# UNIVERSITY INSTITUTE OF COMPUTING

## MASTER OF COMPUTER APPLICATIONS

### DESIGN AND ANALYSIS OF ALGORITHMS

### 24CAT-611

UNIT-2

DISCOVER . LEARN . EMPOWER

# DESIGN AND ANALYSIS OF ALGORITHMS

## Course Outcome

| CO Number | Title | Level |
|---|---|---|
| CO3 | Apply and analyze important algorithmic design paradigms and their applications | Understand |
| CO4 | Implement the major graph algorithms to model engineering problems | Understand |

- **Divide and Conquer**: General method, Binary search, Advantages and disadvantages of divide and conquer, Decrease and conquer approach: Topological sort

# Topics to be covered

- Minimum Spanning Trees
- Prim's Algorithm
- Kruskal's Algorithm

# Spanning Tree

- 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.
- A complete undirected graph can have maximum $\mathbf{n^{n-2}}$ number of spanning trees, where **n** is the number of nodes. In the above addressed example, **n is 3,** hence $3^{3-2}$ **= 3** spanning trees are possible.

# Spanning Tree

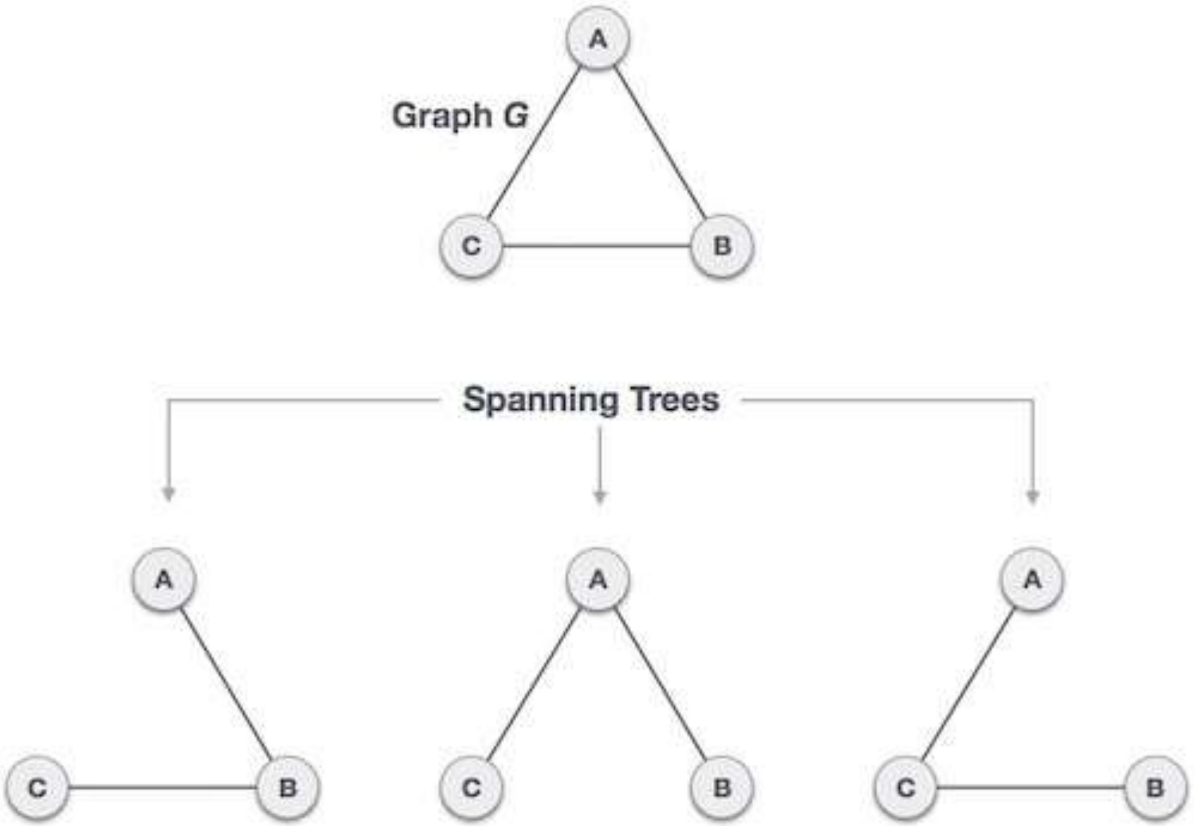



Fig 1: Spanning Trees

# Properties of Spanning Tree

- We now understand that one graph can have more than one spanning tree. Following are a few properties of the spanning tree connected to graph G −
- 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**.

# Minimum Spanning Tree

In a weighted graph, a minimum spanning tree is a spanning tree that has minimum weight than all other spanning trees of the same graph. In real-world situations, this weight can be measured as distance, congestion, traffic load or any arbitrary value denoted to the edges.

# Minimum Spanning Tree

The cost of the spanning tree is the sum of the weights of all the edges in the tree. There can be many spanning trees. Minimum spanning tree is the spanning tree where the cost is minimum among all the spanning trees. There also can be many minimum spanning trees.

# Minimum Spanning Tree

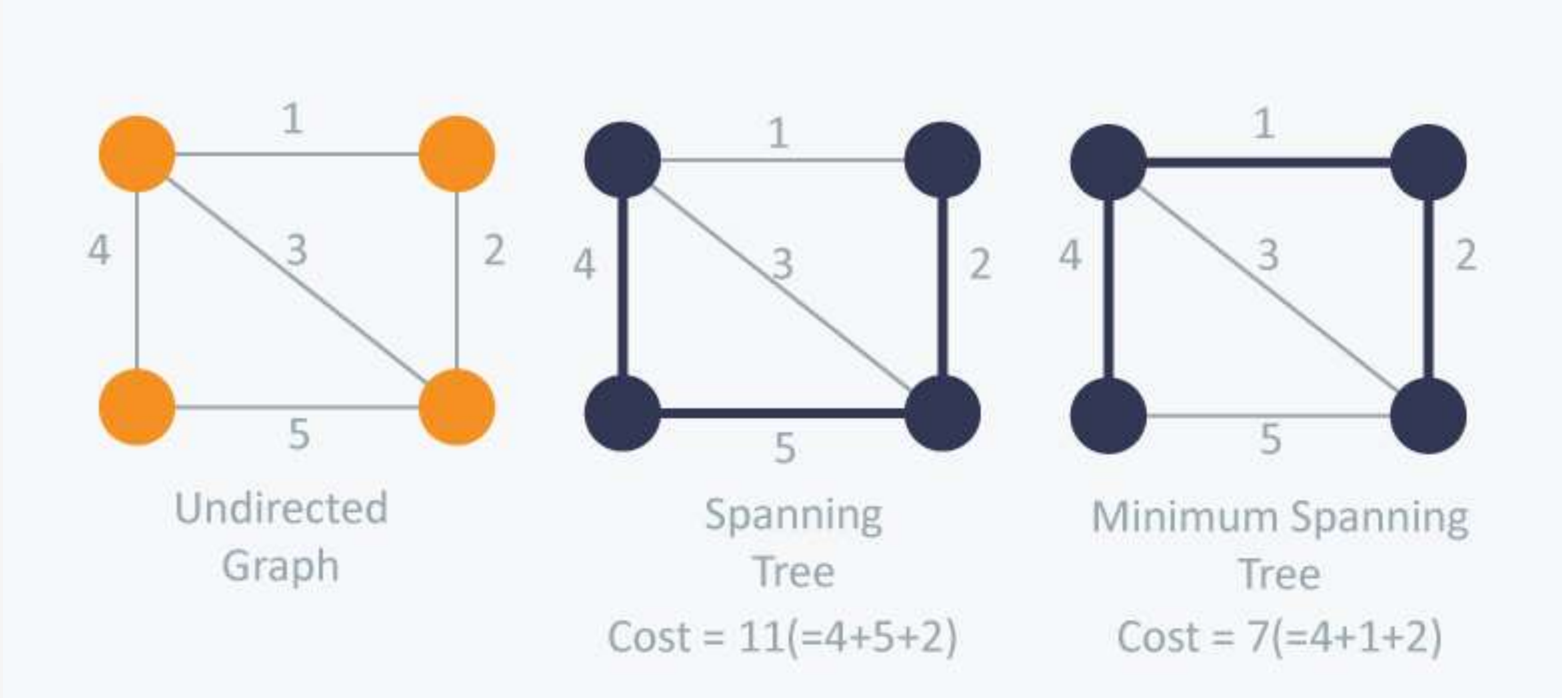



Fig2: Minimum spanning Tree

# Prim’s Minimum Spanning Tree

- ***How does Prim’s Algorithm Work?*** The idea behind Prim’s algorithm is simple, a spanning tree means all vertices must be connected. So the two disjoint subsets (discussed above) of vertices must be connected to make a *Spanning* Tree. And they must be connected with the minimum weight edge to make it a *Minimum* Spanning Tree.

# Prim's Algorithm

- ***Algorithm***

  **1)** Create a set *mstSet* that keeps track of vertices already included in MST.
  **2)** Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.
  **3)** While mstSet doesn't include all vertices
  ….**a)** Pick a vertex *u* which is not there in *mstSet* and has minimum key value.
  ….**b)** Include *u* to mstSet.
  ….**c)** Update key value of all adjacent vertices of *u*. To update the key values, iterate through all adjacent vertices. For every adjacent vertex *v*, if weight of edge *u*-*v* is less than the previous key value of *v*, update the key value as weight of *u*-*v*

# Prim's Method

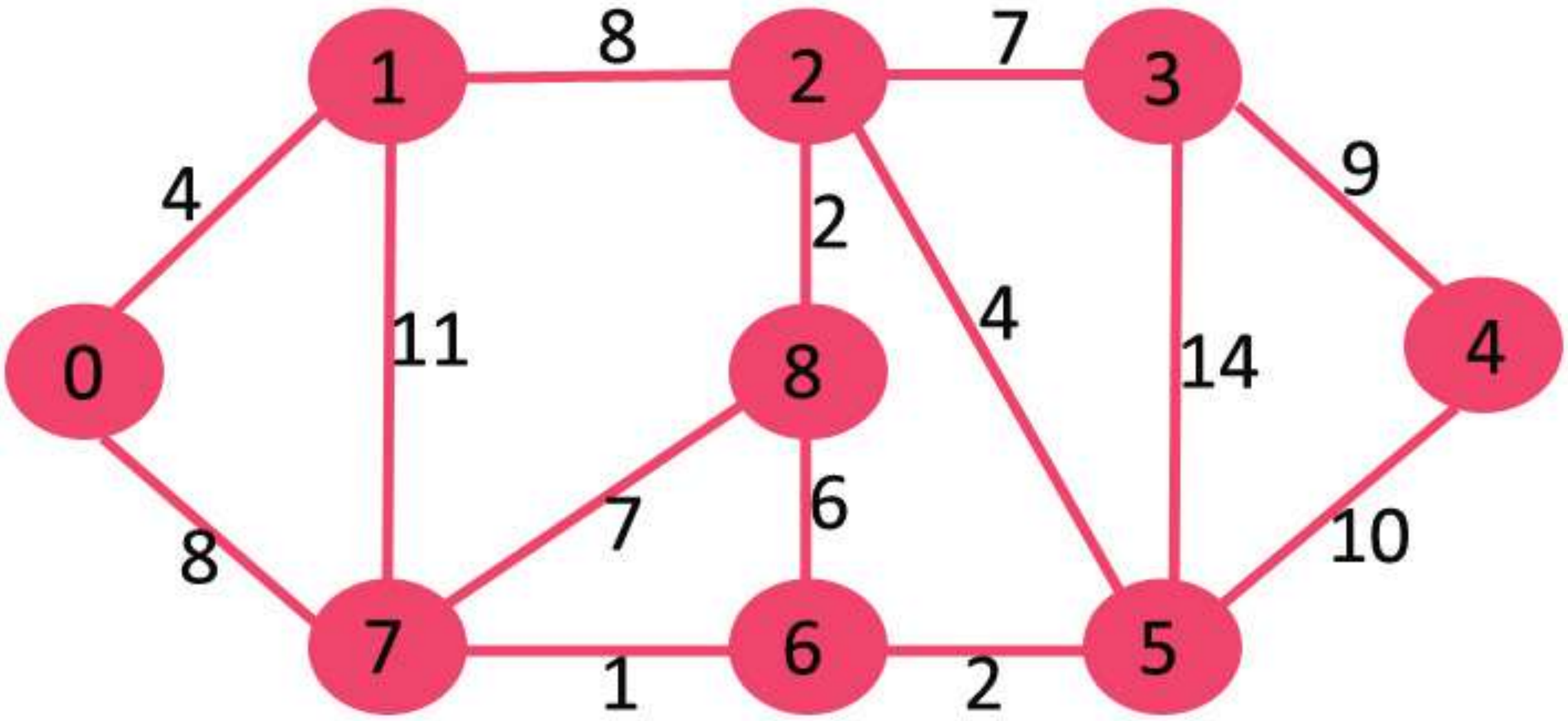



Fig 3: Graph

# Contd…

- The vertices included in MST are shown in green color.

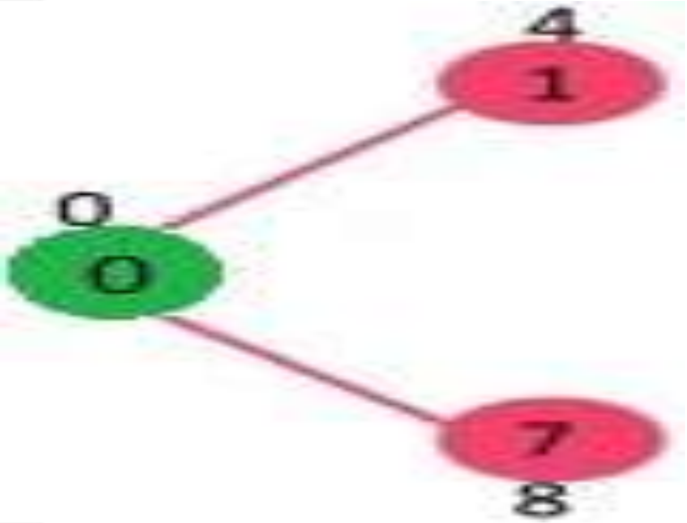



Fig 4: Step 1

# Contd…

Pick the vertex with minimum key value and not already included in MST (not in mstSET). The vertex 1 is picked and added to mstSet. So mstSet now becomes {0, 1}. Update the key values of adjacent vertices of 1. The key value of vertex 2 becomes 8.

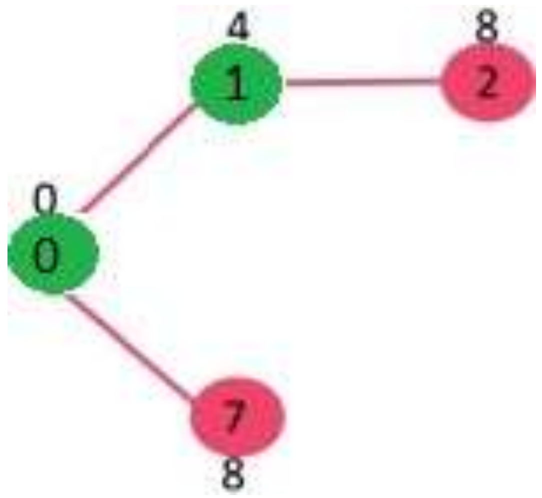
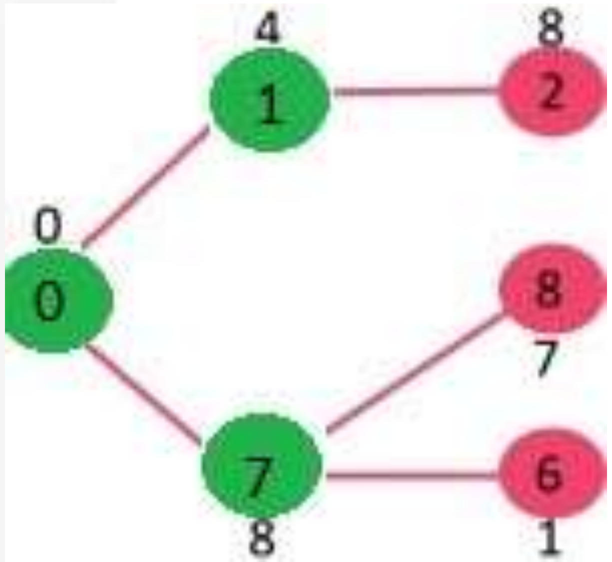



Fig 5: Step 2 of Prim’s method

# Minimum Spanning Tree



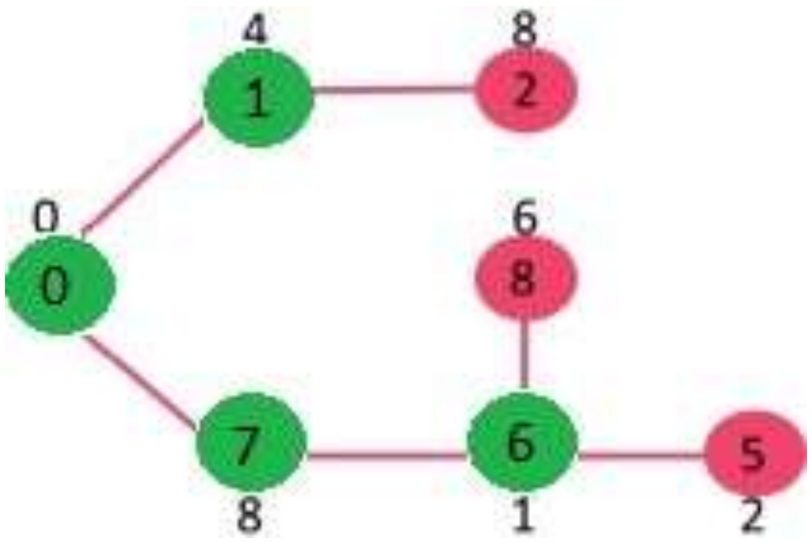




Fig 6: Step 3 of Prim’s method

# Minimum Spanning Tree

We repeat the above steps until *mstSet* includes all vertices of given graph. Finally, we get the following graph.

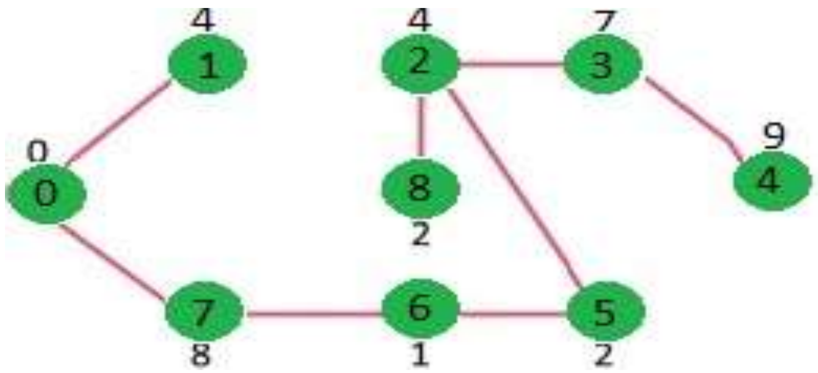



Fig 7: Minimum Spanning Tree

# Frequently asked questions

- Compare Greedy approach and Divide & conquer algorithms with the suitable examples along with their respective complexities

# References


[1] https://www.tutorialspoint.com/data_structures_algorithms/images/spanning_trees.jpg

[2] https://www.tutorialspoint.com/data_structures_algorithms/images/spanning_trees.jpg

[3] https://www.geeksforgeeks.org/wp-content/uploads/Fig-11.jpg

[4] https://www.geeksforgeeks.org/wp-content/uploads/MST1.jpg

[5] https://www.geeksforgeeks.org/wp-content/uploads/MST1.jpg

[6] https://www.geeksforgeeks.org/wp-content/uploads/MST1.jpg

[7] https://www.geeksforgeeks.org/wp-content/uploads/MST1.jpg

Books:

1. Introduction to Algorithms by Coreman, Leiserson, Rivest, Stein.
2. Fundamentals of Algorithms by Ellis Horwitz, Sartaj Sahni, Sanguthevar Rajasekaran

# THANK YOU

**For any Queries-**
**E-Mail:** uttamae7426@cumail.in