# MIT World Peace University
# Advanced Data Structures

*Assignment 6*

Naman Soni Roll No. 10

# Contents

# 1 Problem Statement

A business house has several offices in different countries; they want to lease phone lines to connect them with each other and the phone company charges different rent to connect different pairs of cities. Business house wants to connect all its offices with a minimum total cost. Solve the problem using Prim's algorithm.

# 2 Objective

1. To study data structure Graph and its representation using cost adjacency Matrix
2. To study and implement algorithm for minimum cost spanning tree
3. To study and implement Prim's Algorithm for minimum cost spanning tree
4. To study how graph can be used to model real world problems

# 3 Theory

## 3.1 *What is Spanning tree. Explain with example.*

A spanning tree is a subset of edges in an undirected graph that connects all vertices in the graph without forming any cycles. In other words, it is a tree that spans all vertices in the graph.

A spanning tree can be obtained from any connected undirected graph. The spanning tree will have the same number of vertices as the original graph, but a smaller number of edges. The minimum number of edges required to form a spanning tree is n-1, where n is the number of vertices in the graph.

Here is an example:

Consider the following undirected graph:

```
   A
  / \
   B - C
  / \ / \
 D - E - F
```

To obtain a spanning tree from this graph, we can start with any vertex (let's say vertex A) and perform a depth-first search or breadth-first search. As we visit each vertex, we add the edges that connect that vertex to its unvisited neighbors to the spanning tree. We continue this process until we have visited all vertices in the graph.

One possible spanning tree for this graph is:

```
  A
  \
   B - C
  /   /
   D - E
    \
     F
```

## 3.2 *Example of weighted graph and its cost adjacency matrix (with diagrams)*

A weighted graph is a graph in which each edge has a numerical weight assigned to it. The weight can represent any kind of cost, distance, or value associated with the edge. Here is an example of a weighted graph and its cost adjacency matrix:

Consider the following weighted graph:

```
  2
  A --- B
  |     | \
   3|     |1 \4
  |     |   \
  C --- D --- E
     5     2
```

In this graph, each edge is labeled with its weight. For example, the edge between A and B has weight 2, and the edge between B and D has weight 4.

To represent the weights in the form of a cost adjacency matrix, we can create a matrix where each row and column corresponds to a vertex in the graph. The entry in row i and column j represents the cost of the edge connecting vertex i to vertex j, if such an edge exists. If there is no edge between i and j, the entry is typically set to infinity.

For the above graph, the cost adjacency matrix would be:

```
  A    B    C    D    E
  +-----------------------
  A |   0    2    3   Inf  Inf
  B |   2    0  Inf    4    1
  C |   3  Inf   0    5  Inf
  D | Inf   4    5    0    2
  E | Inf   1  Inf    2    0
```

The entries corresponding to edges with weights are filled in with the weight values, while the entries corresponding to non-existent edges are set to infinity. For example, the entry in row A and column E is Inf, since there is no direct edge between A and E.

This cost adjacency matrix can be used to perform various graph algorithms that require knowledge of edge weights, such as Dijkstra's shortest path algorithm or Prim's minimum spanning tree algorithm.

### 3.3 *Explanation of Prim's algorithm using the above example graph*

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. The algorithm starts with a single vertex and then adds the edge of minimum weight that connects the tree to a vertex that is not yet in the tree. It repeats this process until all vertices are in the tree.

Here's how Prim's algorithm would work on the example graph given in the previous question:

```
  2
  A --- B
  |     | \
   3|     |1 \4
  |     |   \
  C --- D --- E
     5     2
```

1. Choose a starting vertex, say A, and add it to the minimum spanning tree (MST). The MST now contains only one vertex: A.

```
  2
  A --- B
  |     | \
   3|     |1 \4
  |     |   \
  C --- D --- E
     5     2

  {A}
```

2. Find the minimum weight edge that connects any vertex in the MST to a vertex that is not yet in the MST. In this case, the minimum weight edge is the edge between A and B, with weight 2. Add this edge to the MST.

```
  2
A --- B
      \
     4 \
        \
C --- D --- E
   5     2

{A, B}
```

3. Find the minimum weight edge that connects any vertex in the MST to a vertex that is not yet in the MST. In this case, there are two options: the edge between B and D with weight 4, and the edge between B and E with weight 1. Choose the edge between B and E since it has the smaller weight. Add this edge to the MST.

```
  2
A --- B
      \
     1 \
        \
C --- D --- E
   5     2

{A, B, E}
```

4. Find the minimum weight edge that connects any vertex in the MST to a vertex that is not yet in the MST. In this case, there are two options: the edge between B and D with weight 4, and the edge between D and C with weight 5. Choose the edge between D and C since it has the smaller weight. Add this edge to the MST.

```
  2
A --- B
      \
     1 \
        \
C --- D    E
   5     2

{A, B, E, D, C}
```

5. All vertices are now in the MST, so the algorithm terminates. The minimum spanning tree for the given graph is:

```
  2
A --- B
      \
     1 \
        \
C --- D    E
```

The minimum spanning tree has total weight 5+2+1=8, which is the sum of the weights of the three edges in the tree.

# 4 Implementation

## 4.1 *Platform*

- 64-bit Mac OS
- Open Source C++ Programming tool like Visual Studio Code

subsection*Test Conditions*

1. Input at least 5 nodes.
2. Display Minimum cost spanning tree and Minimum cost for the graph.

# 5 Conclusion

Thus, we have represented graph using cost adjacency matrix and implemented Prim's algorithm for MCST.

# 6 FAQ's

## 6.1 *Explain two applications of minimum cost spanning tree.*

Minimum cost spanning trees (MSTs) have a wide range of applications in various fields. Here are two common applications:

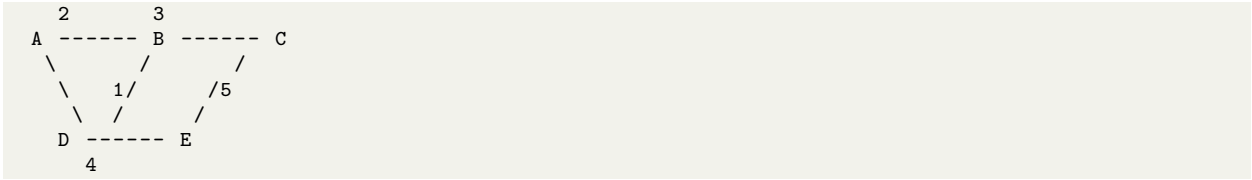
1. Network Design: In network design, MSTs are used to minimize the total cost of constructing a communication network, while ensuring that all nodes are connected. This can be useful in designing electrical power grids, telecommunications networks, computer networks, and transportation systems. By finding an MST for a given set of nodes and edges, we can determine the minimum cost of constructing a network that connects all nodes.

2. Clustering: MSTs can also be used for clustering analysis in data mining and machine learning. In this application, MSTs are used to identify clusters or groups of similar data points based on their pairwise distances. The MST is constructed by treating the data points as nodes in the graph, and the pairwise distances between them as the weights of the edges. The MST can then be used to identify clusters of data points that are closely related to each other, based on the distance between them. This can be useful in image segmentation, customer segmentation, and anomaly detection.

Overall, MSTs are a powerful tool in graph theory that have numerous practical applications in many fields.

## 6.2 *Explain the difference between Prim's and Kruskal's Algorithm for finding minimum cost spanning tree with a small example graph.*

Prim's algorithm and Kruskal's algorithm are two popular algorithms used to find a minimum cost spanning tree for a weighted undirected graph. Here is a small example graph to explain the difference between the two algorithms:




**Prim's Algorithm:** Prim's algorithm starts with a single vertex and grows the minimum spanning tree by adding the edge with the minimum weight that connects the tree to a vertex that is not yet in the tree.

- Choose an arbitrary vertex, say A, and add it to the minimum spanning tree (MST). The MST now contains only one vertex: A.

- Find the minimum weight edge that connects any vertex in the MST to a vertex that is not yet in the MST. In this case, the minimum weight edge is the edge between A and B, with weight 2. Add this edge to the MST.

- Find the minimum weight edge that connects any vertex in the MST to a vertex that is not yet in the MST. In this case, the minimum weight edge is the edge between B and D, with weight 1. Add this edge to the MST.

- Find the minimum weight edge that connects any vertex in the MST to a vertex that is not yet in the MST. In this case, the minimum weight edge is the edge between D and E, with weight 4. Add this edge to the MST.

- Find the minimum weight edge that connects any vertex in the MST to a vertex that is not yet in the MST. In this case, the minimum weight edge is the edge between B and C, with weight 3. Add this edge to the MST.

The final MST obtained by Prim's algorithm is:

```
   2      3
A ------ B ------ C
      |
      |5
      |
      D ------ E
          4
```

**Kruskal's Algorithm:** Kruskal's algorithm starts by sorting all the edges in the graph by weight and then iteratively adding the edges with the minimum weight that do not create a cycle in the MST.

- Sort all the edges in the graph by weight: (B,D), (A,B), (B,C), (D,E), (C,E), (A,D).
- Pick the edge with the minimum weight, (B,D), and add it to the MST.
- Pick the next edge with the minimum weight, (A,B), and add it to the MST.
- Pick the next edge with the minimum weight, (B,C), and add it to the MST.
- Pick the next edge with the minimum weight, (D,E), and add it to the MST.
- Pick the next edge with the minimum weight, (C,E), and add it to the MST.

The final MST obtained by Kruskal's algorithm is:

```
      3
A ------ B ------ C
      |
      |5
      |
      D ------ E
```

### 6.3 *Which algorithmic strategy is used in Prim's algorithm. Explain that algorithmic strategy in brief.*

Prim's algorithm uses a greedy algorithmic strategy to find the minimum cost spanning tree in a weighted undirected graph. A greedy algorithm always makes the locally optimal choice at each step with the hope of finding a global optimum.

The algorithm starts with a single vertex and grows the minimum spanning tree by adding the edge with the minimum weight that connects the tree to a vertex that is not yet in the tree. The steps for Prim's algorithm are as follows:

- Choose an arbitrary vertex, say A, and add it to the minimum spanning tree (MST). The MST now contains only one vertex: A.
- Find the minimum weight edge that connects any vertex in the MST to a vertex that is not yet in the MST. Add this edge to the MST and the vertex to the set of vertices in the MST.
- Repeat step 2 until all vertices are in the MST.

The key idea behind the algorithm is that at each step, we add the edge with the minimum weight that connects the MST to a vertex that is not yet in the MST. This ensures that the MST is always growing by the minimum amount possible at each step.

Prim's algorithm is a computationally efficient way to find the minimum cost spanning tree for a weighted undirected graph, especially when the graph is dense (i.e., has many edges). The time complexity of Prim's algorithm is O(ElogV) using a binary heap or Fibonacci heap to maintain the set of vertices that are not yet in the MST, where E is the number of edges and V is the number of vertices in the graph.