Book

A Simplified Approach
to

Data Structures

Prof.(Dr.)Vishal Goyal, Professor, Punjabi University Patiala
Dr. Lalit Goyal, Associate Professor, DAV College, Jalandhar
Mr. Pawan Kumar, Assistant Professor, DAV College, Bhatinda

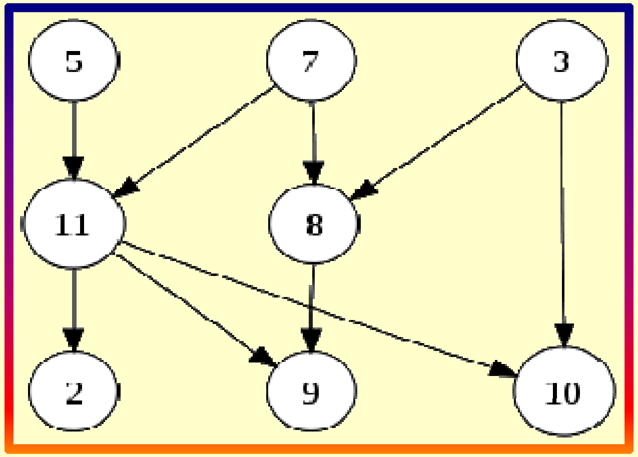
Shroff Publications and Distributors

Edition 2014

Applications of the Graph

Applications of the Graph

- Finding the reachability
- Finding the shortest path
- Spanning Trees



A labeled simple graph:-
Vertex set $V = \{2,3,5,7,8,9,10,11\}$
Edge set $E = \{\{3,8\}, \{3,10\}, \{5,11\},\{7,8\}, \{7,11\}, \{8,9\}, \{11,2\},\{11,9\},\{11,10\}\}$.

Reachability

•It means that whether a particular vertex is reachable from other vertices of the graph or not.

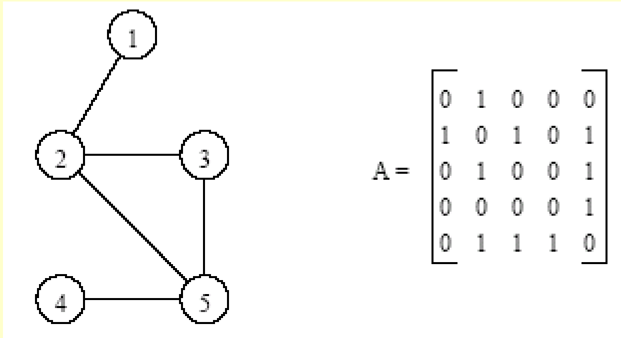
•With the help of reachability matrix of a graph,we can find which vertex of a graph is reachable from which vertex of a graph by 2 ways:-

- ❖ Matrix Multiplication Method
- ❖ Warshall's Algorithm

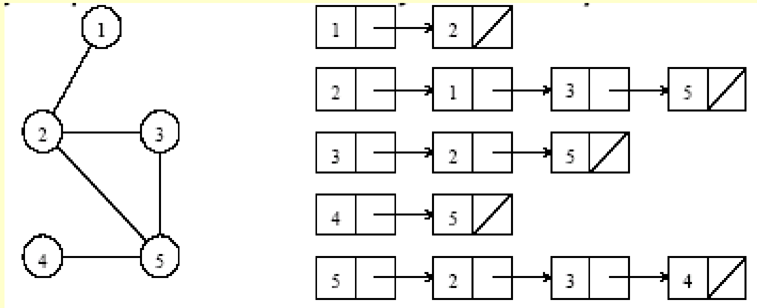
Adjacency Matrix and Adjacency List

Adjacency Matrix:-

The standard adjacency matrix stores a matrix as a 2-D array with each slot in A[i][j] being a 1 if there is an edge from vertex i to vertex j, or storing a 0 otherwise.

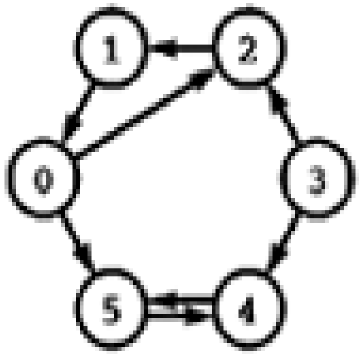


An undirected graph and its adjacency matrix representation.



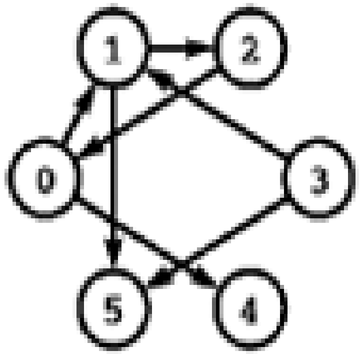
An undirected graph and its adjacency list representation.

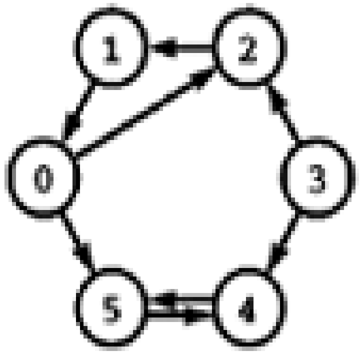
Matrix Multiplication Method



	0	1	2	3	4	5
0	0	0	1	0	0	1
1	1	0	0	0	0	0
2	0	1	0	0	0	0
3	0	0	1	0	1	0
4	0	0	0	0	0	1
5	0	0	0	0	1	0

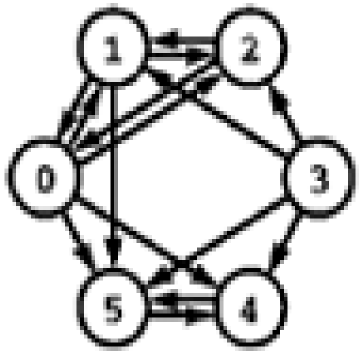
	0	1	2	3	4	5
0	0	1	0	0	1	0
1	0	0	1	0	0	1
2	1	0	0	0	0	0
3	0	1	0	0	0	1
4	0	0	0	0	1	0
5	0	0	0	0	0	1





	0	1	2	3	4	5
0	1	0	1	0	0	1
1	1	1	0	0	0	0
2	0	1	1	0	0	0
3	0	0	1	1	1	0
4	0	0	0	0	1	1
5	0	0	0	0	1	1

	0	1	2	3	4	5
0	1	1	1	0	1	1
1	1	1	1	0	0	1
2	1	1	1	0	0	0
3	0	1	1	1	1	1
4	0	0	0	0	1	1
5	0	0	0	0	1	1



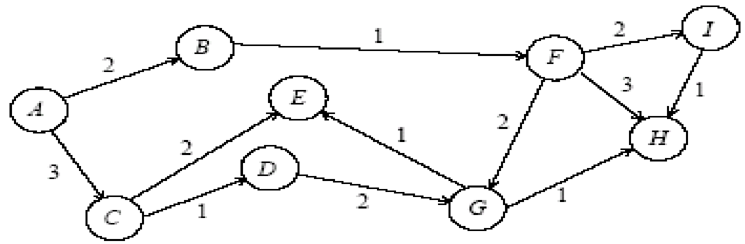
Matrix-Multiplication Algorithm

•Consider the multiplication of the weighted adjacency matrix with itself.

•The product of weighted adjacency matrix with itself returns a matrix that contains shortest paths of length 2 between any pair of nodes.

•It follows that A^n contains all shortest paths.

Matrix-Multiplication Based Algorithm



$$A^1 = \begin{pmatrix} 0 & 2 & 3 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty & 1 & \infty & \infty & \infty \\ \infty & \infty & 0 & 1 & 2 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & 2 & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & 2 & 3 & 2 \\ \infty & \infty & \infty & \infty & 1 & \infty & 0 & 1 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

$$A^2 = \begin{pmatrix} 0 & 2 & 3 & 4 & 5 & 3 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & \infty & 1 & 3 & 4 & 3 \\ \infty & \infty & 0 & 1 & 2 & \infty & 3 & \infty & \infty \\ \infty & \infty & \infty & 0 & 3 & \infty & 2 & 3 & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 3 & 0 & 2 & 3 & 2 \\ \infty & \infty & \infty & \infty & 1 & \infty & 0 & 1 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

$$A^4 = \begin{pmatrix} 0 & 2 & 3 & 4 & 5 & 3 & 5 & 6 & 5 \\ \infty & 0 & \infty & \infty & 4 & 1 & 3 & 4 & 3 \\ \infty & \infty & 0 & 1 & 2 & \infty & 3 & 4 & \infty \\ \infty & \infty & \infty & 0 & 3 & \infty & 2 & 3 & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 3 & 0 & 2 & 3 & 2 \\ \infty & \infty & \infty & \infty & 1 & \infty & 0 & 1 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

$$A^8 = \begin{pmatrix} 0 & 2 & 3 & 4 & 5 & 3 & 5 & 6 & 5 \\ \infty & 0 & \infty & \infty & 4 & 1 & 3 & 4 & 3 \\ \infty & \infty & 0 & 1 & 2 & \infty & 3 & 4 & \infty \\ \infty & \infty & \infty & 0 & 3 & \infty & 2 & 3 & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 3 & 0 & 2 & 3 & 2 \\ \infty & \infty & \infty & \infty & 1 & \infty & 0 & 1 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 1 & 0 \end{pmatrix}$$

Continue..

•A^n is computed by doubling powers - i.e., as A, A^2, A^4, A^8, and so on.

•We need $\log n$ matrix multiplications, each taking time $O(n^3)$.

•The serial complexity of this procedure is $O(n^3 \log n)$.

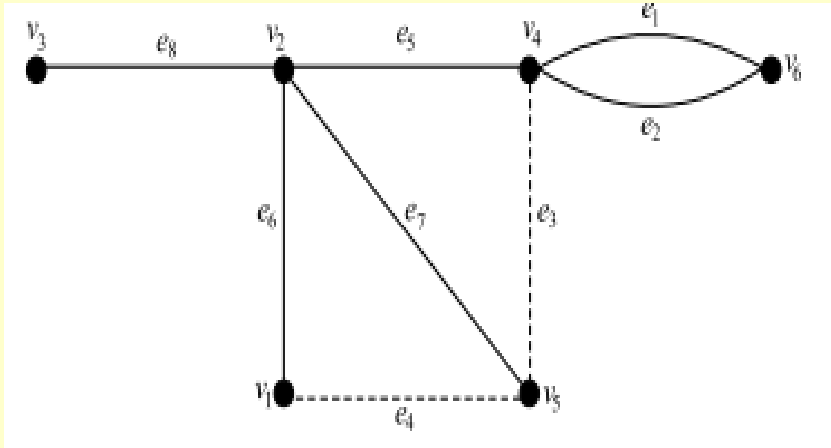
•This algorithm is not optimal, since the best known algorithms have complexity $O(n^3)$.

Path Matrix

Let G be a graph with m edges,u and v vertices.The path matrix P(u, v) = [pij]q×m, where q is the number of different paths between u and v.

pij =1,if jth edge lies in the ith path,

0, otherwise .



The different paths between the vertices v_3 and v_4 are

$$p_1 = \{e_8, e_5\},\ p_2 = \{e_8, e_7, e_3\} \text{ and } p_3 = \{e_8, e_6, e_4, e_3\}.$$

The path matrix for v_3, v_4 is given by

$$P(v_3, v_4) = \begin{array}{c} \begin{matrix} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array}.$$

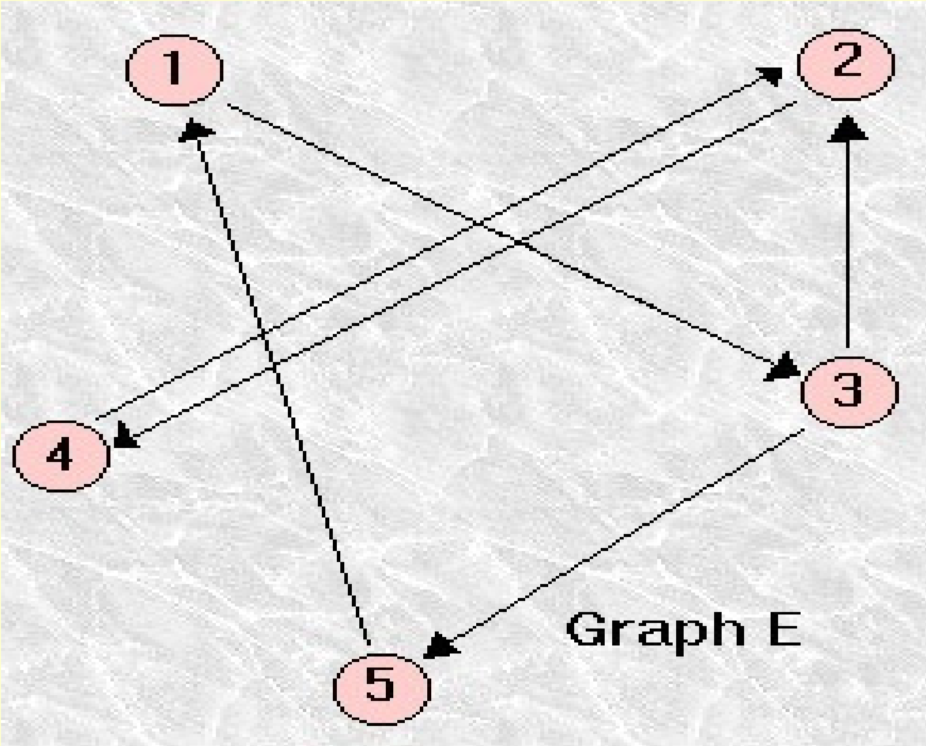
Warshall’s Algorithm

• Warshall's Algorithm is used to compute the existence of paths within a digraph using Boolean operators and matrices.

• It is used for finding shortest paths in a weighted graph with positive or negative edge weights (but with no negative cycles) and also for finding transitive closure of a relation R .

•Complexity of the algorithm is O(|N^3|),where N is number of nodes of the graph.

Warshall’s Algorithm



A	1	2	3	4	5
1	0	0	1	0	0
2	0	0	0	1	0
3	0	1	0	0	1
4	0	1	0	0	0
5	1	0	0	0	0

Begin by creating an adjacency matrix **A** for Graph **E** - instead of using weights,we will use Boolean operators.If there is a path, enter a 1 in matrix **A**, and enter 0 if no path exists.

Continue..

•This matrix tells us whether or not there is a path ***p*** of length 1 between two adjacent nodes.

•Building upon matrix **A**, we will create a new matrix $\mathbf{A^1}$, for which we will choose 1 vertex to act as a *pivot* - an intermediate point between 2 other vertices.

•Initially, we will chose vertex 1 as pivot for $\mathbf{A^1}$.

•For vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$,

$\boldsymbol{p}^{(1)}{}_{i\,j}$ is 1, if there exists an edge between vertices $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$, or if there is a path of length ≥ 2 from $\boldsymbol{v}_i$ to $\boldsymbol{v}_1$ and from $\boldsymbol{v}_1$ to $\boldsymbol{v}_j$.

else 0, if there is no path.

Matrix A^1

- Begin by scanning column 1 of matrix **A**;only vertex 5 connects $\mathbf{v}_i$ to $\mathbf{v}_1$.

- Now scan row 1,the only path from $\mathbf{v}_1$ to $\mathbf{v}_j$ is to vertex 3.

- So, path of length 2 lies between $\mathbf{v}_5$ and $\mathbf{v}_3$, we update matrix $\mathbf{A}^1$ accordingly.

A^1	1	2	3	4	5
1	0	0	1	0	0
2	0	0	0	1	0
3	0	1	0	0	1
4	0	1	0	0	0
5	1	0	1	0	0

Matrix A^2

- Next create matrix $\mathbf{A}^2$, using vertex 2 as the pivot point.

- Begin by scanning *column 2* of matrix **A**; the $\boldsymbol{v}_i$ which connect to $\boldsymbol{v}_2$ are vertices 3 and 4.

- Now scan *row 2*; only 1 path from $\boldsymbol{v}_2$ exists to $\boldsymbol{v}_j$ = vertex 4.

- Newly added paths have been highlighted in gray.

$\mathbf{A}^2$	1	2	3	4	5
1	0	0	1	0	0
2	0	0	0	1	0
3	0	1	0	1	1
4	0	1	0	1	0
5	1	0	1	0	0

Matrix A^3

•Matrix $\mathbf{A}^3$ use vertex 3 as the pivot point.

•Vertices 1 and 5 have a path to 3.

•Now, scanning row 3, v_3 connects to vertices 2, 4, 5. Paths established :-

v_1 to v_2
v_1 to v_4
v_1 to v_5
v_5 to v_2
v_5 to v_4
v_5 to v_5

$\mathbf{A}^3$	1	2	3	4	5
1	0	1	1	1	1
2	0	0	0	1	0
3	0	1	0	1	1
4	0	1	0	1	0
5	1	1	1	1	1

Matrix A^4

Some paths have exceeded length 2 because the newly established paths are not using just 3 as a pivot point, but also the previous pivots points.

•Now,we will be creating 2 more adjacency matrices, $\mathbf{A}^4$ and $\mathbf{A}^5$.

•For $\mathbf{A}^4$, first scan column 4.

•All vertices now have a path to vertex 4.

•Scanning row 4, we see that 4 has a path only to vertex 2, indicating that all vertices have a path to 2.

•The only vertex which doesn't already have a path to vertex 2 is 2 itself.

A^4	1	2	3	4	5
1	0	1	1	1	1
2	0	1	0	1	0
3	0	1	0	1	1
4	0	1	0	1	0
5	1	1	1	1	1

If a graph has ***n*** vertices, it will require ***n*** matrices to produce $\mathbf{A}^n = \mathbf{P}^n$, where $\mathbf{P}^n$ is the path matrix.

Matrix A^5

•Now, scan column 5 to see that vertices 1, 3 and 5 all have paths to vertex 5.

• Scanning row 5 indicates that 5 has a path to all other vertices.

•Consequently, we add 1's to rows 1, 3 and 5 to reflect that vertices 1, 3 and 5 have paths to all other vertices.

A^5	1	2	3	4	5
1	1	1	1	1	1
2	0	1	0	1	0
3	1	1	1	1	1
4	0	1	0	1	0
5	1	1	1	1	1

This completes the path matrix for Graph E.

Warshall's Algorithm for computing a path matrix

procedure *Warshall*
(A: BoolMatrix; /*Input, the adjacency matrix of a given graph*/
var P: BoolMatrix; /*Output, the path matrix of the graph*/
n: **integer**); /*Input, the size of the matrix (i.e., the number of vertices)*/
int i, j, k;
Begin
for i :=1 **to** n **do**
    for j := 1 **to** n **do**
        P[i, j] := A[i, j]; /*Step 1: Copy adjacency into path matrix*/
for k := 1 **to** n **do** /*Step 2: Allow vertex k as a pivot point*/
for i := 1 **to** n **do** /*Step 3: Process rows*/
  for j := 1 **to** n **do** /*Step 4: Process columns*/
 P[i, j] := P[i, j] **or** (P[i, k] **and** P[k, j]) /*Step 5*/
end;