VE281

Data Structures and Algorithms

Recitation Class

Graph Theory Tutorial

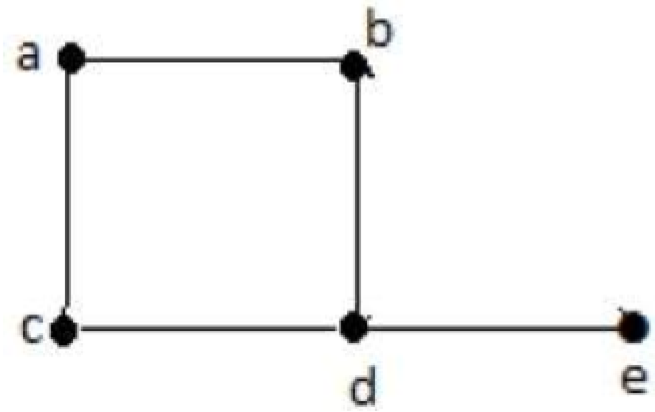
TA Group

What is a Graph?

- A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as **vertices,** and the links that connect the vertices are called **edges**.
- Formally, a graph is a pair of sets **(V, E),** where **V** is the set of vertices and **E** is the set of edges, connecting the pairs of vertices.

Example

- In the above graph,
  - V = {a, b, c, d, e}
  - E = {ab, ac, bd, cd, de}



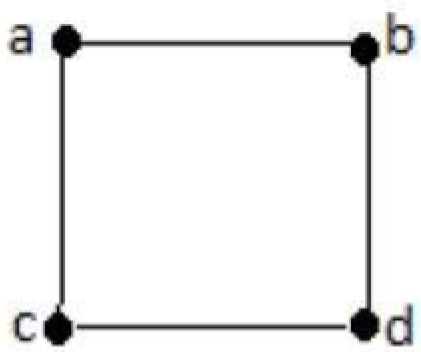
Terminology

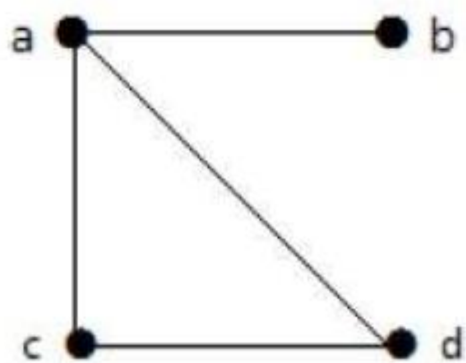
- Vertex
  - A vertex is a point where multiple lines meet. It is also called a **node**. Similar to points, a vertex is also denoted by an alphabet.

- Edge
  - An edge is the mathematical term for a line that connects two vertices. Many edges can be formed from a single vertex.

Terminology

- Graph
  - A graph 'G' is defined as $G = (V, E)$ Where V is a set of all vertices and E is a set of all edges in the graph.



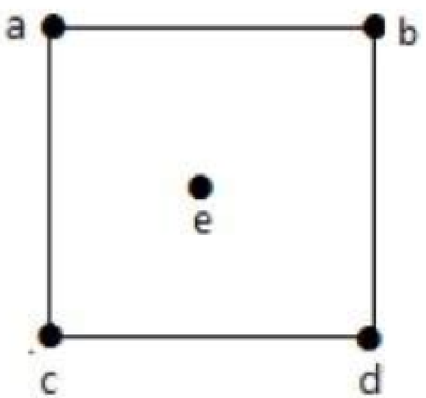


Terminology

- Loop
  - In a graph, if an edge is drawn from vertex to itself, it is called a loop.

Terminology

- Degree of Vertex
  - It is the number of vertices adjacent to a vertex V.
  - **Notation** − deg(V).
- Degree in Undirected Graph
  - deg(a) = 2, deg(b) = 2, deg(c) = 2, deg(d) = 2, and deg(e) = 0.
  - The vertex ‘e’ is an isolated vertex. The graph does not have any pendent vertex.



Terminology

- Degree in Directed Graph
  - Indegree of a Graph
    - Indegree of vertex V is the number of edges which are coming into the vertex V.
    - **Notation** − $\deg^{-}(V)$.
  - Outdegree of a Graph
    - Outdegree of vertex V is the number of edges which are going out from the vertex V.
    - **Notation** − $\deg^{+}(V)$.

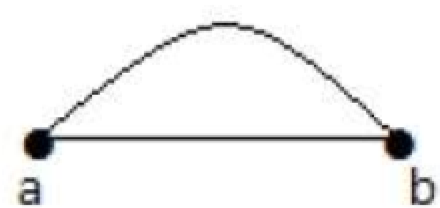
Terminology

- Pendent Vertex
  - A vertex with degree one is called a pendent vertex.

- Isolated Vertex
  - A vertex with degree zero is called an isolated vertex.

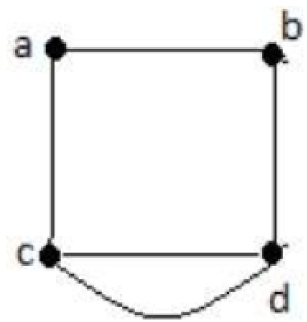
Terminology

- Adjacency
  - In a graph, two vertices are said to be **adjacent,** if there is an edge between the two vertices.
  - In a graph, two edges are said to be **adjacent**, if there is a common vertex between the two edges.
- Parallel Edges
  - In a graph, if a pair of vertices is connected by more than one edge, then those edges are called parallel edges.



Terminology

- Multi Graph
  - A graph having parallel edges is known as a Multigraph.



- Degree Sequence of a Graph
  - If the degrees of all vertices in a graph are arranged in **descending** or **ascending** order, then the sequence obtained is known as the degree sequence of the graph.

Basic Properties

- Distance between Two Vertices
  - It is number of edges in a shortest path between Vertex U and Vertex V. If there are multiple paths connecting two vertices, then the **shortest** path is considered as the distance between the two vertices.
  - **Notation** – d(U,V)

Dijkstra

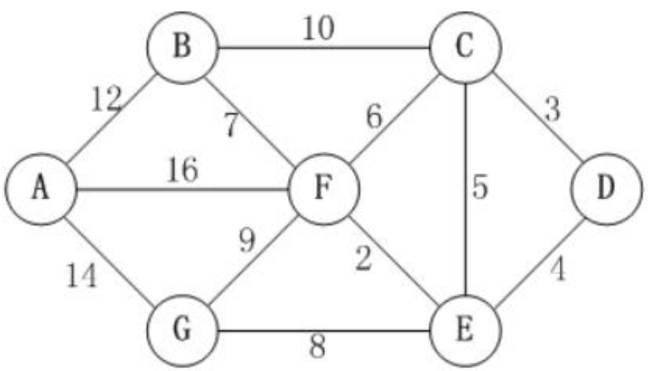
- Step1: Think of the initial points on the graph as one set S, and the other points as another set.
- Step2: According to the initial point, calculate the distance d[i] from other points to the initial point (if adjacent, d[i] is the edge weight; if not adjacent, d[i] is infinite).
- Step3: Select the smallest d[i] (denoted as d[x]) and add the point corresponding to this d[i] edge (denoted as x) into the set S.
- Step4: According to x, update d[y] value of y adjacent to x: d[y] = min{d[y], d[x] + edge weight w[x][y]}
- Repeat (3), (4) until the target point also joins the set. At this time, the corresponding d[i] of the target point is the shortest path length.

Floyd

- Step1: Initialize the matrix S. The distances between two vertices are the weights of the edges. If there is no edge connected between the two vertices, the weight is infinite.
- Step2: Each node w is the intermediate vertex in turn. For every pair of vertices u and v, update S(u,v) by min(S(u,v), S(u,w)+S(w,v)).

Floyd

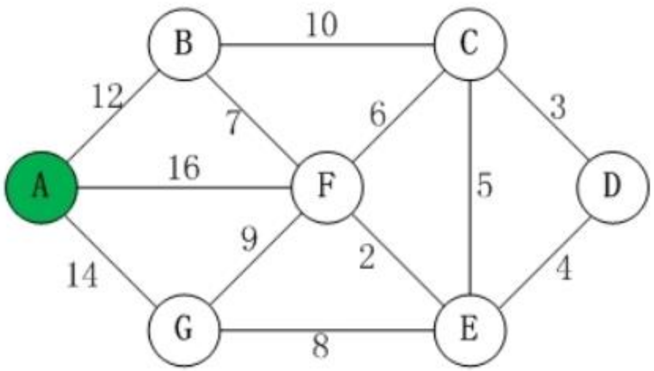
1. Initialize the matrix S



$$
\begin{array}{c c}
 & \begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\
\begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \\ G \end{array} &
\begin{bmatrix}
0 & 12 & INF & INF & INF & 16 & 14 \\
12 & 0 & 10 & INF & INF & 7 & INF \\
INF & 10 & 0 & 3 & 5 & 6 & INF \\
INF & INF & 3 & 0 & 4 & INF & INF \\
INF & INF & 5 & 4 & 0 & 2 & 8 \\
16 & 7 & 6 & INF & 2 & 0 & 9 \\
14 & INF & INF & INF & 8 & 9 & 0
\end{bmatrix}
\end{array}
$$

Floyd

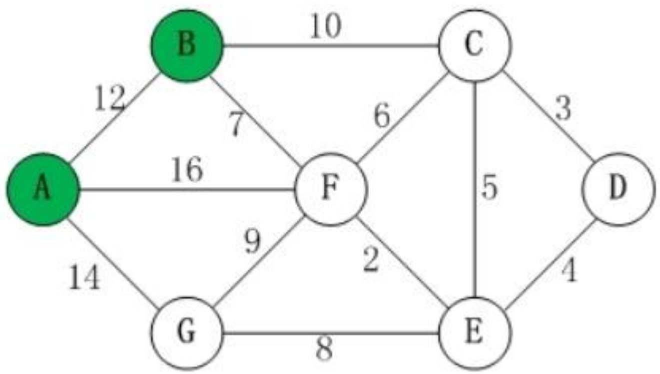
2. Take A as the intermediate vertex and update S



	A	B	C	D	E	F	G
A	0	12	INF	INF	INF	16	14
B	12	0	10	INF	INF	7	26
C	INF	10	0	3	5	6	INF
D	INF	INF	3	0	4	INF	INF
E	INF	INF	5	4	0	2	8
F	16	7	6	INF	2	0	9
G	14	26	INF	INF	8	9	0

Floyd

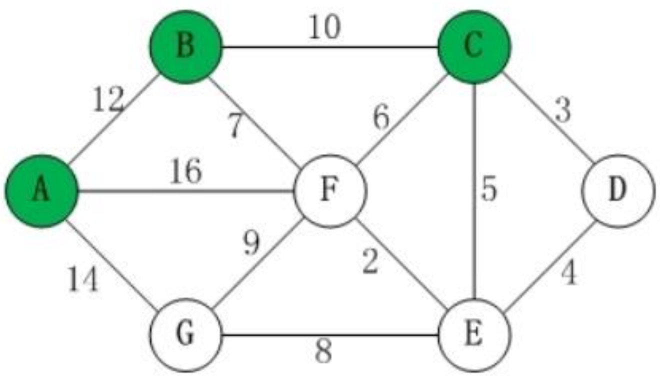
3. Take B as the intermediate vertex and update S



	A	B	C	D	E	F	G
A	0	12	22	INF	INF	16	14
B	12	0	10	INF	INF	7	26
C	22	10	0	3	5	6	36
D	INF	INF	3	0	4	INF	INF
E	INF	INF	5	4	0	2	8
F	16	7	6	INF	2	0	9
G	14	26	36	INF	8	9	0

Floyd

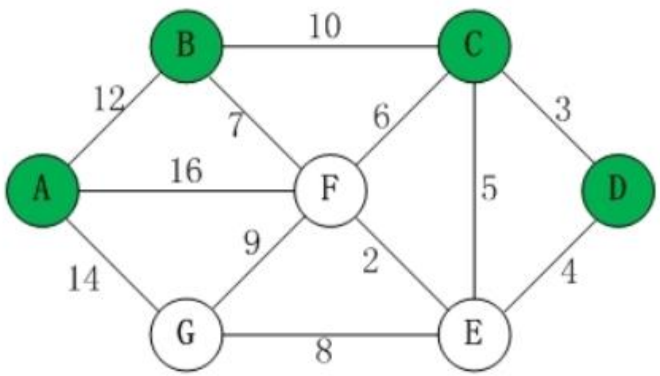
4. Take C as the intermediate vertex and update S



	A	B	C	D	E	F	G
A	0	12	22	25	27	16	14
B	12	0	10	13	15	7	26
C	22	10	0	3	5	6	36
D	25	13	3	0	4	9	39
E	27	15	5	4	0	2	8
F	16	7	6	9	2	0	9
G	14	26	36	39	8	9	0

Floyd

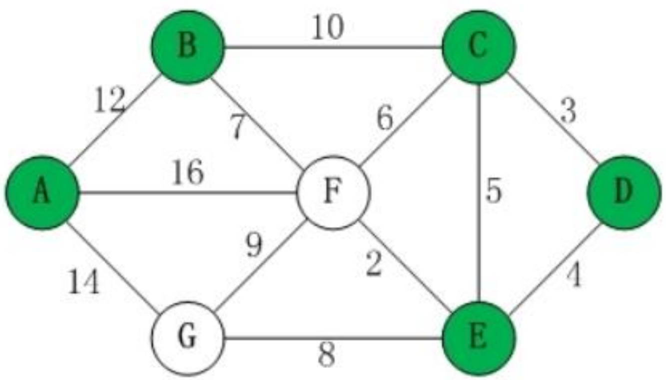
5. Take D as the intermediate vertex and update S



$$
\begin{array}{c}
 \\ A \\ B \\ C \\ D \\ E \\ F \\ G
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\
\begin{bmatrix}
0 & 12 & 22 & 25 & 27 & 16 & 14 \\
12 & 0 & 10 & 13 & 15 & 7 & 26 \\
22 & 10 & 0 & 3 & 5 & 6 & 36 \\
25 & 13 & 3 & 0 & 4 & 9 & 39 \\
27 & 15 & 5 & 4 & 0 & 2 & 8 \\
16 & 7 & 6 & 9 & 2 & 0 & 9 \\
14 & 26 & 36 & 39 & 8 & 9 & 0
\end{bmatrix}
\end{array}
$$

Floyd

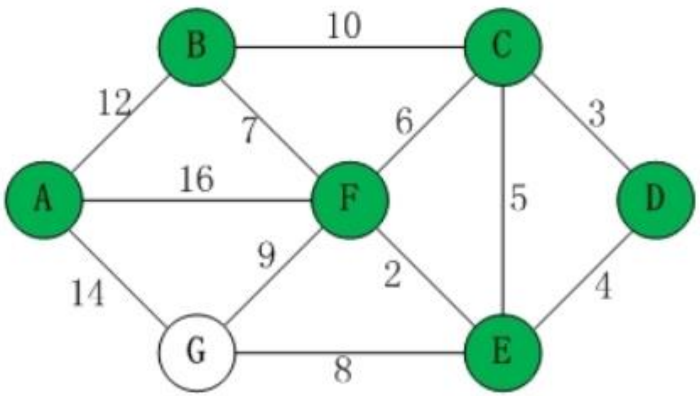
6. Take E as the intermediate vertex and update S



$$
\begin{array}{c}
 \\ A \\ B \\ C \\ D \\ E \\ F \\ G
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\
\left[\begin{array}{ccccccc}
0 & 12 & 22 & 25 & 27 & 16 & 14 \\
12 & 0 & 10 & 13 & 15 & 7 & 23 \\
22 & 10 & 0 & 3 & 5 & 6 & 13 \\
25 & 13 & 3 & 0 & 4 & 6 & 12 \\
27 & 15 & 5 & 4 & 0 & 2 & 8 \\
16 & 7 & 6 & 6 & 2 & 0 & 9 \\
14 & 23 & 13 & 12 & 8 & 9 & 0
\end{array}\right]
\end{array}
$$

Floyd

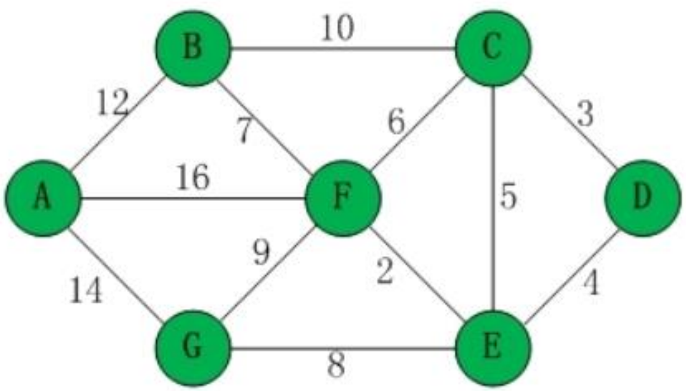
7. Take F as the intermediate vertex and update S



$$
\begin{array}{c}
\\
\begin{array}{c} A \\ B \\ C \\ D \\ E \\ F \\ G \end{array}
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} A & B & C & D & E & F & G \end{array} \\
\begin{bmatrix}
0 & 12 & 22 & 22 & 18 & 16 & 14 \\
12 & 0 & 10 & 13 & 9 & 7 & 16 \\
22 & 10 & 0 & 3 & 5 & 6 & 13 \\
22 & 13 & 3 & 0 & 4 & 6 & 12 \\
18 & 9 & 5 & 4 & 0 & 2 & 8 \\
16 & 7 & 6 & 6 & 2 & 0 & 9 \\
14 & 16 & 13 & 12 & 8 & 9 & 0
\end{bmatrix}
\end{array}
$$

Floyd

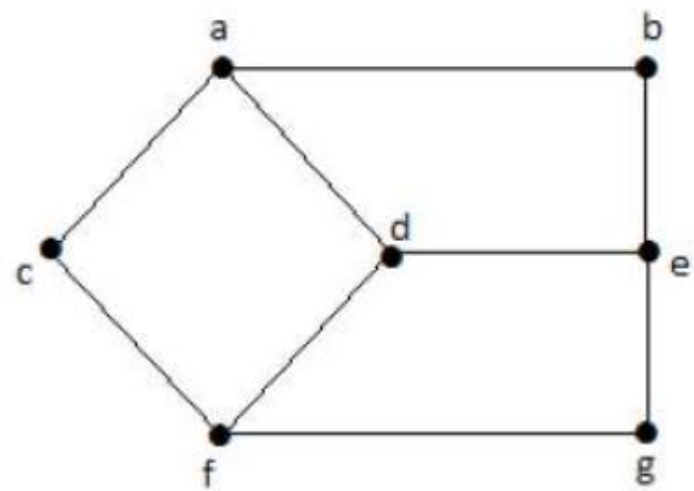
8. Take G as the intermediate vertex and update S



	A	B	C	D	E	F	G
A	0	12	22	22	18	16	14
B	12	0	10	13	9	7	16
C	22	10	0	3	5	6	13
D	22	13	3	0	4	6	12
E	18	9	5	4	0	2	8
F	16	7	6	6	2	0	9
G	14	16	13	12	8	9	0

Basic Properties

- Eccentricity of a Vertex
  - The maximum distance between a vertex to all other vertices is considered as the eccentricity of vertex.
  - **Notation** − e(V)
  - e(a) = 3, e(b) = 3, e(c) = 3, e(d) = 2, e(e) = 3, e(f) = 3, e(g) = 3



Basic Properties

- Radius of a Connected Graph
  - The minimum eccentricity from all the vertices is considered as the radius of the Graph G.
  - **Notation** − r(G)
- Diameter of a Graph
  - The maximum eccentricity from all the vertices is considered as the diameter of the Graph G.
  - **Notation** − d(G)

Basic Properties

- Central Point
  - If the eccentricity of a graph is equal to its radius, i.e. e(V) = r(V), then it is known as the central point of the graph.
- Centre
  - The set of all central points of 'G' is called the centre of the Graph.

Types of Graphs

- Null Graph
  - A graph having no edges
- Trivial Graph
  - A graph with only one vertex
- Non-Directed Graph
  - A non-directed graph contains edges but the edges are not directed ones.
- Directed Graph
  - In a directed graph, each edge has a direction.

Types of Graphs

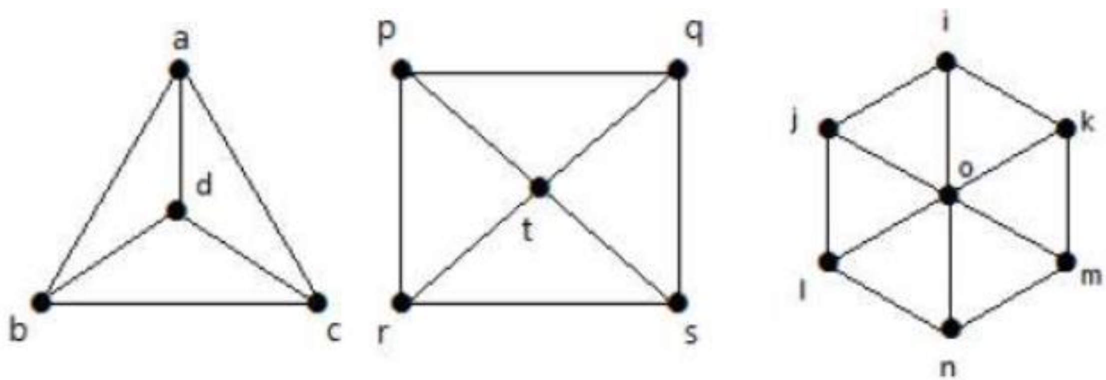
- Simple Graph
  - A graph **with no loops** and **no parallel edges** is called a simple graph.
- Connected Graph
  - A graph G is said to be connected **if there exists a path between every pair of vertices**.
- Disconnected Graph
  - A graph G is disconnected, if it does not contain at least two connected vertices.

Types of Graphs

- Regular Graph
  - A graph G is said to be regular, **if all its vertices have the same degree**. In a graph, if the degree of each vertex is 'k', then the graph is called a 'k-regular graph'.
- Complete Graph
  - In the graph, **a vertex should have edges with all other vertices,** then it called a complete graph.
- Cycle Graph
  - If the **degree of each vertex in the graph is two,** then it is called a Cycle Graph.

Types of Graphs

- Wheel Graph
  - A wheel graph is obtained from a cycle graph C_{n-1} by adding a new vertex. That new vertex is called a **Hub** which is connected to all the vertices of C_n.



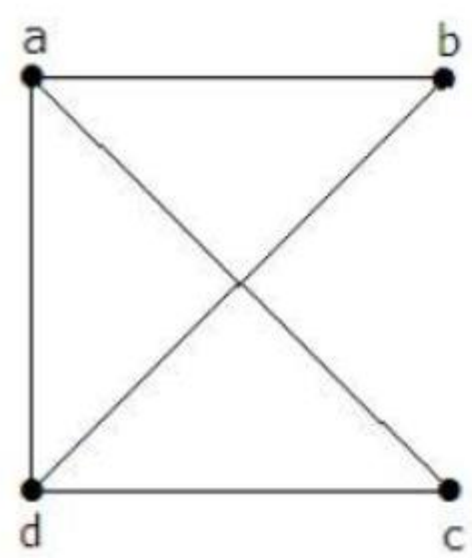
I(W_4) II(W_5) III(W_7)

Types of Graphs

- Cyclic Graph
  - A graph **with at least one** cycle is called a cyclic graph.
- Acyclic Graph
  - A graph **with no cycles** is called an acyclic graph.
- Tree
  - A **connected acyclic graph** is called a tree.
- Spanning Trees
  - Let G be a connected graph, then the sub-graph H of G is called a spanning tree of G if
    - H is a tree
    - H contains all vertices of G

Kirchoff’s Theorem

- Kirchoff’s theorem is useful in finding the number of spanning trees that can be formed from a connected graph.
- Example



Kirchoff's Theorem

$$A = \begin{array}{c} \\ \\ \\ \\ \\ \end{array}\left|\begin{array}{ccccc} 0 & a & b & c & d \\ a & 0 & 1 & 1 & 1 \\ b & 1 & 0 & 0 & 1 \\ c & 1 & 0 & 0 & 1 \\ d & 1 & 1 & 1 & 0 \end{array}\right| = \begin{vmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{vmatrix}$$

By using kirchoff's theorem, it should be changed as replacing the principle diagonal values with the degree of vertices and all other elements with -1.A

$$= \begin{vmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{vmatrix} = M$$

$$M = \begin{vmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{vmatrix}$$

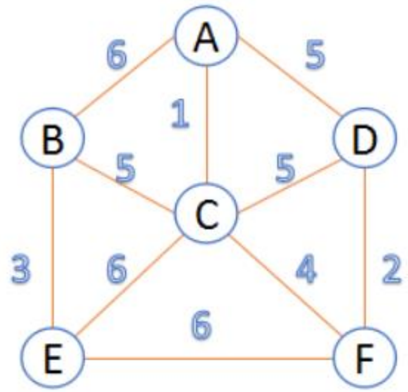
$$\text{Cofactor of m1} = \begin{vmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ -1 & -1 & 3 \end{vmatrix} = 8$$

Thus, the number of spanning trees = 8.

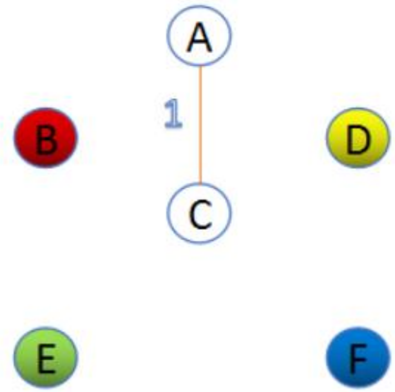
Kruskal

- Step1: Sort all the edges in the graph by cost, from smallest to largest
- Step2: Consider the n vertices in the graph as a forest of n individual trees
- Step3: Select edges from small to large by weight. The two vertices that the selected edge connects, ui,vi, should belong to two different trees. Combine the two trees as one.
- Step4: Repeat (3) until all vertices are in a tree or the tree has n-1 edges.

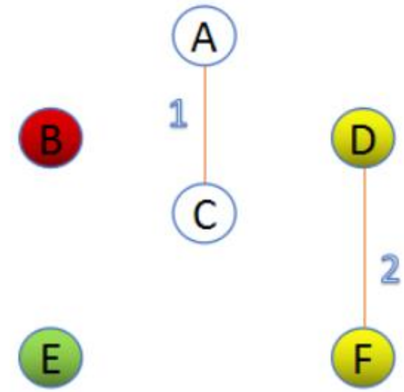
Kruskal



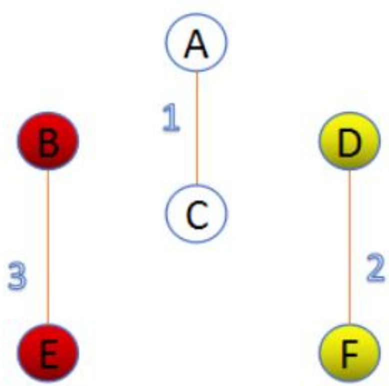
Graph



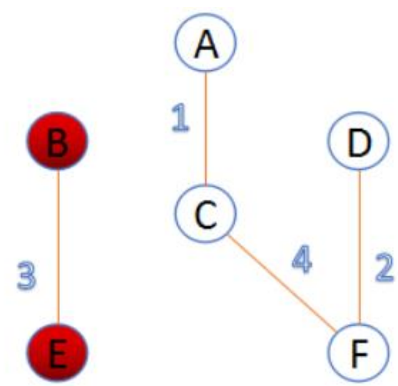
Choose edge AC



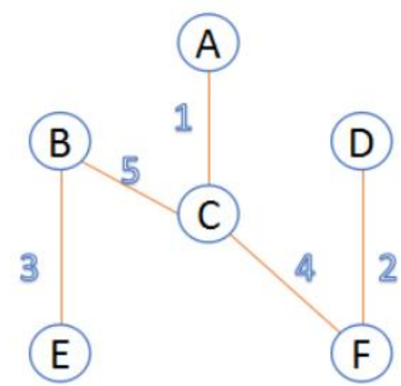
Choose edge DF



Choose edge BE



Choose edge CF

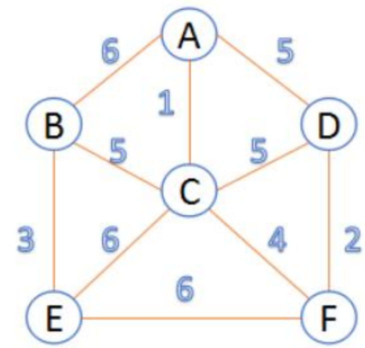


Choose edge BC. Stop

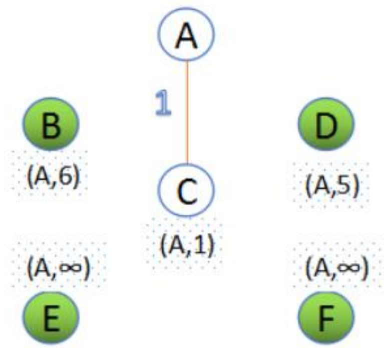
Prim

- Step1: The set of all vertices of the graph is V. Set u={s},v=v −u.
- Step2: Of the edges that two sets u and v can form, select the least costly edge (u0,v0), add to the minimum spanning tree, and add v0 into the set u.
- Step3: Repeat (2) until the minimum spanning tree has n-1 edges or n vertices.

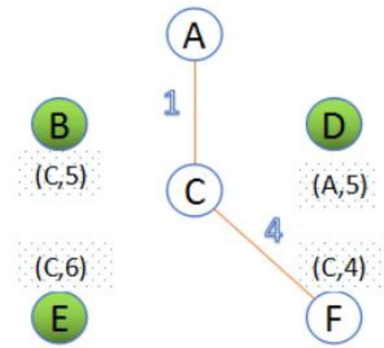
Prim



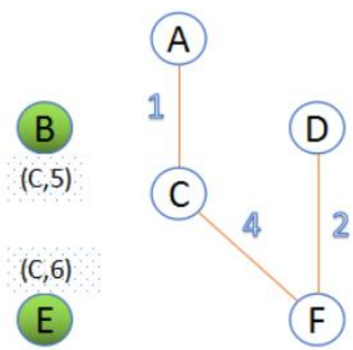
Graph



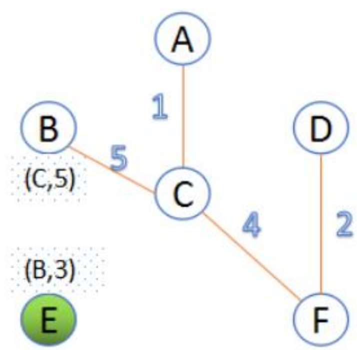
Choose vertex C



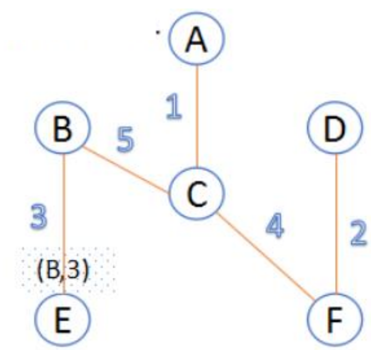
Choose vertex F



Choose vertex D



Choose vertex B



Choose vertex E.
Stop