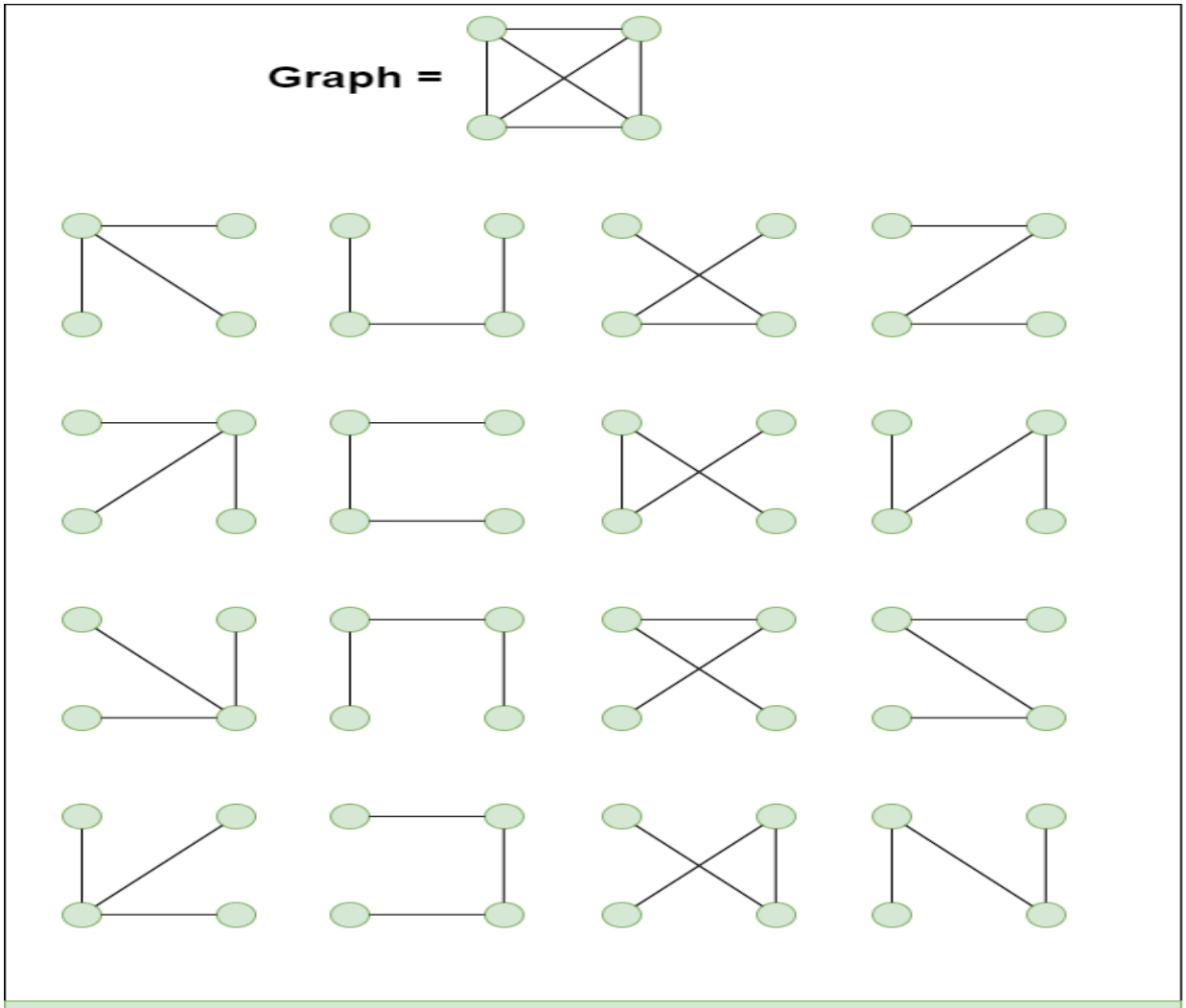
What is a spanning tree?

A spanning tree can be defined as the subgraph of an undirected connected graph. It includes all the vertices along with the least possible number of edges. If any vertex is missed, it is not a spanning tree. A spanning tree is a subset of the graph that does not have cycles, and it also cannot be disconnected.

A spanning tree consists of (n-1) edges, where 'n' is the number of vertices (or nodes). Edges of the spanning tree may or may not have weights assigned to them. All the possible spanning trees created from the given graph G would have the same number of vertices, but the number of edges in the spanning tree would be equal to the number of vertices in the given graph minus 1.

The total number of spanning trees with n vertices that can be created from a complete graph is equal to $\boldsymbol{n^{n-2}}$
If we have n = 4, the maximum number of possible spanning trees is equal to 16. Thus, 16 spanning trees can be formed from a complete graph with 4 vertices.



All Possible Spanning Trees of the Graph

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.

Prim's Algorithm

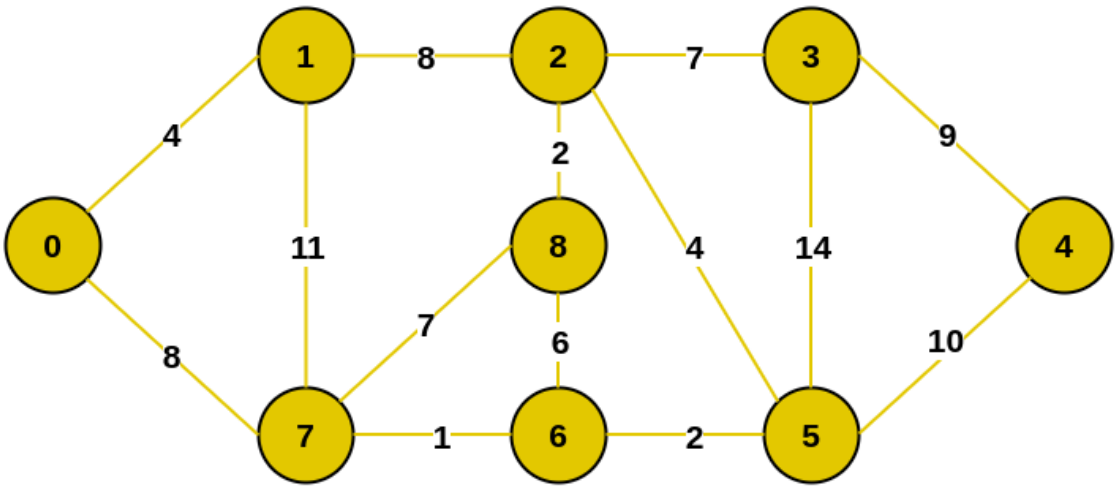
Prim's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which

- form a tree that includes every vertex
- has the minimum sum of weights among all the trees that can be formed from the graph

- **Prim's Algorithm** is a greedy algorithm that is used to find the minimum spanning tree from a graph. Prim's algorithm finds the subset of edges that includes every vertex of the graph such that the sum of the weights of the edges can be minimized.
- Prim's algorithm starts with the single node and explores all the adjacent nodes with all the connecting edges at every step. The edges with the minimal weights causing no cycles in the graph got selected.

Algorithm:
Prim's algorithm is a greedy algorithm that starts from one vertex and continue to add the edges with the smallest weight until the goal is reached. The steps to implement the prim's algorithm are given as follows

- First, we have to initialize an MST with the randomly chosen vertex.
- Now, we have to find all the edges that connect the tree in the above step with the new vertices. From the edges found, select the minimum edge and add it to the tree.
- Repeat step 2 until the minimum spanning tree is formed.



Example of a Graph

Advantages:

1. Prim’s algorithm is guaranteed to find the MST in a connected, weighted graph.
2. It has a time complexity of O(E log V) using a binary heap or Fibonacci heap, where E is the number of edges and V is the number of vertices.
3. It is a relatively simple algorithm to understand and implement compared to some other MST algorithms.

Disadvantages:

1. Like Kruskal’s algorithm, Prim’s algorithm can be slow on dense graphs with many edges, as it requires iterating over all edges at least once.
2. Prim’s algorithm relies on a priority queue, which can take up extra memory and slow down the algorithm on very large graphs.
3. The choice of starting node can affect the MST output, which may not be desirable in some applications

Kruskal's Algorithm

Kruskal's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which

- form a tree that includes every vertex
- has the minimum sum of weights among all the trees that can be formed from the graph

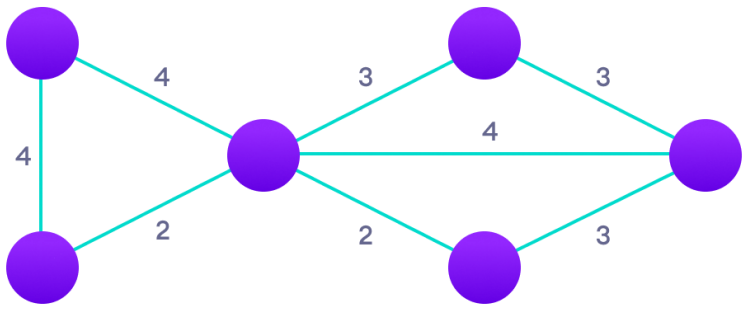
How Kruskal's algorithm works

It falls under a class of algorithms called greedy algorithms that find the local optimum in the hopes of finding a global optimum.
We start from the edges with the lowest weight and keep adding edges until we reach our goal.

The steps for implementing Kruskal's algorithm are as follows:

1. Sort all the edges from low weight to high

2. Take the edge with the lowest weight and add it to the spanning tree. If adding the edge created a cycle, then reject this edge.
3. Keep adding edges until we reach all vertices.



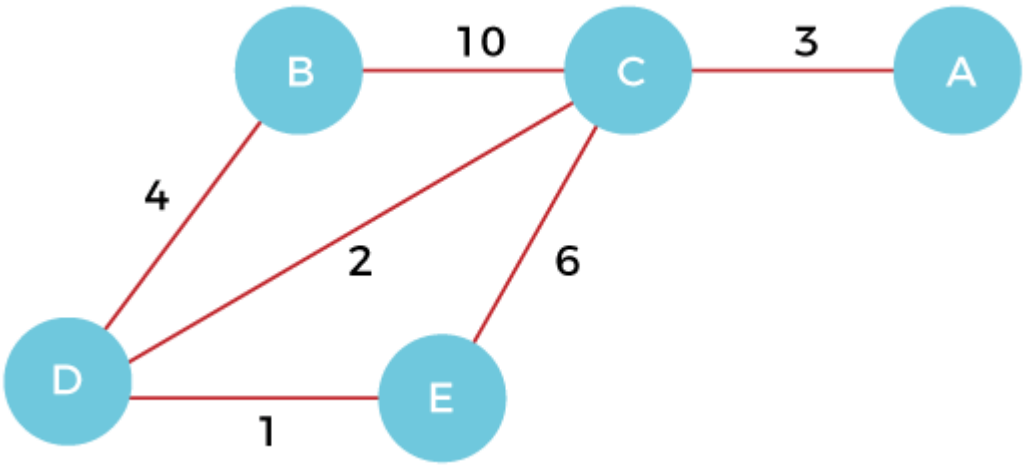
Step: 1

Kruskal's Algorithm Complexity

The time complexity Of Kruskal's Algorithm is: `O(E log E)`.

Kruskal's Algorithm Applications

- In order to layout electrical wiring
- In computer network (LAN connection)




```
# Prim's Algorithm in Python


INF = 9999999
# number of vertices in graph
V = 5
# create a 2d array of size 5x5
# for adjacency matrix to represent graph
G = [[0, 9, 75, 0, 0],
     [9, 0, 95, 19, 42],
     [75, 95, 0, 51, 66],
     [0, 19, 51, 0, 31],
     [0, 42, 66, 31, 0]]
# create a array to track selected vertex
# selected will become true otherwise false
selected = [0, 0, 0, 0, 0]
# set number of edge to 0
no_edge = 0
# the number of egde in minimum spanning tree will be
# always less than(V - 1), where V is number of vertices in
# graph
# choose 0th vertex and make it true
selected[0] = True
# print for edge and weight
print("Edge : Weight\n")
while (no_edge < V - 1):
```

```
# For every vertex in the set S, find the all adjacent vertices
#, calculate the distance from the vertex selected at step 1.
# if the vertex is already in the set S, discard it otherwise
# choose another vertex nearest to selected vertex  at step 1.
minimum = INF
x = 0
y = 0
for i in range(V):
    if selected[i]:
        for j in range(V):
            if ((not selected[j]) and G[i][j]):
                # not in selected and there is an edge
                if minimum > G[i][j]:
                    minimum = G[i][j]
                    x = i
                    y = j
print(str(x) + "-" + str(y) + ":" + str(G[x][y]))
selected[y] = True
no_edge += 1
```