			
d			1	

$P^{(3)}$

	a	b	c	d
a		3		3
b			1	3
c	2			
d		3	1	

$P^{(4)}$

	a	b	c	d
a		3		3
b			1	3
c	4			
d		3	1	

c→a
c->d -> a

Number of iterations = number of vertices

Floyd’s Algorithm (Pseudocode and Analysis)

```
ALGORITHM  Floyd(W[1..n, 1..n])
    //Implements Floyd's algorithm for the all-pairs shortest-paths problem
    //Input: The weight matrix W of a graph with no negative-length cycle
    //Output: The distance matrix of the shortest paths' lengths
    D ← W //is not necessary if W can be overwritten
    for k ← 1 to n do
        for i ← 1 to n do
            for j ← 1 to n do
                D[i, j] ← min{D[i, j], D[i, k] + D[k, j]}
    return D
```

Time efficiency: $\Theta(n^3)$

Space efficiency: Matrices can be written over their predecessors

Shortest Path Reconstruction

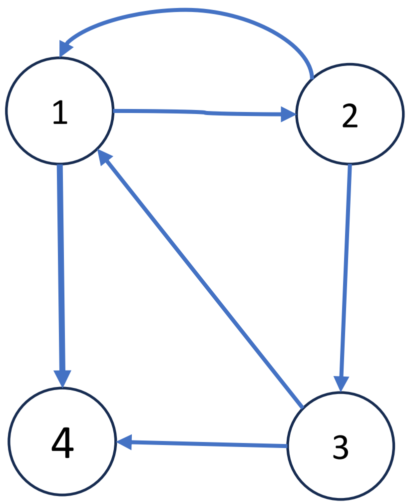
```
Algorithm path_reconstruct(P,i, j)
//Input: Path matrix P
//Output: Recostructed Path
{
   if (i==j)  output i;
   else if (P[i,j]==0)
          {   output 'No path exists';}
          else
               path_reconstruct(P, i, P[i,j])
               output the value of j
    endif
}
```

Here, P[][] is a path matrix which stores the predecessor node for path from I to j.

It is computed alongwith computation of matrix D. It records the value of *k* which gives the minimum value at every iteration.

P[i][j] = 0 if i=j   and if no path exists between i and j

Apply Warshall algorithm and find transitive closure for the graph shown below:

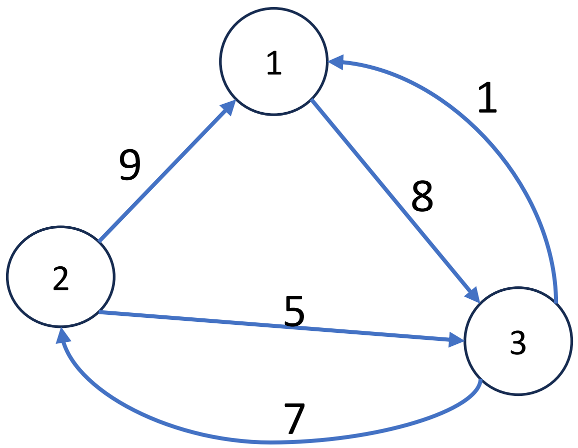


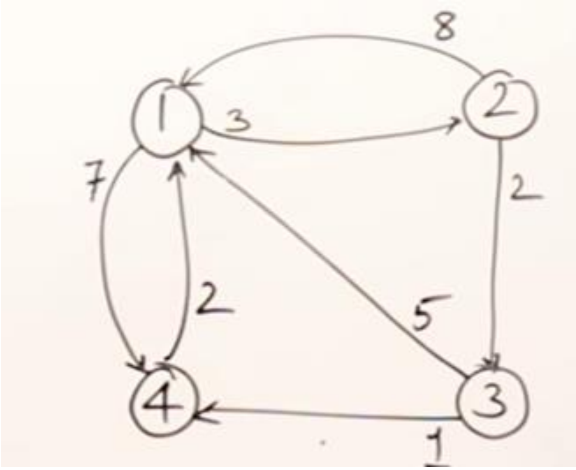
$R^{(0)}$

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

$$R^{(k)}[i,j] = R^{(k-1)}[i,j] \text{ or } (R^{(k-1)}[i,k] \text{ and } R^{(k-1)}[k,j])$$

Floyd's Algorithm Practice:





$$D^{(k)}[i,j] = \min\{D^{(k-1)}[i,j],\ D^{(k-1)}[i,k] + D^{(k-1)}[k,j]\}$$