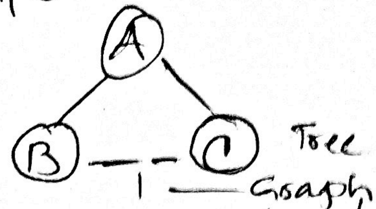
GRAPHS

Example



A, B, C – vertices/nodes

(A, B), (A, C) – Edges

B, C

→ SET of (V, E) pairs

V – set of vertices

E – " " Edges

Vertecs — Represented as Circles also Known as nodes

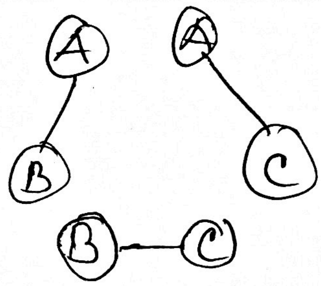
Edge —— Represented as lines connecting to vertices/nodes

TERMINOLOGY

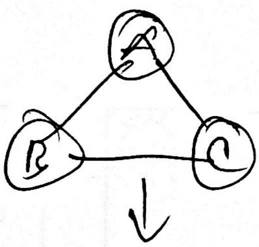
① Node

② Edge

③ Adjecent nodes



④ Degree of node → no. of edges connected to that node.



D (A) – 2

D (B) – 1

D (C) – 1

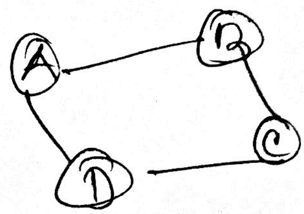
⑤ size of a graph – Total no of edges in graph

SIZE = 3

⑥ Path – sequence of vertices from source node to Destination node.

A → C

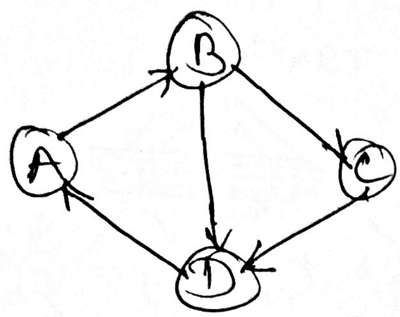
Source Destination



Path → A → B → C | A → D → C

Types of Graphs

① Directed Graph

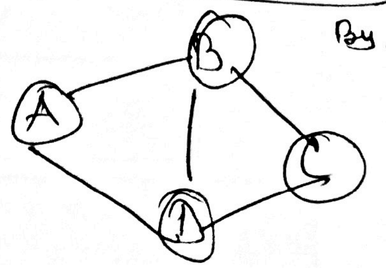


$(A,B) \neq (B,A)$

UNI-DIRECTIONAL

UnDirected Graph

By direction

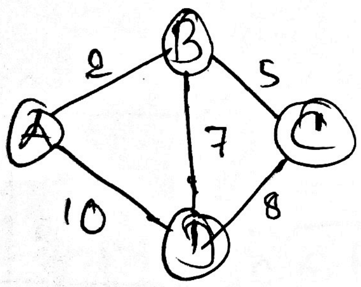


$(A,B) = (B,A)$

$(B,C) = (C,A)$

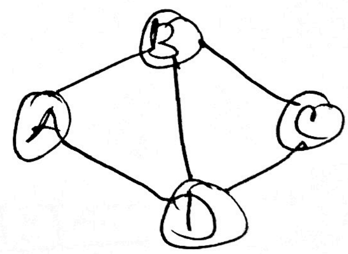
BI-DIRECTIONAL

② WEIGHTED G



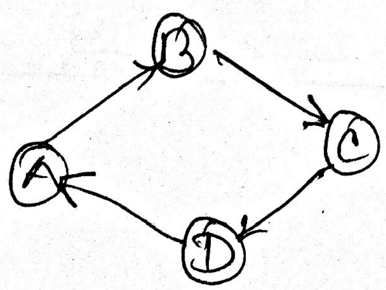
→ weight is specifed for every edge

UN WEIGHTED GRAPH



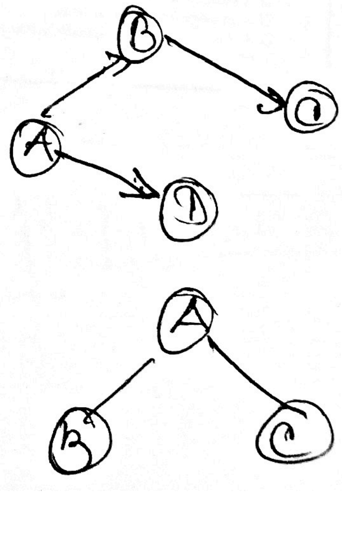
→ no weight is specified

③ CYCLIC GRAPH



$A \to B \to C \to D \to A$

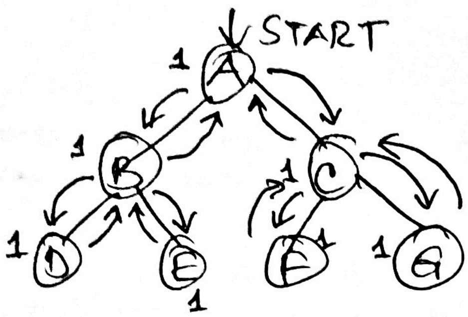
ACYCLIC GRAPH



GRAPH TRAVERSALS

① DEPT FIRST SEARCH (DFS) → STACK

② BREADTH FIRST SEARCH (BFS) → QUEUE



DFS

| A | C | G | F | B | D | E |

STEP -1

Stack: A

DFS: (empty)

STEP-2

Stack: C, B

DFS: A

STEP -3

Stack: G, F, B

DFS: A | C

STEP-4

Stack: F, B

DFS: A | C | G

STEP -5

Stack: B

DFS: A C G F

STEP -6

Stack: D, E

DFS: A C G F B

STEP -7

Stack: E

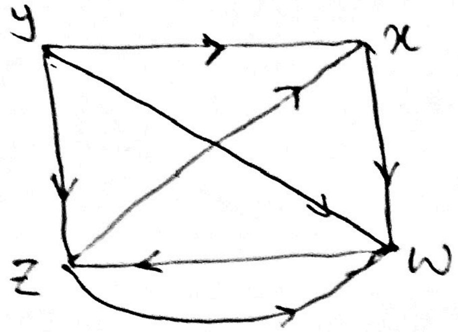
DFS: A C G F B D

STEP -8

Stack: empty

DFS: A C G F B D E

Warshalls Algorithm to find Path Matrix Example:



$$A = P = \begin{array}{c|cccc} & x & y & z & w \\ \hline x & 0 & 0 & 0 & 1 \\ y & 1 & 0 & 1 & 1 \\ z & 1 & 0 & 0 & 1 \\ w & 0 & 0 & 1 & 0 \end{array}$$

$$P_x = \begin{array}{c|cccc} & x & y & z & w \\ \hline x & 0 & 0 & 0 & 1 \\ y & 1 & 0 & 1 & 1 \\ z & 1 & 0 & 0 & 1 \\ w & 0 & 0 & 1 & 0 \end{array} \qquad P_y = \begin{array}{c|cccc} & x & y & z & w \\ \hline x & 0 & 0 & 0 & 1 \\ y & 1 & 0 & 1 & 1 \\ z & 1 & 0 & 0 & 1 \\ w & 0 & 0 & 1 & 0 \end{array}$$

$$P_z = \begin{array}{c|cccc} & x & y & z & w \\ \hline x & 0 & 0 & 0 & 1 \\ y & 1 & 0 & 1 & 1 \\ z & 1 & 0 & 0 & 1 \\ w & 1 & 0 & 1 & 1 \end{array} \qquad P_w = \begin{array}{c|cccc} & x & y & z & w \\ \hline x & 1 & 0 & 1 & 1 \\ y & 1 & 0 & 1 & 1 \\ z & 1 & 0 & 1 & 1 \\ w & 1 & 0 & 1 & 1 \end{array}$$

$$\begin{tikzcd} z \arrow[rr, "0"] \arrow[dr, "1"'] & & y \\ & x \arrow[ur, "0"'] & \end{tikzcd} \qquad \begin{tikzcd} w \arrow[rr, "0"] \arrow[dr, "1"'] & & w \\ & z \arrow[ur, "1"'] & \end{tikzcd}$$

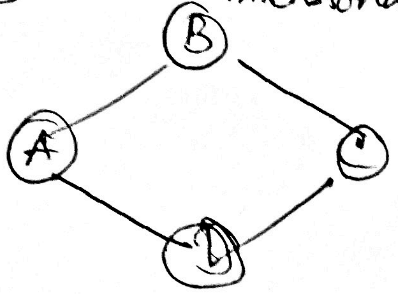
A Directed Graph G with N nodes is maintained in memory by the adjacency matrix A. The Algorithm finds the (Boolean) path matrix P of the Graph G.

```
Repeat for I, J = 1, 2, ..., N          // Initialize P
   if (A[I,J] = 0 then P[I,J] = 0
   else P[I,J] = 1                       // End to Loop

Repeat for K = 1, 2, ..., N             // Updates P
  Repeat for I = 1, 2, ..., N
  Repeat for J = 1, 2, ..., N
        P[I,J] = P[I,J] or [P[I,K] and P[K,J])).
```

REPRESENTATION OF GRAPH

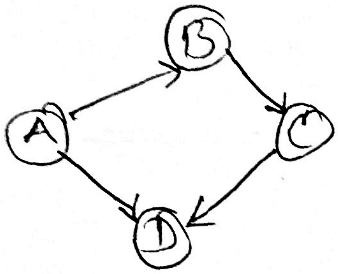
Using multi-Dimensional Array.



$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 0 & 1 \\ B & 1 & 0 & 1 & 0 \\ C & 0 & 1 & 0 & 1 \\ D & 1 & 0 & 1 & 0 \end{array}$$

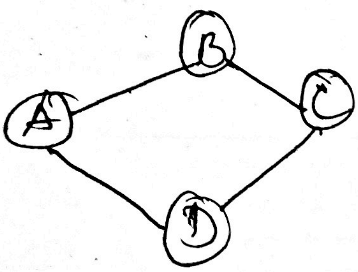
D - nodes

Graph - adjacent matrix



$$\begin{array}{c|cccc} & A & B & C & D \\ \hline A & 0 & 1 & 0 & 1 \\ B & 0 & 0 & 1 & 0 \\ C & 0 & 0 & 0 & 1 \\ D & 0 & 0 & 0 & 0 \end{array}$$

→ Using list

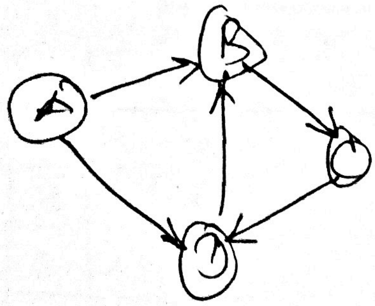


A : [B |] → [D | x]   A : B, D

B : [A |] → [C | x]   B : A, C

C : [B |] → [D | x]   C : B, D

D : [A |] → [B |] → [C | x]   D : A, B, C



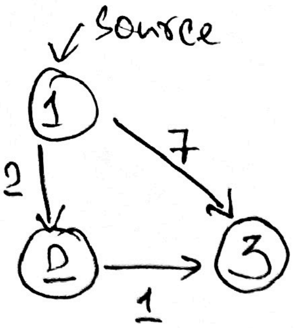
A : [B |] → [D | x]   A : B, D

B : [C | x]   B : C

C : [D | x]   C : D

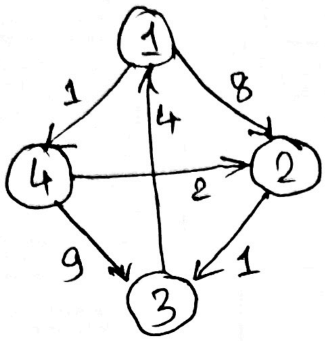
D : [B | x]   D : B

Floyd Warshall Alg.



① → ③ = 7

① → 2 → ③ = 3



D0:

	1	2	3	4
1	0	8	∝	1
2	∝	0	1	∝
3	4	∝	0	∝
4	∝	2	9	0

D1:

	1	2	3	4
1	0	8	∝	1
2	∝	0	1	∝
3	4	12	0	5
4	∝	2	9	0

∝ — $\boxed{1}$ — ∝

D1:

3 − 1 − 2 = 12

3 − 1 − 4 = 5

D2:

	1	2	3	4
1	0	8	9	1
2	∝	0	1	∝
3	4	12	0	5
4	∝	2	3	0

D3:

	1	2	3	4
1	0	8	9	1
2	5	0	1	6
3	4	12	0	5
4	7	2	3	0

D2

∝ — $\boxed{2}$ — ∝

∝ $\boxed{1,2}$ — ∝

∝ $\boxed{2,1}$ — ∝

4 − 2 − 3 = 3

1 − 2 − 3 = 9

D3

2 − 3 − 1 = 5

2 − 3 − 1 − 4 = 6

4 − 2 − 3 − 1 = 7

4 − 3 − 1 = 13 ×

D4

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

Shortest length for 3 to 4

3 — 4 = 5

D4

1 − 4 − 3 = 10 ×

1 − 4 − 2 = 3

3 − 1 − 4 − 2 = 7

1 − 4 − 2 − 3 = 4