

UNIT-4**GRAPHS****Q. Define Graph. Explain types of graphs.****-- 5 Marks****Ans: Graphs:-** A graph is a non linear data structure.

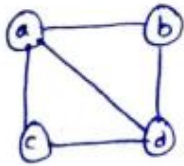
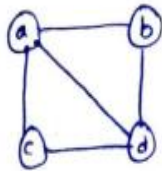
* A graph is denoted as a collection of vertices and edges.

* It is denoted by $G=\{V,E\}$

Where 'V' represents set of vertices

'E' represents set of Edges

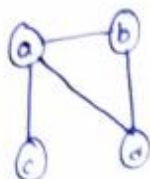
* A vertex is also known as "Node".

* An edge is a line joining of two vertices. It is denoted by a pair of vertices $E=(u,v)$ Example: $V = \{a, b, c, d\}$ $E = \{(a,b), (b,d), (c,d), (c,a), (a,d)\}$ **Degree of a node:** Total number of edges connected to a node or vertex is known as "Degree of a node". $V = \{a, b, c, d\}$ $E = \{(a,b), (b,d), (c,d), (c,a), (a,d)\}$

In the above graph, the degree of "a" is 3, degree of "b" is 2, degree of "c" is 2, degree of "d" is 3.

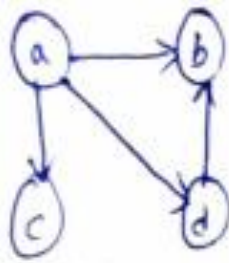
Types of graphs: In computer programming, different types of graphs are :-

1. Undirected Graph
2. Directed Graph
3. Directed Acyclic graph (DAG)
4. Simple Graphs
5. Complete Graph

1. Undirected Graphs:- Undirected Graph is a Graph in which the edge has "No Direction Marked".**Example:** $V = \{a, b, c, d\}$ $E = \{(a,b), (a,c), (a,d), (b,d), (c,d)\}$

2. Directed Graph: Directed graph in which the edge has "Direction Marked" . It is also known a "Di-Graph".

Example:

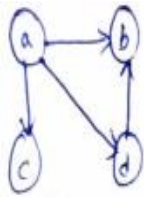


$$V = \{a, b, c, d\}$$

$$E = \{(a, c), (a, b), (a, d), (d, b)\}$$

3. Directed Acyclic Graph:- Directed Acyclic graph is a graph in which it has "No cycle" between two nodes. It is also known as DAG (Directed Acyclic Graph).

Example:

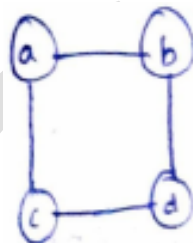


$$V = \{a, b, c, d\}$$

$$E = \{(a, c), (a, b), (a, d), (d, b)\}$$

4. Simple Graph: A graph that has "Parallel edges" is known as simple graph. Simple graph is always undirected graph.

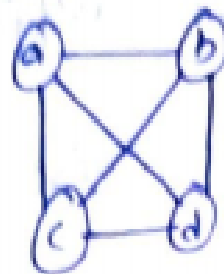
Example:



$$E = \{(a, b), (a, c), (c, d), (b, d)\}$$

5. Complete Graph: A Graph that has edges from "one vertex to all other vertices in a graph"., is known as complete graph. It is also known as connected graph.

Example:



Q. Define Graph. Explain graph representations.**--- 10 Marks(V.Imp)****Ans: Graphs:-** A graph is a non linear data structure.

* A graph is denoted as a collection of vertices and edges.

* It is denoted by $G=\{V,E\}$

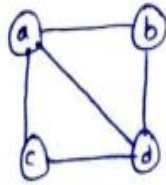
Where 'V' represents set of vertices

'E' represents set of Edges

* A vertex is also known as "Node".

* An edge is a line joining of two vertices. It is denoted by a pair of vertices $E=(u,v)$

Example:



$$V = \{a, b, c, d\}$$

$$E = \{(a,b), (b,d), (c,d), (c,a), (a,d)\}$$

Graph Representation:- There are two types of graph representation. They are :-

Linear representation (or) Adjacent Matrix

ii) Linked List Representation (or) Adjacent List

Linear Representation: Representation of graph using Array is known as linear representation. It is also known as Adjacency Matrix representation. For adjacency matrix we use a 2D array.

* create a 2D array 'a' with size "n X n" where n is the no. of vertices or nodes.

* A graph may be directed graph or undirected graph.

* Elements of 2D array are defined as $a[i][j] = 1$, if there is an edge, otherwise $a[i][j] = 0$

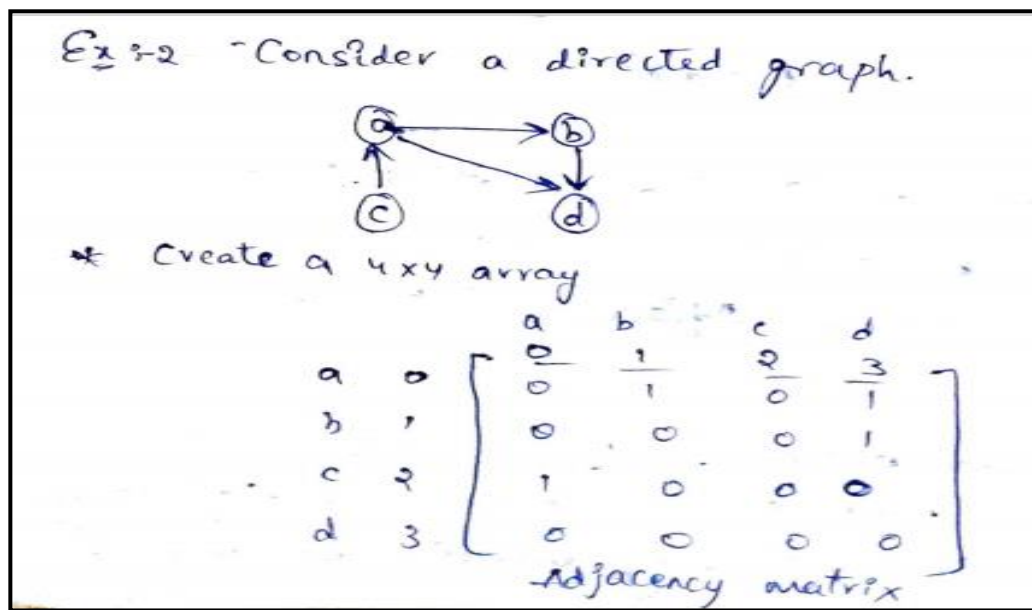
Ex: Consider an undirected graph



* Create a 4x4 array

	a	b	c	d
a	0	1	1	1
b	1	0	0	1
c	1	0	0	0
d	1	1	0	0

Adjacency Matrix

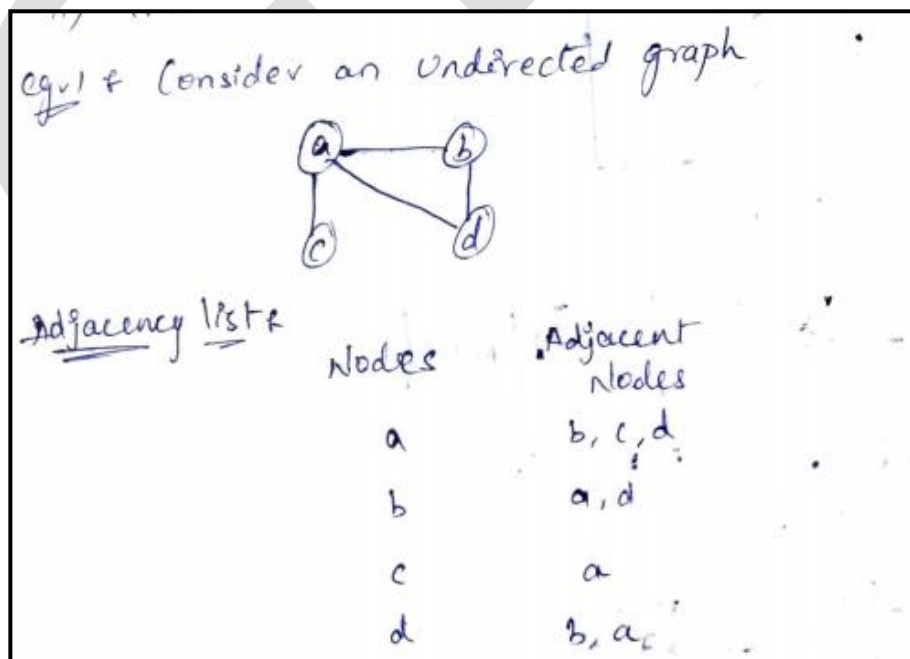


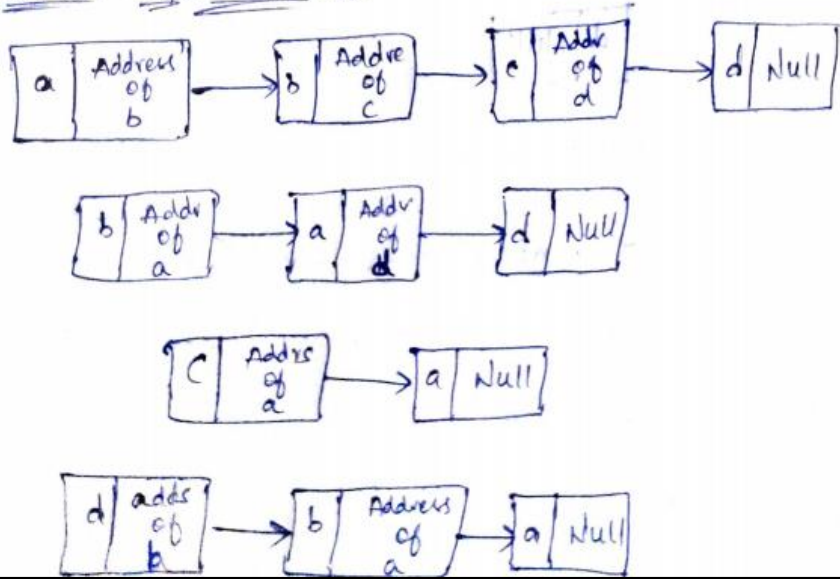
Linked list Representation: Representation of Graph using single Linked List is known as Linked List Representation. It is also known as "adjacency list representation".

* In adjacency list representation it has a 2 parts .

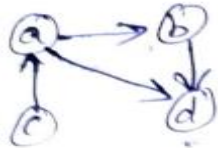
They are :-

1. Adjacency table (or) adjacency list
2. Linked list representation for each node in graph

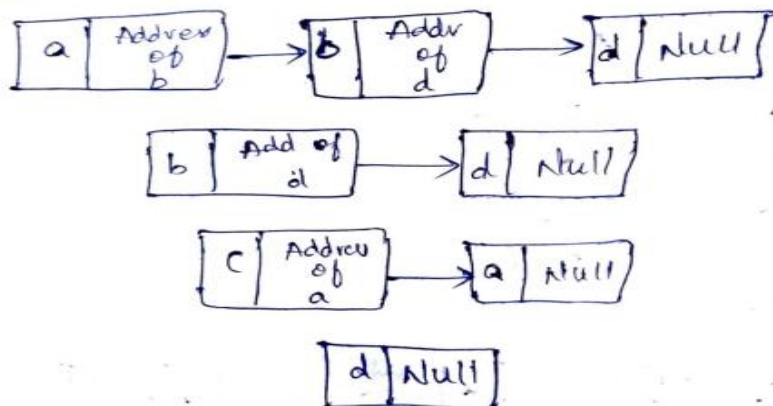


Linkedlist Representation

Eg. consider a directed graph

Adjacency list

Nodes	Adjacent Nodes
a	b, d
b	d
c	a
d	-

Linkedlist Representation

Q. What are graph traversal techniques Or Graph Searching Techniques.

Explain Depth First Search.

----- 10 Marks (V.Imp)

Ans: Graph traversal techniques (or) Graph searching techniques : A graph traversal technique is used to visit all the nodes or vertices " exactly one time without any loop".

There are two graph traversal techniques. They are :-

- i) Depth first search(DFs)
- ii) Breath first search (BFs)

Depth Traversal techniques (or) Graph searching Techniques: A graph traversal is used to visit all the nodes or vertices "exactly once without any loop".

there are two graph traversal techniques. They are:-

- i) Depth first Search (DFs)
- ii) Breadth first search (BFs)

Depth first search(DFs):A DFS use "Stack data Structure" to implement a traversal. create a stack with size equal to no.of vertices in a graph.

* DFS traversal gives a "spanning tree" as final result.

Algorithm for DFs:-

Step 1: Create a stack with size equal to no.of vertices in a graph.

Step 2: Select any vertex as initial node for the traversal visit the node and "Push it into stack".

Step 3: Select adjacent vertices of the node which are not visited.

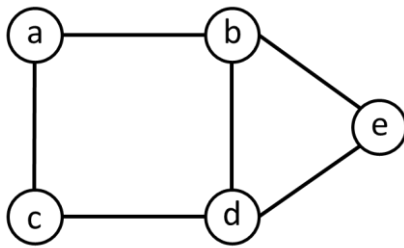
Step 4: Push them into stack

Step 5: Repeat steps 3,4 stack is full

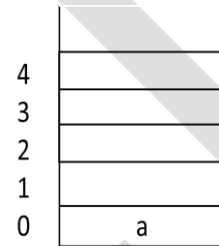
Step:5 Pop all the elements until stack is empty, and write to output.

Step:6 Use back tracking (reverse process) to print elements as final output or final spanning tree.

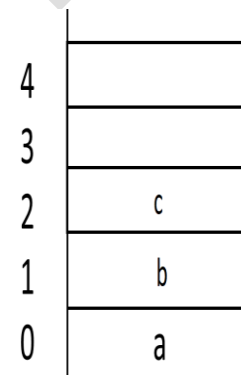
Example. consider a graph



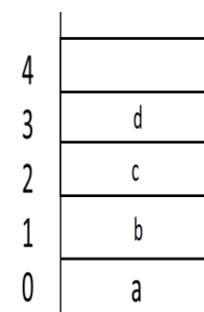
- 1) Create stack with size 5.
- 2) let us start from vertex "a" and push into stack.



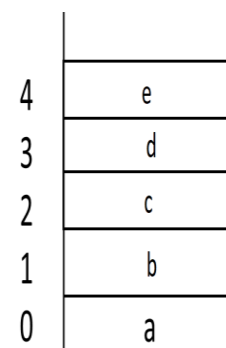
- 3) Adjacent nodes of (a) which are not visited are [b,c].
Push nodes which are not visited into stack..



- 4) Adjacent node of [c] which are not visited are [d].
Push nodes which are not visited into stack.



- 5) Adjacent nodes of [d] which are not visited are [e].
Push nodes which are not visited into stack.



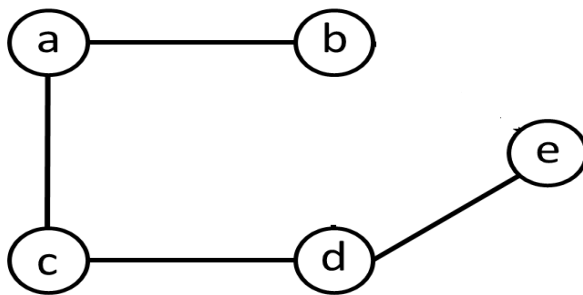
6) since, the stack is full, stop visiting the nodes.

7) pop all nodes from the stack. Hence the elements are $e \rightarrow d \rightarrow c \rightarrow b \rightarrow a$.

8) Print the elements in the reverse order to get the final spanning tree. Hence the elements are

$a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$
backtracking

Hence the final spanning tree for the given graph i.e.,



9) We can generate more than spanning tree for given graph.

Note: If only DFS or BFS ASKED WRITE COMPLETE ANS FOR 10MARKS.

IF BOTH ASKED AS TRAVERSAL TECHNIQUES , YOU WRITE BOTH DFS & BFS ONLY EXAMPLES.(NO NEED OF ALGORITHM)

Q. What are graph traversal techniques Or Graph Searching Techniques. Explain Breadth First Search . ----- 10 Marks (V.Imp)

Ans: Graph traversal techniques (or) Graph searching techniques : A graph traversal technique is used to visit all the nodes or vertices " exactly one time without any loop".

There are two graph traversal techniques. They are :-

- i) Depth first search(DFs)
- ii) Breath first search (BFs)

Breadth first search traversal(BFs): In BFs, we use "Queue Data Structure" to implements a traversal.

- * Create a queue with size equal to no.of vertices in a graph.
- * BFs traversal gives a spanning trees as a final result.

Algorithms for BFs:-

Step 1: Define a queue with size is equal to no.of vertices in a graph.

Step 2: Select any vertex as starting point. Insert the vertex into the queue.

Step 3: consider node at "Front position" and find adjacent nodes which are not visited.

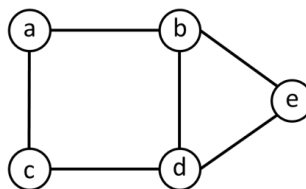
Step 4: Insert all adjacent nodes into queue.

Step 5: Delete node from queue & write it to output.

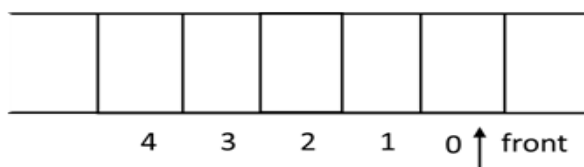
Step 6: Repeat step3,4,5 until queue becomes full.

Step 7: Delete all nodes until queue is empty and write them to output to give a final spanning tree.

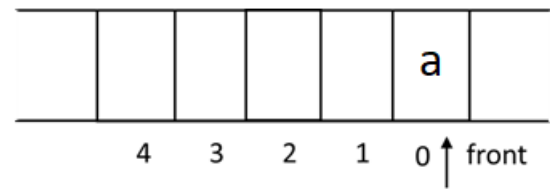
Example: Consider the graph



1) create a queue with size 5 and assume front =0

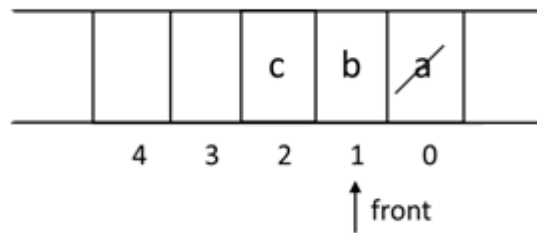


2) consider [a] as starting vertex and insert [a] into a queue



3) Adjacent vertices of [a] which are not visited are [c,b]

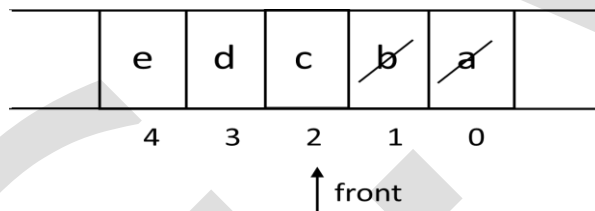
Insert [c, b] into queue and delete [a] from queue to output.



4) vertex at front=[b]

Adjacent vertex of [b] which are not visited = [d,e]

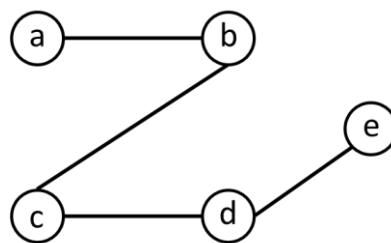
Insert [d,e] into queue and delete [b] from queue to output.



5) since queue is full delete all elements to output.

output = a--> b-->c-->d-->e

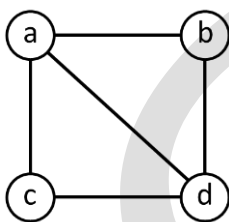
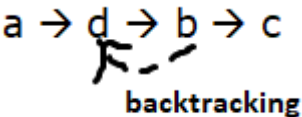
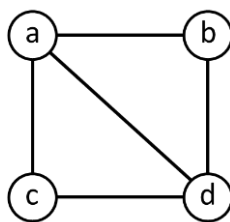
6) Hence final spanning tree is



* A graph can have more than one spanning tree.

Q) Difference between DFS and BFS?

A) Difference between DFS & BFS:

Depth First Search	Breadth First Search
1.DFS use "Stack Data Structure" to implement traversal	1.BFS use "Queue Data Structure" to implement traversal
2. DFS visit nodes of a graph depth wise until it reaches a leaf node.	2. BFS visit nodes of a graph level-by-level.
3. DFS has "back tracking of nodes".	3. BFS doesn't follow "back tracking of nodes".
4. DFS requires less memory space.	4. BFS requires more memory space.
5. DFS is faster process compared to BFS.	5. BFS is slower process compared to DFS.
6. Applications: 1. find shortest path b/w nodes of graph.	6. Applications: i. It is used in solving a Puzzles such a maze. ii. It is used in topological Sorting.
7. Example:  DFS Solution: $a \rightarrow b \rightarrow c \rightarrow d$ or $a \rightarrow c \rightarrow d \rightarrow b$ or $a \rightarrow d \rightarrow b \rightarrow c$  backtracking	7. Example:  BFS Solution: $a \rightarrow b \rightarrow c \rightarrow d$

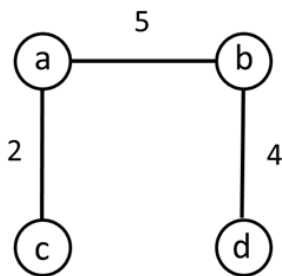
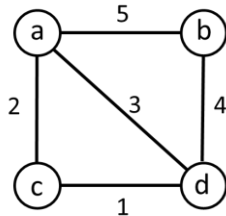
Q. Define spanning tree. Explain about minimum spanning tree?

A. Spanning Tree: A spanning tree is a sub graph which connects all the vertices of a given graph.

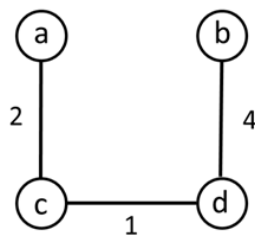
- ✓ A graph can have "more than one spanning tree".
- ✓ A spanning tree does not have cycles.

Minimum spanning tree: A minimum spanning tree of a graph 'G' is its spanning tree, in which "Sum of weights of edges is minimum".

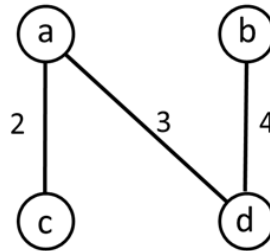
Example:- consider a graph



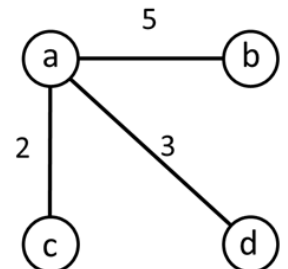
Cost : 12
S1



Cost 7
S2



Cost 9
S3



Cost : 10
S4

Spanning trees for above graphs are:-

Since s2 is the minimum spanning tree.

Applications of Minimum spanning Tree:

1. Minimum spanning tree is used to find shortest path in a graph.
2. Minimum spanning tree is used in google maps designing.
3. Minimum spanning tree is used in network designing.
4. Minimum spanning tree is used electronic circuit designing.

Algorithms used to implement minimum spanning tree are:-

1. Prim's Algorithm (Note: Refer last for Prim's, 1 algorithm is enough for 10 marks)
2. Kruskal's Algorithm

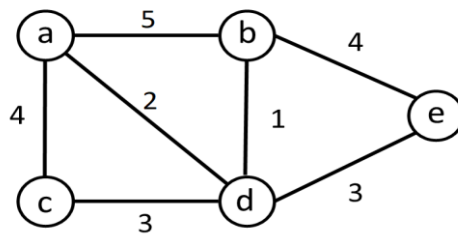
Kruskal's Algorithm:

Step1: Arrange all edges in ascending order of their weights.

Step2: Select the edge with minimum weight. If no cycle is formed, then add edge to spanning tree.

Step 3: Repeat step2, until spanning contains (n-1) edges, where 'n' is no. of nodes of graph.

Example: Consider the graph

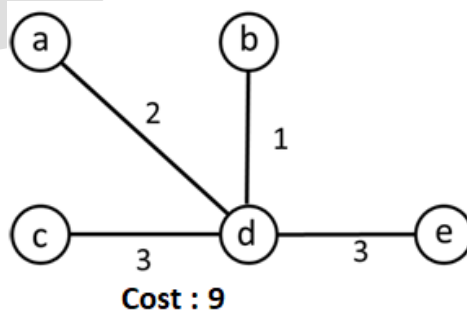


Step 1: Given graph contains "5" vertices. So no.of edges for final spanning tree is $(n-1)$ i.e, $(5-1) = 4$ edges. Where 'n' is no.of vertices.

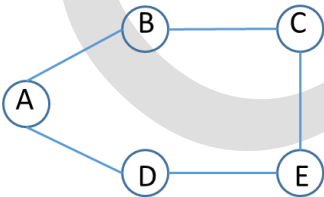
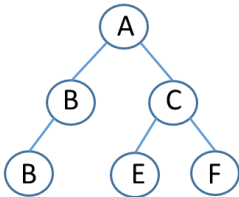
Step 2:

Edges	Weights
bd	1
ad	2
ed	3
cd	3
ac	4
be	4
ab	5

Step 3: Output of Final Minimum Spanning tree



Q. Difference between graph & tree.

GRAPH	TREE
1. A graph is a non-linear data structure. It is a collection of vertices and edges.	1. A tree is a non-linear data structure. It is a collection of nodes. It is represented in hierarchical order
2. Graph consists of vertices and edges	2. Tree consists of root node, parent node, leaf node etc.
3. There are 5 types of graph. They are: <ul style="list-style-type: none"> • Undirected graph • Directed graph • DAG • Simple graph • Connected graph (or) complete graph 	3. There are 6 types of graph. They are: <ul style="list-style-type: none"> • Complete binary tree • Binary Search Tree • Skew tree • AVL tree • Threaded Binary tree • Heap tree
4. Graph can be represented in 2 ways <ul style="list-style-type: none"> • Adjacency matrix • Adjacency list 	4. Tree can be represented in 2 ways <ul style="list-style-type: none"> • Linear representation using arrays • Linked list representation using DLL
5. Graph can be visited in 2 methods. They are BFS & DFS	5. Tree can be visited in 3 ways. They are <ul style="list-style-type: none"> • In-order • Pre-order • Post-order
6. Ex: 	6. They are no cycles or loops in a tree 
7. Graphs are used in google maps, circuit designing	7. Trees are used in searching and sorting techniques.

Q. Explain applications of graph. (Note: first write graph Definition)

ANS. Applications: Graphs are used in computer networks like local area network, internet connection.

- Graphs are used in designing google maps
- Graphs are used in routing algorithms in flight navigation and transportation.
- Graphs are used in circuit designing, database designing software engineering designing.
- Graphs are used in finding “molecular structures” in chemistry
- Graphs are used in mathematical geometry
- Graphs are used in compiler designing
- Graphs are used in designing robots
- Graphs are used in social networking
- Graphs are used in finding shortest path from one city to another city.

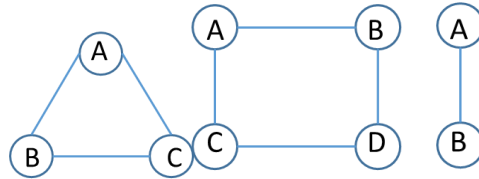
Note: Never asked in exam, but may ask for 5 marks . Refer DataStructures text book for more Explanation.

Q. Explain about connected components

A. Graph def., diagram [refer back]

Connected component: A connected graph is a undirected graph where there is a path between every pair of vertices in a graph. This is also known as "complete graph".

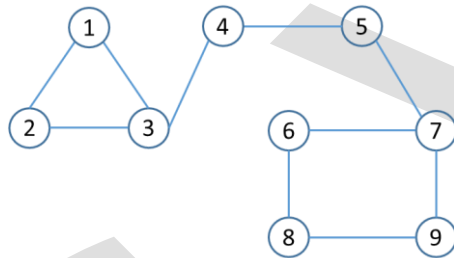
Eg:



Connected components in a graph:-

In a graph theory, a connected component of a graph is an edge which divides a graph into two subgraphs that edge and vertices are also "Articulation point"

Eg:



Here (3,4) & (5,7) & (4,5) are known as articulation points.

Q. Write about Prim's Algorithm. (Note: Refer Data Structures Text book for more explanation.)

Ans: Prim's Algorithm: Prim's is an algorithm used to find minimum spanning tree for the given graph.

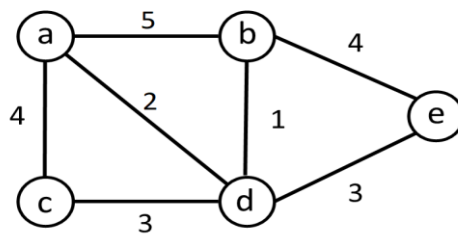
Algorithm:

Step 1: Select any node from given graph and add to Final spanning tree

Step 2: Find all adjacent edges of node which are not visited, and select edge with less cost. Add edge to Final spanning tree **when it doesn't form a cycle or loop.**

Step 3: Repeat step 2, until spanning contains $(n-1)$ edges, where 'n' is no. of nodes of graph.

Example: Consider the graph

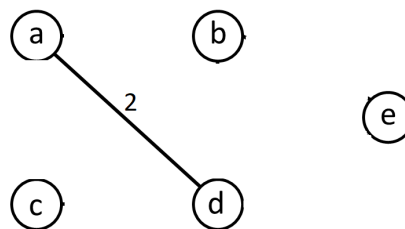


Step 1: Given graph contains "5" vertices. So no. of edges for final spanning tree is $(n-1)$ i.e., $(5-1) = 4$ edges. Where 'n' is no. of vertices.

Consider node 'a'.

Adjacent edges of 'a' with weights are: [ad - 2, ac - 4, ab - 5].

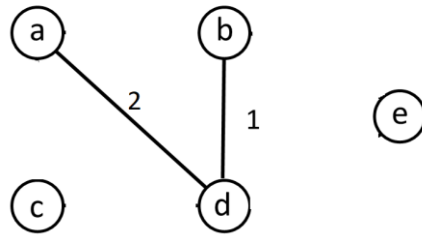
Select edge with minimum cost. i.e., "ad", add to spanning tree



Step 2: Consider node 'd'.

Adjacent edges of 'd' with weights are: [db-1, dc-3, de-3].

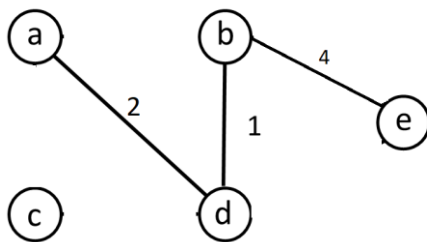
Select edge with minimum cost. i.e., "db", add to spanning tree



Step 3: Consider node 'b'.

Adjacent edges of 'd' with weights are: [be-4].

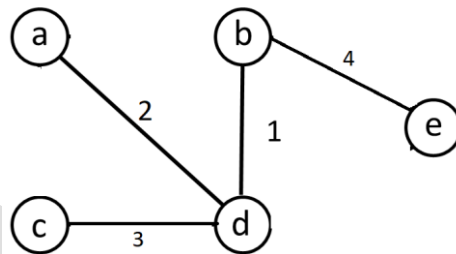
Select edge with minimum cost. i.e., "e", add to spanning tree



Step 4: Consider node 'd' since all nodes of 'b' are visited but spanning tree doesn't contain (n-1) edges, so we go to previous node i.e., 'd'

Adjacent edges of 'd' with weights are: [dc-3]

Select edge with minimum cost. i.e., "c", add to spanning tree.



Step 5: Since spanning tree contains (n-1) i.e., 4 edges, we stop. Hence final minimum spanning tree is :

