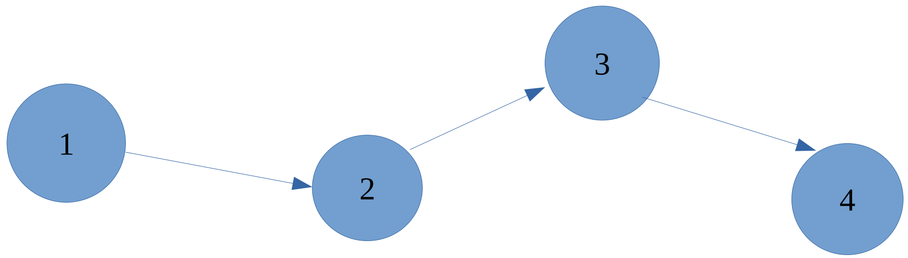
Transitive Closure and Warshall's algorithms

What is ?



Is there exist path from 1 to 4 ?

>> Direct path, >> indirect path (via, other nodes.)

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

3	0	0	0	1
4	0	0	0	0

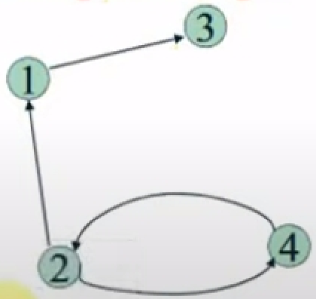
Transitive closure of above graph

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

Warshall’s algorithm: For Transitive closure

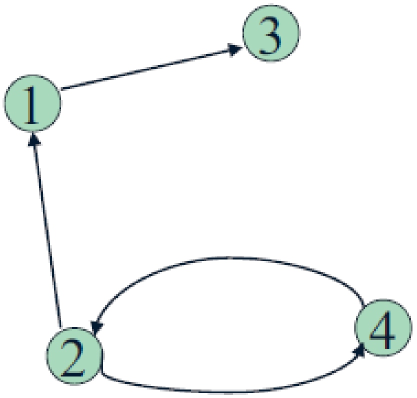
- Computes the transitive closure of a relation
- (Alternatively: **all paths in a directed graph**)
- Example of transitive closure

Example Graph

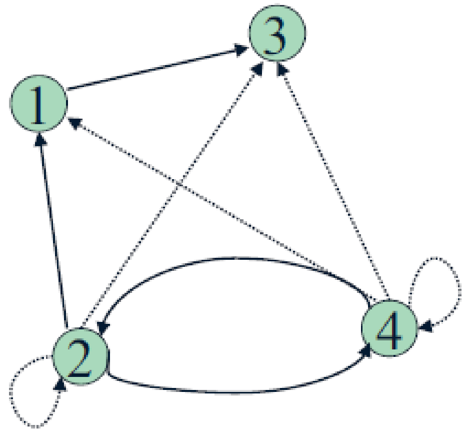


$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

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

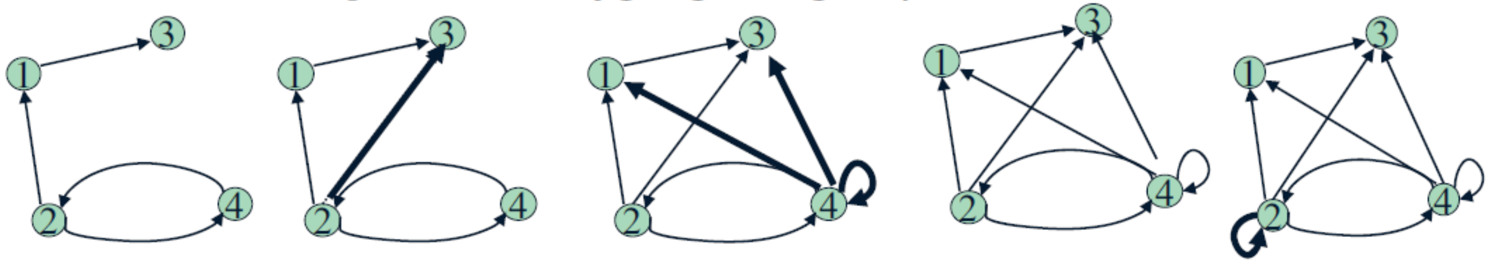


$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$



$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & \mathbf{1} & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 \\ \mathbf{1} & 1 & \mathbf{1} & \mathbf{1} \end{matrix}$$

- **Main idea: a path exists between two vertices i, j, iff**
  - **there is an edge from i to j; or**
  - **there is a path from i to j going through vertex 1; or**
  - **there is a path from i to j going through vertex 1 and/or 2; or**
  - **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
$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

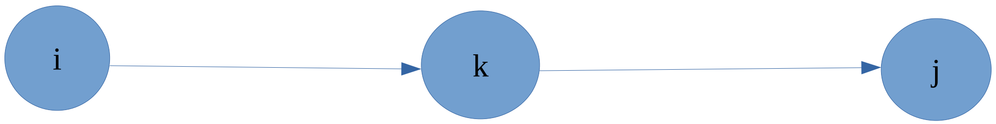
R_1
$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

R_2
$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ \mathbf{1} & 1 & \mathbf{1} & \mathbf{1} \end{matrix}$$

R_3
$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}$$

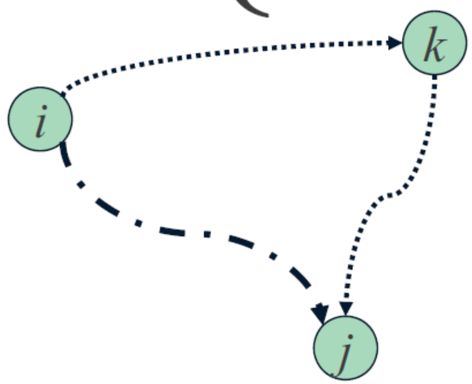
R_4
$$\begin{matrix} 0 & 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}$$

i to j => via k

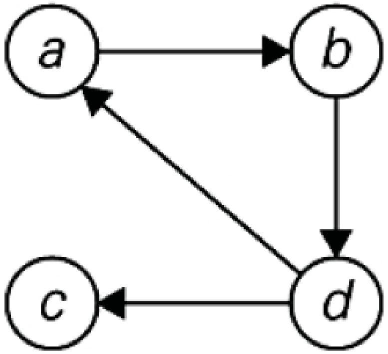


• **On the k^{th} iteration, the algorithm determine if a path exists between two vertices i, j using just vertices among 1,…,k allowed as intermediate**

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



k^{th} iteration



(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)

Here,
(a) Directed Graph
(b) Its Adjacency Matrix
(c) Its Transitive Closure

Warshall's Algorithm

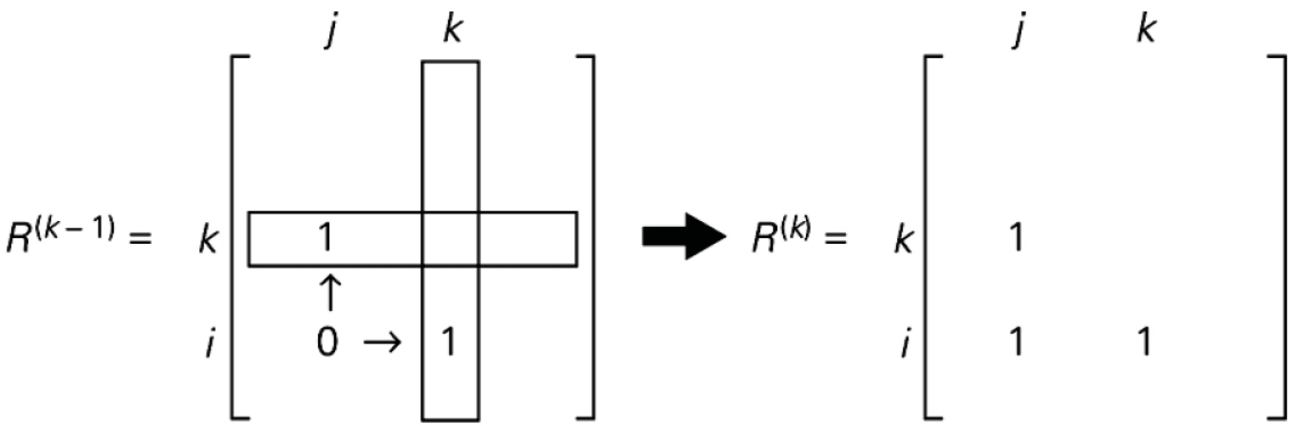
Recurrence relating elements $R^{(k)}$ to elements of $R^{(k-1)}$ is:

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

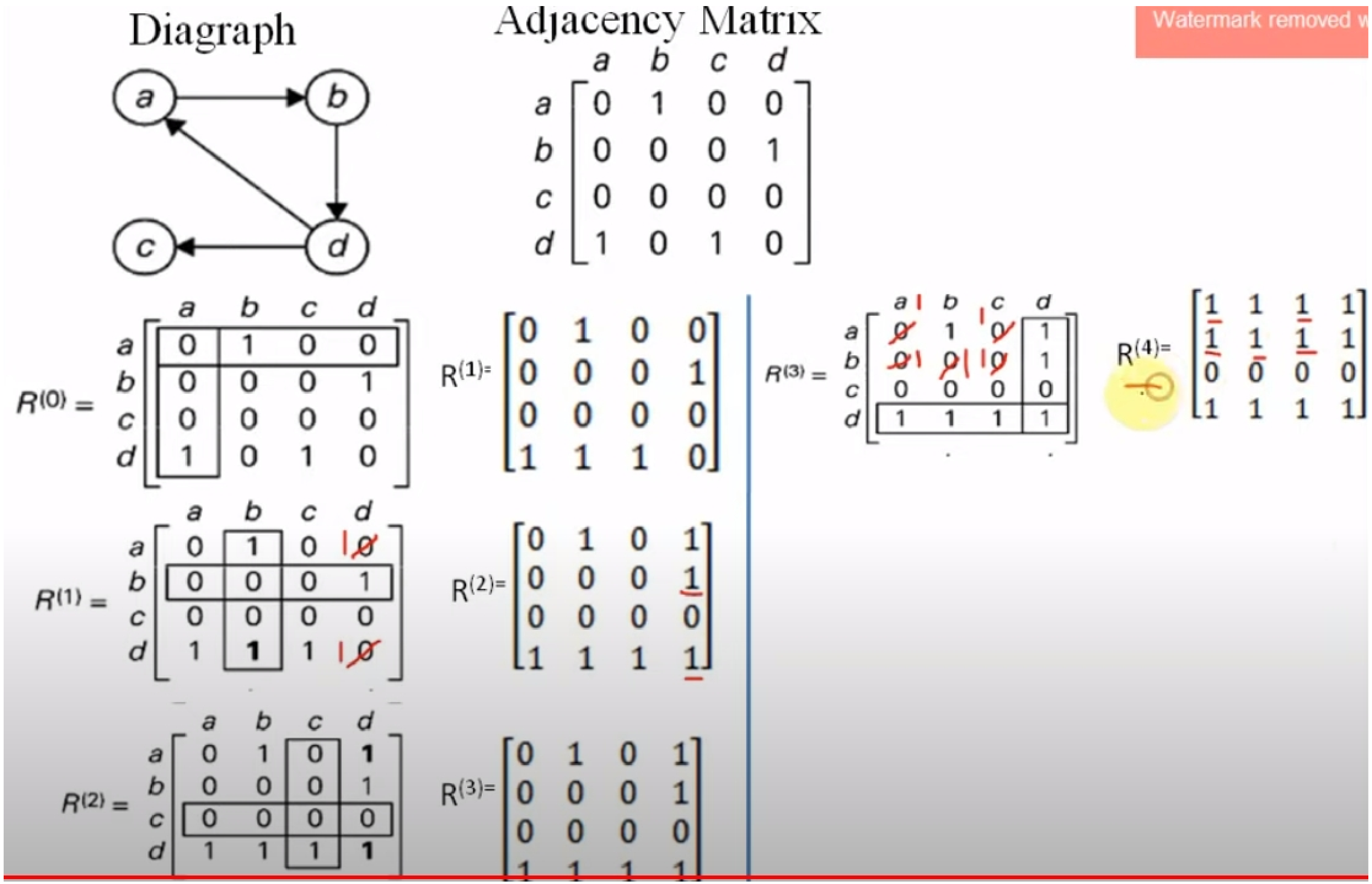
It implies the following rules for generating $R^{(k)}$ from $R^{(k-1)}$:

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 $R^{(k-1)}$



Example:


Diagraph
Adjacency Matrix
Watermark removed w
a
b
c
d
R(0) =
R(1) =
R(2) =
R(1)=
R(2)=
R(3)=
R(3) =
R(4)=

Warshall's Algorithm Pseudo-code

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: $\Theta(n^3)$

Graph Traversals

Two Ways

a. Depth First Traversals [DFS]
   >> Deeper Nodes visited first.
   >> Employed Stacks (Data structure)

b. Bread First Traversals [BFS]
   >> Closer Nodes visited first.
   >> Employed Queues

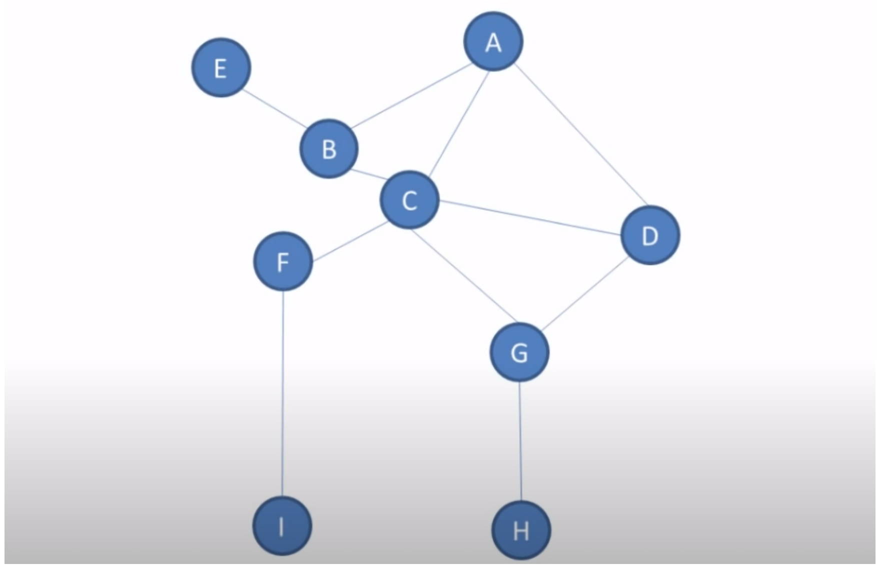
DFS

```
Push the first vertex onto the stack
Mark this vertex as visited
Repeat
    Visit the next vertex adjacent to the one on top of the stack
    Push this vertex onto the stack
    Mark this vertex as visited
    If there isn't a vertex to visit
        Pop this vertex off the stack
    End if
Until the stack is empty
```

BFS

```
Enqueue the first vertex
Mark the first vertex as visited
Repeat
    Visit next vertex adjacent to the first vertex
    Mark this vertex as visited
    Enqueue this vertex
Until all adjacent vertices visited
Repeat
    Dequeue next vertex from the queue
    Repeat
        Visit next unvisited vertex adjacent to that at the front of the queue
        Mark this vertex as visited
        Enqueue this vertex
    Until all adjacent vertices visited
Until the queue is empty
```

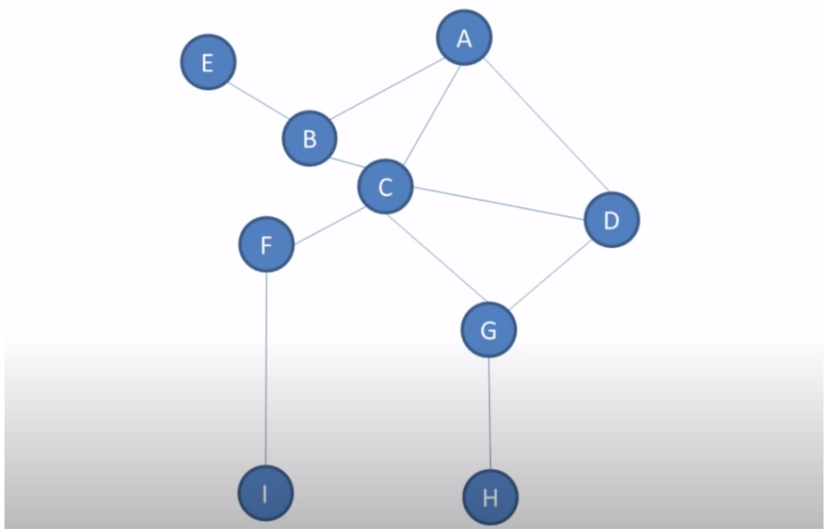
Example



DFS
Root A,

Visited: A, D, G, H, C, F, I, B, E

Stack: [|||||||||||||]



BFS
Root A,

Visited:A, B, C, D, E, F, G, I, H

Queue[]