DEEMED TO BE UNIVERSITY

Advanced Algorithms & Data Structures

Department of CSE

ADVANCED ALGORITHMS AND DATA STRUCTURES

23CS03HF

Topic:

Minimum Cost Spanning Tree

Session - 25

AIM OF THE SESSION

To familiarize students with the concept of Spanning Tree

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1.Demonstrate :- Minimum Cost Spanning Tree.

2.Describe :- Minimum spanning tree and its related algorithms like Prim's and Kruskal's methods

LEARNING OUTCOMES

At the end of this session, you should be able to:

1. Define :- Minimum Cost Spanning Tree.
2. Describe :- Minimum spanning tree and its related algorithms like Prim's and Kruskal's methods
3. Summarize:- How to build an MST for a given graph using Prim's and Kruskal's Algorithm.

Spanning Tree

- **Graph**: Set of Vertices and edges G: { V,E }
- A *tree* is a connected undirected graph that contains no cycles.
- i.e **Tree** : Tree is Graph T: { V,E}

  ⍰ Acyclic

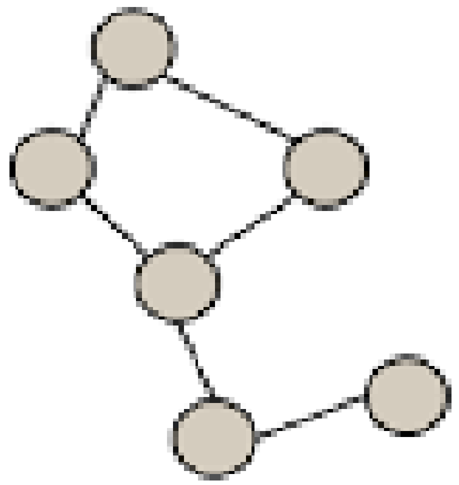
  ⍰ N nodes

  ⍰ N-1 edges
- Note: Every Tree is a Graph , But Every Graph is not a Tree

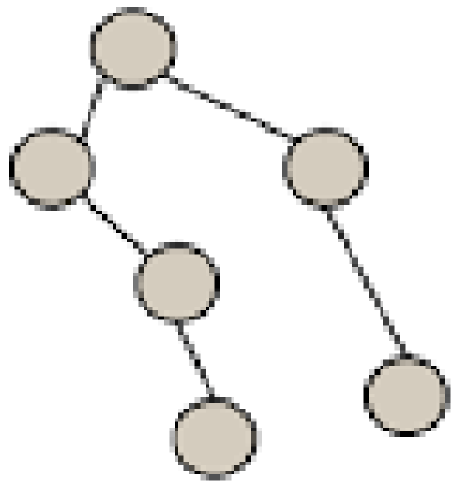
Properties of a Spanning Tree

- A connected graph G can have **more than** one spanning tree.
- All possible spanning trees of graph G, have the **same number of edges and vertices**.
- The spanning tree **does not** have any cycle (loops).
- **Removing one** edge from the spanning tree will make the graph disconnected, i.e. the spanning tree is **minimally connected**.
- **Adding one edge** to the spanning tree will create a circuit or loop, i.e. the spanning tree is **maximally acyclic**.

Graphs & Trees

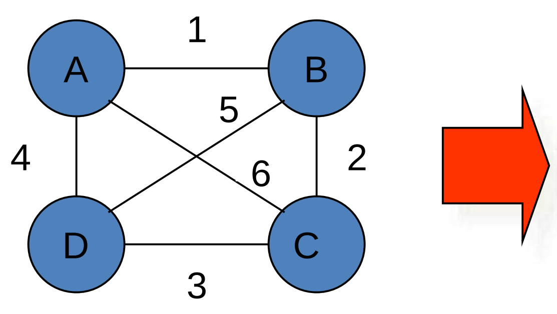


Graph

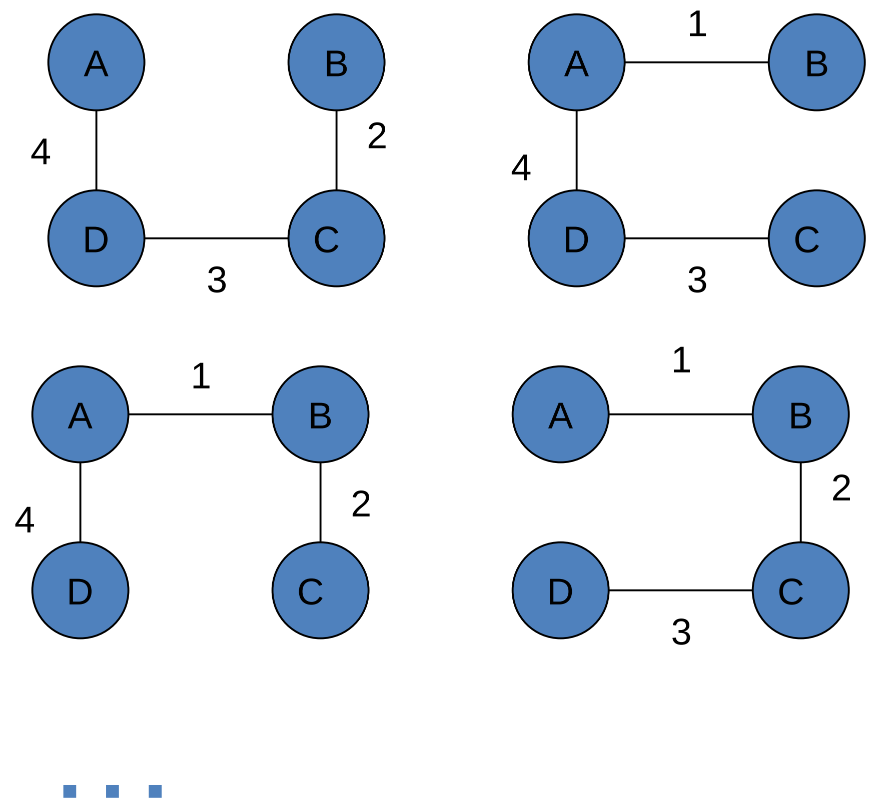


Tree

:



Undirected Graph



Some Spanning Trees

Spanning Tree : A spanning tree of G is a sub graph T that is

➢Connected.

➢ Acyclic.

➢ Includes all of the vertices in given graph and vertices-1 edges.

- $T \subseteq G \qquad T: \{V' = V, E' \subseteq E\}$

  $V' = V \qquad E = |V| - 1$

Spanning Tree properties:

- A graph may have many spanning trees. If the Graph G: { V,E } then
the possible spanning Trees are ECV-1

•Removing one edge from the Spanning Tree Will make it as Disconnected graph

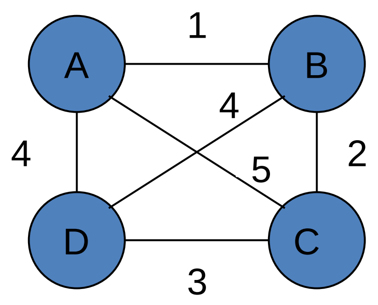
•Adding one edge to the Spanning Tree will create a cycle.

- If each edge has distinct weight then there will be only one & unique
Minimum cost Spanning Tree.

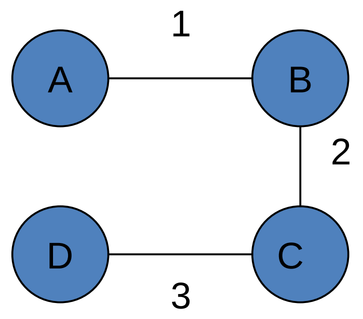
•Disconnected graph doesn't have any Spanning Tree.

Minimum Cost Spanning Tree (MST)

- A minimum spanning tree is the one among all the spanning trees with the smallest total cost or A Spanning tree with minimum weight



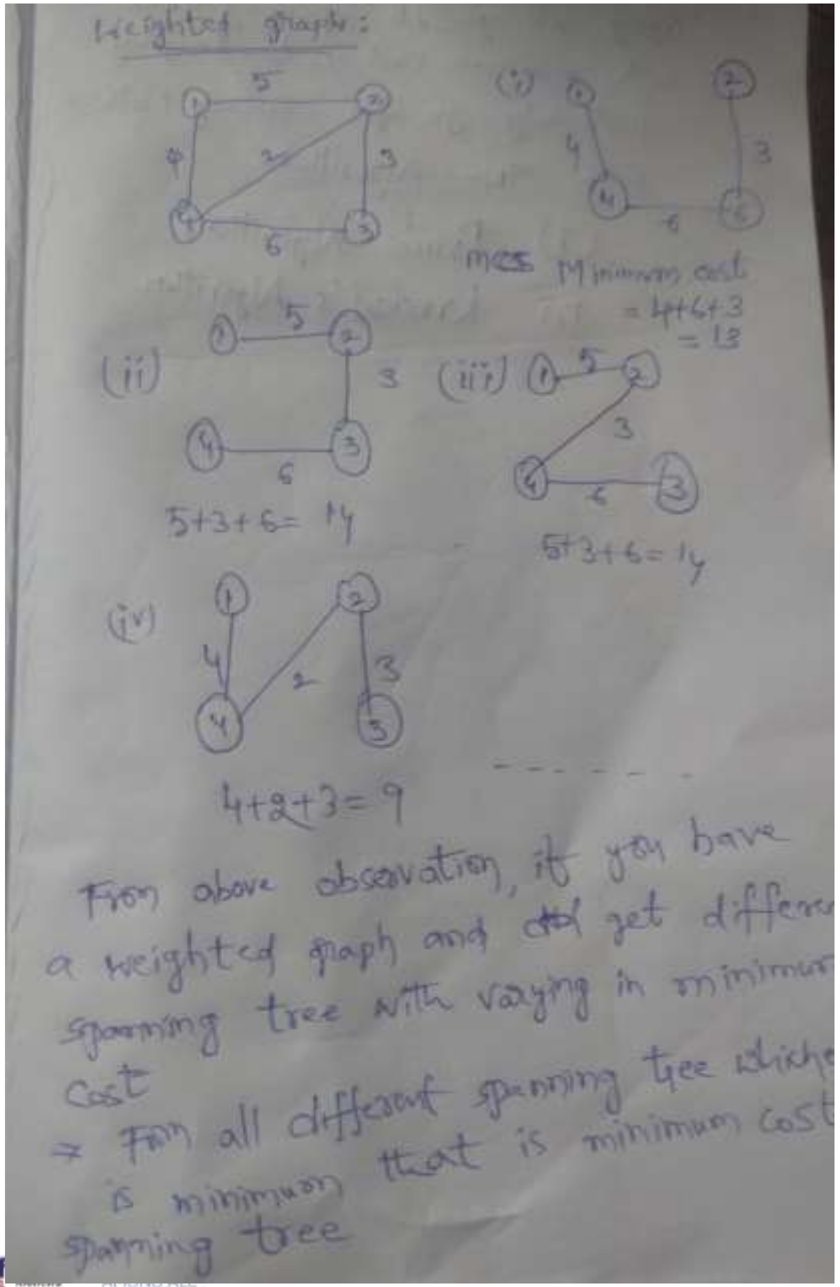
Undirected Graph



Minimum Spanning Tree

Applications of MSTs

- Computer Networks
  - How to connect a set of computers using the minimum amount of wire..
- Civil Network Planning
- Computer Network Routing Protocol
- Cluster Analysis



Weighted graph:

(i)

mcs Minimum cost = 4+6+3 = 13

(ii)

5+3+6 = 14

(iii)

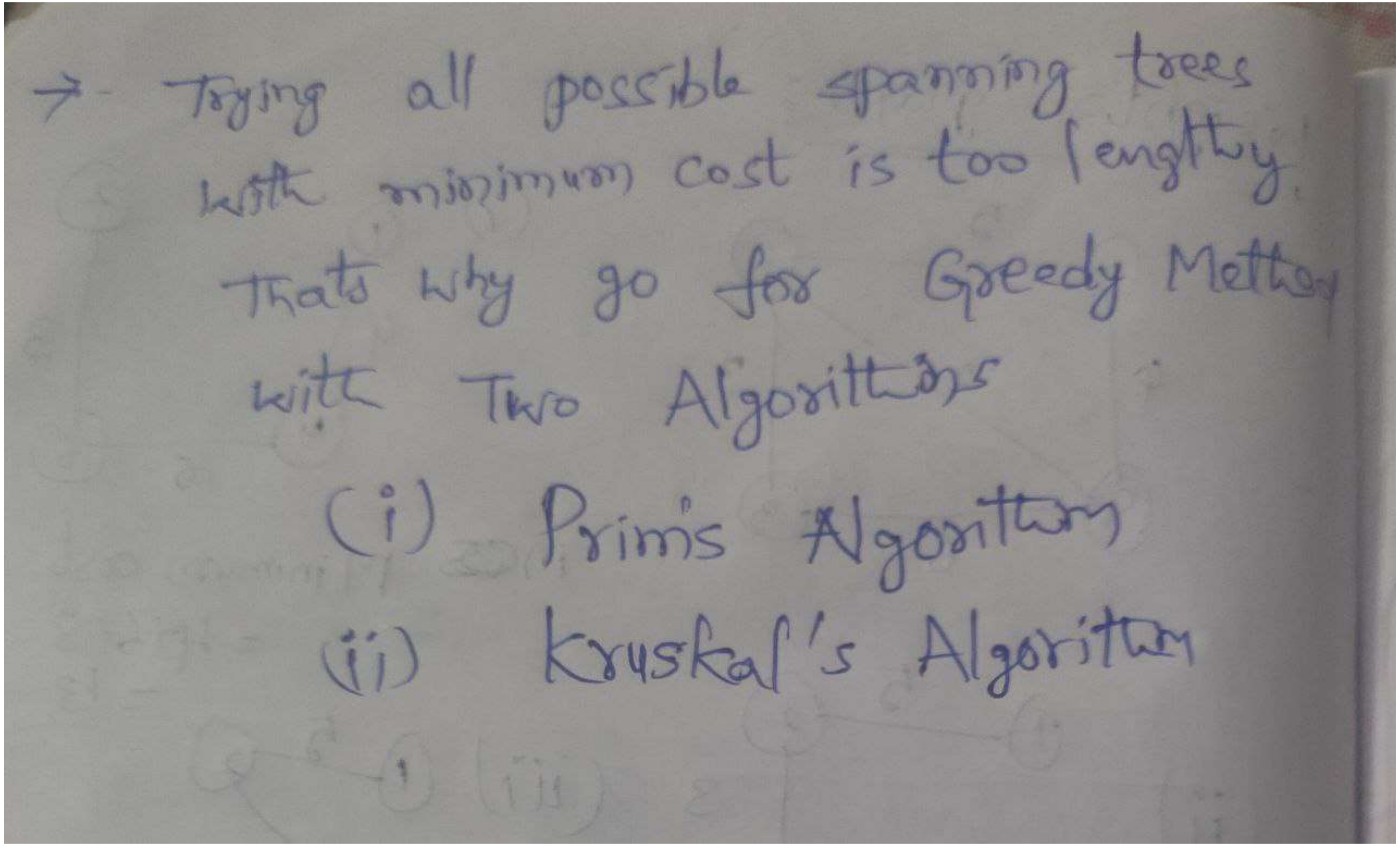
5+3+6 = 14

(iv)

4+2+3 = 9

From above observation, if you have a weighted graph and ~~[illegible]~~ get differe
spanning tree with varying in minimum cost

⇒ From all different spanning tree which is minimum that is minimum cost spanning tree



→ Trying all possible spanning trees with minimum cost is too lengthy.
Thats why go for Greedy Metho
with Two Algorithms

(i) Prim's Algorithm

(ii) Kruskal's Algorithm

Kruskal’s Algorithm

- Kruskal's algorithm is a greedy algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph.
- This algorithm treats every node as an independent tree and connects one with another only if it has the lowest cost compared to all other options available.
- **Steps to Kruskal’s algorithm:**

  ▯ Sort the graph edges with respect to their weights.

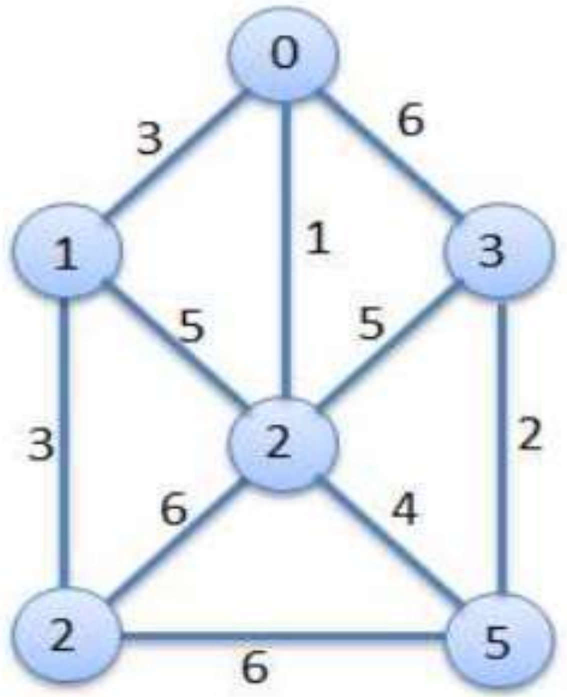
  ▯ Start adding edges to the minimum spanning tree from the edge with the smallest weight until the edge of the largest weight.

  ▯ Only add edges which don’t form a cycle—edges which connect only disconnected components.

Kruskal's Algorithm:

```
MST KRUSKAL(G, w)
T← Ø
cost← 0
for each vertex v V[G]
     do MAKE-SET(v)
sort the edges of E into nondecreasing order by weight w
for each edge (u, v) E, taken in nondecreasing order by weight
        do if FIND-SET(u) ≠ FIND-SET(v)
             then T.E← T.E ∪ {(u, v)}
             UNION(u, v)
             cost ← cost+ cost(u,v)
return cost
```

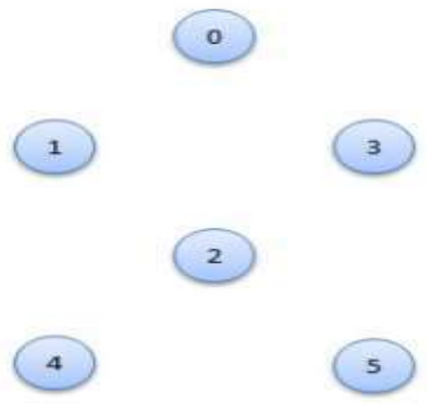
Example: Find Minimum spanning tree cost using kruskal's method for the following Graph



Step1. Edges are sorted in ascending order by weight.

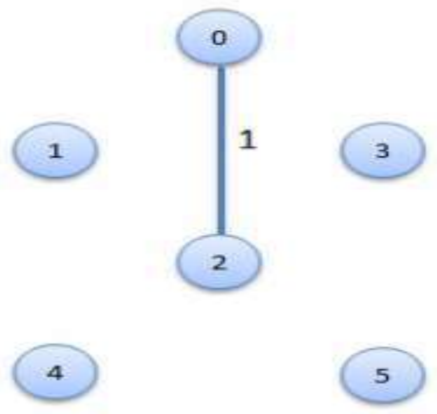
Edge No.	Vertex Pair	Edge Weight
E1	(0,2)	1
E2	(3,5)	2
E3	(0,1)	3
E4	(1,4)	3
E5	(2,5)	4
E6	(1,2)	5
E7	(2,3)	5
E8	(0,3)	6
E9	(2,4)	6
E10	(4,5)	6

Step2. Edges are added in sequence. T.E= { Ø } cost=0

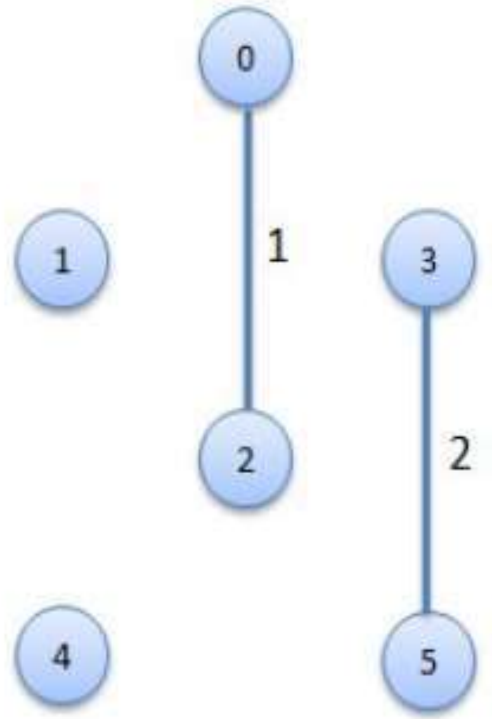


Step3: Consider the minimum edge if it's not connected add the edge and consider the cost

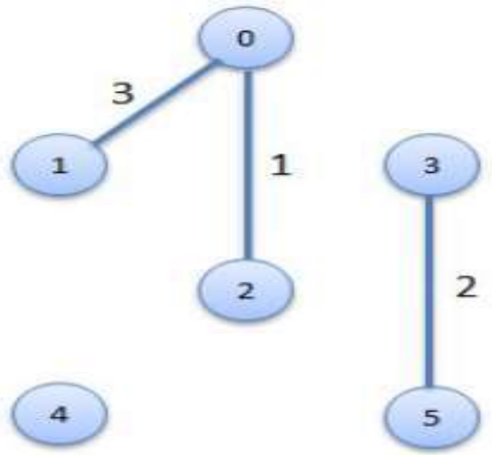
Edge E1 T.E= { (0,2) } cost=1



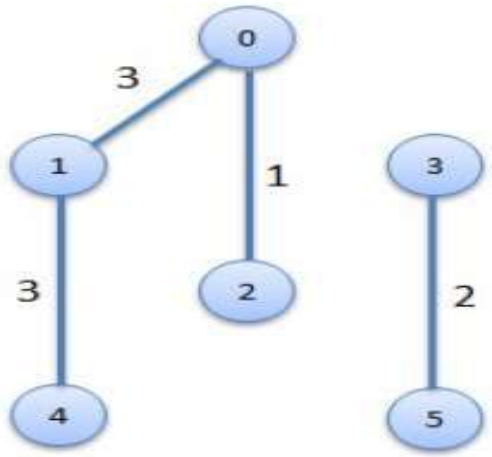
Step4: consider the Edge E2 T.E= { (0,2),(3,5) } cost=1+2



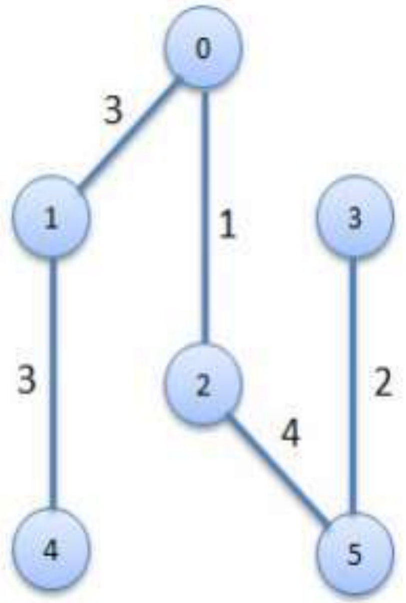
Step5: consider the Edge E3 T.E= { (0,2),(3,5),(1,0) } cost=1+2+3



Step6: consider the Edge E4 T.E= { (0,2),(3,5),(1,0),(1,4) } cost=1+2+3+3



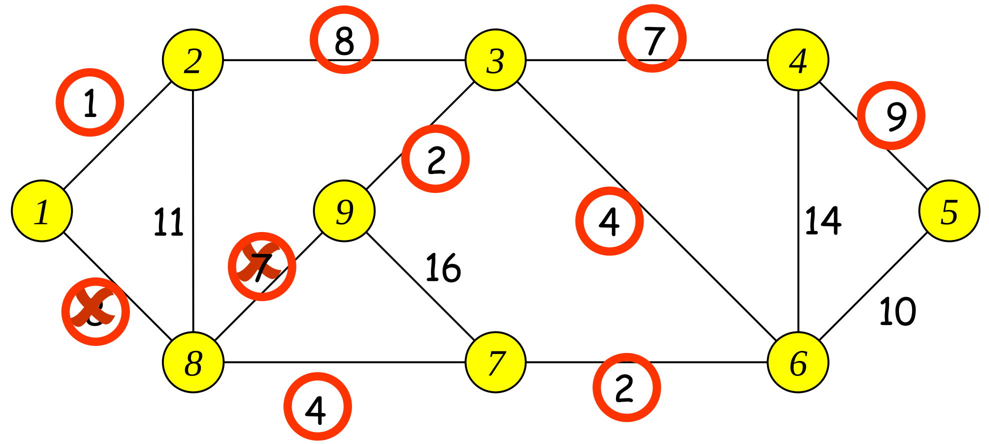
Step7: consider the Edge E5 T.E={ (0,2),(3,5),(1,0),(1,4),(2,5) } cost=1+2+3+3+4

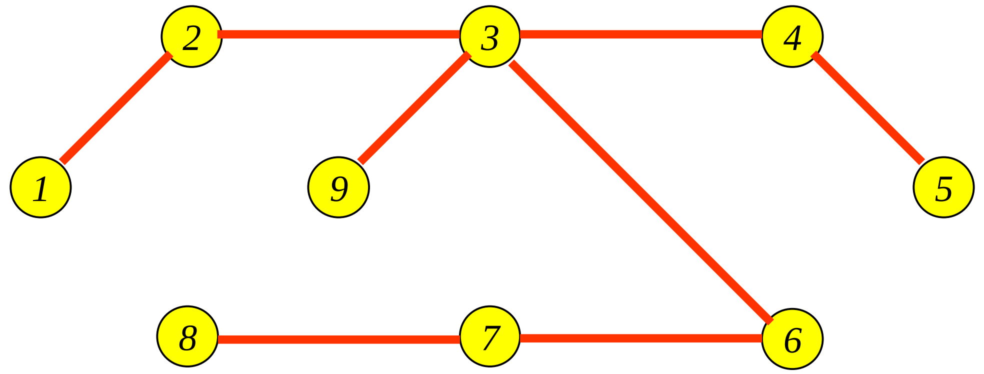


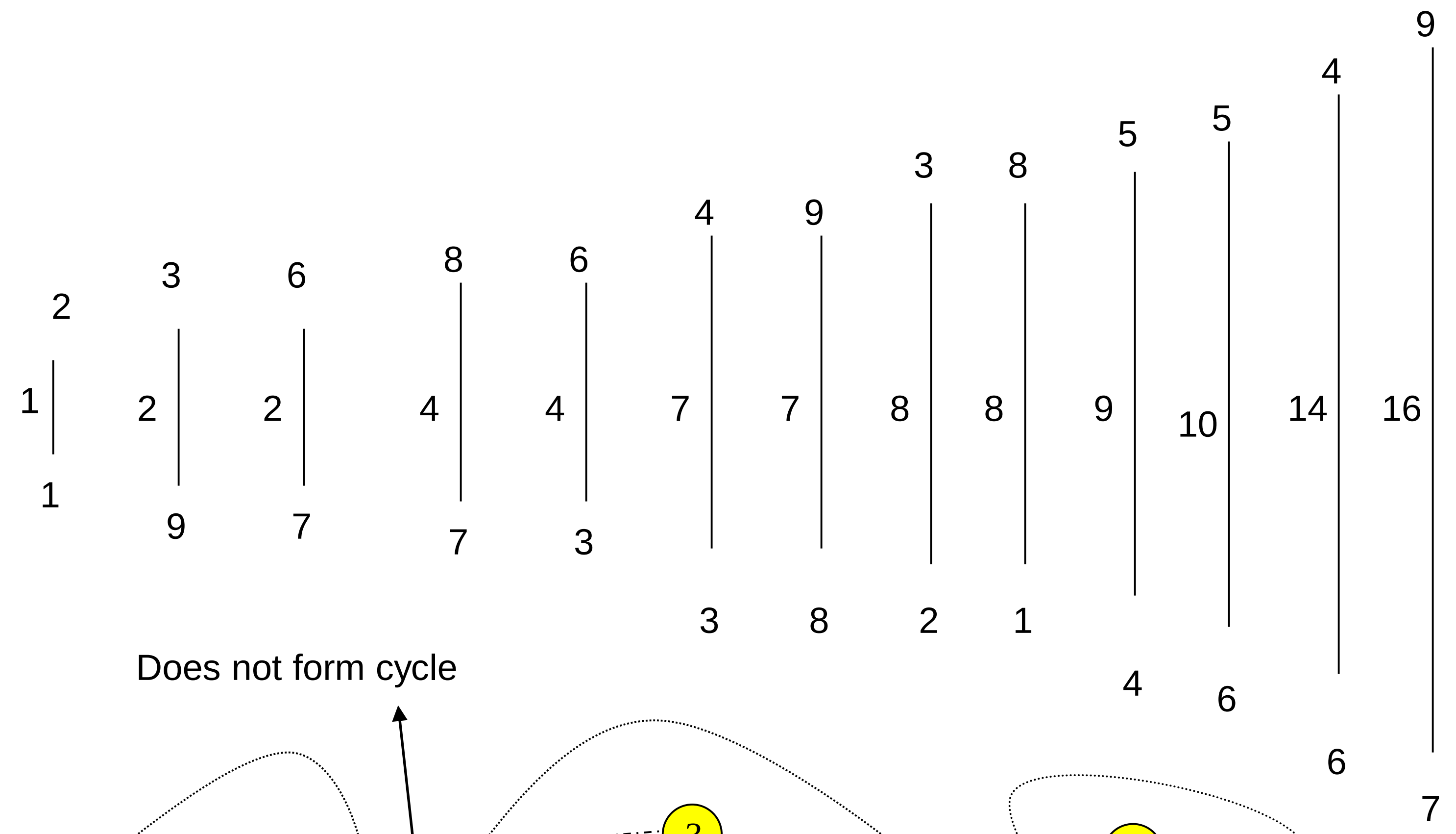
Step 8: check all the remaining edges such that it doesn't form any cycle, for all remaining if we check it will form cycle and its already connected component so the cost of the Tree is:

Total Cost = 1+2+3+3+4 = 13

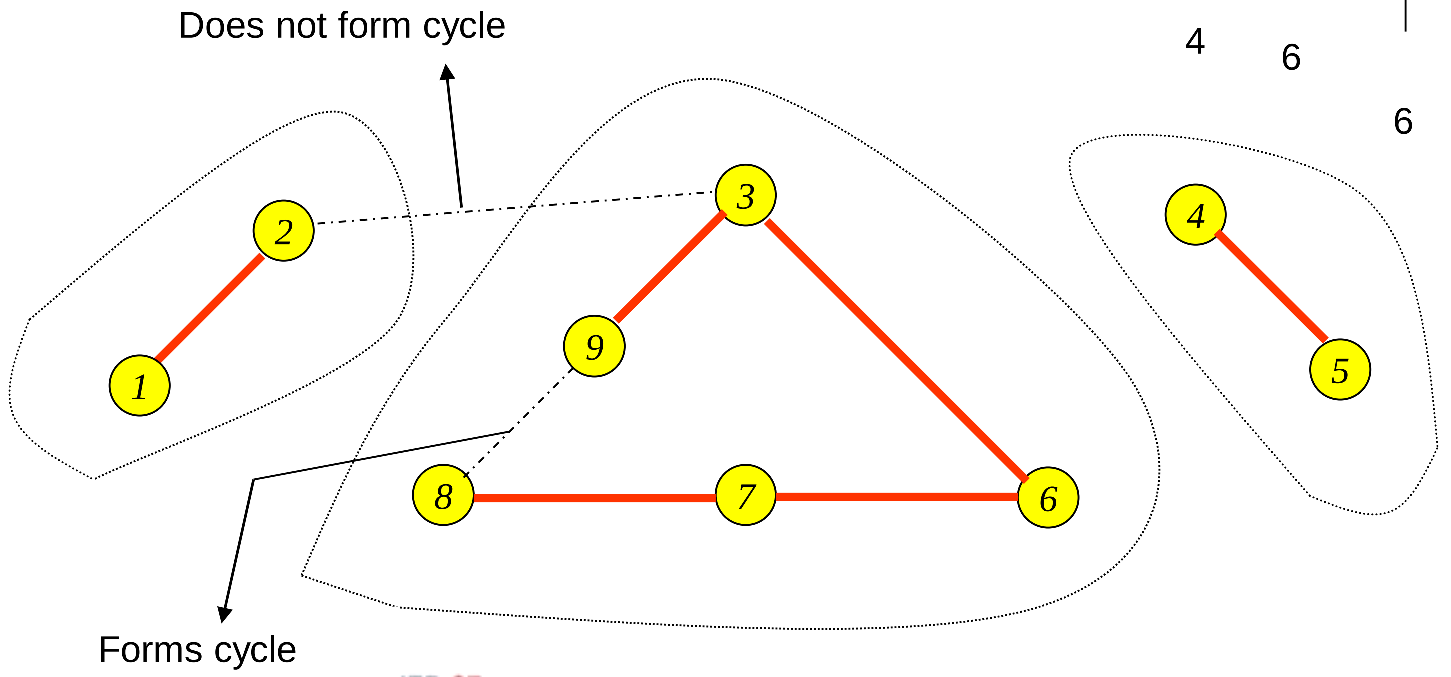
Kruskal's Algorithm







2 1 1
3 2 9
6 2 7
8 4 7
6 4 3
4 7 3
9 7 8
3 8 2
8 8 1
5 9 4
5 10 6
4 14 6
9 16 7



Does not form cycle
Forms cycle
1
2
3
4
5
6
7
8
9

Time complexity of kruskal’s algorithm

- With an efficient Find-set and union algorithms, the running time of kruskal’s algorithm will be dominated by the time needed for sorting the edge costs of a given graph.

- Hence, with an efficient sorting algorithm(merge sort), the complexity of kruskal’s algorithm is

  O(ElogE).

MST-Prim’s Algorithm

- Prim’s Algorithm is another greedy algorithm used for finding the Minimum Spanning Tree (MST) of a given graph.
- The graph must be weighted, connected and undirected
- Start with minimum cost edge.
- For rest of the procedure, always select a minimum cost edge from graph make sure that already connected to the selected vertices.
- Continue this process until the tree has n - 1 edges.

Algorithm for prim's

```
root  ←  {Any arbitrary node in the G.V}
Prim(G={ V,E},root) {
T ← Ø
T.V← T.V U root
Cost ← 0
While (G.V ≠ T.V)   do
{    find an edge e: u→v  of min cost
     such that u∈ T.V and v ∉ T.V
      T← T.E U (e)
      T.V← T.V U (v)
     cost ← cost +cost(u,v)
}
return cost
}
```

Steps for implementing Prim's Algorithm-

Step-01: Remove all loops and parallel edges in graph if there are any in given graph then apply prim's algorithm

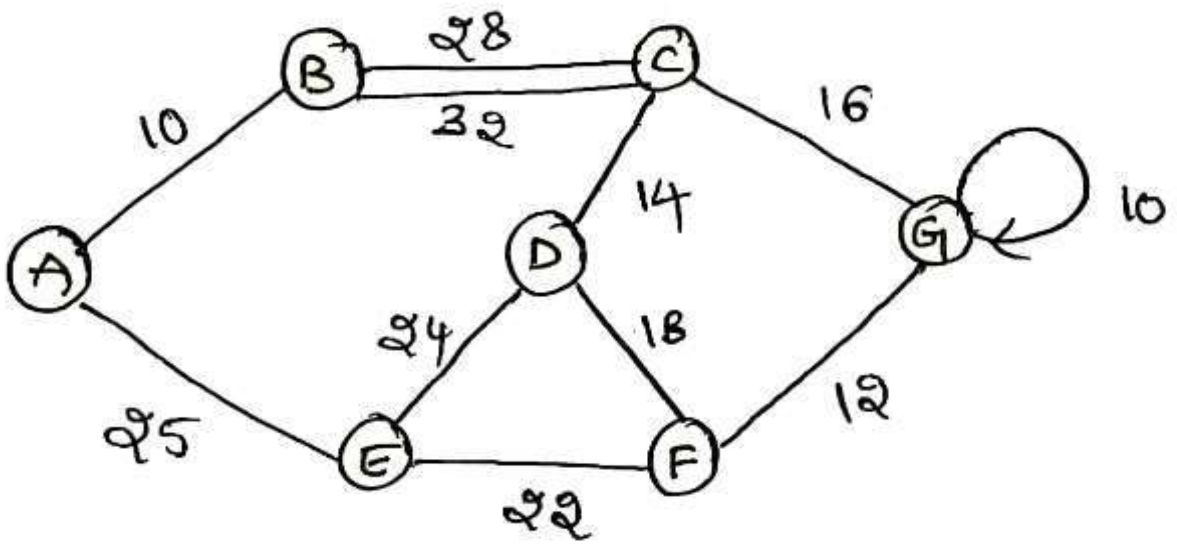
Step-02: Randomly choose any vertex. We usually select and start with a vertex that connects to the edge having least weight.

Step-03: Find all the edges that connect the tree to new vertices, then find the least weight edge among those edges and include it in the existing tree. If including that edge creates a cycle, then reject that edge and look for the next least weight edge.

Step-04: Keep repeating step-03 until all the vertices are included and Minimum Spanning Tree (MST) is obtained.

Question : Construct the minimum spaning tree

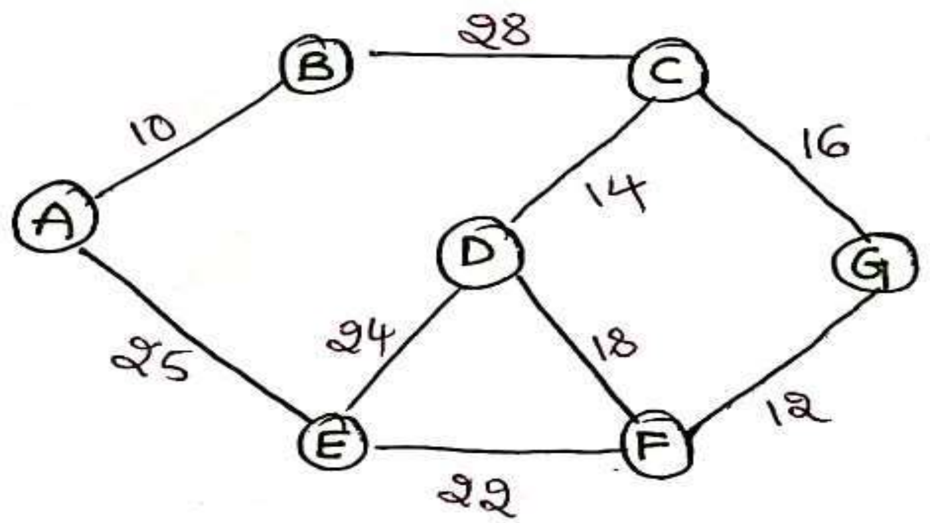
(Question : Construct the minimum spaning tree (MST) for the given graph using prims algorithm?



Solution:

Solution:

Step 1: Remove self loops and parallel edge, if any present in the graph. If any parallel edge is there consider minimum cost edge, redraw the graph.



Step 2 : Select arbitrary node as root here

Step 2 : Select arbitrary node as root here

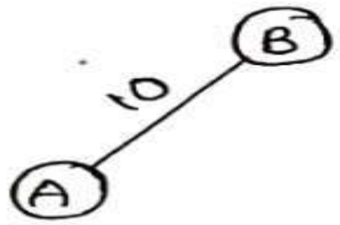
root ← A

T.V = {A} i.e visited node

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

T.E = {(A,B)}

cost = 10



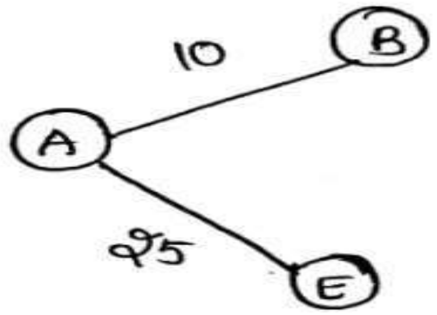
Step 3:

T.V = { A, B, E }

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

T.E = { (A,B), (A,E) }

cost = 10+25



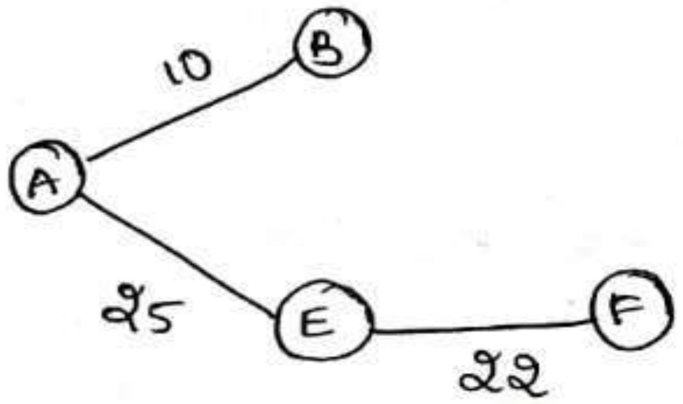
step 4:

T.V = { A, B, E, F }

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

T.E = { (A,B), (A,E), (E,F) }

Cost = 10 + 25 + 22



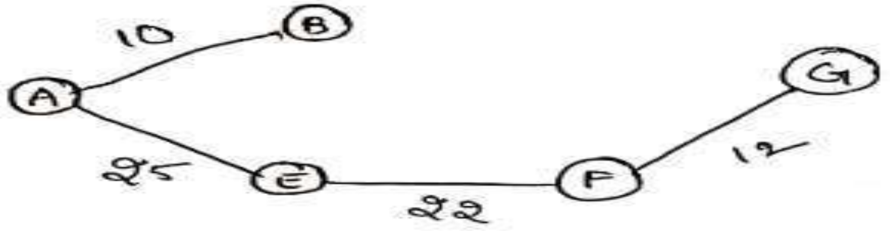
step5: T.V = { A, B, E, F, G }

step5: T.V = { A, B, E, F, G }

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

T.E = { (A,B), (A,E), (E,F), (F,G) }

Cost = 10 + 25 + 22 + 12



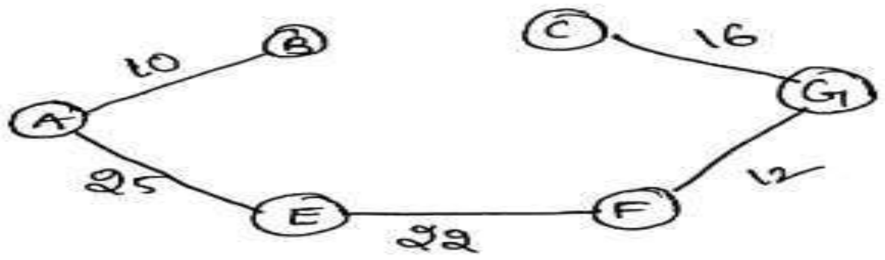
Step 6:

T.V = { A, B, E, F, G, C }

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

T.E = { (A,B), (A,E), (E,F), (F,G), (G,C) }

cost = 10 + 25 + 22 + 12 + 16



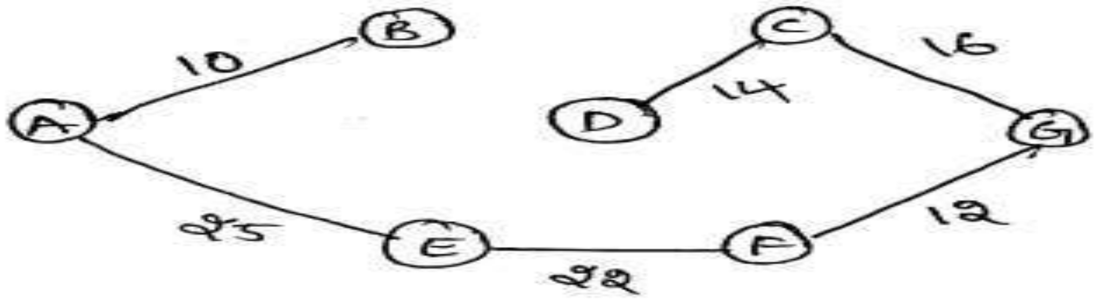
Step 7 :

T·V = { A, B, E, F, G, C }

G·V = { A, B, C, D, E, F, G }

T·E = { (A,B), (A,E), (E,F), (F,G), (G,C), (C,D) }

Cost = 10 + 25 + 22 + 12 + 16 + 14



Step 8 : All vertices covered the final cost is = 10 + 25 + 22 + 12 + 16 + 14

= 99 units

Question-01:

Will both Prim's Algorithm and Kruskal's Algorithm always produce the same Minimum Spanning Tree (MST) for any given graph?

Question-01:

Will both Prim's Algorithm and Kruskal's Algorithm always produce the same Minimum Spanning Tree (MST) for any given graph?

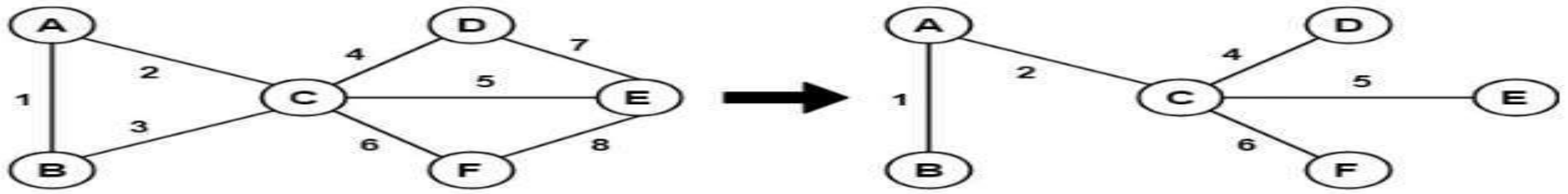
The following two cases are possible-

Case-01: When all the edge weights are distinct-

If all the edge weights are distinct, then both the algorithms are guaranteed to find the same i.e. unique MST.

Example-

Consider the following example-



Minimum Spanning Tree (MST)
(Cost = 18 units)

The application of both the algorithms on the above graph will produce the same MST as shown.

This example clearly illustrates when all the edge weights are distinct, both the algorithms always produces the same MST having the same cost as shown.

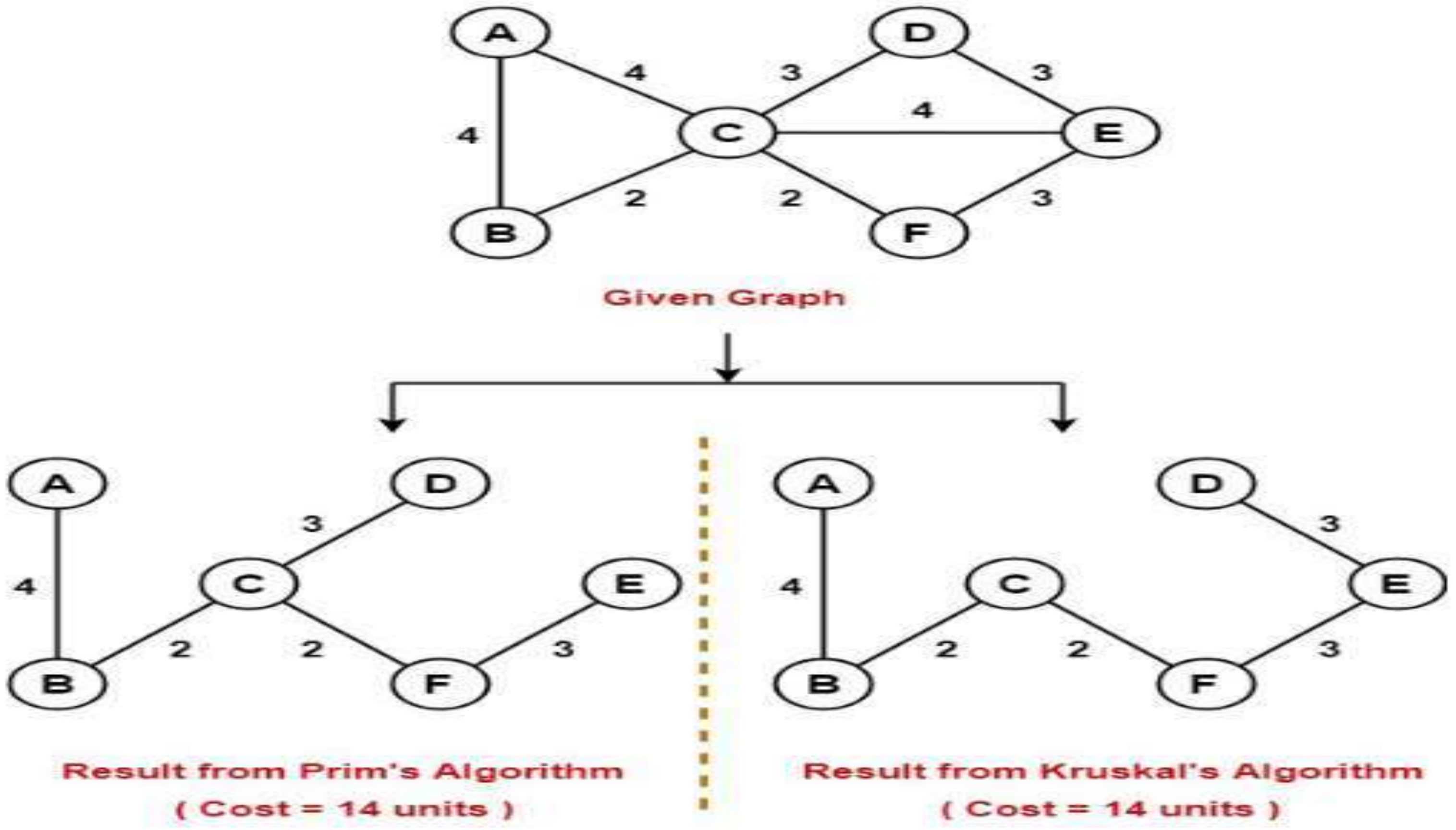
Case-02: When all the edge weights are not distinct-

Case-02: When all the edge weights are not distinct-

If all the edge weights are not distinct, then both the algorithms may not always produce the same i.e. unique MST but the cost of the MST produced would always be same in both the cases.

Example-

Consider the following example-



Differences

Kruskal's Algorithms	Prim's Algorithm
1. Select the shortest edge in a network 2. Select the next shortest edge which does not create a cycle 3. Repeat step 2 until all vertices have been connected 4. Kruskal's Begins with forest and merge into tree. **Complexity** O(NlogN) comparison sort for edges.	1. Select any vertex 2. Select the shortest edge connected to that vertex 3. Select the shortest edge connected to any vertex already connected 4. Repeat step 3 until all vertices have been connected 5. Prim's always stays as a tree. **Complexity** O(NlogN) search the least weight edge for every vertices.

SUMMARY

- A spanning tree is a subset of Graph G, which has all the vertices covered with minimum possible number of edges. Hence, a spanning tree does not have cycles and it cannot be disconnected..
- By this definition, we can draw a conclusion that every connected and undirected Graph G has at least one spanning tree. A disconnected graph does not have any spanning tree, as it cannot be spanned to all its vertices.
- Prim’s and Kruskal's algorithm are used to construct the minimal spanning tree for a given graph topology.

SELF-ASSESSMENT QUESTIONS

Which data structure is commonly used to implement Prim's algorithm?

(a) Array

(b) Linked List

(c) Heap

(d) Stack

In Kruskal's algorithm, edges are added to the spanning tree in which order?

(a) Decreasing order of weights.

(b) Random order.

(c) Non-decreasing order of weights

(d) Alphabetical order.

TERMINAL QUESTIONS

1. Compare and contrast Prim's algorithm and Kruskal's algorithm for finding the MST of a weighted graph.

1. Explain the concept of cycle detection in the context of MST algorithms. Why is it important to avoid cycles when constructing a minimum spanning tree, and how do Prim's and Kruskal's algorithms ensure acyclic structures?

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books :

1. Introduction to Algorithms, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein., 3rd, 2009, The MIT Press.

2 Algorithm Design Manual, Steven S. Skiena., 2nd, 2008, Springer.

3 Data Structures and Algorithms in Python, Michael T. Goodrich, Roberto Tamassia, and Michael H. Goldwasser., 2nd, 2013, Wiley.

4 The Art of Computer Programming, Donald E. Knuth, 3rd, 1997, Addison-Wesley Professiona.

MOOCS :

1. https://www.coursera.org/specializations/algorithms?=

2.https://www.coursera.org/learn/dynamic-programming-greedy-algorithms#modules

THANK YOU