Graph & Application

Introduction to Graphs

- A Graph is a collection of nodes, which are called vertices 'V', connected in pairs by line segments, called Edges E.
- Sets of vertices are represented as V(G) and sets of edges are represented as E(G).

So a graph is represented as G = (V,E).

- There are two types of Graphs

  - Undirected Graph

  - Directed Graph

Undirected Graph

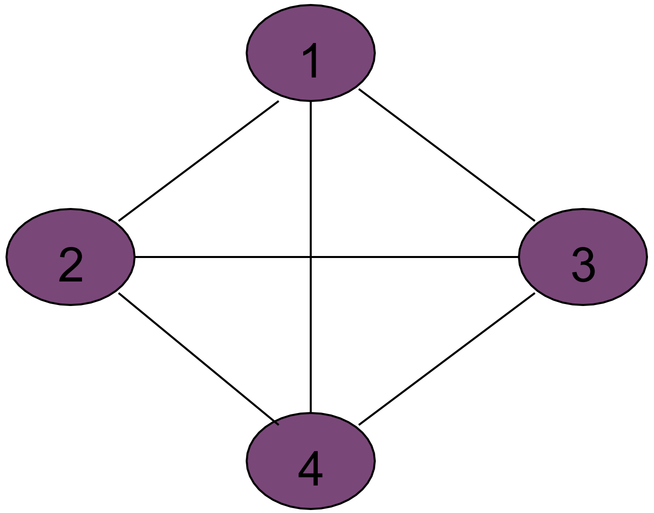
- An undirected Graph is one, where each edge E is an unordered pair of vertices. Thus pairs (v1, v2) & (v2, v1) represents the same edge.

G1

V(G1) = { 1, 2, 3, 4 }

E(G1) = { (1,2), (1,3), (1,4),

(2, 3), (2, 4), (3, 4) };



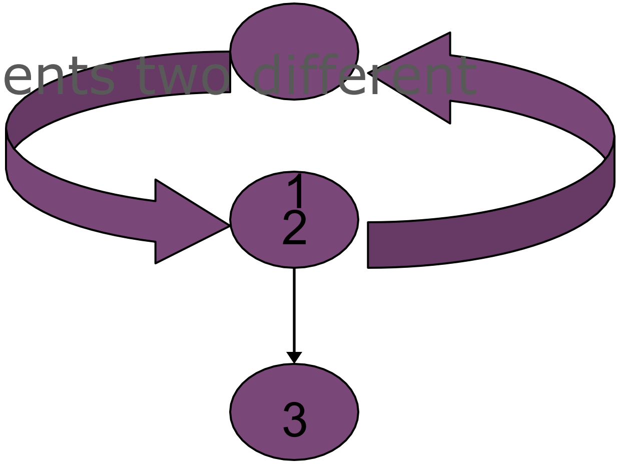
Directed Graph

- Directed Graph are usually referred as Digraph for Simplicity.
- A directed Graph is one, where each edge is represented by a specific direction or by a directed pair <v1, v2>.
- Hence <v1, v2> & <v2, v1> represents two different edges.

G2

E(G2) = { < 1, 2 >, < 2 , 1> , < 2, 3 >};

V(G2) = { 1, 2, 3 };



Definitions Related To Graphs

- Out – degree: The number of Arc exiting from the node is called the out degree of the node.
- In- Degree: The number of Arcs entering the node is the In degree of the node.
- Sink node: A node whose out-degree is zero is called Sink node.
- Path: path is sequence of edges directly or indirectly connected between two nodes.
- Cycle: A directed path of length at least L which originates and terminates at the same node in the graph is a cycle

Definitions Related To Graphs

- Adjacent: Two vertices in an undirected Graph are called adjacent if there is an edge from the first to the second.

- Incident: In an Undirected graph if e=(v, w) is an edge with vertices v and w, then v and W are said to lie on e, and e is said to be incident with v and w.

- Sub-Graph: A sub-graph of G is G1 if

  V(G1) is subset of V(G) .

  and E(G1) is subset of E(G)

Implementing Graph

- The graphs can be implemented in two ways

1. Array Method
2. Linked List

Array Method

- In this an array is used to store the values of nodes.

  e.g. V[] = { a, b, c, d };

  It means that node 1 has value a, 2 has value b, 3 has value c and 4 has value d.

- To show the connectivity between nodes a two dimensional array is used.

- This matrix will have an entry '1' if there is an edge between the two nodes, otherwise '0'.

- This matrix is known as "adjacency matrix".

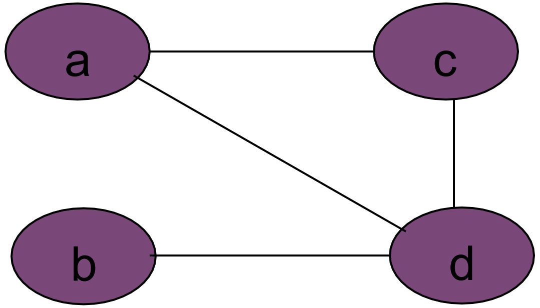
- In undirected graph this matrix will be symmetric.

Example – Array Method

- Char v[] = { a, b, c, d}

- Adj – Matrix adj[][4] =

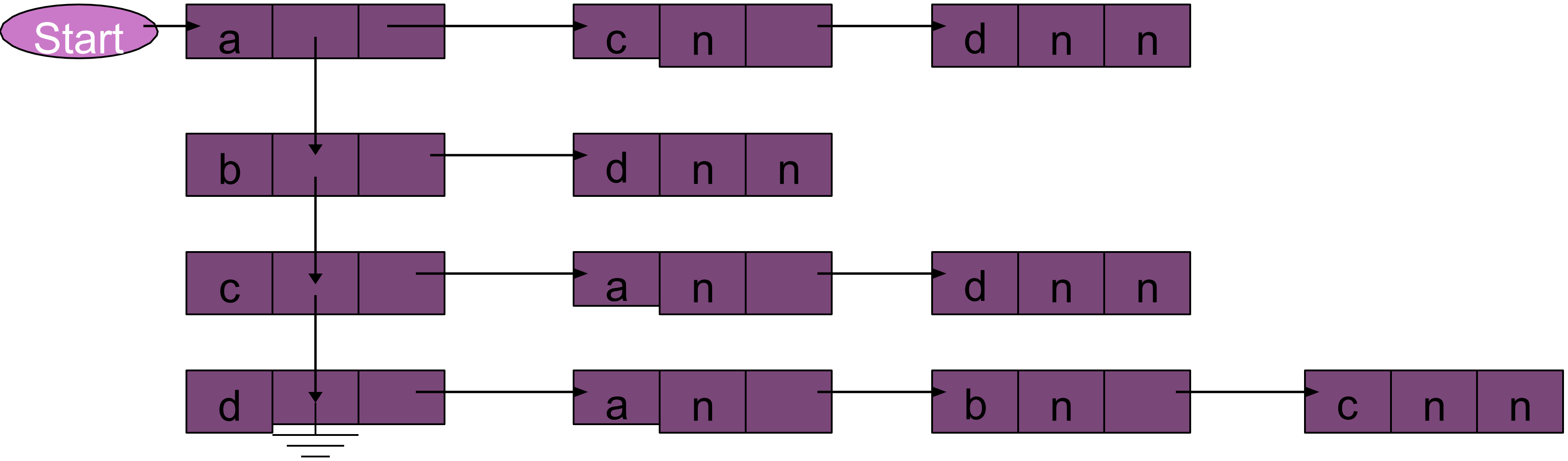
$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 & 1 \\ 4 & 1 & 1 & 0 & 1 \end{array}$$



Linked List Method

- The information and edges must be stored in the node structure.
- Consider the same graph in last example.
- The linked list representation is:

Linked List Representation of graphs



Graph traversal

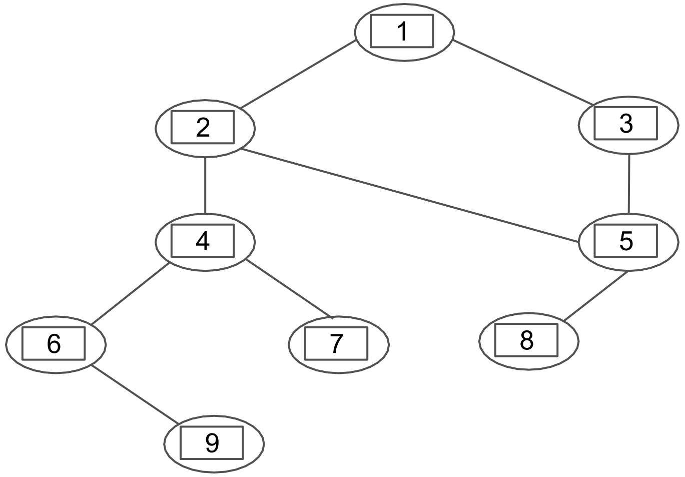
- Depth first search
- Breadth first search

Algorithm for Depth first search

- Step 1 : Select any node in the graph. Visit that node. Mark this node as visited and push this node onto the stack.
- Step 2: Find the adjacent node to the node on top of the stack, and which is not yet visited. Visit this new node. Make this node as visited and push it onto the stack.
- Step 3: Repeat the step 2 until no adjacent node to the top of stack node can be found. When no new adjacent node can be found, pop the top of stack.
- Step 4 : Repeat step 2 and 3 till stack becomes empty.
- Step 5: Repeat above steps if there are any more nodes which are still unvisited.

+ Depth first search

- Find out DFS for following graph.



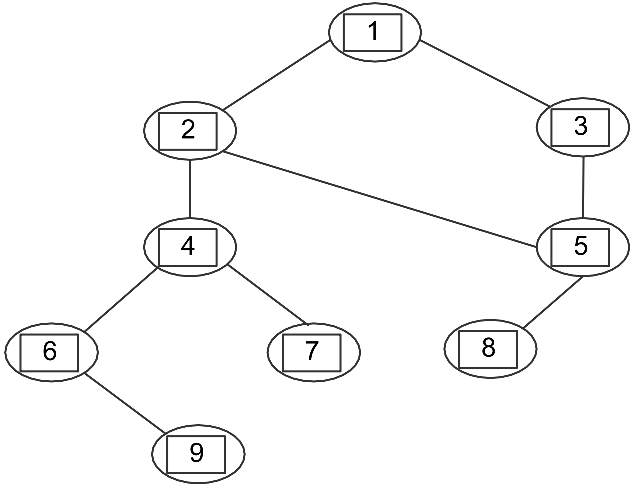
- 1 2 4 6 9 7 5 8 3
-

Algorithm for Breadth first search

- Step 1 : Start with any node and mark it as visited & place it into the queue.
- Step 2: Find the adjacent nodes to the node marked in step 1 and place it in the queue.
- Step 3: Visit the node at the front of the queue. Delete from the queue, place it’s adjacent nodes in the queue.
- Step 4 : Repeat step 3 till the queue is not empty.
- Step 5: Stop.

Breadth first search

- Find out BFS for following graph.



- 1 2 3 4 5 6 7 8 9

Applications of graph

- Game theory
- Telephone networking
- Scheduling of interrelated tasks for job
- Routing from one location to another

Spanning Tree

- A Sub-graph of a graph ‘g’ is a tree containing all the nodes of ‘g’ but less number of links with minimum cost.

- Minimal spanning tree is a spanning tree with smallest possible weight values

Prim’s algorithm

- This method builds minimum spanning tree edge by edge

- Select an edge from the graph whose cost is minimum among all the edges.

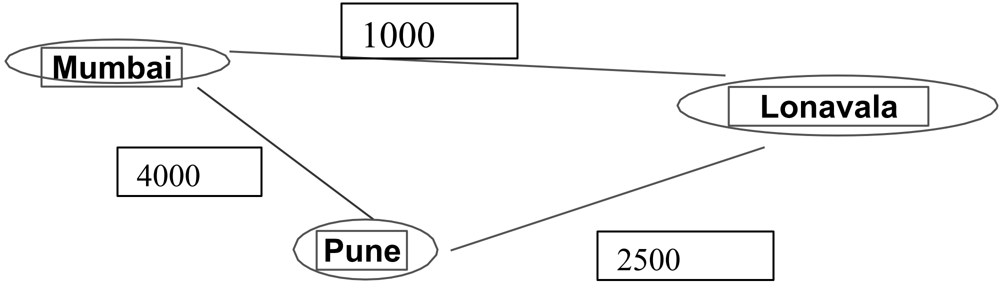
- Add the next edge (i,j) to the tree such that vertex i is already in the tree & vertex j is new one & cost of the edge (i,j) is minimum among all the edges (k.l) such that k is in the tree & l is not in the tree.

- While including any edge ensure that it doesn’t form a cycle.

- Time complexity: O(n2)

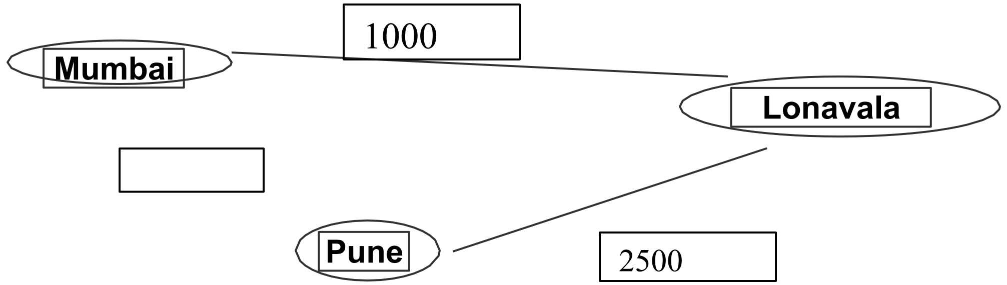
+ Prim's algorithm

- Find out minimum cost & spanning tree for following graph by applying Prim's algorithm



+ Prim’s algorithm

- Spanning tree using Prim’s algorithm is



- Total weight = 3500 This is also minimal spanning tree

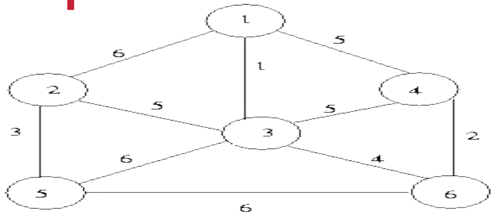
+ Kruskal’s algorithm

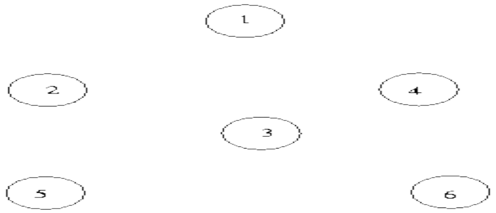
- It is another method for finding minimum cost spanning tree.
- Edges are added to the spanning tree in increasing order of cost.
- If the edge to added forms a cycle then it is discarded.
- Time complexity of algorithm:
  - O(e log e) + O(e log n)

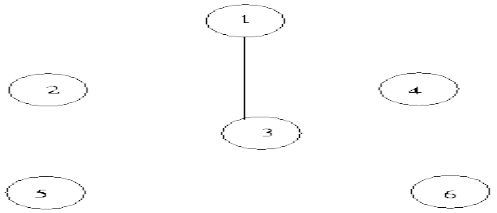
Algorithm

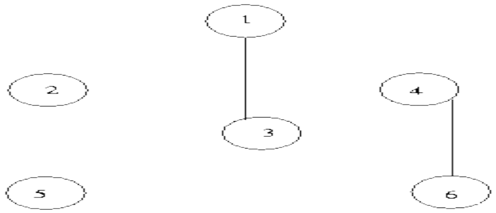
- create a forest *F* (a set of trees), where each vertex in the graph is a separate tree
- create a set *S* containing all the edges in the graph
- while *S* is nonempty
  - remove an edge with minimum weight from *S*
  - if that edge connects two different trees, then add it to the forest, combining two trees into a single tree
  - otherwise discard that edge
- At the termination of the algorithm, the forest has only one component and forms a minimum spanning tree of the graph.

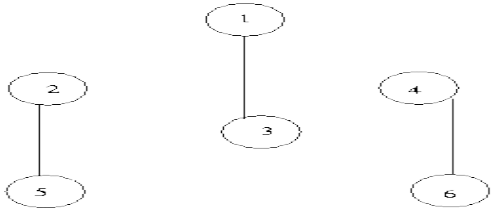
Example

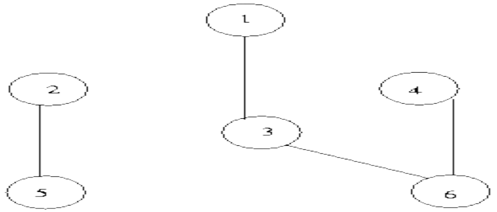


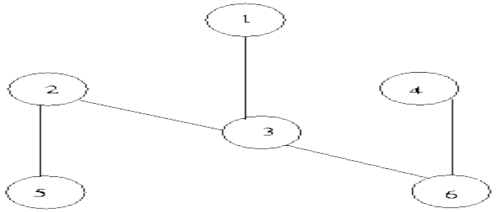

Initial Configuration


step1. choose (1,3)


step2. choose (4,6)


step3. choose (2,5)


step4. choose (3,6)


step5. choose (2,3)