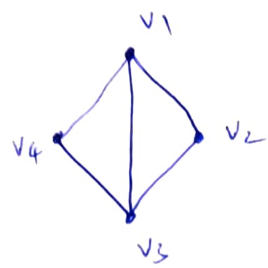
Graph :-

- A graph G consists of two sets as,
    - A set V, called set of vertices/nodes
    - A set E, called set of Edges/arcs.
- Ex:-



here, $V = \{v_1, v_2, v_3, v_4\}$

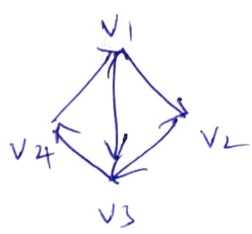
$E = \{(v_1 v_2), (v_1 v_3), (v_1 v_4), (v_2 v_3), (v_3 v_4)\}$

Terminology :-

i) Digraph :-

It is also called as directed graph, where all vertices are connected through directions as edges.

Ex:-

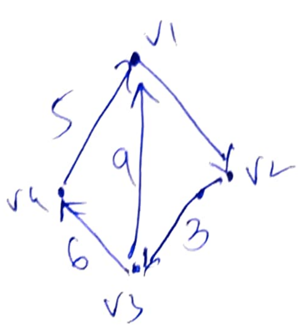


here, $V = \{v_1, v_2, v_3, v_4\}$

$E = \{(v_1 v_2) (v_2 v_3) (v_1 v_3) (v_3 v_4) (v_4 v_1)\}$

ii) weighted Graph :-

If a graph have all the edges labelled with weights, then it is called weighted Graph.



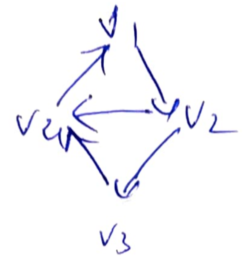
here,

$$\left.\begin{aligned} v_1 v_4 &= 5 \\ v_3 v_4 &= 6 \\ v_2 v_3 &= 3 \\ v_1 v_3 &= 9 \end{aligned}\right\} \text{weights}$$

iii) Adjacent vertices :-

A vertex which acts as adjacent (or) neighbour vertex are adjacent vertices.

Ex:



here,

V_2 is adjacent to V_3, V_4

V_1 is adjacent to V_2

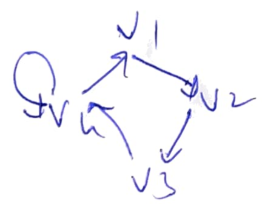
V_3 is adjacent to V_4

V_4 is adjacent to V_1.

iv) loop :-

- In a graph G, if any edge starts & ends at the same vertex then it is called as loop.

Ex:-



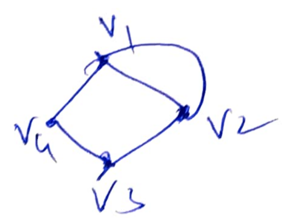
here, V_4 has the self loop.

(with edge starting & ending at the same vertex)

v) parallel edges :-

If there are more than one edges b/w same pair of vertices, then it is called parallel edge.

Ex:-



here,

$V_1 V_2$ has more than one edge

So they have parallel edges.

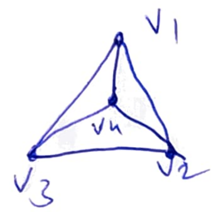
vi) Simple graph:-

It is as same as digraph, but doesn't contain any self loops (or) parallel edges.

vii) Complete graph :-

A digraph is said to be complete if each vertex is adjacent to other vertex. ie, edges from any vertex to all other vertices.

Ex:-



viii) Acyclic graph:-

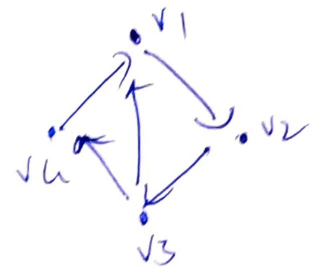
- Any path having one (or) more edges which starts from a vertex and terminates into same vertex then path is called cycle.
- And any graph which doesn't have any cycle is called acyclic graph.

ix) Isolated vertex :-

A vertex is isolated if there is no edge connected from any vertex to other vertex.

x) Degree of vertex :-

The no. of edges are connected as both indegree and outdegree.



here,

indegree - no. of incoming edges
outdegree - no. of outgoing edges

indegree(v1) = 2	outdegree(v1) = 1
indegree(v2) = 1	outdegree(v2) = 1
indegree(v3) = 1	outdegree(v3) = 2
indegree(v4) = 1	outdegree(v4) = 1

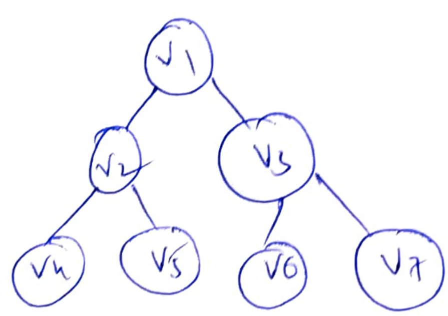
xi) Pendant vertex :-

A vertex with its indegree as '1' & outdegree as '0' is called pendant vertex.

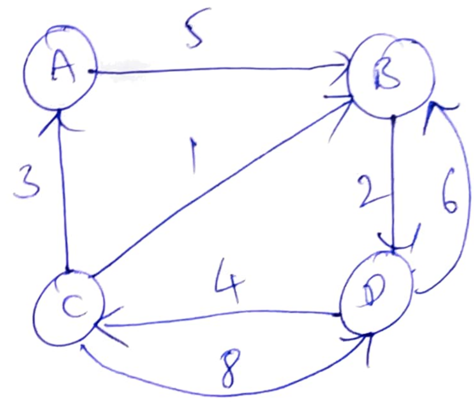
=> Representations of Graphs :-

- Graphs can be represented by many ways. Some of these representations are,

  i) set representation
  ii) Linked representation
  iii) sequential / matrix representation.



G1



G2

i) Set Representation :-

- for G_1 given above the vertices & Edges are,

$$V(G_1) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$$

$$E(G_1) = \{(v_1 v_2)(v_1 v_3)(v_2 v_4)(v_2 v_5)(v_3 v_6)(v_3 v_7)\}$$

- for G_2 given above the vertices & Edges are,

$$V(G_2) = \{A, B, C, D\}$$

$$E(G_2) = \{(5,A,B)(2,B,D)(3,A,C)(1,C,B)(8,C,D)(4,D,C)(6,D,B)\}$$

ii) Linked Representation :-

- This is other way for space saving way of graph representation.
- Here in linked way the node structures are given as either of these

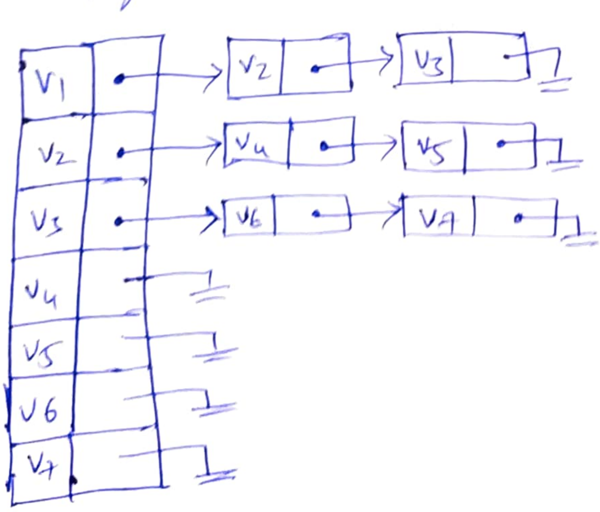
Node label	Adj-List

fig :- node structure for nonweighted graph

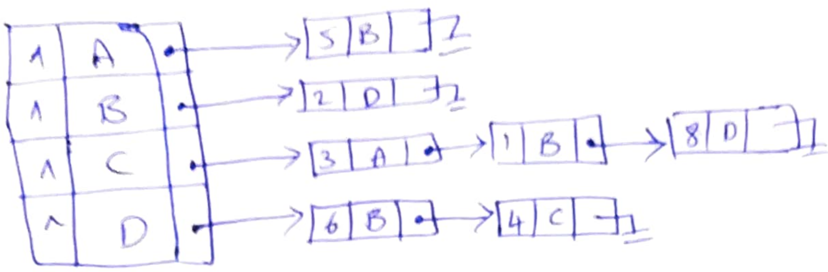
weight	Node label	Adj-list

fig :- node structure for weighted graph

- for G_1 given above the linked node structure is given as,



→ For G_2 given above the linked node structure is given as,



iii) Matrix representation :-

- This uses square matrix of order $n \times n$, as no. of vertices in a graph.
- This matrix is called adjacency matrix (∵ it stores info. of two vertices whether adjacent or not.
- This is also called bit matrix or Boolean matrix
- For G_1 the matrix representation (adjacency matrix) is given as,

$$
\begin{array}{c c}
 & \begin{array}{ccccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{array} \\
\begin{array}{c} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{array} &
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

G_1

→ For G_2 the matrix representation (adjacency matrix) is,

$$
\begin{array}{c c}
 & \begin{array}{cccc} A & B & C & D \end{array} \\
\begin{array}{c} A \\ B \\ C \\ D \end{array} &
\begin{bmatrix}
0 & 5 & 0 & 0 \\
0 & 0 & 0 & 2 \\
3 & 1 & 0 & 8 \\
0 & 6 & 4 & 0
\end{bmatrix}
\end{array}
$$

G_2

⇒ Operations of Graphs :-

- The important operations possible on graphs are,

i) Insertion
   - insert a vertex
   - insert an edge

ii) Deletion
   - delete a vertex
   - delete an edge

iii) Merging

iv) Traversal

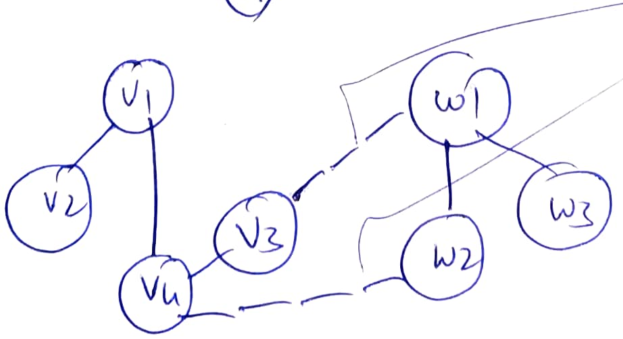
iii) Merging :-

- Consider two graphs G_1 and G_2. By merging these two graphs into a single component is done by establishing one (or) more edges b/w vertices in G_1 and G_2.

Ex :-

G_1 G_2

⇓



→ newly added edges b/w vertices v_3 & w_1 and v_4 & w_2.

$G_1 \oplus G_2$

⇒ Applications of Graphs :-

⇒ Shortest Path problem :

- The different algorithms used here are,
  - Warshall's algorithm
  - Floyd's algorithm
  - Dijkstra's algorithm

• Warshall's algorithm :-

This determines whether there is a path b/w one vertex to other either directly (or) through one (or) more intermediate vertices.

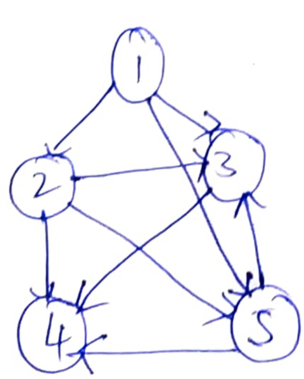
- The path matrix $P[i][j]$ entries whether there is direct path from v_i to v_j

Steps to be followed :-

1. Initialize path matrix P as $P[i][j]$ from v_i to v_j
2. Here calculate with 'N' entries as if k, which decides path for all $i, j = 1, 2 \dots N$ directly (or) through k i.e., from v_i to v_k and from v_k to v_j by entering either 0 (or) 1.

$$P[i][j] = P[i][j] \vee \left(P[i][k] \wedge P[k][j] \right)$$

Ex :-



adjacency matrix is,

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

Now let $k=1$ then, $P[i][j] = P[i][j] \vee (P[i][k] \wedge P[k][j])$

$P[1][1] = P[1][1] \vee (P[1][1] \wedge P[1][1]) = 0 \vee (0 \wedge 0) = 0 \vee (0) = 0$

$P[1][4] = P[1][4] \vee (P[1][1] \wedge P[1][4]) = 0 \vee (0 \wedge 0) = 0 \vee (0) = 0$

$P[2][1] = P[2][1] \vee (P[2][1] \wedge P[1][1]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[2][2] = P[2][2] \vee (P[2][1] \wedge P[1][2]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[3][1] = P[3][1] \vee (P[3][1] \wedge P[1][1]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[3][2] = P[3][2] \vee (P[3][1] \wedge P[1][2]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[3][3] = P[3][3] \vee (P[3][1] \wedge P[1][3]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[3][5] = P[3][5] \vee (P[3][1] \wedge P[1][5]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[4][1] = P[4][1] \vee (P[4][1] \wedge P[1][1]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[4][2] = P[4][2] \vee (P[4][1] \wedge P[1][2]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[4][3] = P[4][3] \vee (P[4][1] \wedge P[1][3]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[4][4] = P[4][4] \vee (P[4][1] \wedge P[1][4]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[4][5] = P[4][5] \vee (P[4][1] \wedge P[1][5]) = 0 \vee (0 \wedge 1) = 0 \vee 0 \equiv 0$

$P[5][1] = P[5][1] \vee (P[5][1] \wedge P[1][1]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[5][2] = P[5][2] \vee (P[5][1] \wedge P[1][2]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[5][5] = P[5][5] \vee (P[5][1] \wedge P[1][5]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

after $k=1$:-

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

Now let $k=2$ then, $P[i][j] = P[i][j] \vee (P[i][k] \wedge P[k][j])$

$P[1][1] = P[1][1] \vee (P[1][2] \wedge P[2][1]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[1][4] = P[1][4] \vee (P[1][2] \wedge P[2][4]) = 0 \vee (1 \wedge 1) = 0 \vee 1 = 1$

$P[2][1] = P[2][1] \vee (P[2][2] \wedge P[2][1]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[2][2] = P[2][2] \vee (P[2][2] \wedge P[2][2]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0.$

$P[3][1] = P[3][1] \vee (P[3][2] \wedge P[2][1]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[3][2] = P[3][2] \vee (P[3][2] \wedge P[2][2]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[3][3] = P[3][3] \vee (P[3][2] \wedge P[2][3]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[3][5] = P[3][5] \vee (P[3][2] \wedge P[2][5]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[4][1] = P[4][1] \vee (P[4][2] \wedge P[2][1]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[4][2] = P[4][2] \vee (P[4][2] \wedge P[2][2]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[4][3] = P[4][3] \vee (P[4][2] \wedge P[2][3]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[4][4] = P[4][4] \vee (P[4][2] \wedge P[2][4]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

$P[4][5] = P[4][5] \vee (P[4][2] \wedge P[2][5]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0.$

$P[5][1] = P[5][1] \vee (P[5][2] \wedge P[2][1]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[5][2] = P[5][2] \vee (P[5][2] \wedge P[2][2]) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$

$P[5][5] = P[5][5] \vee (P[5][2] \wedge P[2][5]) = 0 \vee (0 \wedge 1) = 0 \vee 0 = 0$

after $k=2$:-

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

Similarly do the same process for $k=3,4,5$ ($\because k \le 5$)

& the final path matrix we obtain is,

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

• Floyd's algorithm :-

- To find the shortest path b/w any pair of vertices, a person named Robert Floyd introduced "Floyd's algorithm".
- In Floyd's algorithm two functions are used as,
    - Min (x, y), which returns min. value b/w x and y.
    - Combine (P_1, P_2), which returns concatenation of two paths P_1 and P_2 resulting in single path.

Steps :-

1) Initially draw the graph's adjacency matrix.
2) For the no paths represent '∞' (infinity).
3) Then compute the all pairs shortest paths as,

$$Q[i][j] = \min\left(Q[i][j],\ Q[i][k] + Q[k][j]\right)$$

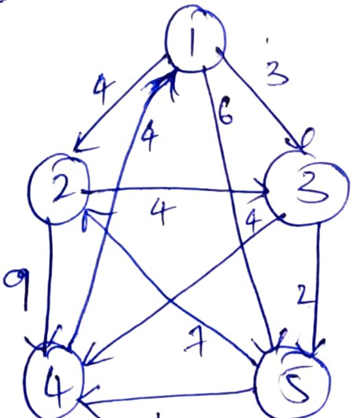
and if, $\left(Q[i][k] + Q[k][j]\right) < Q[i][j]$ then,

assign, $P_1 = \text{Paths}[i][k]$ &

$P_2 = \text{Paths}[k][j]$

Then Combine (P_1, P_2) & store in Path[i][j].

Ex :-



$\Longrightarrow$

	1	2	3	4	5
1	-	4	3	-	6
2	-	-	4	9	-
3	-	-	-	4	2
4	4	-	-	-	-
5	-	7	-	1	-

After initialization of Q & Paths,

	1	2	3	4	5
1	∞	4	3	∞	6
2	∞	∞	4	9	∞
3	∞	∞	∞	4	2
4	4	∞	∞	∞	∞
5	∞	7	∞	1	∞

	1	2	3	4	5
1	-	[1-2]	[1-3]	-	[1-5]
2	-	-	[2-3]	[2-4]	-
3	-	-	-	[3-4]	[3-5]
4	[4-1]	-	-	-	-
5	-	[5-2]	-	[5-4]	-

Let $k=1$ then, $Q[i][j] = \min(Q[i][j],\ Q[i][k] + Q[k][j])$

$Q[1][1] = \min(Q[1][1],\ Q[1][1] + Q[1][1]) = \min(\cancel{\infty}, \infty+\infty) = \infty$

$Q[1][2] = \min(Q[1][2],\ Q[1][1] + Q[1][2]) = \min(4, \infty+4) = 4$

$Q[1][3] = \min(Q[1][3],\ Q[1][1] + Q[1][3]) = \min(3, \infty+3) = 3$

$Q[1][4] = \min(Q[1][4],\ Q[1][1] + Q[1][4]) = \min(\infty, \infty+\infty) = -$

$Q[1][4] = \min(Q[1][4],\ Q[1][1] + Q[1][5]) = \min(6, \infty+6) = 6$

$Q[1][5] = \min(Q[1][5],\ Q[1][1] + Q[1][5]) = \min(\infty, \infty+\infty) = -$

$Q[2][1] = \min(Q[2][1],\ Q[2][1] + Q[1][1]) = \min(\infty, \infty+4) = \infty$

$Q[2][2] = \min(Q[2][2],\ Q[2][1] + Q[1][2]) = \min(4, \infty+3) = 4$

$Q[2][3] = \min(Q[2][3],\ Q[2][1] + Q[1][3]) = \min(9, \infty+\infty) = 9$

$Q[2][4] = \min(Q[2][4],\ Q[2][1] + Q[1][4]) = \min(\infty, \infty+6) = \infty$

$Q[2][5] = \min(Q[2][5],\ Q[2][1] + Q[1][5])$

$Q[3][1] = \min(Q[3][1],\ Q[3][1] + Q[1][1]) = \min(\infty, \infty+\infty) = \infty$

$Q[3][2] = \min(Q[3][2],\ Q[3][1] + Q[1][2]) = \min(\infty, \infty+4) = \infty$

$Q[3][3] = \min(Q[3][3],\ Q[3][1] + Q[1][3]) = \min(\infty, \infty+3) = \infty$

$Q[3][4] = \min(Q[3][4],\ Q[3][1] + Q[1][4]) = \min(4, \infty+\infty) = 4$

$Q[3][5] = \min(Q[3][5],\ Q[3][1] + Q[1][5]) = \min(2, \infty+6) = 2$

$Q[4][1] = \min(Q[4][1],\ Q[4][1] + Q[1][1]) = \min(4, 4+\infty) = 4$

$Q[4][2] = \min(Q[4][2],\ Q[4][1] + Q[1][2]) = \min(\infty, 4+4) = 8$

$Q[4][3] = \min(Q[4][3],\ Q[4][1] + Q[1][3]) = \min(\infty, 4+3) = 7$

$Q[4][4] = \min(Q[4][4],\ Q[4][1] + Q[1][4]) = \min(\infty, 4+\infty) = \infty$

$Q[4][5] = \min(Q[4][5],\ Q[4][1] + Q[1][5]) = \min(\infty, 4+6) = 10$

Q[5][1] = min(Q[5][1], Q[5][1] + Q[1][1]) = min(∞, ∞ + ∞) = ∞
Q[5][2] = min(Q[5][2], Q[5][1] + Q[1][2]) = min(7, ∞ + 4) = 7
Q[5][3] = min(Q[5][3], Q[5][1] + Q[1][3]) = min(∞, ∞ + 3) = ∞
Q[5][4] = min(Q[5][4], Q[5][1] + Q[1][4]) = min(1, ∞ + ∞) = 1
Q[5][5] = min(Q[5][5], Q[5][1] + Q[1][5]) = min(∞, ∞ + 6) = ∞

Now, Q & Path is,

	1	2	3	4	5
1	∞	4	3	∞	6
2	∞	∞	4	9	∞
3	∞	∞	∞	4	2
4	4	8	7	∞	10
5	∞	7	∞	1	∞

	1	2	3	4	5
1	-	[1-2]	[1-3]	-	[1-5]
2	-	-	[2-3]	[2-4]	-
3	-	-	-	[3-4]	[3-5]
4	[4-1]	[4-1-2]	[4-1-3]	-	[4-1-5]
5	-	[5-2]	-	[5-4]	-

- Similarly find Q & Path based on min% values using same process with k = 2, 3, 4, 5.
- (Also while doing k = 2, 3, 4, 5 consider the updated values of Q & Path matrix too).
- So finally, after k = 5 we get Q & Path as min% (or) Shortest path finally as follows,

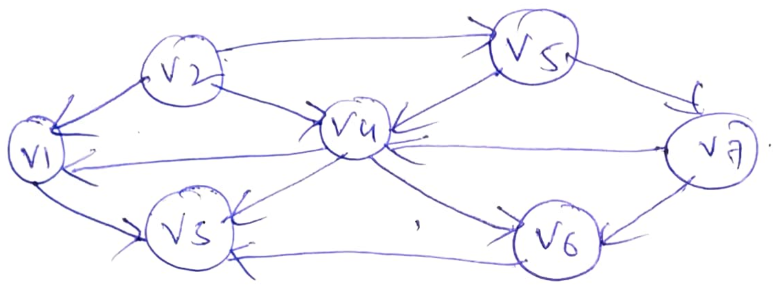
	1	2	3	4	5
1	10	4	3	6	5
2	11	13	4	7	6
3	7	9	10	3	2
4	4	8	7	10	9
5	5	7	8	1	10

	1	2	3	4	5
1	[1-3-5-4-1]	[1-2]	[1-3]	[1-3-5-4]	[1-3-5]
2	[2-3-5-4-1]	[2-3-5-2]	[2-3]	[2-3-5-4]	[2-3-5]
3	[3-5-4-1]	[3-5-2]	[3-5-4-1-3]	[3-5-4]	[3-5]
4	[4-1]	[4-1-2]	[4-1-3]	[4-1-3-5-4]	[4-1-3-5]
5	[5-4-1]	[5-2]	[5-4-1-3]	[5-4]	[5-4-1-3-5]

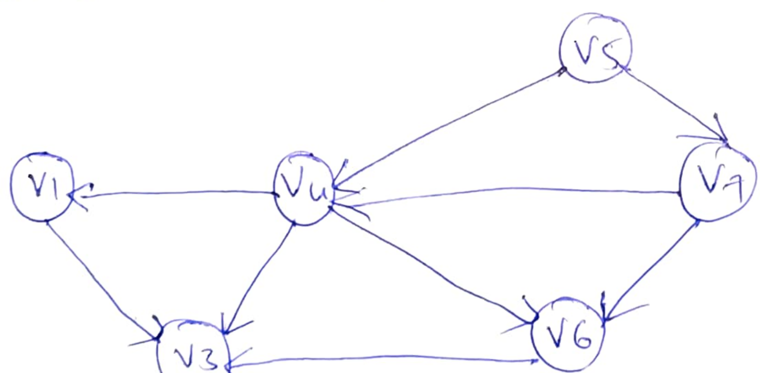
⇒ Topological sorting :-

- It is used to make the vertices of a graph in an order.
- In this sorting, ordering is done by considering the vertex with indegree zero & delete it along with its edge from graph.
- And repeat the same process of selecting vertices with zero indegree until the graph becomes empty.

Ex:-



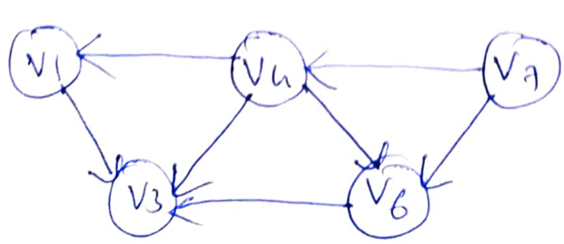
i) In the above graph, the vertex V_2 has indegree as '0'. so delete V_2 along its edges. Then we get,



order

V_2

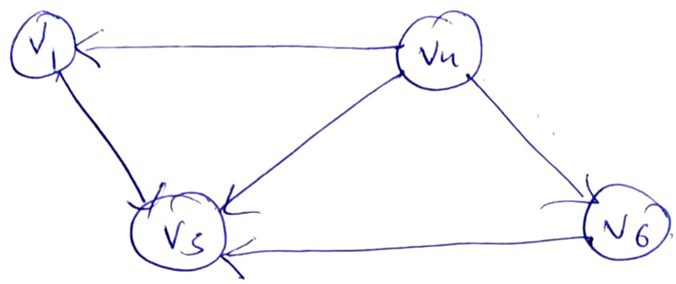
ii) After removing V_2, the other vertex V_5 has indegree as '0'. so delete V_5 along its edges. Then we get,



order

$V_2 - V_5$.

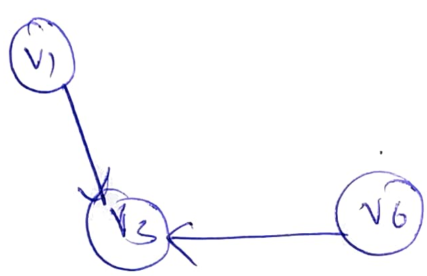
iii) After removing v_5, the other vertex v_7 has indegree as '0'. So delete v_7 along its edges. Then we get,



order

$v_2 - v_5 - v_7$

iv) After removing v_7, the other vertex v_4 has indegree as '0'. So delete v_4 along its edges. Then we get,



order

$v_2 - v_5 - v_7 - v_4$

v) After removing v_4, the other vertexes v_1 & v_6 have indegree as '0'. So, select v_1 initially & remove it along its edges.

Then we get,

order

$v_2 - v_5 - v_7 - v_4 - v_1$

vi) After removing v_1 the other vertex v_6 has indegree as '0'. So, select & delete v_6 along its edges. Then we get,

order

$v_2 - v_5 - v_7 - v_4 - v_1 - v_6$

vii) After removing v_6 the other vertex v_3 has indegree as '0'. So, delete v_3 finally. So, the final topological order of vertices is

($v_2 - v_5 - v_7 - v_4 - v_1 - v_6 - v_3$)