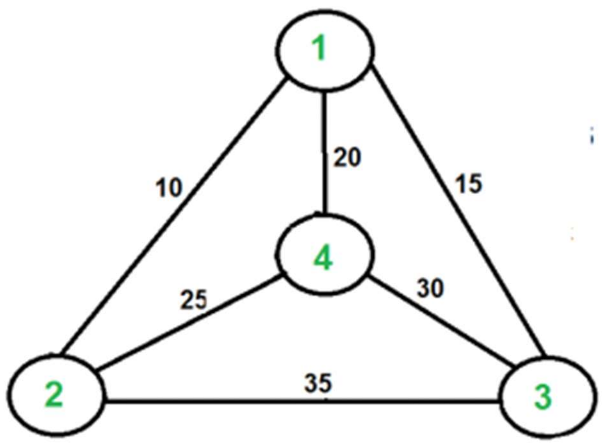
Travelling Salesman Problem (TSP)

- *"Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point."*
- There are two important things in this problem statement,
    - Visit every city exactly once
    - Minimize the total length of the tour (i.e.) find shortest possible route.
- Example: Consider the following set of cities:



- The problem lies in finding a minimal path passing from all vertices once.
- For example, the path Path1 {1, 2, 3, 4, 1} and the path Path2 {1, 2, 4, 3, 1} pass all the vertices but Path1 has a total length of 95 and Path2 has a total length of 80.
- A TSP tour in the graph is 1-2-4-3-1 whose cost is 10+25+30+15 = 80.
- The problem is a famous NP hard problem.
- There is no polynomial time know solution for this problem.
- There are approximate algorithms to solve the problem.
- The approximate algorithms work only if the problem instance satisfies Triangle-Inequality.

Triangle-Inequality:

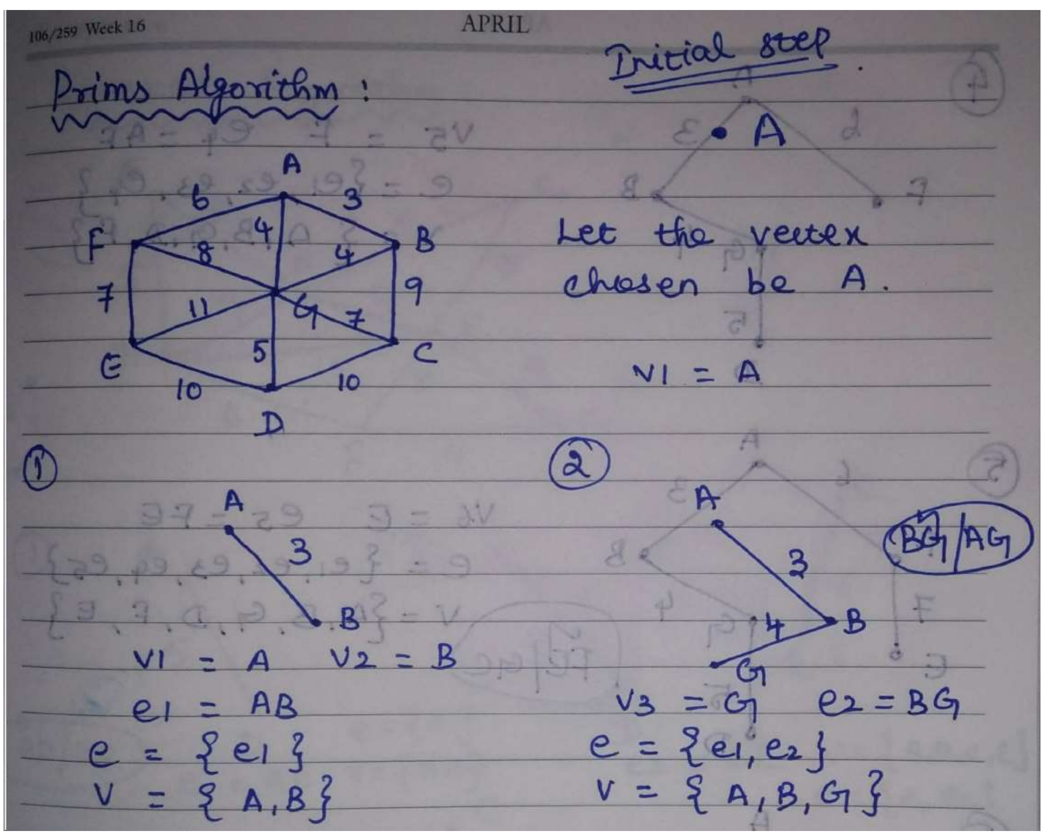
- The least distant path to reach a vertex j from i is always to reach j directly from i, rather than through some other vertex k (or vertices), i.e.,

  dis(i, j) is always less than or equal to dis(i, k) + dist(k, j).
- The Triangle-Inequality holds in many practical situations.
- When the cost function satisfies the triangle inequality, we can design an approximate algorithm for TSP that returns a tour whose cost is never more than twice the cost of an optimal tour.
- The idea is to use **M**inimum **S**panning **T**ree (MST).

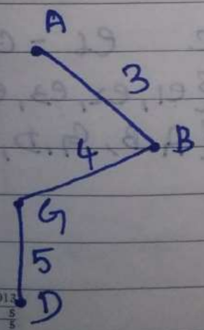
Prim's MST Algorithm Example:

Input: Weighted graph G = V, E

Output: Spanning tree of minimum cost



③

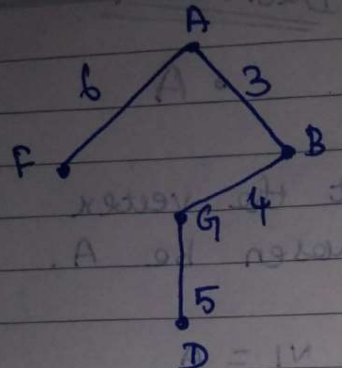


$V_4 = D \quad e_3 = GD$

$e = \{e_1, e_2, e_3\}$

$V = \{A, B, G, D\}$

④

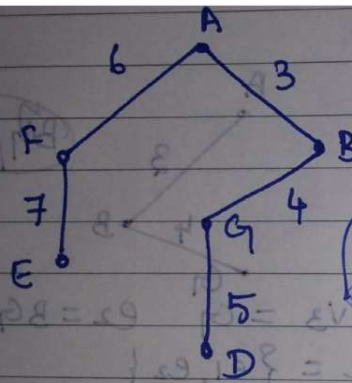


$V_5 = F \quad e_4 = AF$

$e = \{e_1, e_2, e_3, e_4\}$

$V = \{A, B, G, D, F\}$

⑤



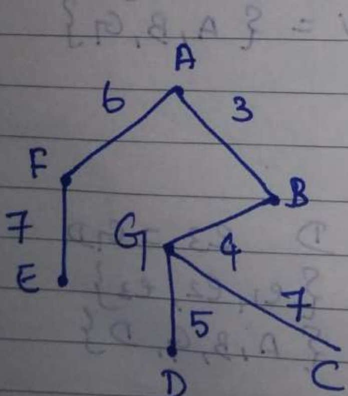
FE | GC

$V_6 = E \quad e_5 = FE$

$e = \{e_1, e_2, e_3, e_4, e_5\}$

$V = \{A, B, G, D, F, E\}$

⑥



$V_7 = C \quad e_6 = GC$

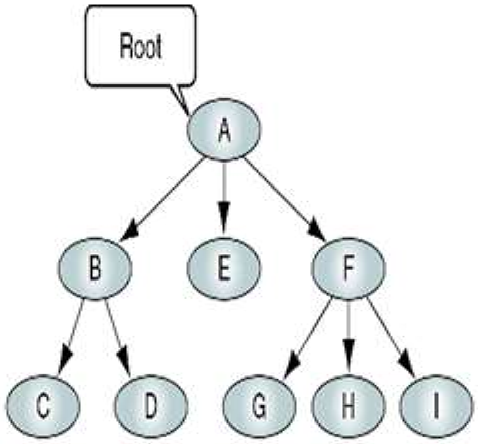
$e = \{e_1, e_2, e_3, e_4, e_5, e_6\}$

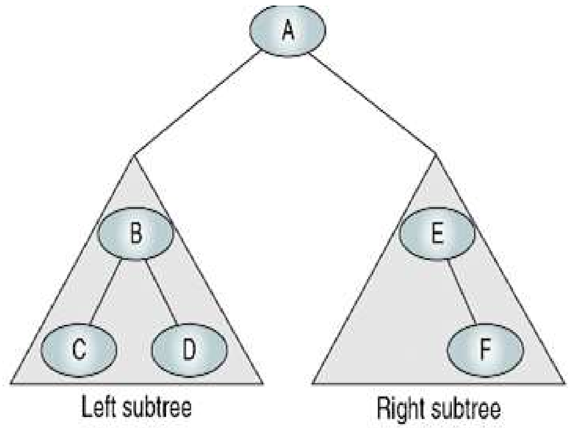
$V = \{A, B, G, D, F, E, C\}$

MST Cost = 32

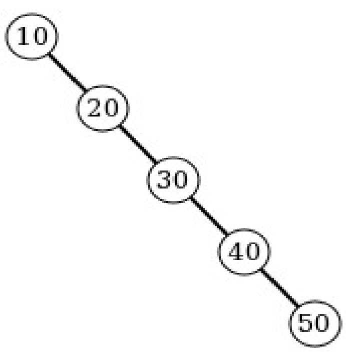
Binary Tree Traversal:

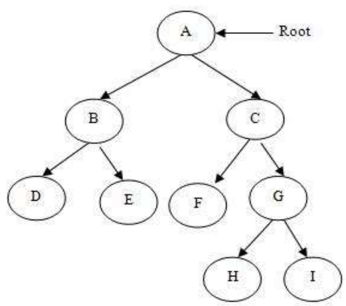
General tree **Binary tree**

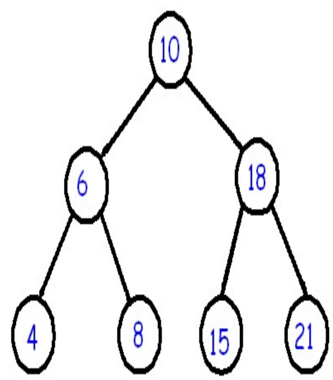




- Say whether the following trees are binary tree?



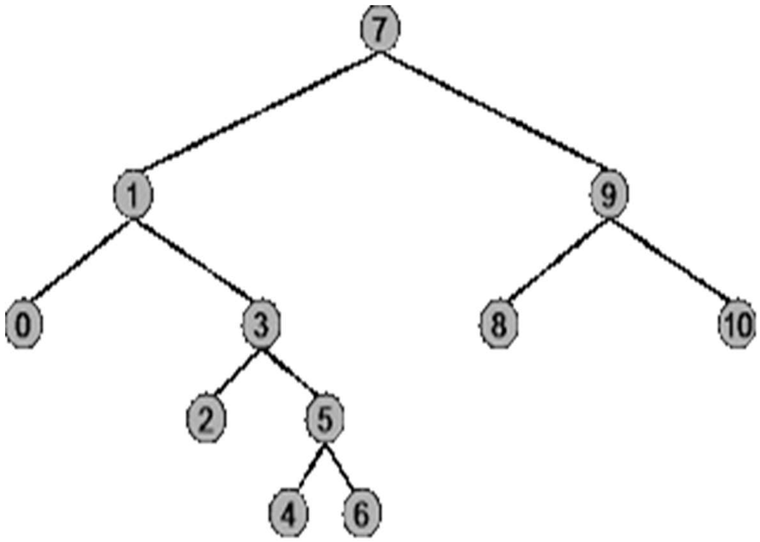




Preorder Traversal of Binary Tree:

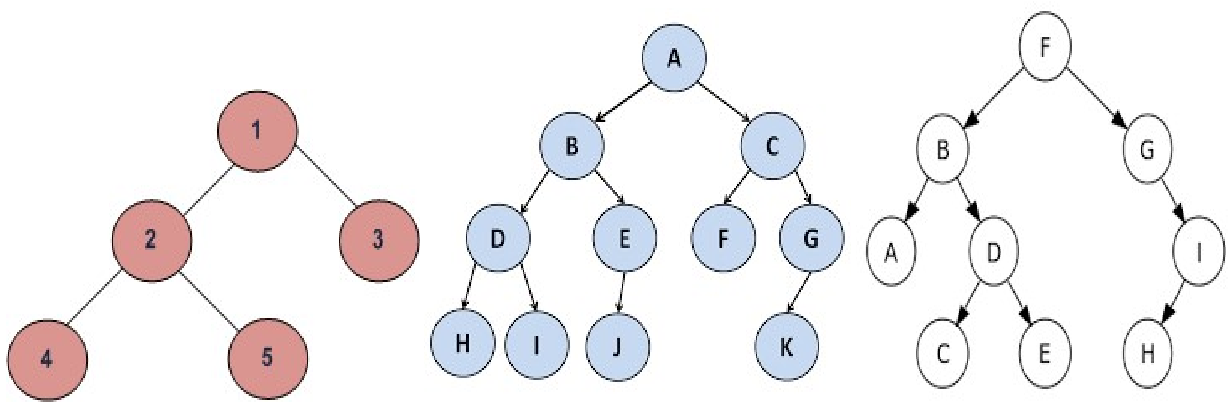
To traverse a binary tree in Preorder, following operations are carried-out

- Visit the root (v)
- Traverse the left subtree recursively (L)
- Traverse the right subtree recursively (R)



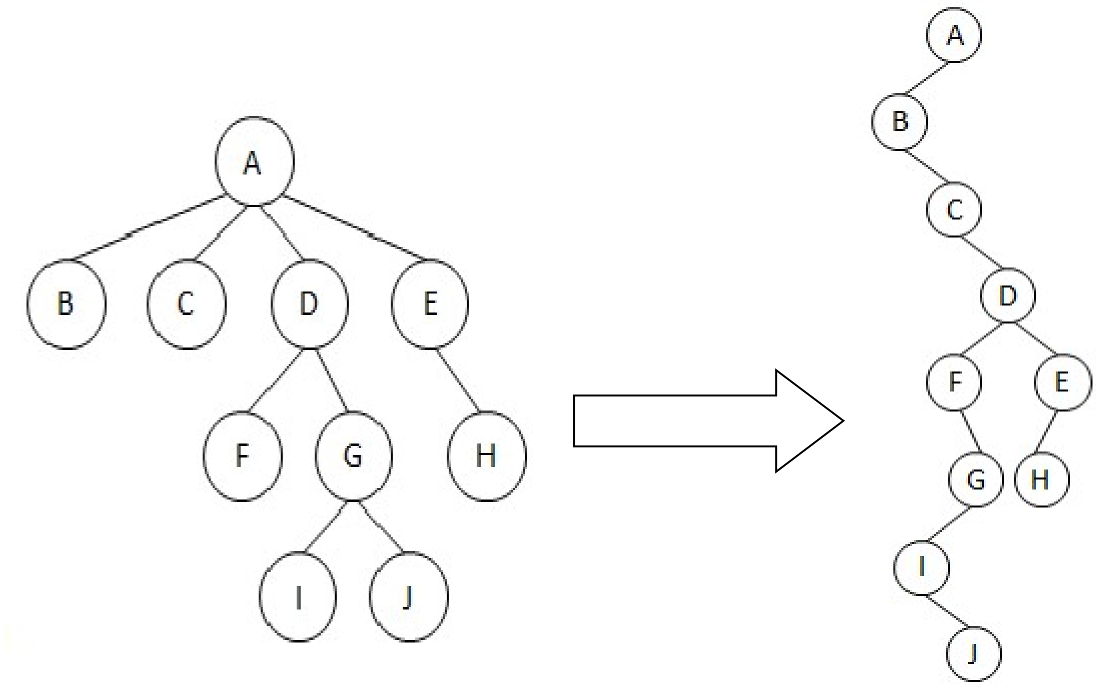
Therefore, the Preorder traversal of the given tree is: 7, 1, 0, 3, 2, 5, 4, 6, 9, 8, 10

Practice Problems: Do the preorder traversal

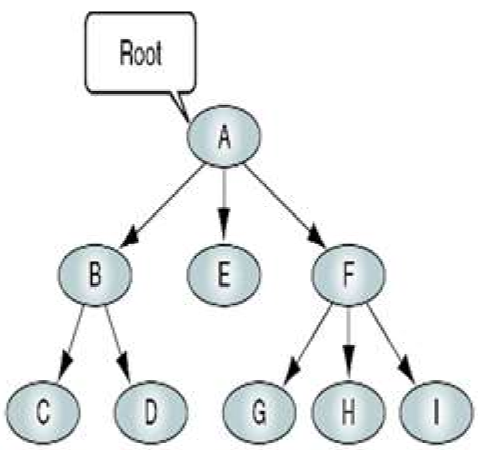


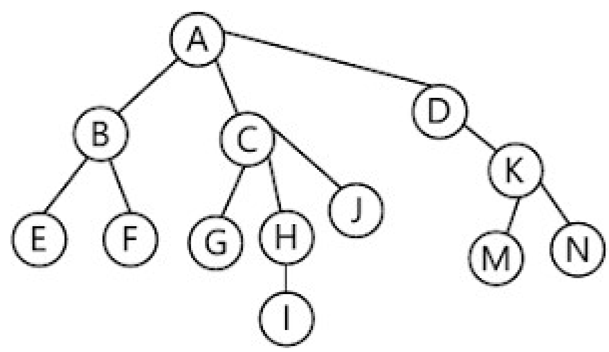
Converting General Tree to a Binary Tree:

- Binary tree left child = leftmost child
- Binary tree right child= right sibling



Practice Problems: Convert into binary tree

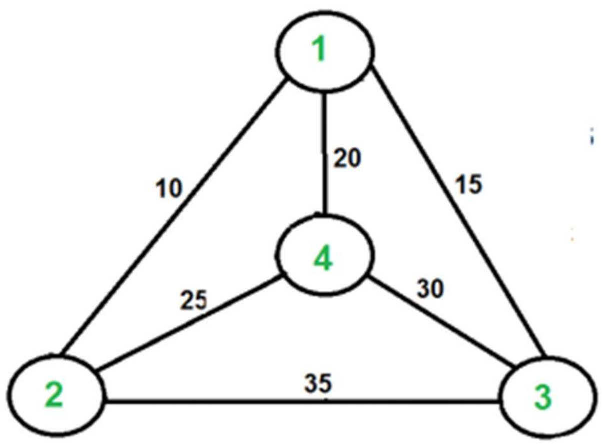




Euclidean TSP Approximation Algorithm:

1. Compute a minimum spanning tree T connecting the cities.
2. Visit the cities in order of a preorder traversal of T.

Approximate TSP Algorithm for the example graph given below:

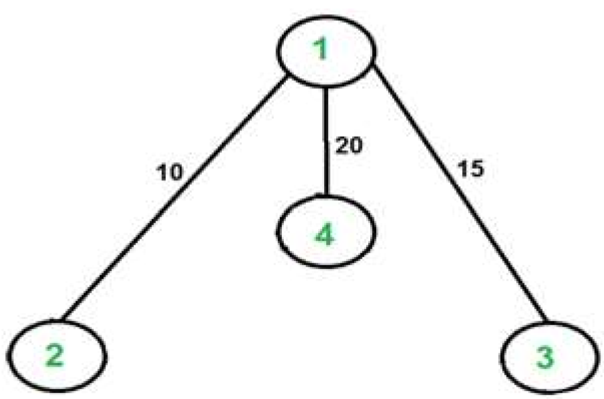


1) Let 1 be the starting and ending point for salesman.

2) Construct MST from with 1 as root using Prim's Algorithm.

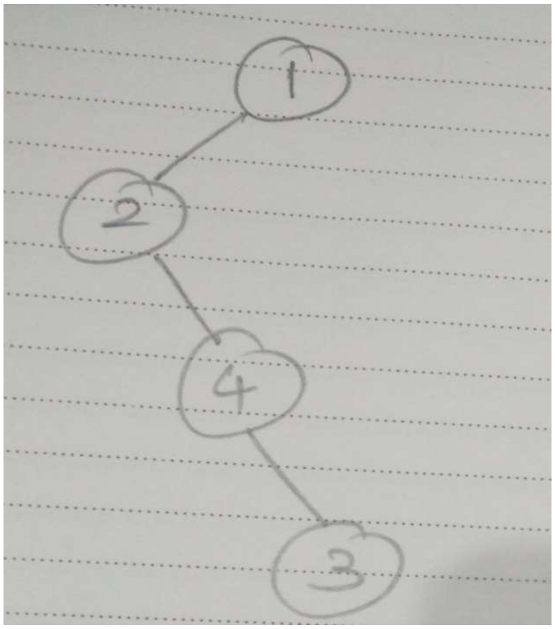
3) List vertices visited in preorder walk of the constructed MST and add 1 at the end.

The MST constructed with 1 as root is given below.



MST constructed is a general tree. So, convert into BT.

Corresponding binary tree is given below.



Now, the preorder traversal of MST is 1-2-4-3.

Adding 1 at the end gives 1-2-4-3-1 which is the output of this algorithm.

Optimal Tour Path = 1-2-4-3-1

Tour cost = 80

- In this case, the approximate algorithm produces the optimal tour, but it may not produce optimal tour in all cases.
- The cost of the output produced by the above algorithm is never more than twice the cost of best possible output.

Note: Approximation algorithms that produce result whose size is never more than twice the size of the optimal solution is called 2-approximation algorithm.