Design and Analysis of Algorithms

CSE 5311

**Project: Minimum Spanning Tree Algorithms**

Name: Maitreyee Avinash Damale

ID: 1001878520

1. **Introduction**

An Algorithm is a sequence of steps to solve a problem. Design and Analysis of Algorithm is important for designing algorithm to solve different types of problems in the branch of computer science and information technology.

The Course CSE 5311 teaches us to implement these various Algorithms.

This Project is for implementing all the given Algorithms for finding minimum spanning tree of graph. They are as follows:

* Kruskal’s Algorithm
* Prim’s Algorithm

The following report will show the details of each algorithm, their comparisons along with its implementation details and user interface.

Kruskal’s Algorithm:

Kruskal's algorithm finds a minimum spanning forest of an undirected edge-weighted graph. If the graph is connected, it finds a minimum spanning tree. (A minimum spanning tree of a connected graph is a subset of the edges that forms a tree that includes every vertex, where the sum of the weights of all the edges in the tree is minimized. For a disconnected graph, a minimum spanning forest is composed of a minimum spanning tree for each connected component.) It is a greedy algorithm in graph theory as in each step it adds the next lowest-weight edge that will not form a cycle to the minimum spanning forest.

Prim’s Algorithm:

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. The algorithm operates by building this tree one vertex at a time, from an arbitrary starting vertex, at each step adding the cheapest possible connection from the tree to another vertex.

1. **Implementation Details**

The code is written in Javascript programming language and HTML for GUI.

The IDE used for editing is VSCode.

The code consists of 2 functions for 2 algorithms which are Prim’s and Kruskal’s Algorithm and another function is for taking user input and generating random matrix.

**PRIMS ALGORITHM:**

Here are steps we use to perform the Algorithm

1. We ask user to enter the number of nodes.
2. We call function prim getPrimsMsts ().
3. We note the start time at the start of function.
4. We create the randomize matrix of nodes\*nodes
5. We create three arrays visited, parent, weights.
6. weight keeps the track of current key of vertices, Parent keeps the track of parent of vertices and visited keeps the track of all the visited nodes.
7. Find the key with minimum value and not yet visited
8. Find all the adjacent node and update the weight of the vertices
9. Repeat till all the nodes are visited

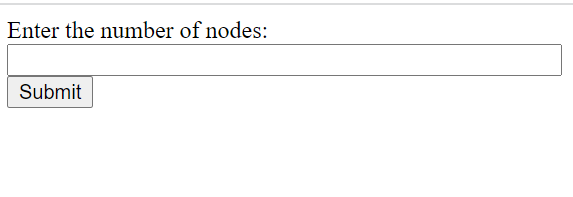
**Kruskal’s Algorithm:**

Here are steps we use to perform the Algorithm

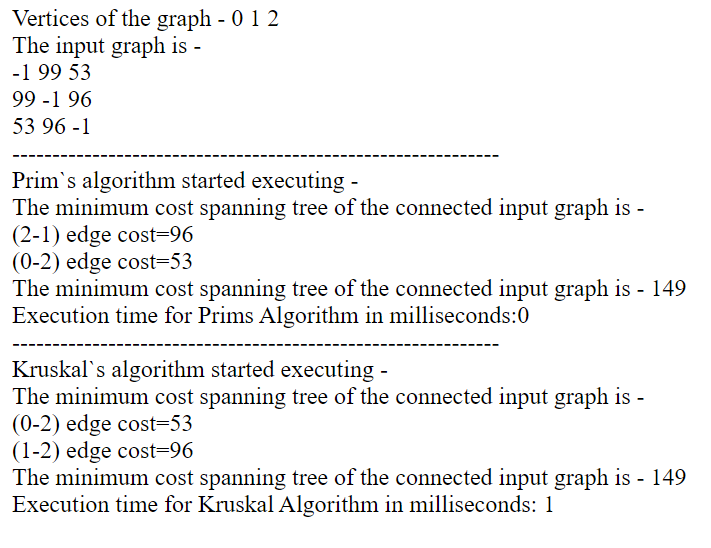
1. Then we ask user to enter the number of nodes.
2. We call function getKruskalsMst ().
3. We note the start time at the start of function.
4. We create the randomize matrix of nodes\*nodes and weight.
5. We sort the weight in ascending order.
6. We keep the track of parent and rank.
7. We keep moving forward and keep checking that no cycles are created
8. **UI:**

For UI implementation, we use the simple html form which takes the input from user. An input contains number of nodes i.e. unput size for algorithm.

Below is the screenshot for input:



Result Screen:



1. **Comparison**

**Time Comparison for 10 nodes**

|  |  |
| --- | --- |
| **Algorithm** | **Time(milliseconds)** |
| **Prims** | **2** |
| **Kruskal** | **1** |

**Run time comparison between Prims and Kruskal:**

1. **Time Complexity:**

Best, worst, and average cases of a given [algorithm](https://en.wikipedia.org/wiki/Algorithm) express what the [resource](https://en.wikipedia.org/wiki/Resource_(computer_science)) usage is at least, at most and on average, respectively. Usually, the resource being considered is running time, i.e. [time complexity](https://en.wikipedia.org/wiki/Time_complexity), but could also be memory or other resource. Best case is the function which performs the minimum number of steps on input data of n elements. Worst case is the function which performs the maximum number of steps on input data of size n. Average case is the function which performs an average number of steps on input data of n elements.

Big O is used to denote either the time complexity of an algorithm or how much space it takes up. This blog focuses mainly on the time complexity part of this notation. The way people can calculate this is by identifying the worst case for the targeted algorithm and formulating a function of its performance given an n amount of elements.

Time Complexity for Prim’s algorithm:

Prim’s algorithm has a time complexity of O(V2), V being the number of vertices and can be improved up to O(E log V) using Fibonacci heaps.

Time Complexity for Kruskal’s algorithm:

Kruskal’s algorithm’s time complexity is O(E log V), V being the number of vertices.