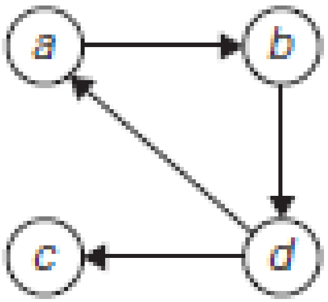
# Unit III – Dynamic Programming

- Dynamic Programming
  - Computing a Binomial Coefficient
  - **Warshall’s algorithm**
  - Floyd’s algorithm
  - Optimal Binary Search Trees
  - Knapsack Problem and Memory functions

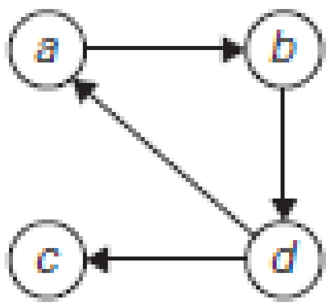
# Warshall's algorithm

- Compute the *Transitive closure* of a directed graph
- The ***transitive closure*** of a directed graph with $n$ vertices can be defined as the $n \times n$ boolean matrix $T = \{tij\}$, in which the element in the $i$th row and the $j$th column is 1 if there exists a nontrivial path (i.e., directed path of a positive length) from the $i$th vertex to the $j$th vertex; otherwise, $tij$ is 0.
- *Example: directed graph (digraph)*



# Warshall's algorithm

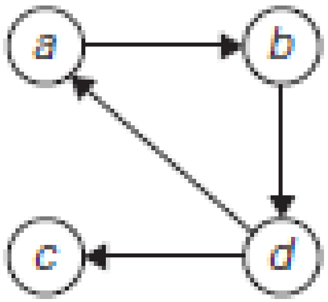
- *Adjacency Matrix* - A= $\{a_{ij}\}$ of a digraph is the boolean matrix that has 1 in the ith row and jth column if and only if there is a directed edge from $i^{th}$ vertex to $j^{th}$ vertex.




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

# Warshall's algorithm

- Transitive closure

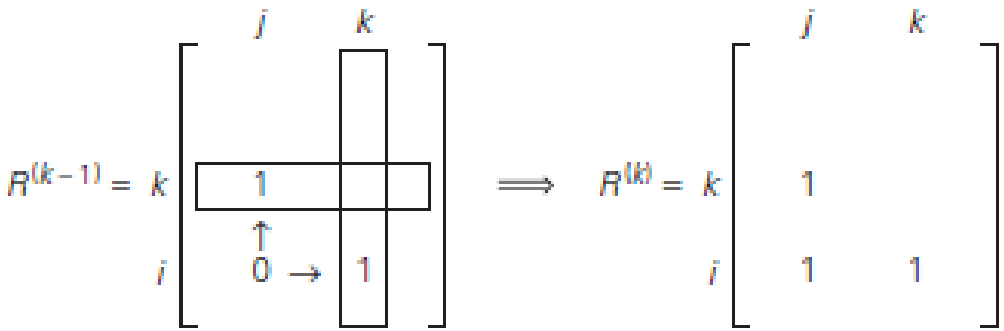



| | a | b | c | d |
|---|---|---|---|---|
| a | 1 | 1 | 1 | 1 |
| b | 1 | 1 | 1 | 1 |
| c | 0 | 0 | 0 | 0 |
| d | 1 | 1 | 1 | 1 |

# Warshall's algorithm

- series of $n \times n$ boolean matrices: $R(0), \ldots, R(k-1), R(k), \ldots R(n)$.
- Matrix value is 1 – using the formula as follows

$$r_{ij}^{(k)} = r_{ij}^{(k-1)} \text{ or } \left( r_{ik}^{(k-1)} \text{ and } r_{kj}^{(k-1)} \right)$$




Rule for changing zeros in Warshall's algorithm.

# Warshall's algorithm – Example

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

| $\mathbf{R_0}$ | a | b | c | D |
|---|---|---|---|---|
| a | **0** | **1** | **0** | **0** |
| b | **0** | 0 | 0 | 1 |
| c | **0** | 0 | 0 | 0 |
| d | **1** | 0 | 1 | 0 |

| $\mathbf{R_1}$ | a | b | c | D |
|---|---|---|---|---|
| a | 0 | **1** | 0 | 0 |
| b | **0** | **0** | **0** | **1** |
| c | 0 | **0** | 0 | 0 |
| d | 1 | **1** | 1 | 0 |

| $\mathbf{R_2}$ | a | b | c | D |
|---|---|---|---|---|
| a | 0 | 1 | **0** | 1 |
| b | 0 | 0 | **0** | 1 |
| c | **0** | **0** | **0** | **0** |
| d | 1 | 1 | **1** | 1 |

| $\mathbf{R_3}$ | a | b | c | D |
|---|---|---|---|---|
| a | 0 | 1 | 0 | **1** |
| b | 0 | 0 | 0 | **1** |
| c | 0 | 0 | 0 | **0** |
| d | **1** | **1** | **1** | **1** |

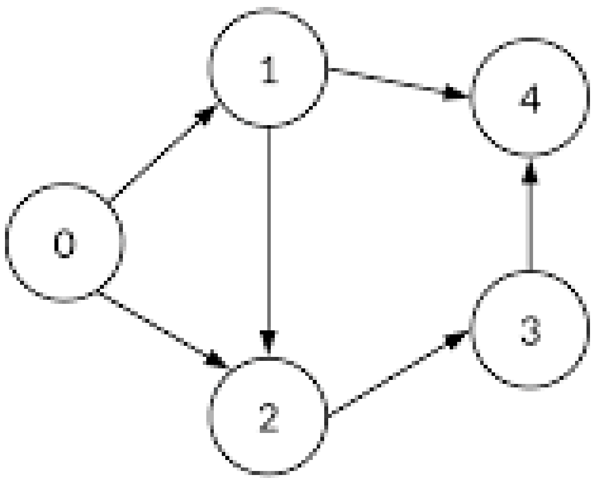
| $\mathbf{R_4}$ | a | b | c | D |
|---|---|---|---|---|
| a | 1 | 1 | 1 | **1** |
| b | 1 | 1 | 1 | **1** |
| c | 0 | 0 | 0 | **0** |
| d | **1** | **1** | **1** | **1** |

# Warshall's algorithm - Algorithm

**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) ← A
for k ← 1 to n do
    for i ← 1 to n do
        for j ← 1 to n do
            R(k)[i, j] ← R(k−1)[i, j] or (R(k−1)[i, k] and R(k−1)[k, j])
return R(n)
```

# Warshall's algorithm - Example




Adjacency Matrix

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 2 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 |