Lesson 2.2

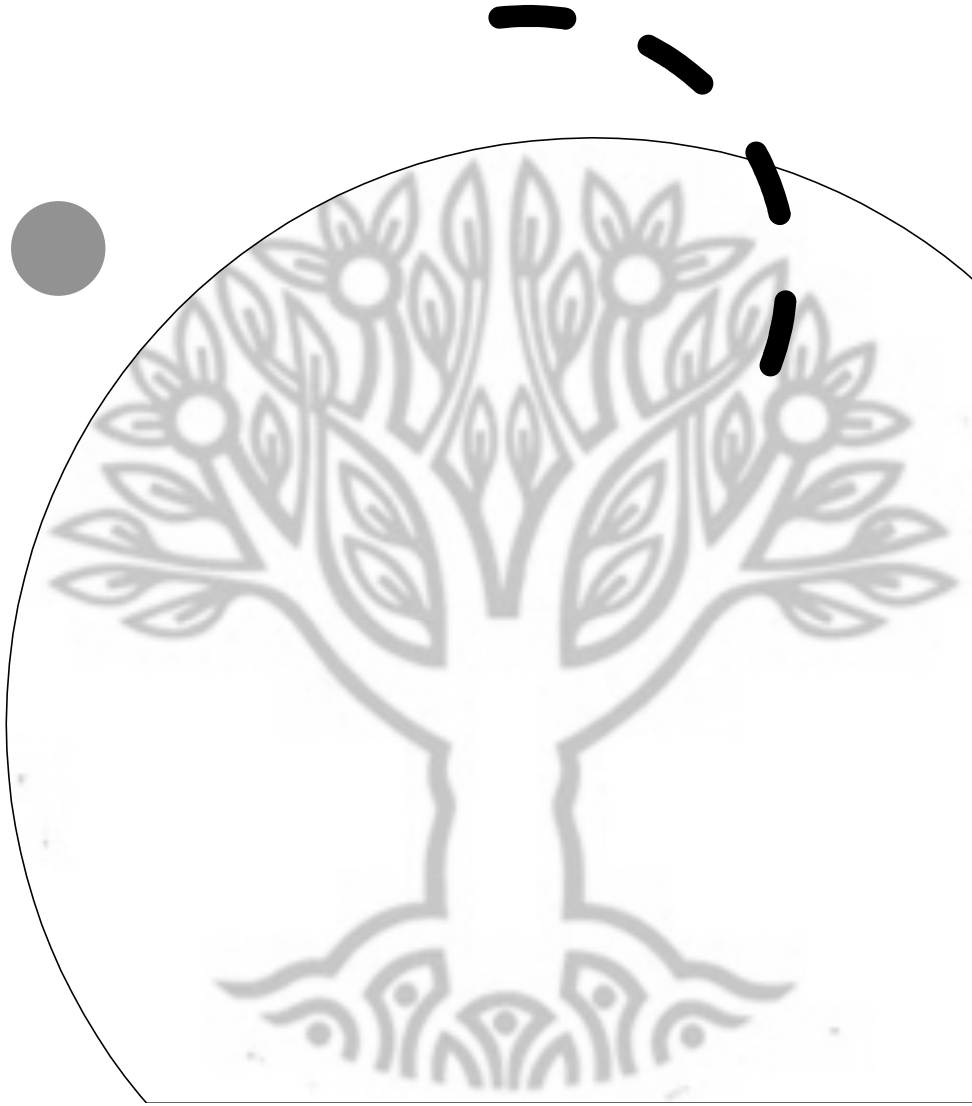
SPANNING TREES
Life is Found in Layers

Wholeness of the lecture: A *spanning tree* is a tree contained in a graph that includes every vertex, but not necessarily every edge, of the graph. Spanning trees give a simpler way to handle all the vertices of a graph. *Science of Consciousness:* Establishing pure consciousness in our awareness is the simplest way to handle all possibilities in life.

Main Points

1. Every connected graph contains a spanning tree that includes every vertex, but not necessarily every edge, of the graph. *Science of Consciousness:* We can locate pure consciousness, the simplest level of life, everywhere.
2. The breadth-first search algorithm can find a spanning tree for an ordinary graph and Prim's algorithm (a greedy algorithm) can find a minimal or maximal spanning tree in a weighted graph. *Science of Consciousness:* The Transcendental Meditation technique is an effortless technique that we use to contact pure consciousness, the supreme level of life.

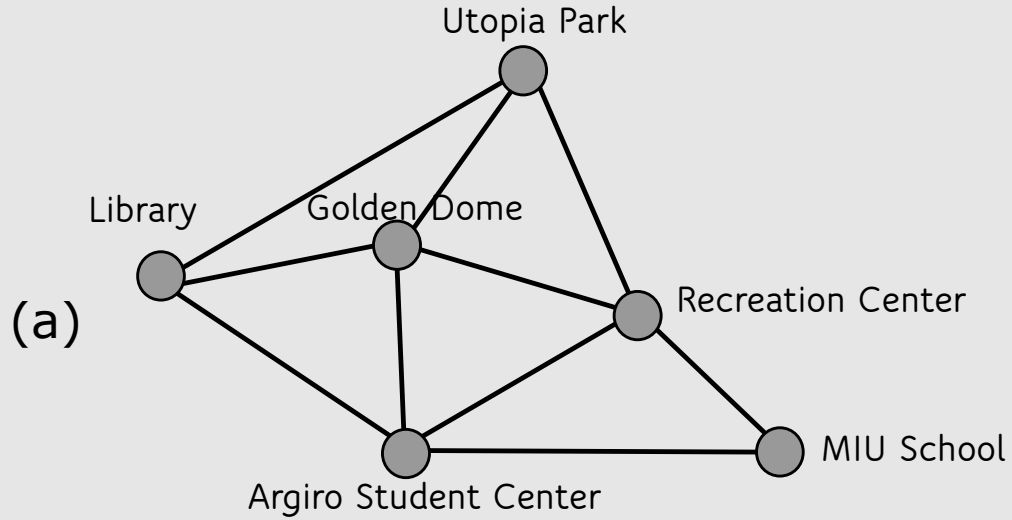
Spanning Trees



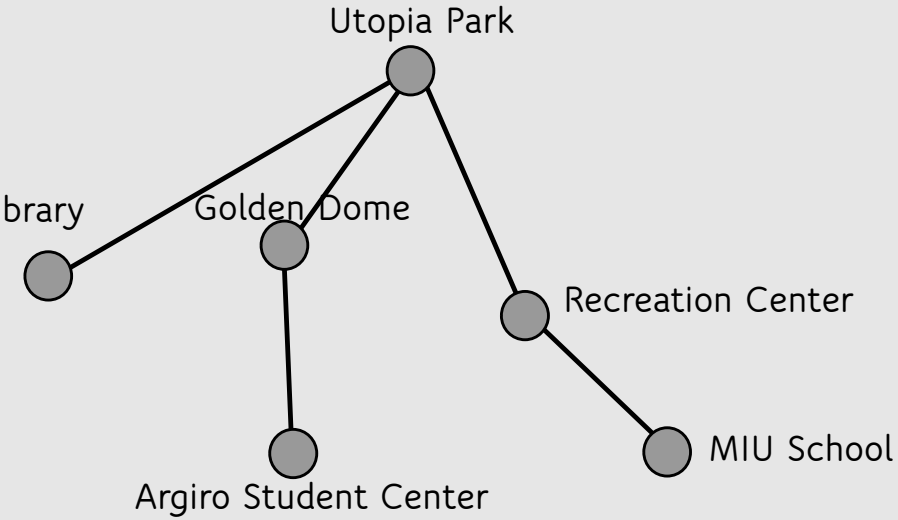
Spanning Trees

1. **Introduction:**

- The connected graph in Fig. (a) represents a road network at MIU Campus. It's important to frequently plow the road during snowfall to keep the road clean. The concerned department wishes to keep only the fewest roads clean while still connecting all the major buildings.
  What are the possible ways?
- We can use the idea of a connected subgraph with the fewest possible edges, containing every vertex in the original graph, to solve this problem. The resulting graph must be a tree. For the main buildings to be connected by cleared roads, this method requires that at least five roads be plowed, as shown in Fig. (b).

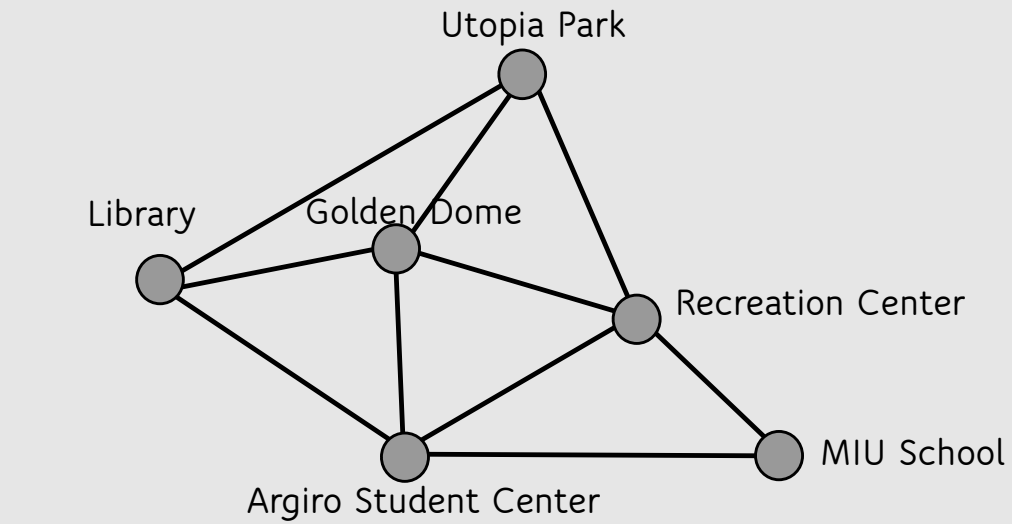


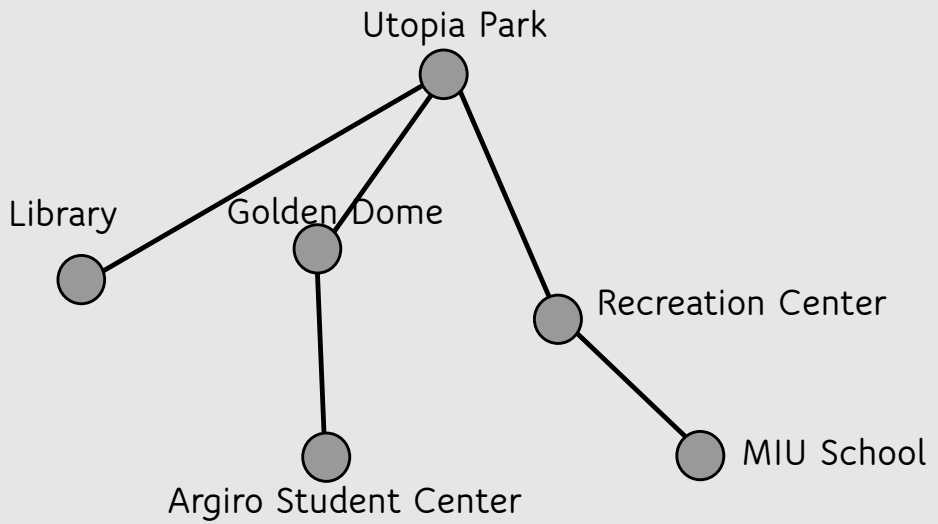
(a)

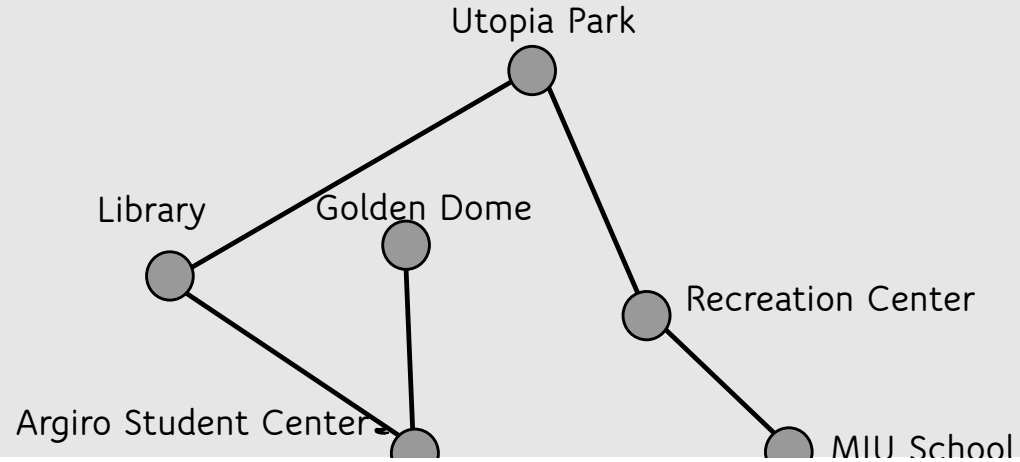


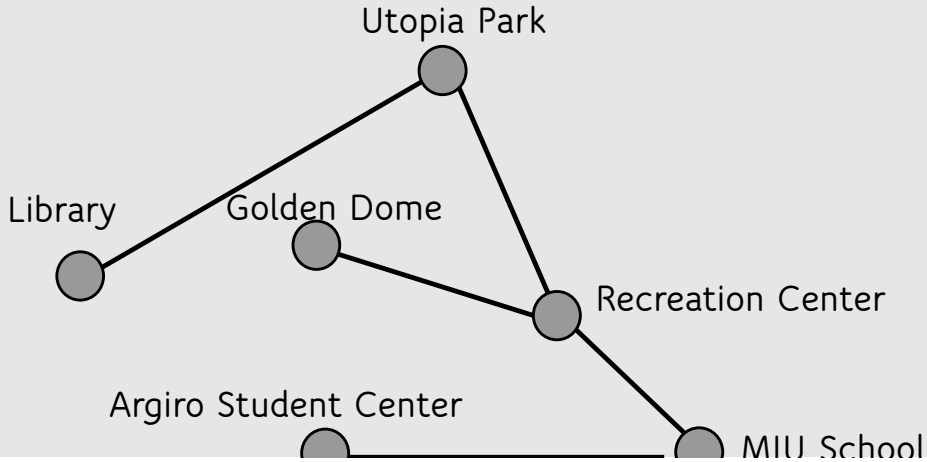
(b)

Spanning Trees









Spanning Trees

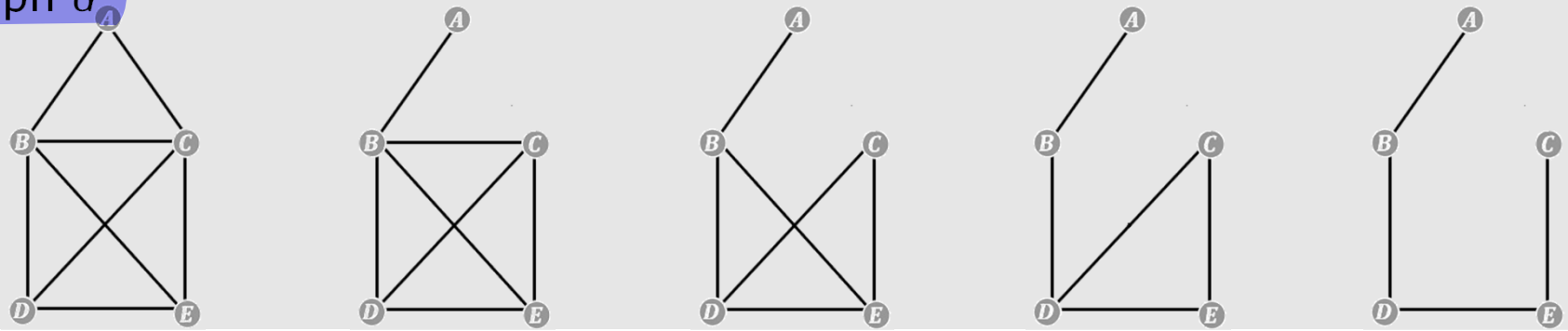
1. Spanning Tree:

- A **spanning tree** of a simple graph G, is a subgraph of G containing every vertex of G, which is a tree is called a spanning tree of G…
  - **i. How to find a spanning Tree?**
    - If G is already a tree, need not to further remove edges.
    - If G is not a tree, it must have cycle, remove an edge from the cycle, this process can be repeated until a tree can be achieved containing all the vertices of the original graph.
  - **ii. How many edges are removed?**
    - If a connected G has v vertices and e edges with $e \geq v$, removing $e - v + 1$ edges will ensure the resulting graph will be a tree with $e - (e - v + 1) = v - 1$ edges and v vertices.
- Given a graph G, a spanning tree is a subgraph of G which (i) is a tree, and (ii) has all the vertices in G.

Spanning Trees

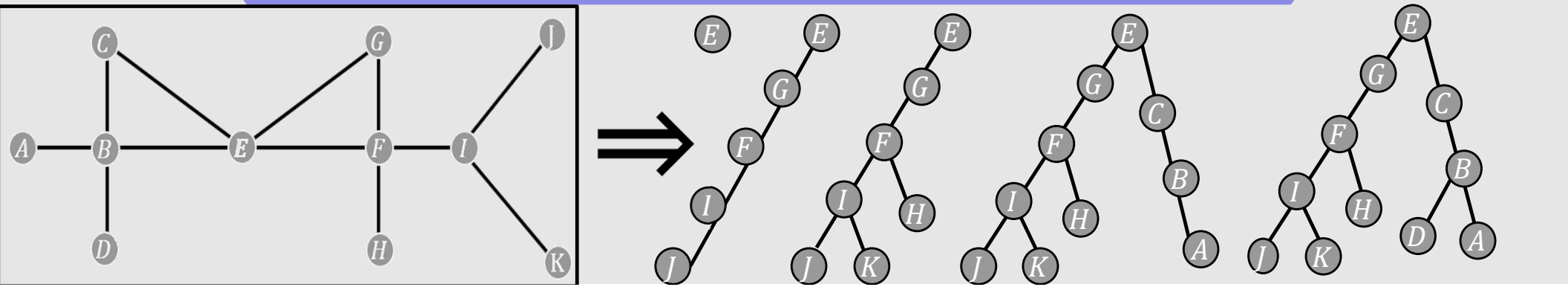
1. **Example:**

- Find a spanning tree of a graph G?.
  i. If a graph has 5 vertices, then the possible number of spanning trees are as large as 5^3.
  ii. Consider a connected graph as shown in Figure, but it is not a tree because it contains simple circuits.
  iii. Removing edges in a sequence produce different spanning trees as shown below.
  iv. Remove the edge $\{A, C\}$, this eliminates one circuit from the graph, but there are still other circuits.
  v. Now, remove the edges $\{B, C\}$, $\{B, E\}$ and $\{D, C\}$ to remove all the circuits to get a simple graph with no simple circuits.
  vi. The final subgraph is a spanning tree and containing all the vertices of the graph G.



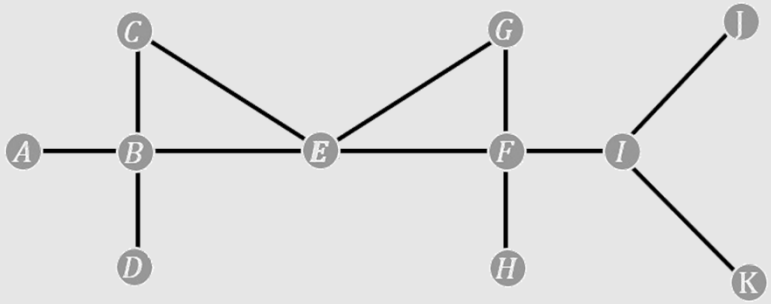
Spanning Tree

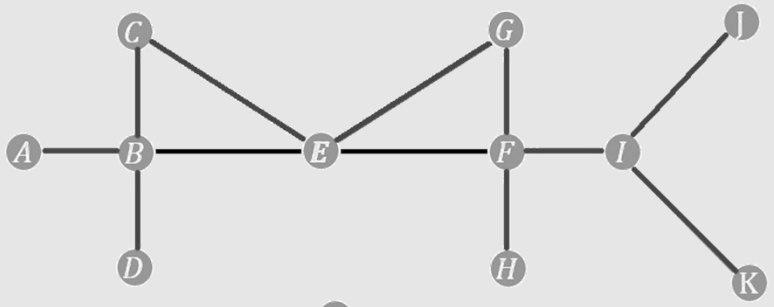
1. **Depth-First Search:**
   - Depth-First Search is a technique which is applied to build a spanning tree for a simple connected graph G.
     i. **Example:**
        - **Firstly, we can choose any arbitrary vertex to start with**.
        - Form a path starting from the selected vertex as a root and then start adding vertices and edges in such a way that new edge is incident with the last vertex in the path and a vertex not already in the path.
        - **We don't want to repeat visited roots**. Continue adding vertices and edges to path, however if path goes through all the vertices, the tree containing this path is a spanning tree.
        - In a similar way, beginning at the last vertex, moving back up the path one vertex at a time, forming new paths that are if possible until no more edges can be added so this process ends with the formation of a spanning tree.

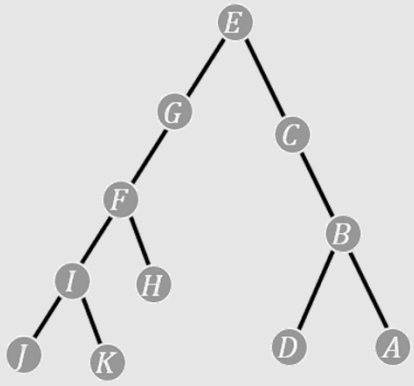


Spanning Trees

1. The edges selected by DFS of a graph are called tree edges. Remaining edges must connect a vertex to an ancestor or descendant of this vertex in the tree. These edges are known as back edges.





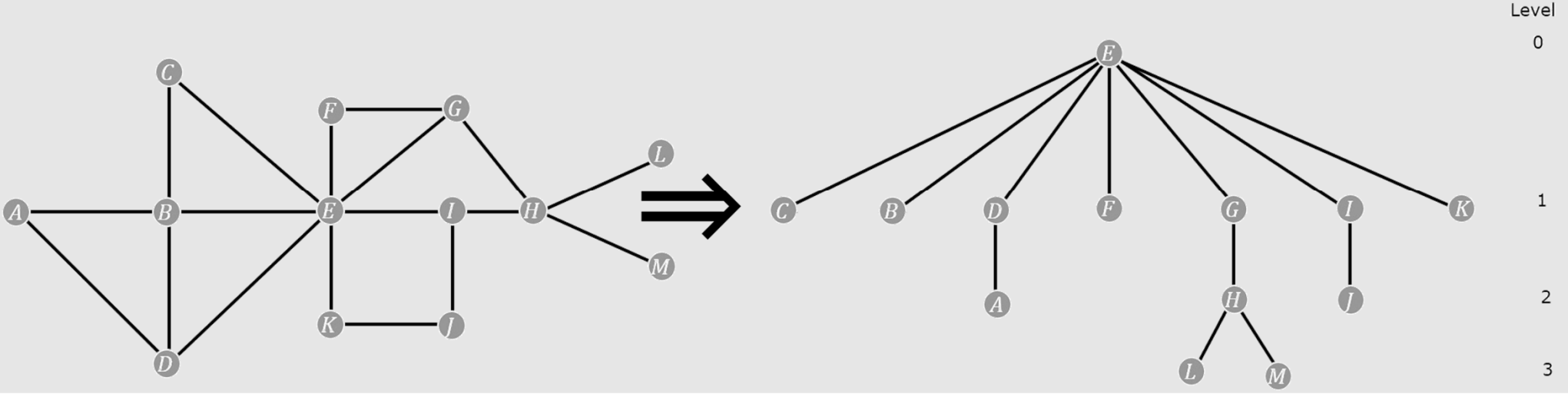


- The tree edges are shown with pink colored lines
- The back edges are shown with black colored lines

Spanning Trees

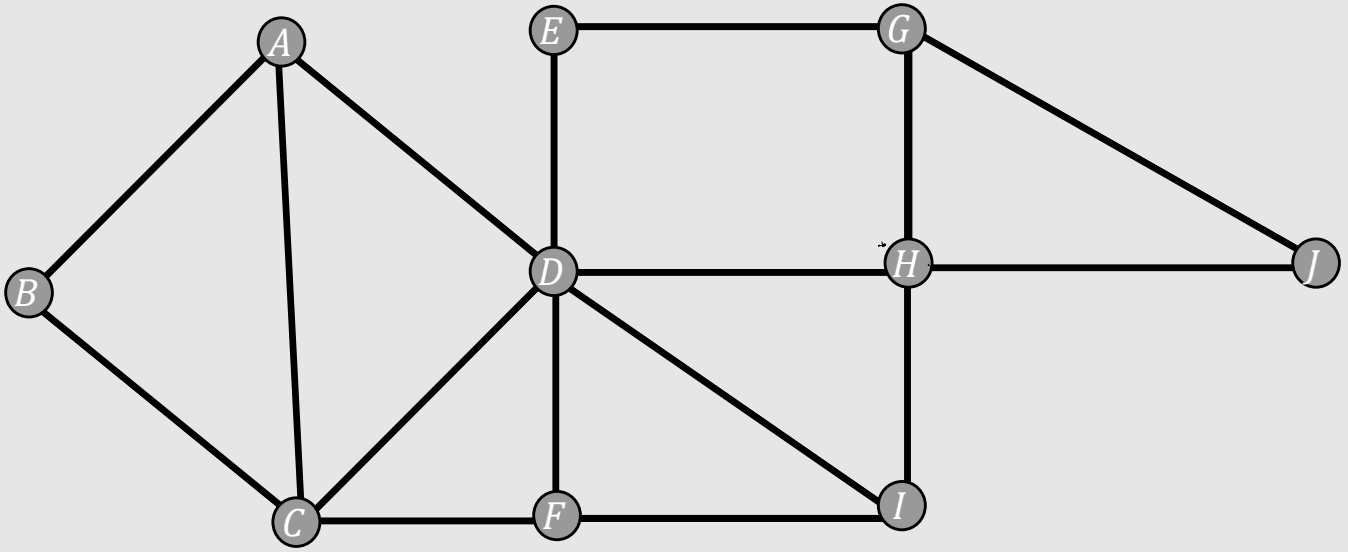
1. Breadth-First Search:

- We can also find a spanning tree by using Breadth-First Search technique.
  - i. **Example:**
    - **Firstly, we can choose any arbitrary vertex to start with** and then start adding vertices incident to this vertex, we are at level 1 in the spanning tree.
    - Next for each vertex at level 1, visited in order, add each edge incident to this vertex to form a tree, we are at level 2 and then continue in similar fashion if it does not produce a simple circuit.



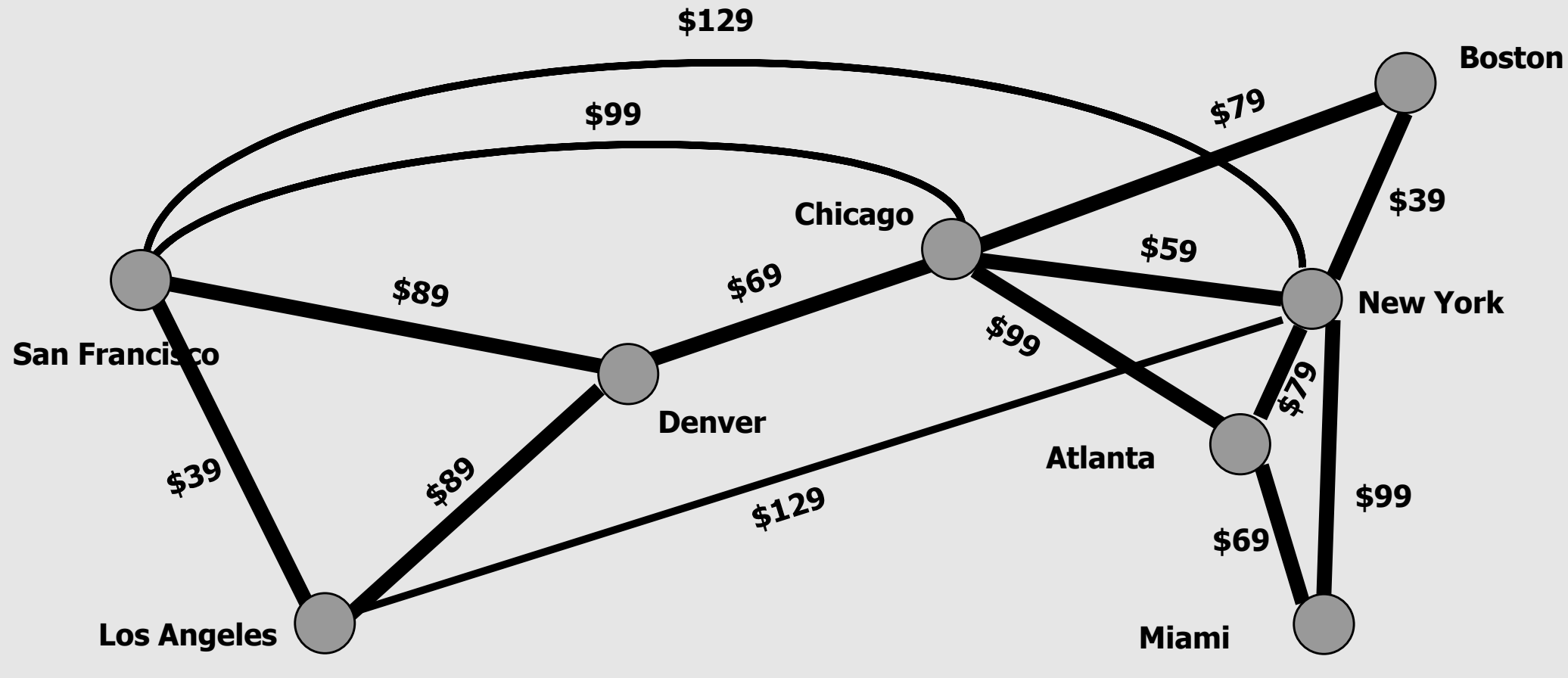
Spanning Trees

1. **Spanning Tree:**
   - Apply the breadth-first and depth-first search spanning tree algorithm to find a spanning tree for the given graph.
   - {A,B}{B,C}{B,E}{E,H}{B,D}{D,G}{G,J}{D,C}{C,F}



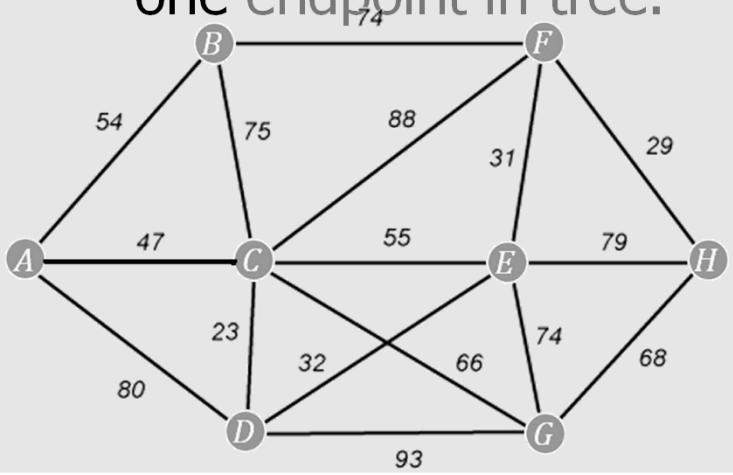
Spanning Trees

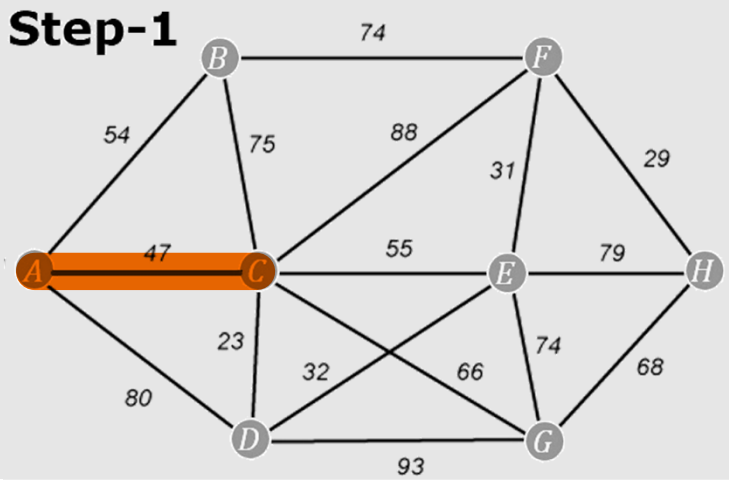
3. Fares:



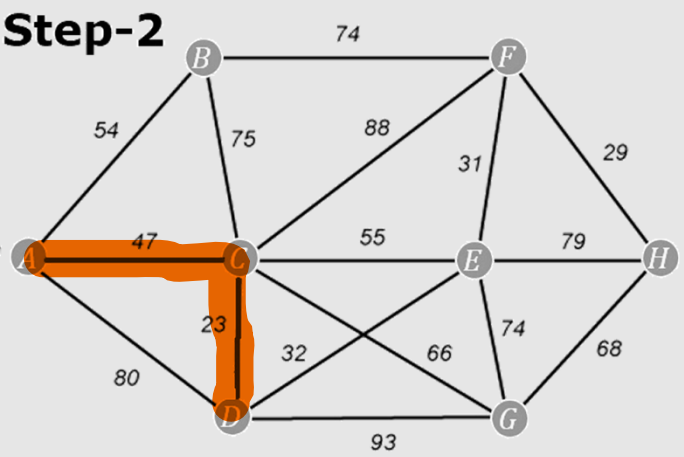
Spanning Trees

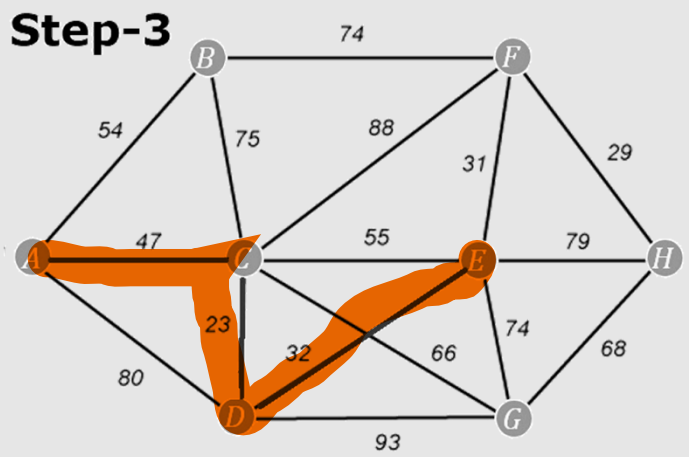
1. **Minimum Spanning Tree:**
   - Let G be a connected weighted graph (each edge has weight ≥ 0) and G is minimum spanning tree having smallest sum of weights of its edges.
     i. **Example:**
        - Use Prim's algorithm to find the minimum spanning tree of Graph G.
        - Construct a tree taking one vertex at a time.
        - Start with a trivial one-point tree.
        - Then start adding the edge of minimum weight among those with one endpoint in tree.

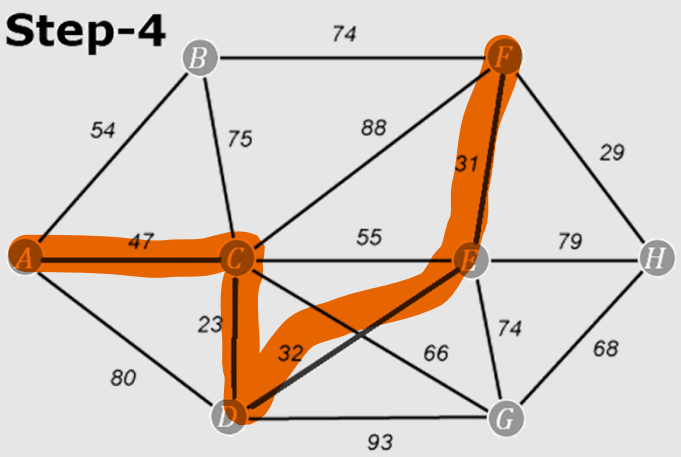


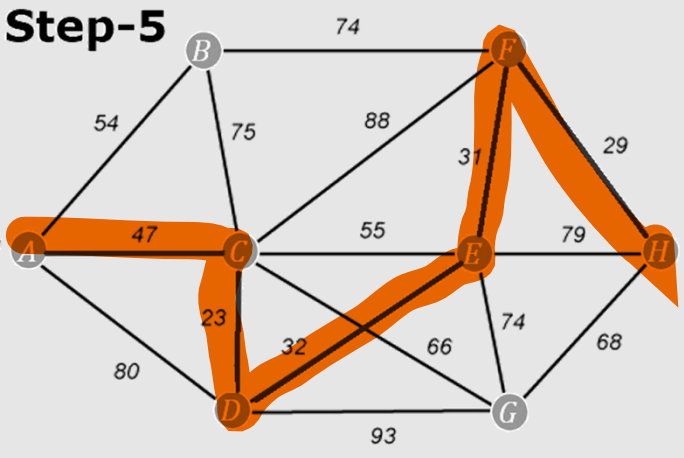


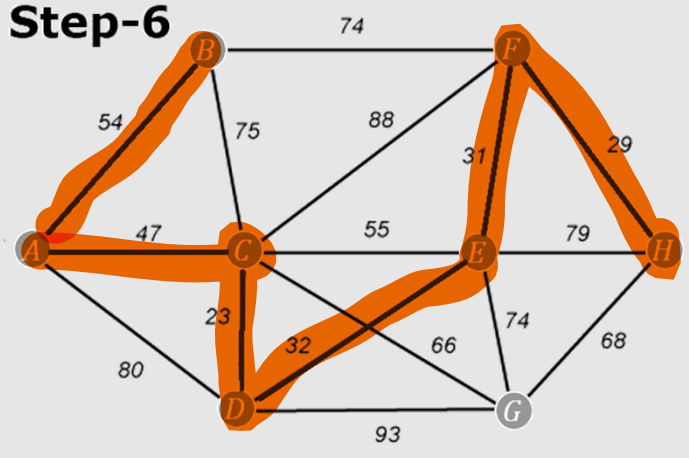
Spanning Trees

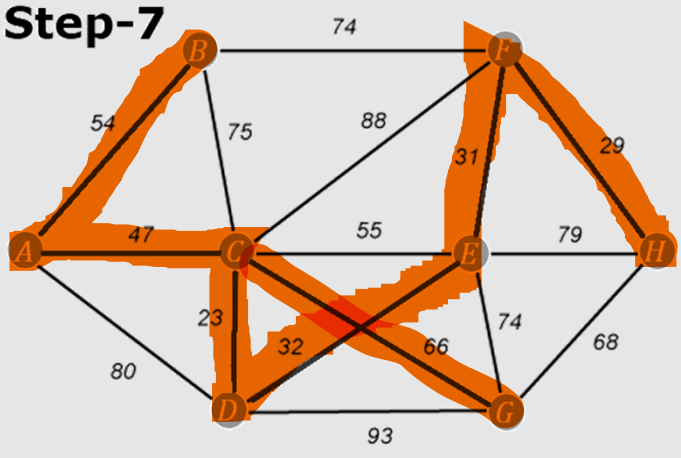












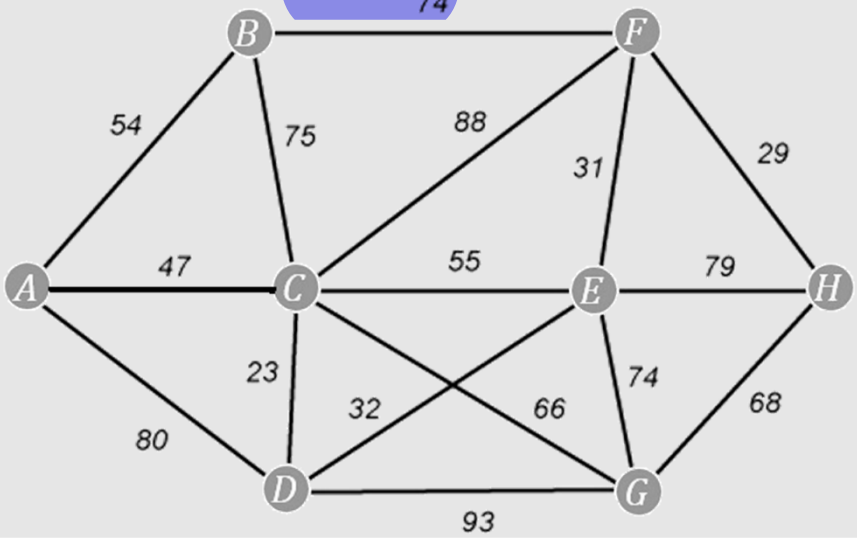
Weight (Tree) = 47 + 23 + 32 + 31 + 29 + 54 + 66 = 282

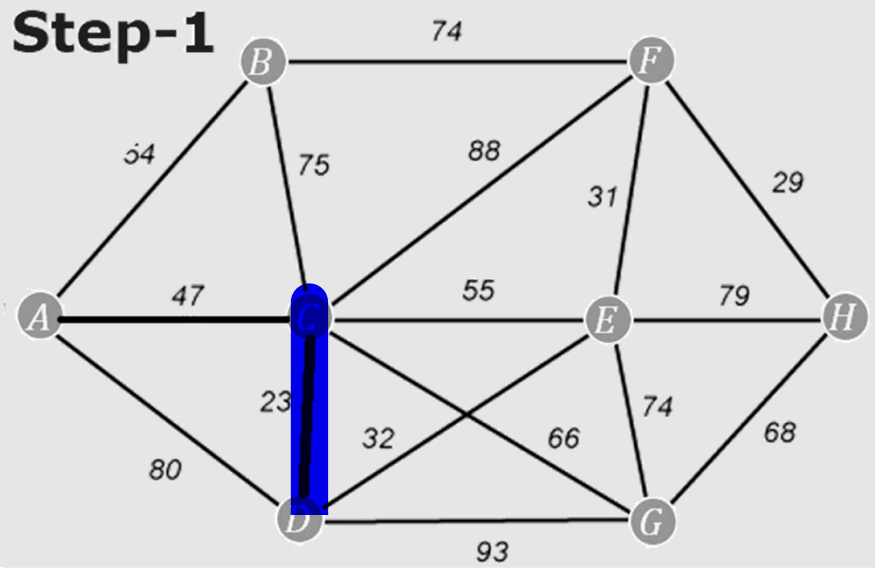
Kruskal's Algorithm

2. Kruskal's Algorithm:

i. Example:

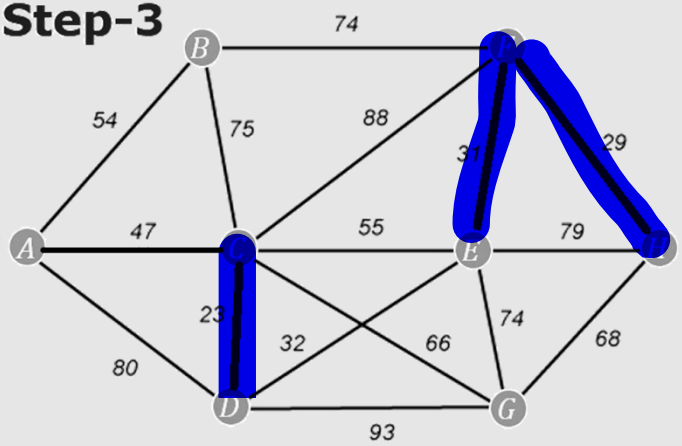
- Use Kruskal's algorithm to find the minimum spanning tree of Graph G.
- Find the edges by weight.
- Construct a minimal spanning tree by adding the edge of the minimum weight when added to those already chosen doesn't form a circuit.



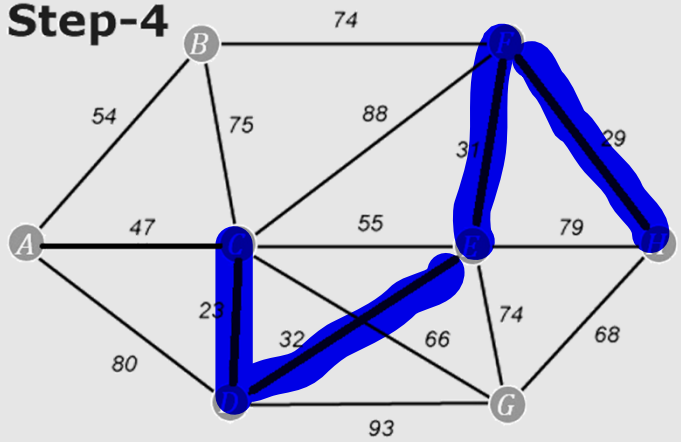


Spanning Trees

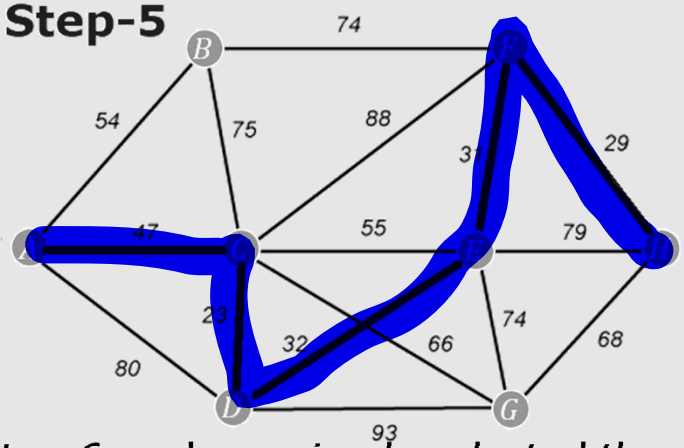
Step-3



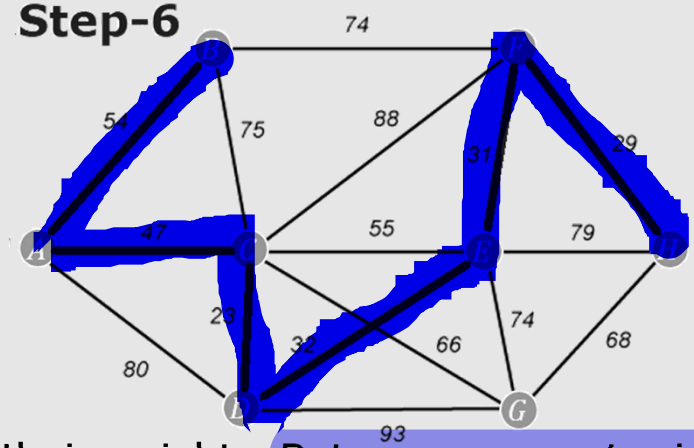
Step-4



Step-5



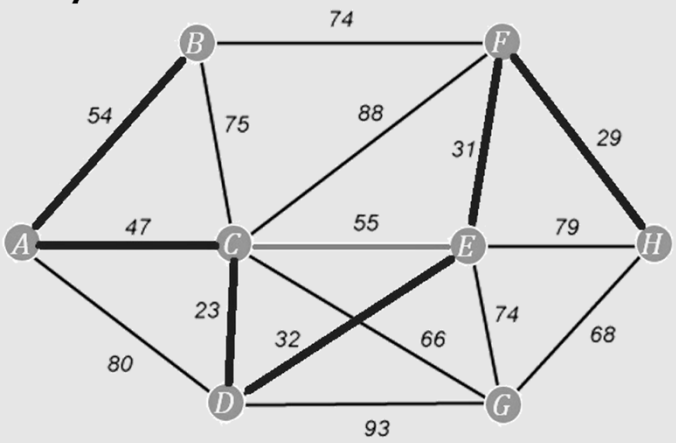
Step-6

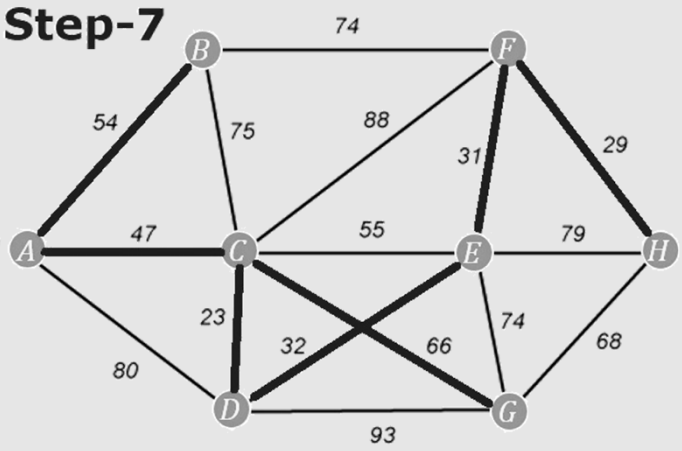


Up to step-6 we have simply selected the edges on the base of their weights. But now we must reject an edge since it forms a circuit when added to the already selected edges.

Kruskal's Algorithm

- An edge {C, E} can't be taken because when added to those already chosen form a circuit.



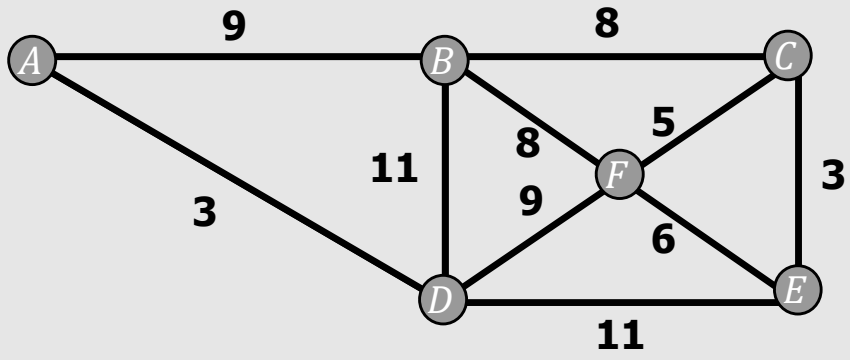


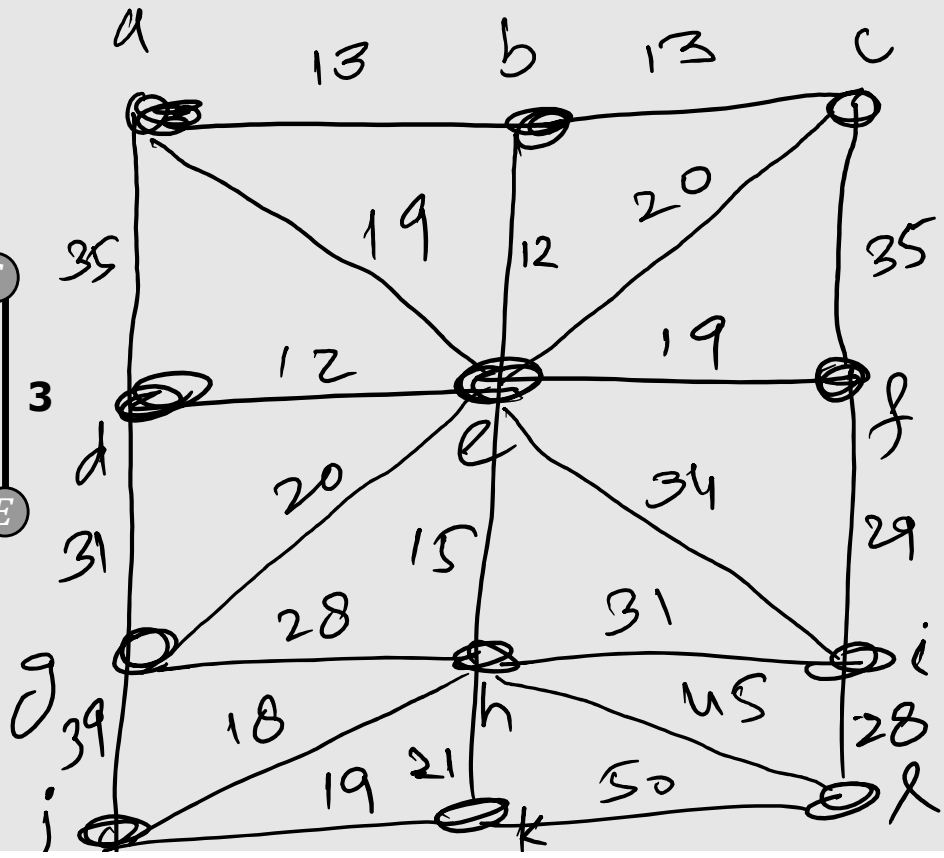
- Weight = 23 + 29 + 31 + 32 + 47 + 54 + 66 = 282 .

1. **Difference between Prim's and Kruskal's algorithms**
   - In Prim's algorithm edges having minimum weight and are incident to a vertex already in a tree, and not forming cycles, are selected, whereas Kruskal's algorithm prefers to select the edges having minimum weight but not necessarily incident to a vertex already in a tree and that don't form a cycle.

Spanning Trees

1. **Example:**
   - Apply Prim's and Kruskal's algorithm to find the minimum spanning tree of graph G.
   - {F, C}, {C, E}, {F, B}, {A, B}, {A, D}





UNITY CHART

Connecting The Parts Of Knowledge With The Wholeness Of Knowledge

Spanning Trees

1. Every connected graph contains a spanning tree that contains all of its vertices.
2. Every weighted connected graph contains a minimal or maximal spanning tree.

---

3. **Transcendental Consciousness** is a field of infinite correlation and perfect communication.
4. **Impulses within the transcendental field** begin to structure the differences we see in the relative world.
5. **Wholeness moving within itself:** In Unity Consciousness, we see every aspect of creation connected to our own Self.