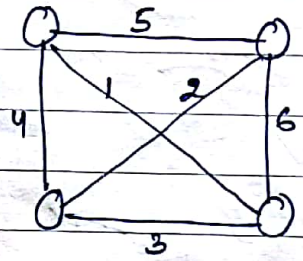
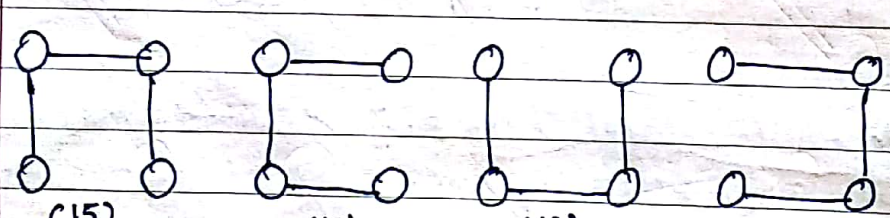
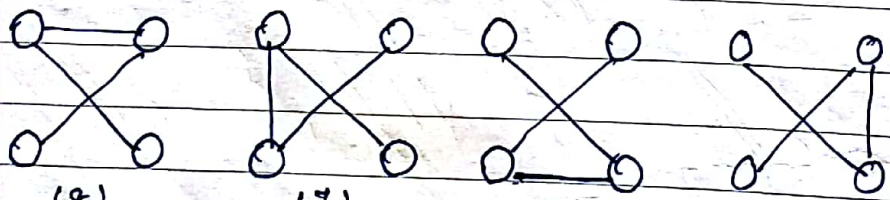
Date
Page

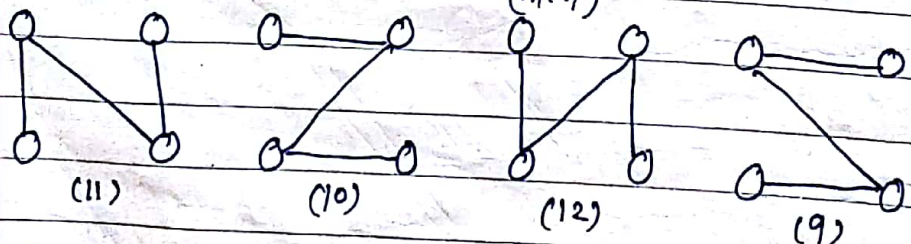
MINIMUM SPANNING TREE (MST) (Greedy Approach)

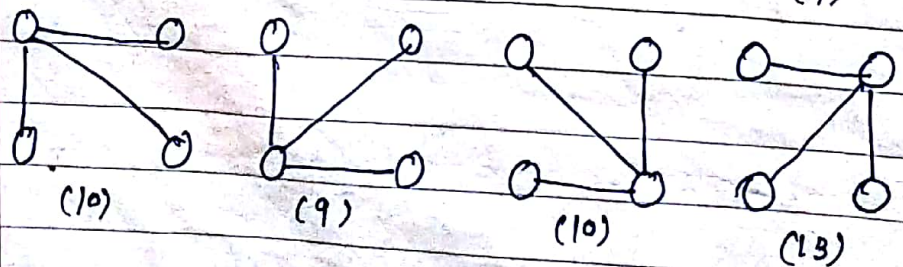
→ A spanning tree of a graph is a sub graph that contains all the vertices and is a tree.
→ A graph may have many spanning trees.
→ A minimum spanning tree is a spanning tree with weight less than or equal to the weight of every other spanning tree.
→ Most generally, any undirected graph has a minimum spanning forest.
→ Spanning tree doesn't contain any cycles or self loop.
→ A graph may have many spanning trees.











→ Let $G = \langle V, E \rangle$ be the undirected graph. T is the spanning tree of G, iff T contains all vertices of graph and subset of edges E' of graph.

$T = G(V, E')$

where E' is subset of E.

→ Let $G = \langle V, E^* \rangle$ be the undirected weighted graph, minimum spanning tree $T = G\langle V, E' \rangle$ contains all the vertices and subset of edges is said to be minimum spanning tree if it contains minimum cost.

→ Minimum spanning tree algorithms are

1. Kruskal's algorithm
2. Prim's algorithm.

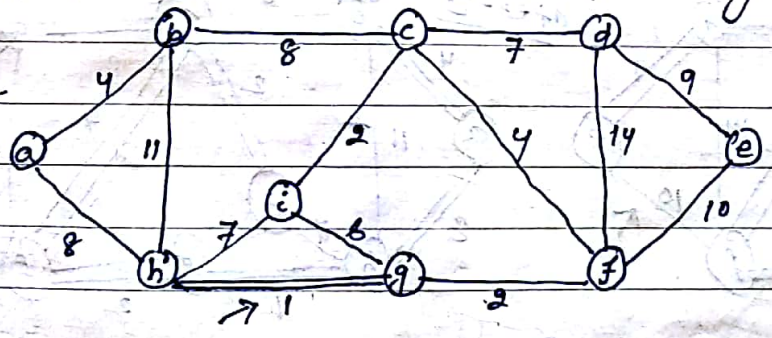
KRUSKAL'S ALGORITHM

→ Kruskal's algorithm deals with edges to determine MST.

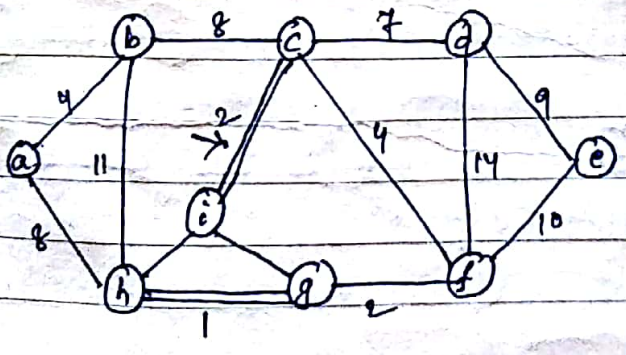
→ If a graph is not connected, then it finds a minimum spanning forest.

→ Example: Find MST of the following graph using Kruskal's Algo.

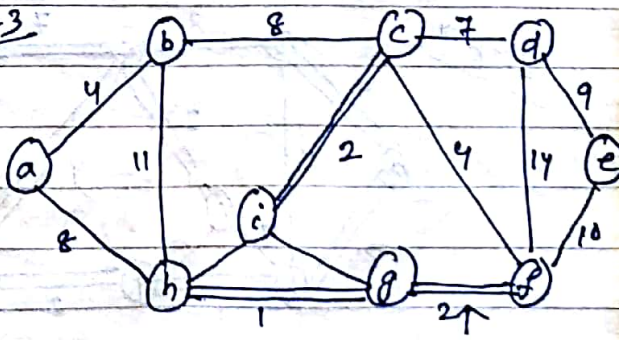
Step-1



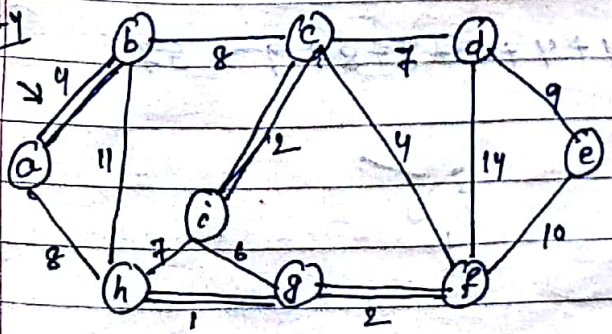
Step-2



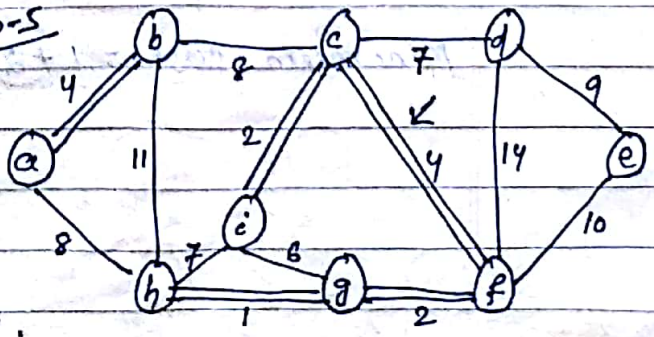
Step-3



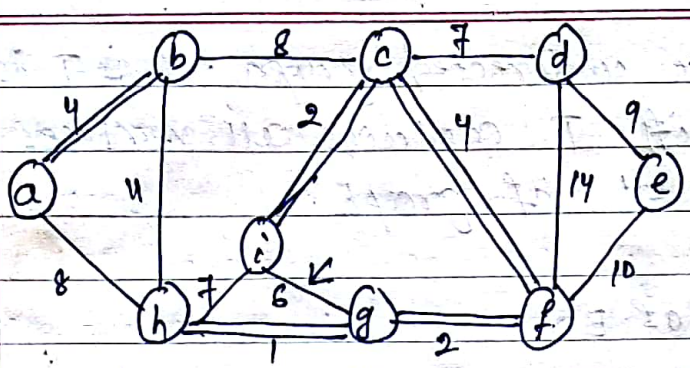
Step-4



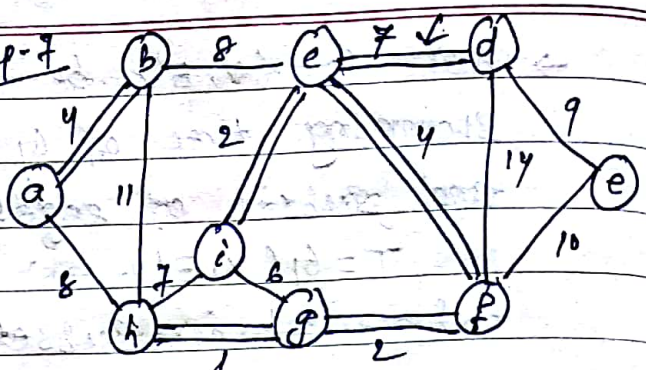
Step-5



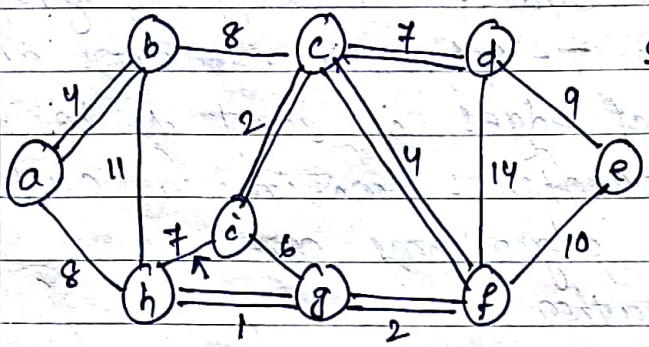
Step-6



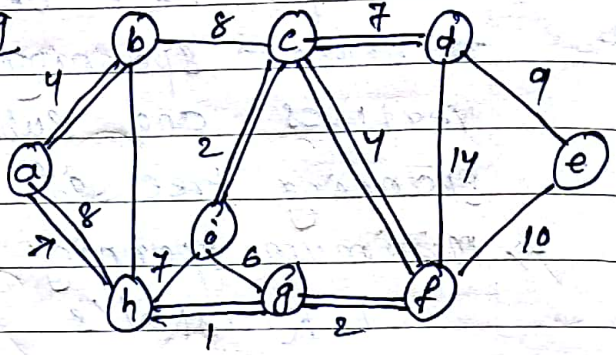
Step-7



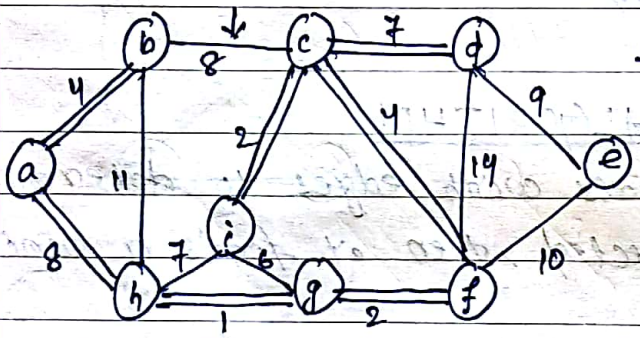
Step-8



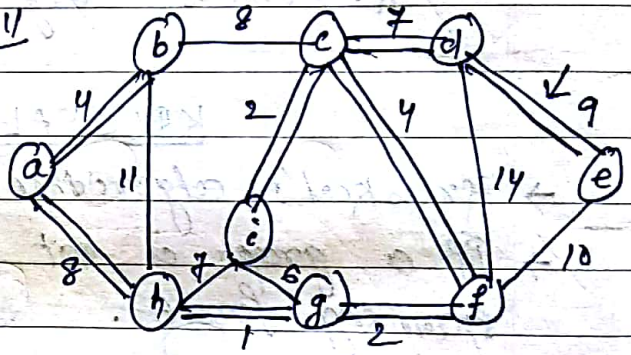
Step-9



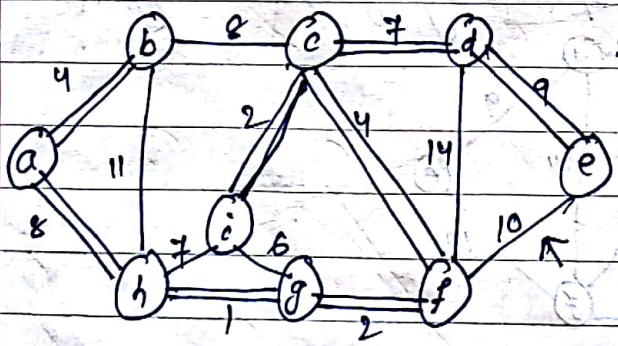
Step-10



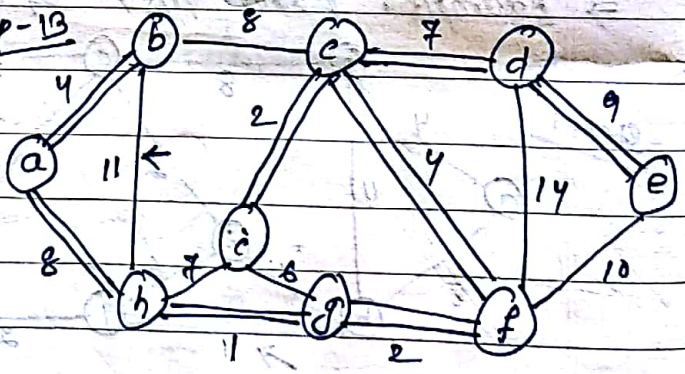
Step-11



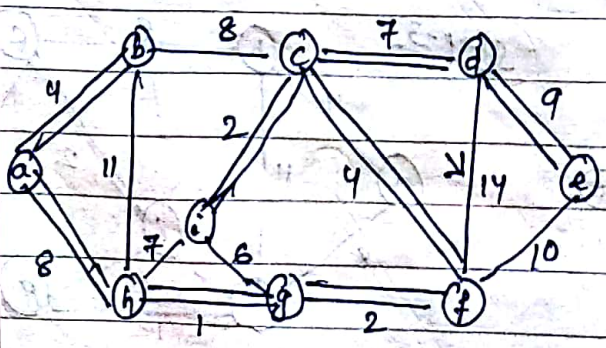
Step-12



Step-13



Step-14



Minimum cost = 1 + 2 + 2 + 4 + 4 + ~~8~~ 7 + 8 + 9 = 37

```
MST-KRUSKAL(G, w)
1. A ← ∅
2. for each vertex v ∈ V[G]
3.     do MAKE-SET(v)
4. Sort the edges of E into nondecreasing order by weight w
5. for each edge (u,v) ∈ E, taken in nondecreasing order
                                    by weight.
6.     do if FIND-SET(u) ≠ FIND-SET(v)
7.         then A ← A ∪ {(u,v)}
8.              UNION(u,v)
9. return A
```

→ Initialize the set A to the empty set.
→ Create |V| trees, one containing each vertex.
⇒ The edges in E are sorted into increasing order by weight.
→ Check, for each edge (u,v), whether the endpoints u and v belong to the same tree.
→ If they are belongs to same tree, then the edge (u,v) can't be added to the forest without creating a cycle and the edge is discarded.
→ If two vertices belong to different trees, then the edge (u,v) is added to A and the vertices in the two trees are merged.
→ Sort the edges by weight using comparison sort in O(ElogE) time
→ Next we use a disjoint-set data structure to keep track of which vertices are in which components.
→ Disjoint-set forests with union by rank can perform O(E) operations in O(Elog V) time.
→ Thus the total time is O(ElogV)

PRIM'S ALGORITHM

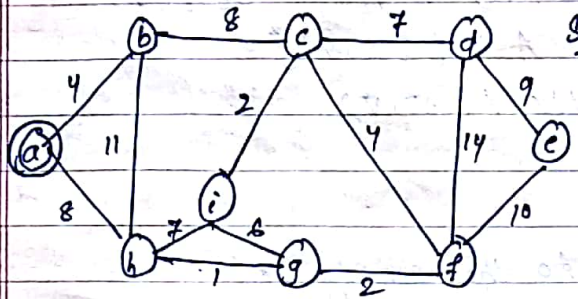
⇒ prim's algorithm uses vertex connections in determining the MST.

→ prim's algorithm has the property that the edges in the set A always form a single tree.

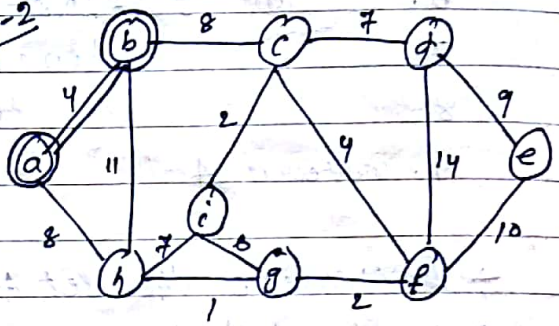
→ The tree starts from an arbitary root vertex r and grows until the tree spans all the vertices in V.

→ At each step, a light edge is added to the tree A.

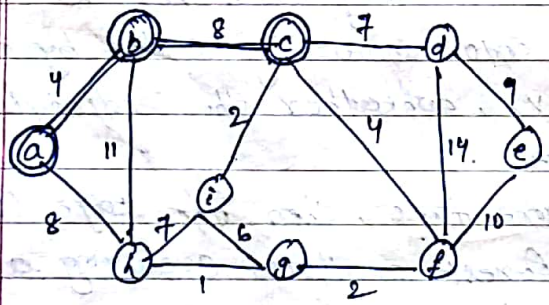
step-1



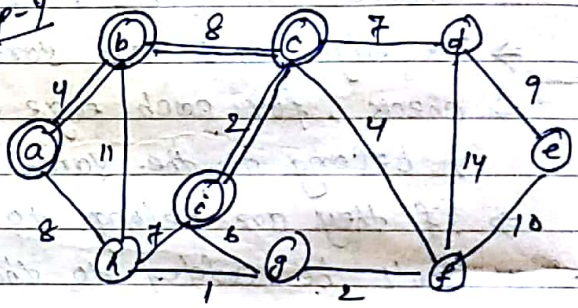
step-2



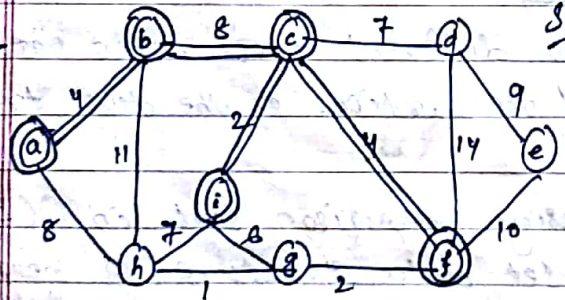
step-3



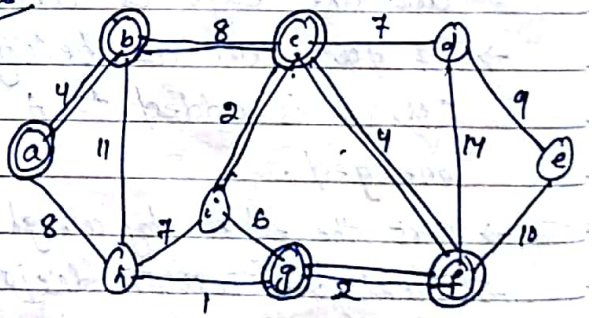
step-4



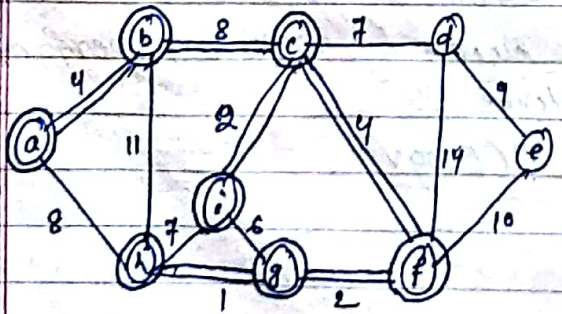
step-5

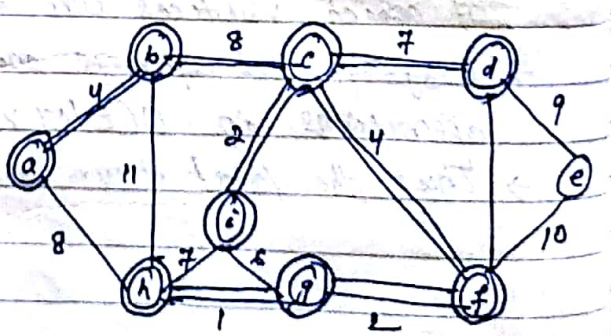


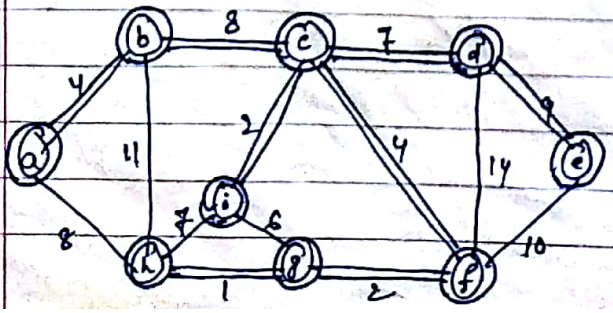
step-6



step-7







Minimum cost =
= 4+8+2+4+2+7+9+1
= 37

17

```
MST-PRIMS(G, w, r)
1. for each u ∈ V[G]
2.     do key[u] ← ∞
3.        π[u] ← NIL
4. key[r] ← 0
5. Q ← V[G]
6. while Q ≠ ∅                                   |V| times
7.     do u ← EXTRACT-MIN(Q)                     log V      O(V log
8.        for each v ∈ Adj[u]                    O(E)
9.            do if v ∈ Q and w(u,v) < key[v]
10.               then π[v] ← u
11.                    key[v] ← w(u,v)           (log V)
```

Complexity:

→ The performance of prim's algorithm depends on how we implement the min priority queue Q.

→ If the Q is implemented as a binary min-heap, we can use BUILD-MIN-HEAP procedure which takes O(V) ti

→ The while loop is executed |V| times and since each ~~Extract~~ EXTRACT-MIN operation takes O(log V) time. so, total time for all calls to EXTRACT-MIN is O(V lg V)

→ The for loop is executed O(E) times

→ The sum of the lengths of all adjacency lists is 2|E|

→ The assignment in line no. 11 involves an implicit DECREASE-KEY operation on the min heap ~~op~~ can be implemented in binary min heap in O(lg V) time.

→ The total time for prim's algorithm is

O(V log V + E log V) = O(E log V)