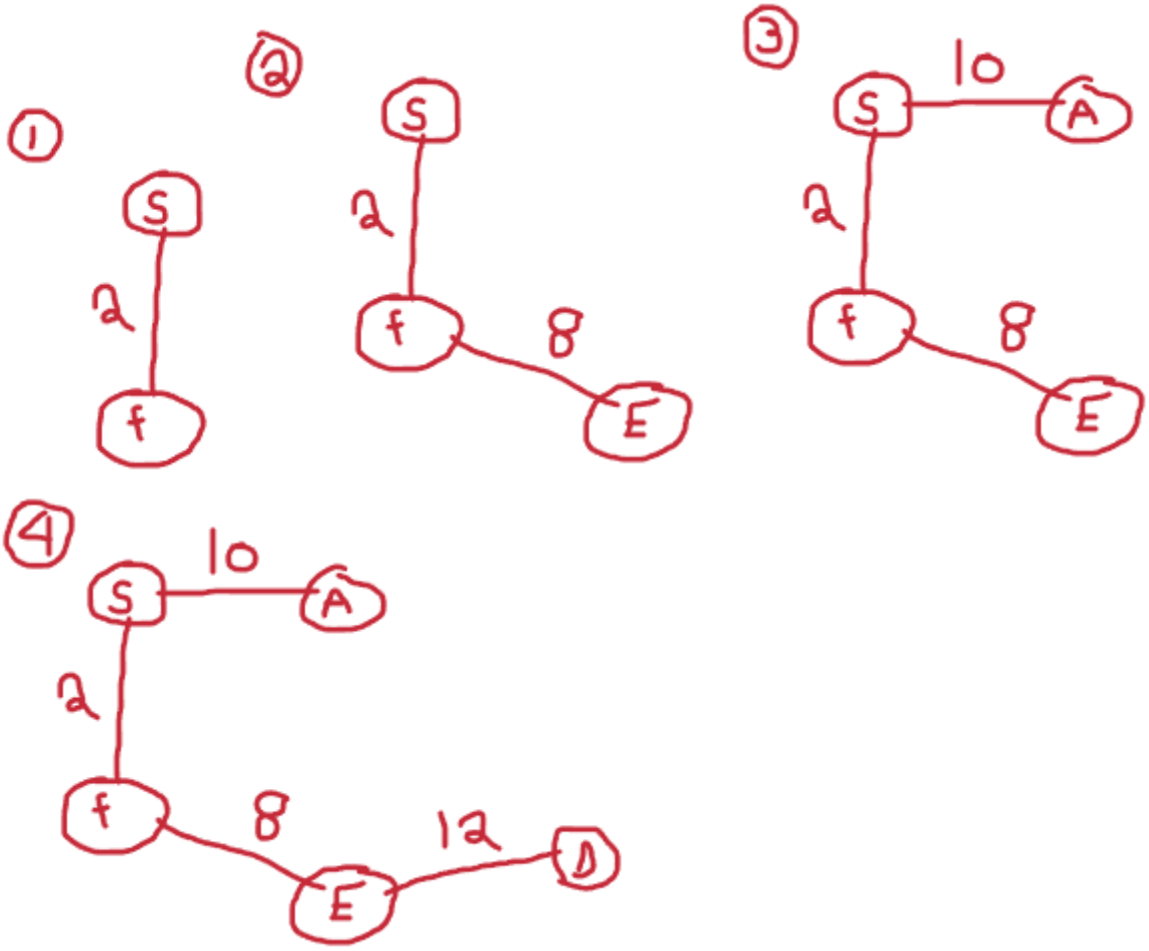
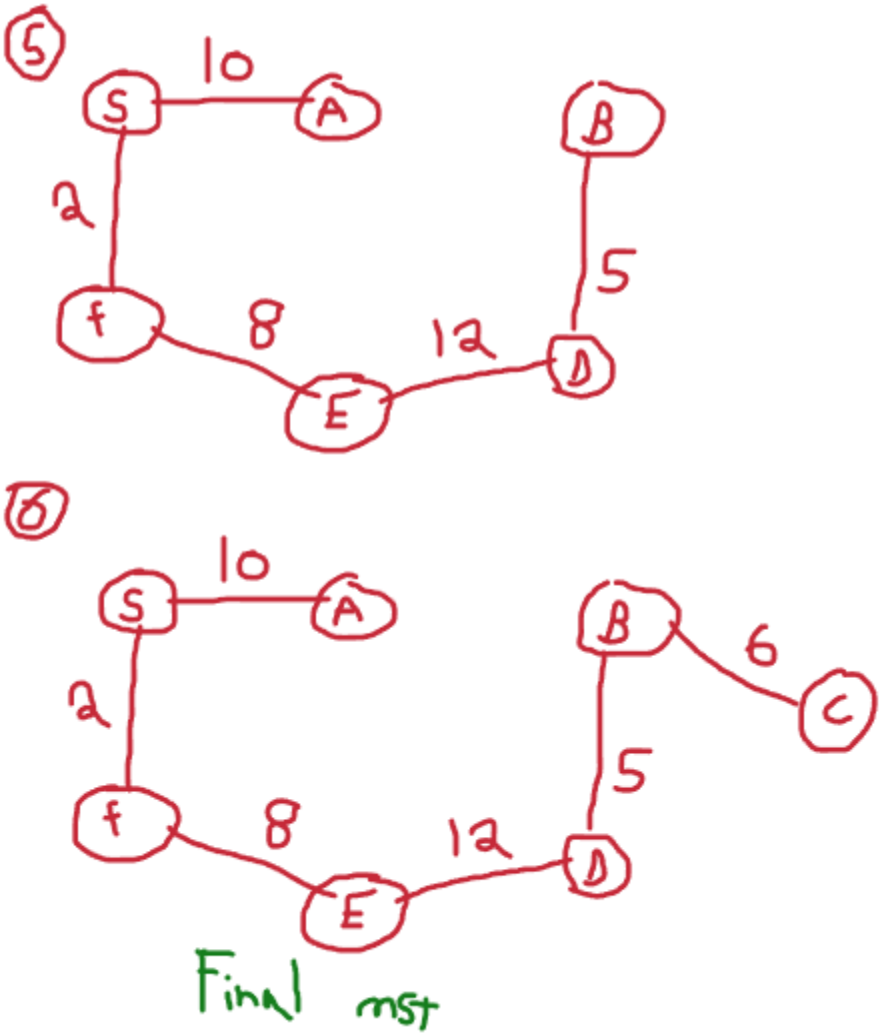
#2





Explanation: Start at Vertex S, pick the edge with less weight and include in mst (SF) , next check edged incident to S and F, pick minimum cost edge (FE), next pick edges S,F,E having less cost (SA), and so on

#3
It takes o(v) time to initiate, it takes o(E log E) time to sort edges by weight, it takes o(Ea(E,V)) to process the edges. If the edge weights are integers of 1 to w, then the sorting would take O(E) time and the total running time would be o(Ea(v))

#4
Tree createMST(G, w) // G is the connected graph with w as the weight matrix
{
for every vertex create a set of vertices which will be included in the MST when condition satisfies.
sort the edges in increasing weight order and store in E array.
Take (u,v) a pair of adjacent edge in increasing order of //weight from E array

check if u and v are valid vertices and in present in the set and they are not the same vertex

if u and v are different vertices add them to the MST

included u,v vertices need to be checked if there is a cycle formation. If cycle formed, discard it.

if cycle not formed MST retains it and the weight is added to the MST cost and total minimum cost is retrieved.

Return mst

}

#5

```
Function MBST(V, E)
    if |E| = 1 then
        Return E
     else
        A, B ?  Partition E in two halves around median
              A is higher half, B is lower half
        F ? Connected components of B
        if |F| = 1 and spans all vertices then
            Return MBST(V, B)
        else
           V' ? create one vertex for each connected component in F
                 plus vertices missing from F
           B' ? Edges from A that connects components in F
                 and edges to vertices not in F
           Return F ? MBST(V', B')
        end if
    end if
 end procedure
```