GROUP 2

FINDING THE SHORTEST PATH USING DIJKSTRA'S ALGORITHM

- dijkstra's algorithm is a graph alggorithm that finds the shortest path from a source vertex to all other vertices in the graph (single source shortest path).
- it is a type of greedy algorithm that only works on weighted graphs having positive weights. the time complexity of this algorithm is $O(V^2)$ with the help of adjacency matrix representation of the graph.

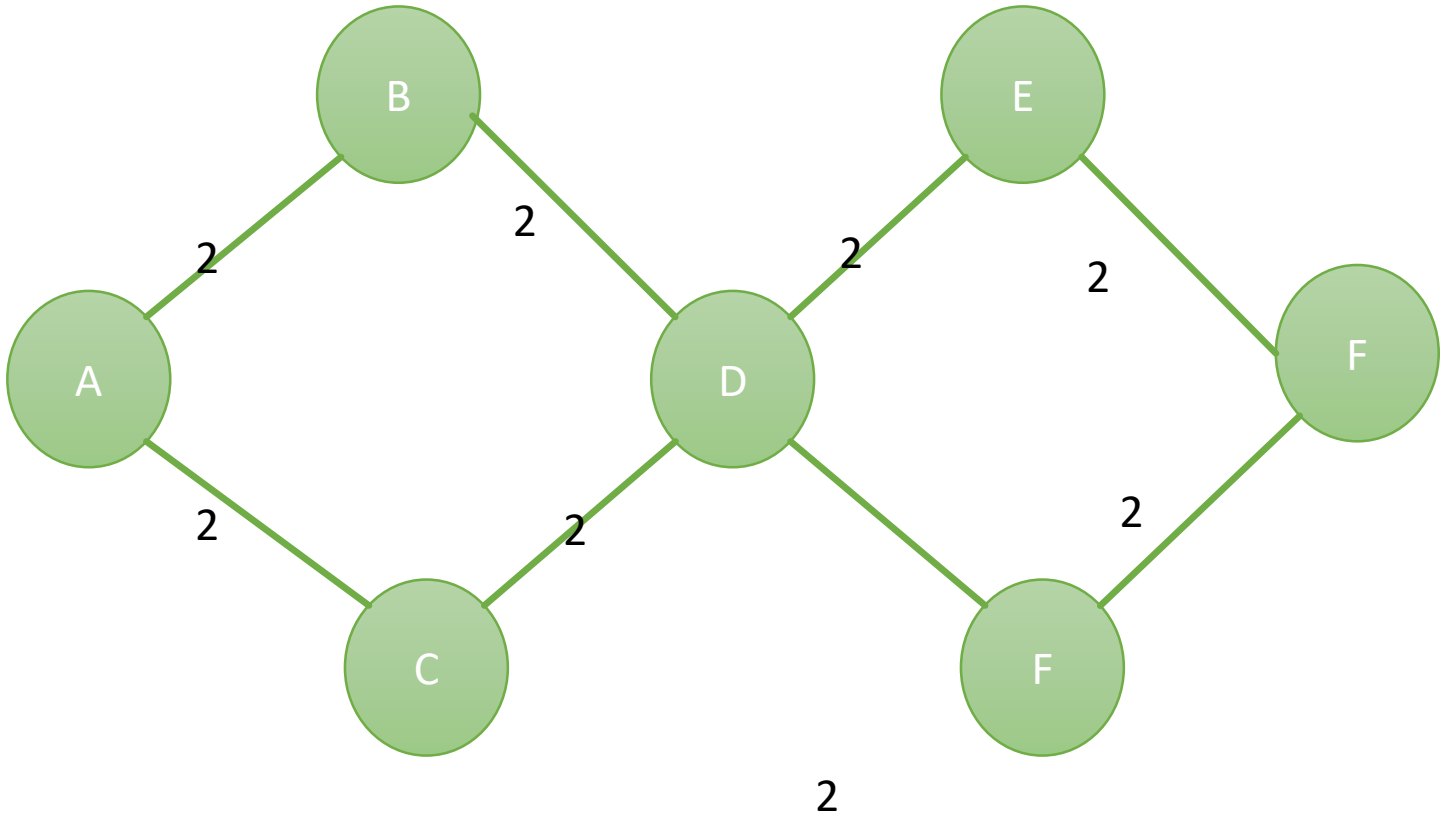
A graph and source vertex are requirements for Dijkstra's Algorithm. This Algorithm is established on Greedy Approach and thus finds the locally optimal choice (local minima in this case) at each step of the Algorithm.

Each Vertex in this Algorithm will have two properties defined for it:

- Visited Property
- Path Property

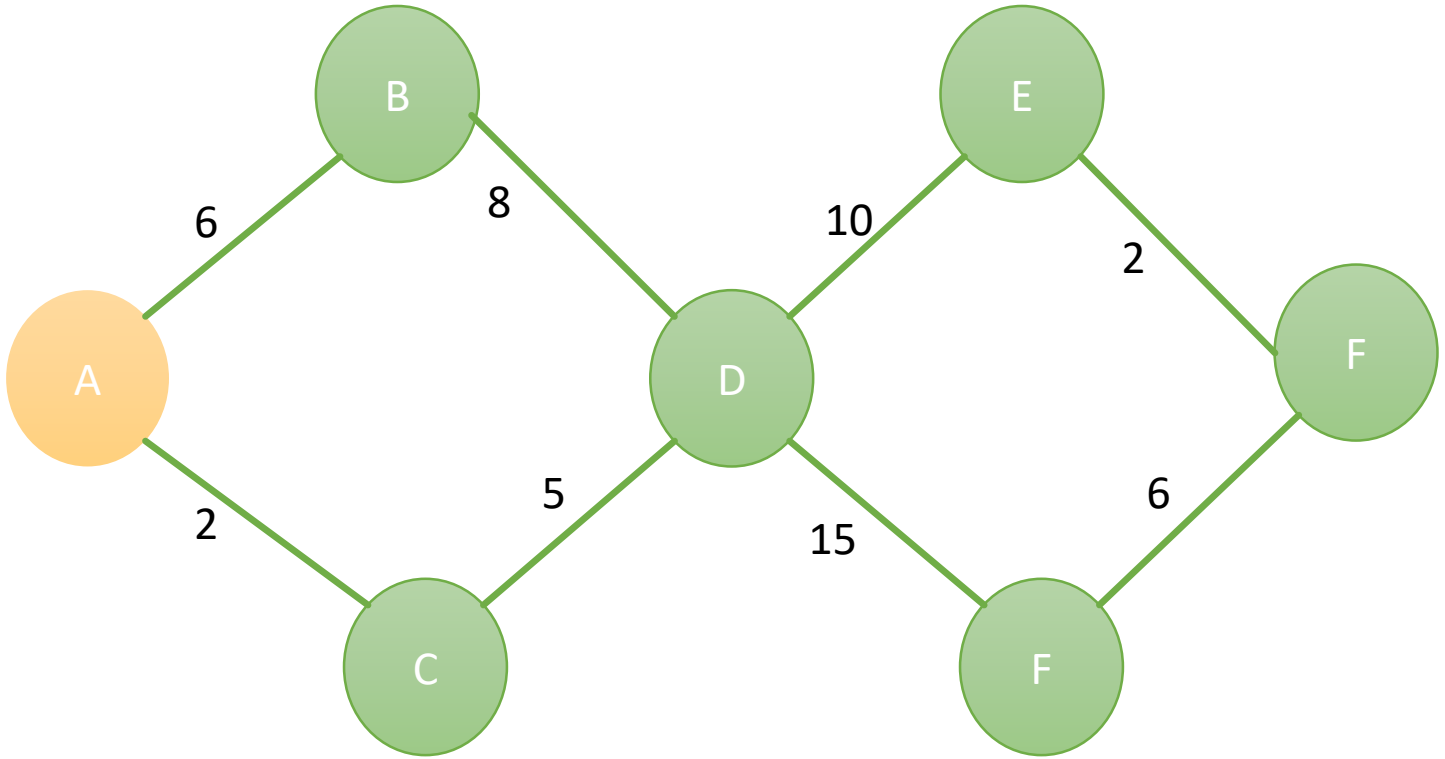
- Visited Property:
- The 'visited' property signifies whether or not the node has been visited.
- We are using this property so that we do not revisit any node.
- A node is marked visited only when the shortest path has been found.
- Path Property:
- The 'path' property stores the value of the current minimum path to the node.
- The current minimum path implies the shortest way we have reached this node till now.
- This property is revised when any neighbor of the node is visited.
- This property is significant because it will store the final answer for each node.

- Example graph



- The algorithm will generate the shortes path from node A to all the other nodes in the graph
- The distance from the source code to all nodes is unknown so we mark all of them as infinity.

- Step 1: Start from Node 0 and mark Node as visited as you can check in below image visited Node is marked orange.



Unvisited Nodes
{A,B,C,D,E,F,G}

NODE	DISTANCE
A	0
B	∞
C	∞
D	∞
E	∞
F	∞
G	∞

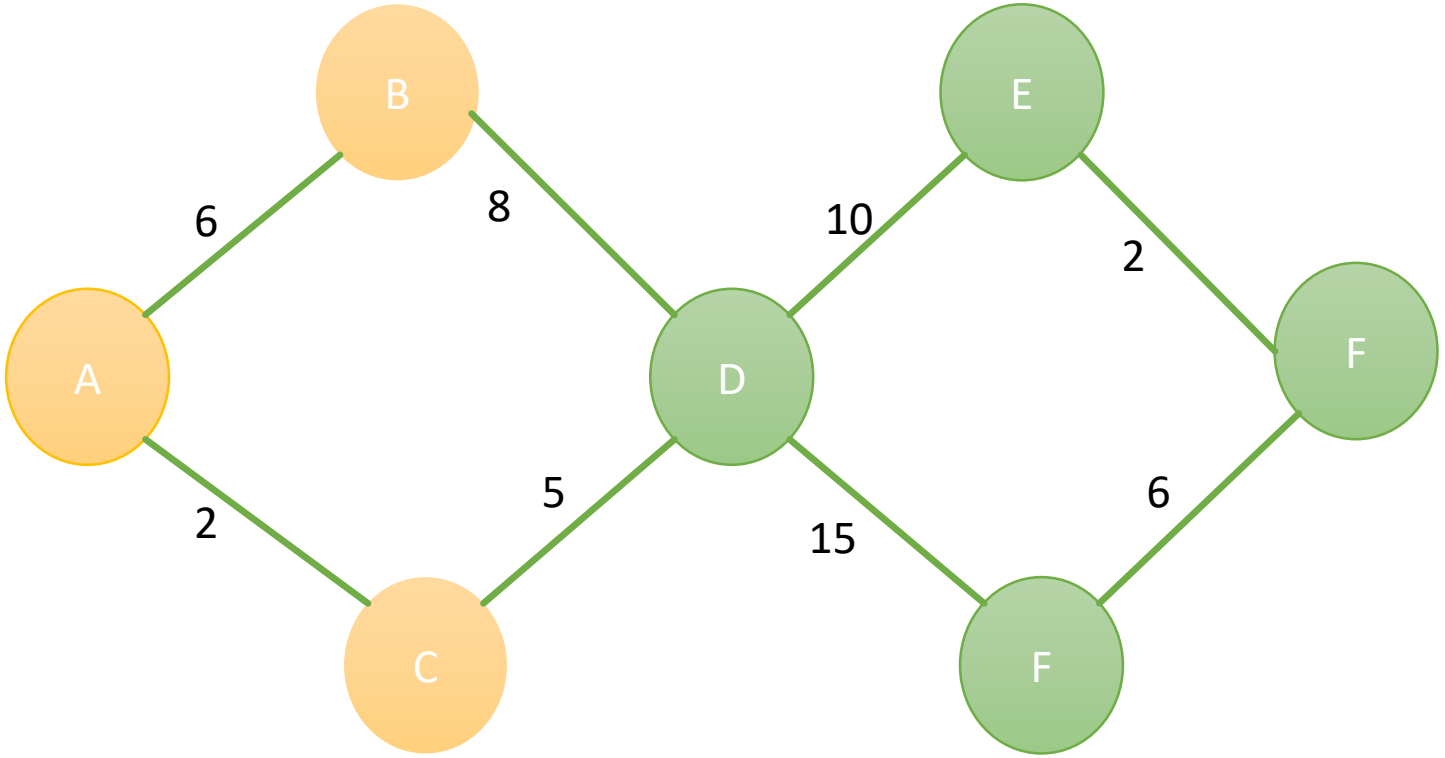
- Step 2 : Check for adjacent nodes, which are C and B

The shortest path to B is

- Distance: Node A -> Node B = 6

The shortest path to C is

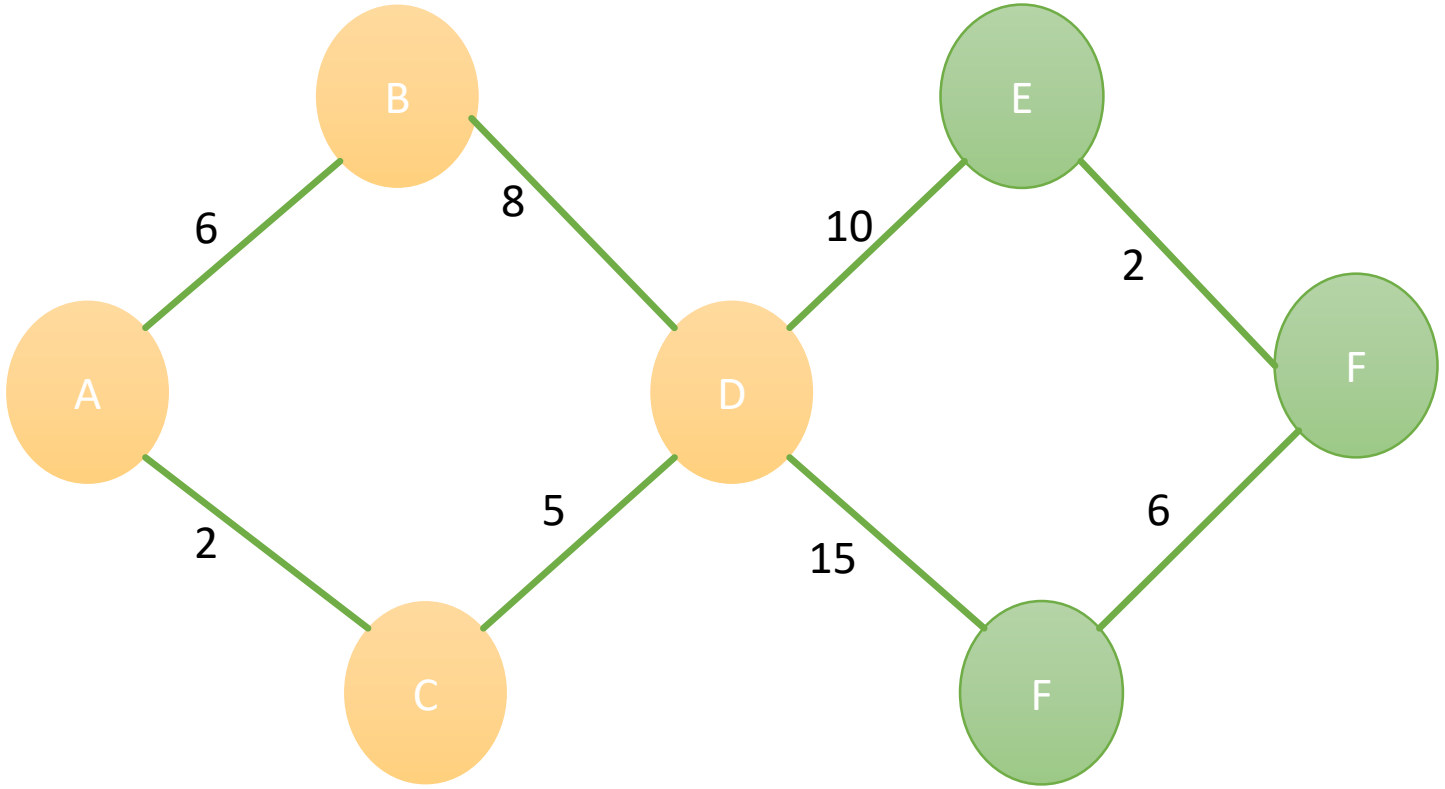
- Distance: Node A -> Node C = 2



Unvisited Nodes
{A,B,C,D,E,F,G}

NODE	DISTANCE
A	0
B	2
C	6
D	∞
E	∞
F	∞
G	∞

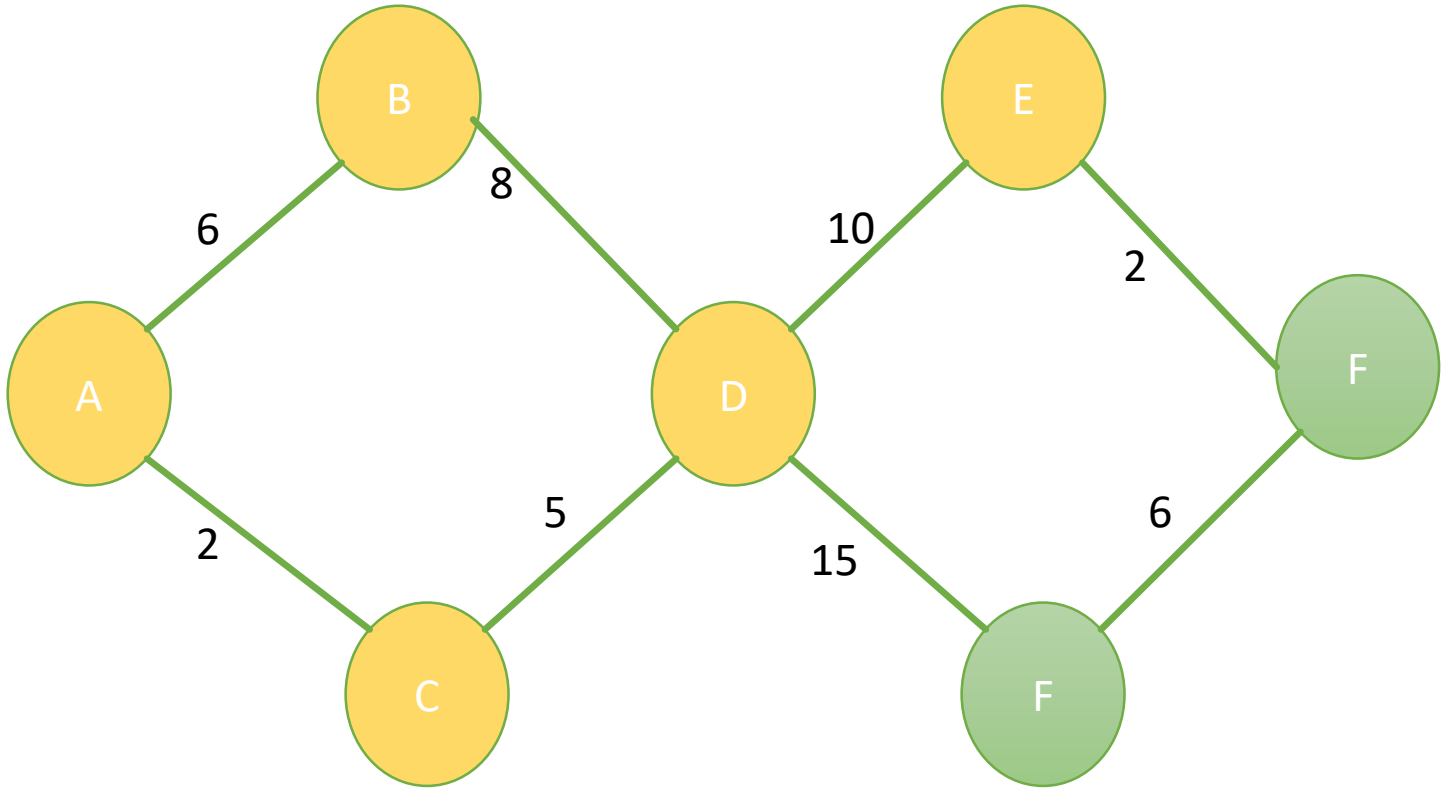
- Step 3:Then Move Forward and check for adjacent Node which is Node 3, so marked it as visited and add up the distance, Now the distance will be:
- Distance: Node A -> Node C -> Node D = 2 + 5 = 7



Unvisited Nodes
{A,B,C,D,E,F,G}

NODE	DISTANCE
A	0
B	2
C	6
D	7
E	∞
F	∞
G	∞

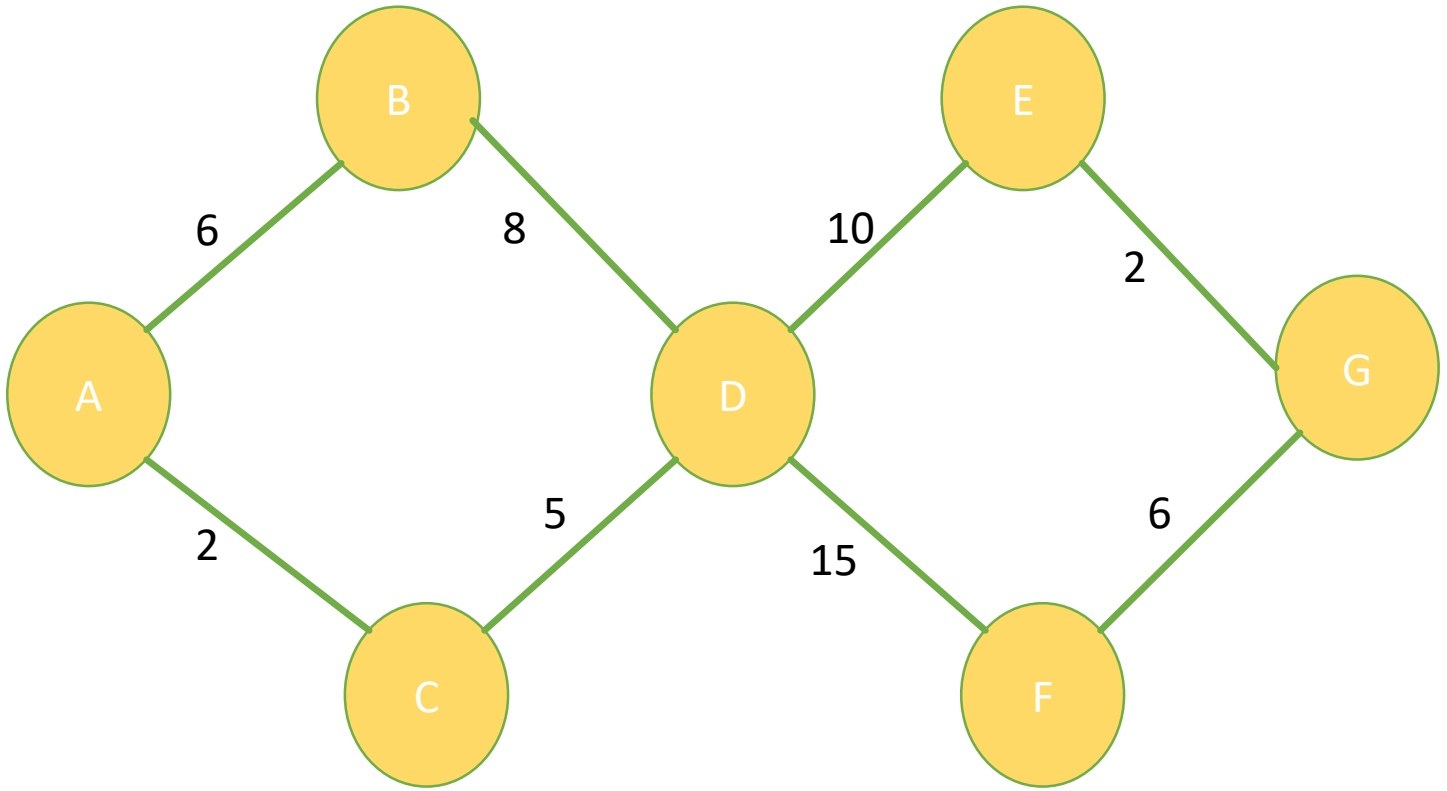
- Step 4: Again we have two choices for adjacent Nodes (Either we can choose Node E with distance 10 or either we can choose Node F with distance 15) so choose Node with minimum distance. In this step Node 4 is Minimum distance adjacent Node, so marked it as visited and add up the distance.
- Distance: Node A -> Node C -> Node D -> Node E = 2 + 5 + 10 = 17



Unvisited Nodes
{A,B,C,D,E,F,G}

NODE	DISTANCE
A	0
B	2
C	6
D	7
E	17
F	∞
G	∞

- Step 4: Again, Move Forward and check for adjacent Node which is Node G, so marked it as visited and add up the distance, Now the distance will be:
- Distance: Node A -> Node C -> Node D -> Node E -> Node G = 2 + 5 + 10 + 2 = 19



Unvisited Nodes
{A,B,C,D,E,F,G}

NODE	DISTANCE
A	0
B	2
C	6
D	7
E	17
F	22
G	19

APPLICATIONS OF DIJKSTRA'S ALGORITHM

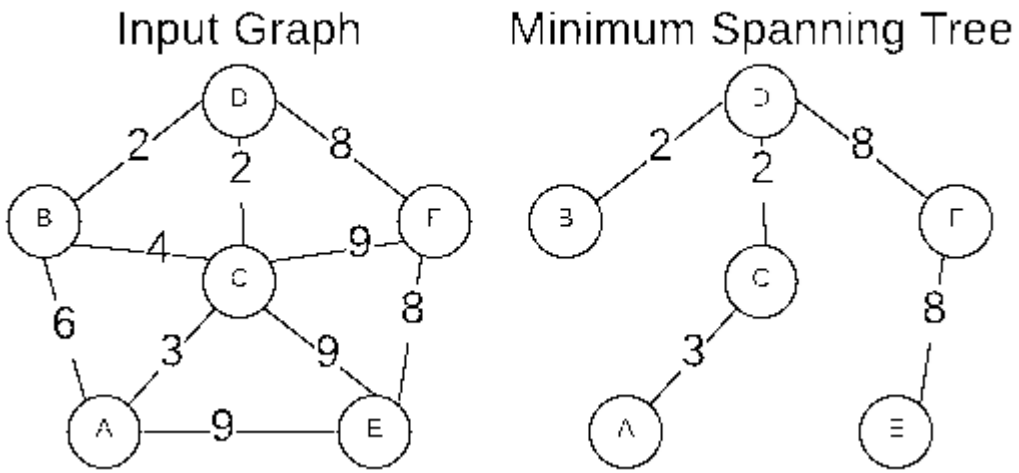
- In maps to get the shortest distance between locations. An example is Google Maps.
- In telecommunications to determine transmission rate.
- In robotic design to determine shortest path for automated robots.

MINIMUM SPANNING TREE PROBLEM

- A spanning tree: its defined as a tree-like graph of connected, undirected graph that includes alll the vertices of the graph.
- or it is asubset of the edges of the graph that forms a tree where every node of the graph is a part of the tree
- example

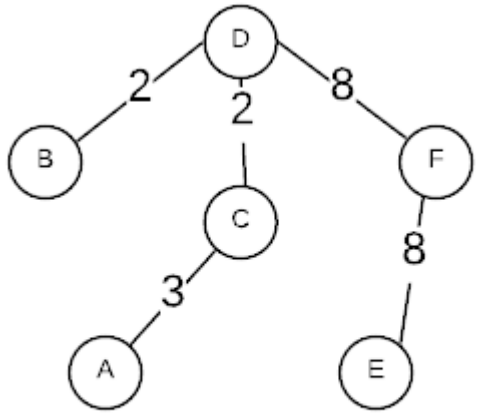
Properties of a spanning tree

- the number of vertices(V) in the graph and the spanning tree should be the same



- there is a fixed number of edges in the spanning tree which is equal to one less than the total number of vertices **(E = V-1)**
- the spanning tree should be acyclic, which means the tree formed should not bmake acycle.
- a path exists between every pair of vertices

- The total cost (or weight) of the spanning tree is defined as the sum of the edge weights of all the edges of the spanning tree.



Sum of Edge Weights = 23

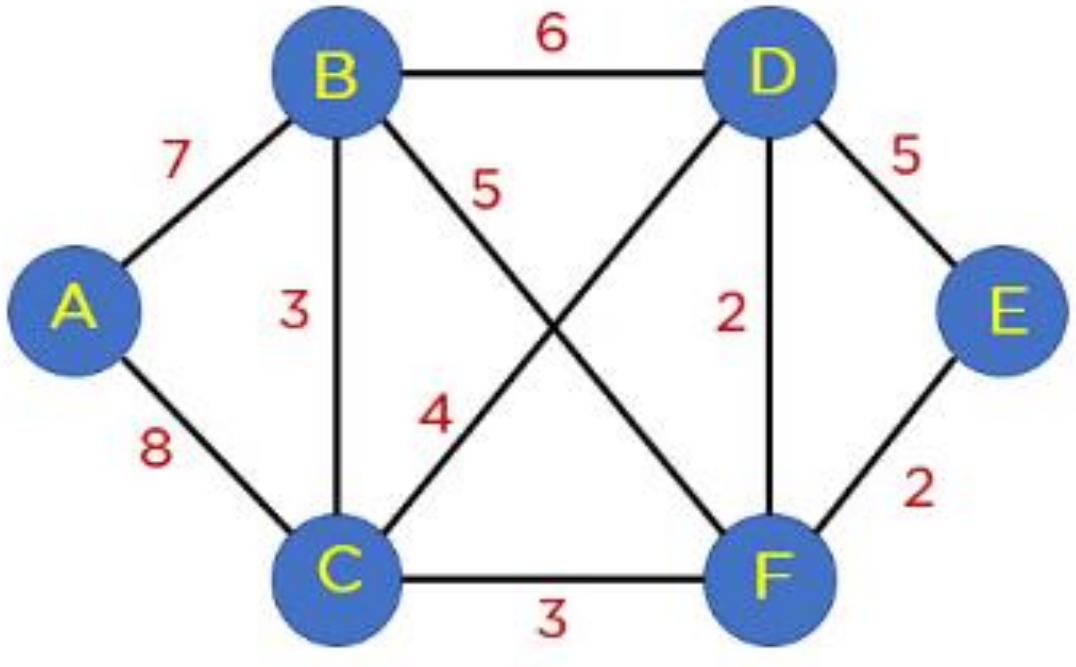
Minimum spanning tree

- Def: its defined as a spanning tree that has the minimum weight among all the possible spanning trees.
- The minimum spanning tree has all the properties of a spanning tree with an added constraint of having the minimum possible weights among all possible spanning trees. Like a spanning tree, there can also be many possible MSTs for a graph.

ALGORITHMS TO FIND MINIMUM SPANNING TREE

- **Kruskal's minimum spanning tree algorithm:** This is one of the popular algorithms for finding the minimum spanning tree from a connected, undirected graph. This is a greedy algorithm. The algorithm workflow is as below:
- First, it sorts all the edges of the graph by their weights,
- Then starts the iterations of finding the spanning tree.
- At each iteration, the algorithm adds the next lowest-weight edge one by one, such that the edges picked until now does not form a cycle.
- This algorithm can be implemented efficiently using a DSU (Disjoint-Set) data structure to keep track of the connected components of the graph. This is used in a variety of practical applications such as network design, clustering, and data analysis.

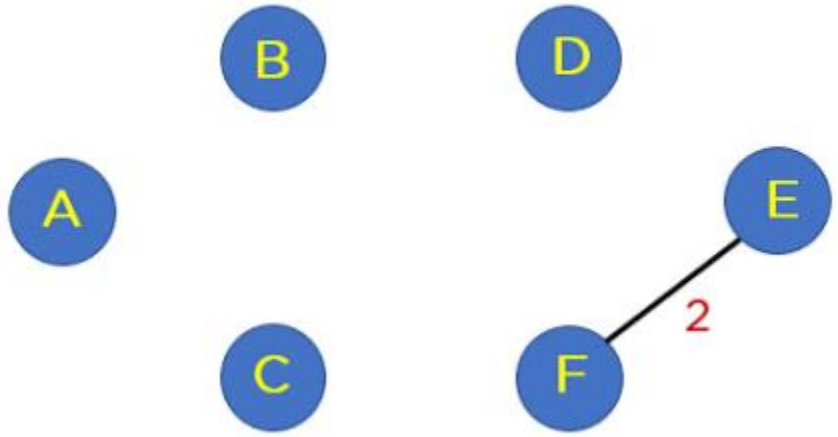
CREATING MINIMUN SPANNING TREE USING KRUSKAL ALGORITHM



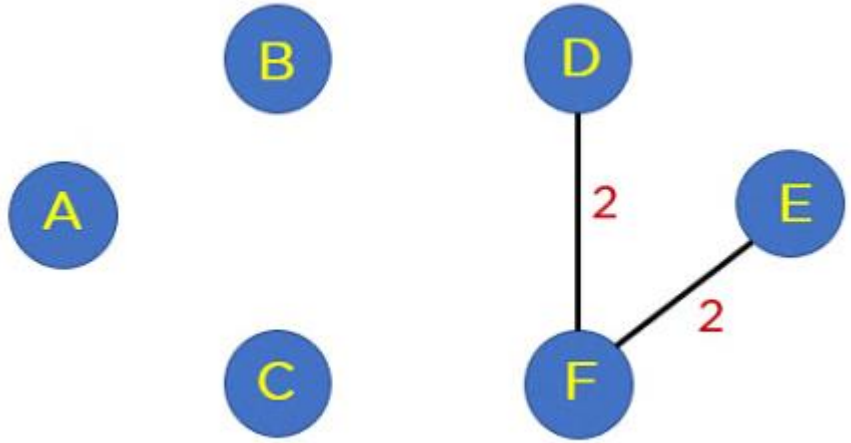
Arrange all edges in a sorted list by their edge weights.

The edges of the graph	edge weight
source vertex	destination vertex
E	F
F	D
B	C
C	F
C	D
B	F
B	D
A	B
A	C

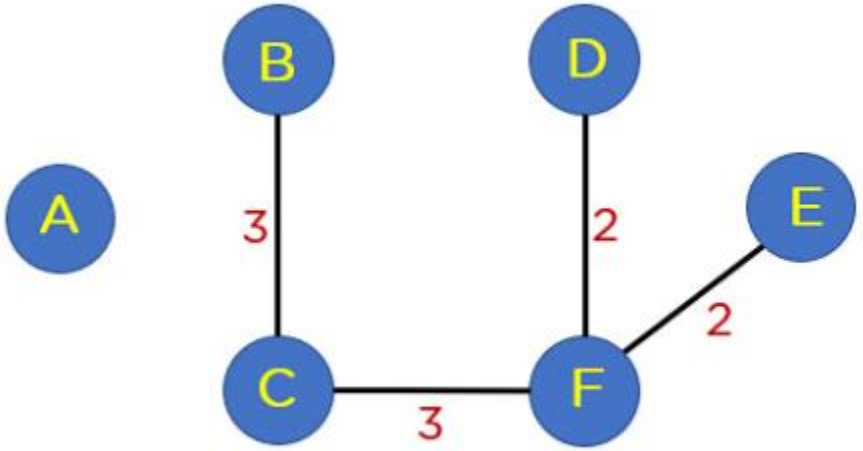
- Step2: After this step, you will include edges in the MST such that the included edge would not form a cycle in your tree structure. The first edge that you will pick is edge EF, as it has a minimum edge weight that is 2.



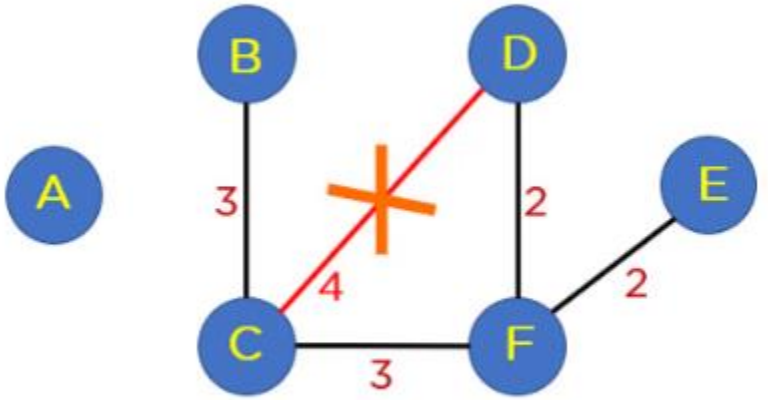
- Step3: Add edge FD to the spanning tree.



- Step 4: Add edge BC and edge CF to the spanning tree as it does not generate any loop.

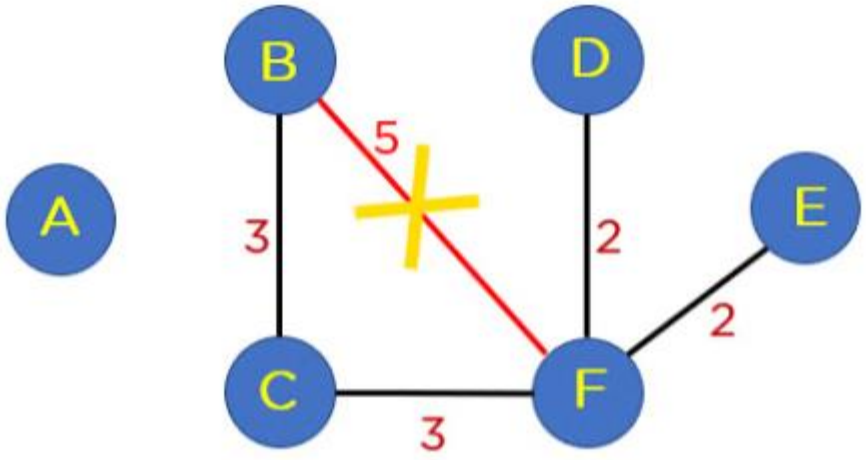


- Step 5: Next up is edge CD. This edge generates the loop in Your tree structure. Thus, you will discard this edge.



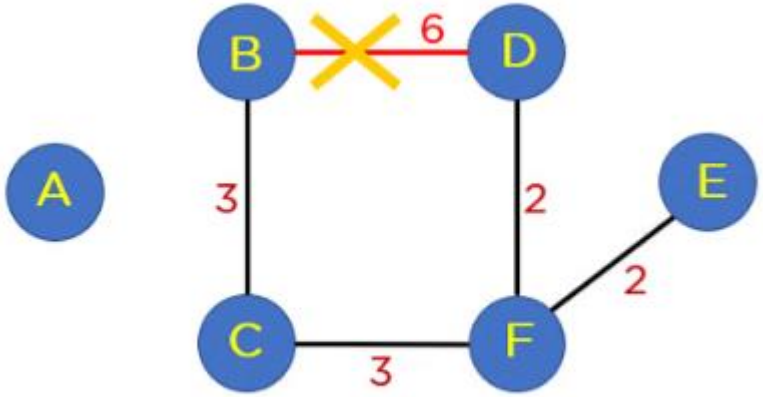
Edge CD should be discarded, as it creates loop.

- Step 6: Following edge CD, you have edge BF. This edge also creates the loop; hence you will discard it.



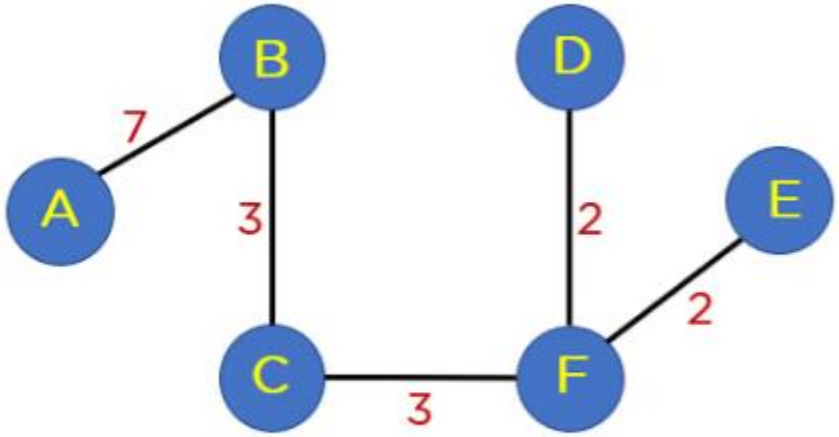
Edge BF should be discarded.

- Step 7: Next up is edge BD. This edge also formulates a loop, so you will discard it as well.



Edge BD should be discarded.

- Step 8: Next on your sorted list is edge AB. This edge does not generate any cycle, so you need not include it in the MST structure. By including this node, it will include 5 edges in the MST, so you don't have to traverse any further in the sorted list. The final structure of your MST is represented in the image below:



Minimum Spanning Tree.

- The summation of all the edge weights in MST T(V', E') is equal to 17, which is the least possible edge weight for any possible spanning tree structure for this particular graph. Moving ahead, you will learn about implementing Kruskal algorithms using the Union Find Algorithm.

- **Prim's minmum spanning tree algorithm:** This is also a greedy algorithm. This algorithm has the following workflow:
- It starts by selecting an arbitrary vertex and then adding it to the MST.
- Then, it repeatedly checks for the minimum edge weight that connects one vertex of MST to another vertex that is not yet in the MST.
- This process is continued until all the vertices are included in the MST.

- To efficiently select the minimum weight edge for each iteration, this algorithm uses priority_queue to store the vertices sorted by their minimum edge weight currently. It also simultaneously keeps track of the MST using an array or other data structure suitable considering the data type it is storing.

- This algorithm can be used in various scenarios such as image segmentation based on color, texture, or other features. For Routing, as in finding the shortest path between two points for a delivery truck to follow.

APPLICATIONS OF MINIMUM SPANNING TREE

- Network design: Spanning trees can be used in network design to find the minimum number of connections required to connect all nodes. Minimum spanning trees, in particular, can help minimize the cost of the connections by selecting the cheapest edges.

- Image processing: Spanning trees can be used in image processing to identify regions of similar intensity or color, which can be useful for segmentation and classification tasks.

- Biology: Spanning trees and minimum spanning trees can be used in biology to construct phylogenetic trees to represent evolutionary relationships among species or genes.

- Social network analysis: Spanning trees and minimum spanning trees can be used in social network analysis to identify important connections and relationships among individuals or groups.