

**DATA STRUCTURES AND ALGORITHM – 3**

**CLUSTERING USING KRUSKAL'S AND PRIM'S ALGORITHM**

TEAM MEMBERS

BATCH – 13

KARTHIKEYAN M - CB.EN.U4AIE20029

KARTHIK RAJA B - CB.EN.U4AIE20028

NALIN M R - CB.EN.U4AIE20041

PRAGADISHWARAN K - CB.EN.U4AIE20051

RISHEKESAN S V- CB.EN.U4AIE20058

**ACKNOWLEGEMENT**

We would like to thank to our professor B. Premjith who gave us his valuable suggestions and ideas when we needed them to work on our project

“**CLUSTERING USING KRUSKAL'S AND PRIM'S ALGORITHM**”

We are also grateful to our university, Amrita Vishwa Vidyapeetham for giving us the opportunity to work on this project. Last but not the least we would like to thank our group members, Nalin MR, Karthikeyan.M, Karthik Raja.B, Pragadishwaran K without their effort and co-operation, we would not have been able to complete the project within the prescribed time.

**ABSTRACT**

In this project our main aim is to form a MST (minimum spanning tree) for any given graph. We should mainly use two algorithms, namely Kruskal’s algorithm and Prim’s algorithm. A minimum spanning tree (MST) or minimum weight spanning tree for a weighted, connected, undirected graph is a spanning tree with a weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

**TABLE OF CONTENTS**

**S. no TITLE Page No.**

1 INTRODUCTION 5

2 KRUSKAL’S ALGORITHM 7

3 PRIM’S ALGORITM 11

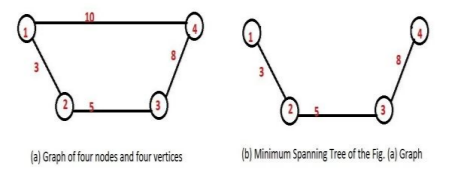
4 CLUSTERING USING PRIM’S AND KRUSKAL’S 15

5 CODE IMPLEMENTATION 17

6 CONCLUSION 21

**INTRODUCTION**

A Minimum Spanning Tree of a weighted graph is a spanning tree in which the sum of the weight of all its edges is a minimum of all such possible spanning tree of the graph. Minimum spanning Tree must be finding from the Graph. A collection of vertices and edges makes a graph, and each edge connects a pair of vertices [1, 2, 3, 4,].



There are two types of graph, Directed graph and undirected graph. A directed graph is graph in which a set of vertices are connected together, where all the edges are directed from one vertex to another. A directed graph is also known as digraph or a directed network. In contrast, a graph where the edges are bidirectional is called an undirected graph. In the directed graph edges have a direction associated with them. An undirected graph is one in which edges have no orientation. The edge (a, b) is identical to the edge (b, a).The maximum number of edges in an undirected graph without a self-loop is n (n - 1)/2 .

**APPLICATION OF MST**

Applications of MST are used in the design of computer and communication networks, telephone networks, links road network, islands connection, pipeline network, electrical circuits, utility circuit printing, obtaining an independent set of circuit equations for an electrical network, etc.

It offers a method of solution to other problems to which it applies less directly, such as network reliability, clustering and classification problems. Used to find the approximation solution for the NP hard problems.

**MST CLASSICAL ALGORITHM**

There are various classical algorithms available which describe below. Kruskal’s , Prim's algorithm is a greedy algorithm which used to find a minimum spanning tree for a connected weighted undirected graph. This means when the total weight of all the edges is minimized in the tree, at that time it finds a subset of the edges which forms a tree which includes every vertex.

**Basic Terms:**

**Spanning tree -** A spanning tree is the subgraph of an undirected

Connected graph.

**Minimum Spanning tree -** Minimum spanning tree can be defined as the spanning tree in which the sum of the weights of the edge is minimum. The weight of the spanning tree is the sum of the weights given to the edges of spanning tree.

In case of spanning tree it should it be a connected graph, undirected and it never contains cycle.

**Kruskal's Algorithm**

Kruskal algorithm: This algorithm first appeared in Proceedings of the American Mathematical Society during 1956, and was written by Joseph Kruskal. In Kruskal’s algorithm all edges are shorted in non decreasing order and selected the lowest edges first for becoming a minimum spanning tree. Using simple data structure Kruskal's algorithm complexity is O (E log E) time, or equivalently, O (E log V) time. Where E is the number of edges in the graph and V is the number of vertices .

Kruskal's Algorithm is used to find the minimum spanning tree for a connected weighted graph. The main target of the algorithm is to find the subset of edges by using which we can traverse every vertex of the graph. It follows the greedy approach that finds an optimum solution at every stage instead of focusing on a global optimum

The algorithm that takes a graph as input and finds the subset of the edges of that graph which

1. Form a tree that includes every vertex

2. Has the minimum sum of weights among all the trees that can be formed from the graph

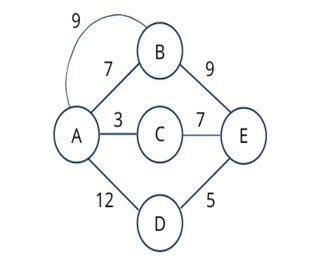
**Kruskal's algorithm steps:**

In Kruskal's algorithm, we start from edges with the lowest weight and keep adding the edges until the goal is reached.

The steps to implement Kruskal's algorithm are listed as follows -

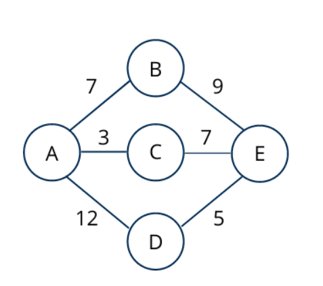
1. First, sort all the edges from low weight to high.
2. Now, take the edge with the lowest weight and add it to the spanning tree. If the edge to be added creates a cycle, then reject the edge.
3. Continue to add the edges until we reach all vertices, and a minimum spanning tree is created.

## Let’s See an Example to understand Kruskal’s algorithm



### Step 1 – Remove all loops and Parallel Edges

After removing parallel edge from the graph, graph will look like below.

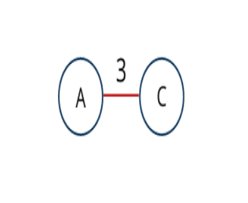


### Step 2 – Sort all the edges in non-decreasing order of their weight

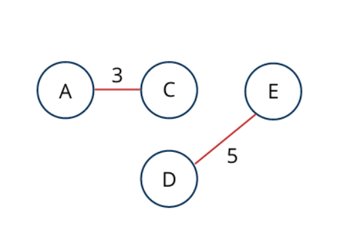
AC – 3, DE – 5, AB – 7, BE – 9, CE – 7, AD – 12

### Step 3 – Pick the least weightage edge and include this edge

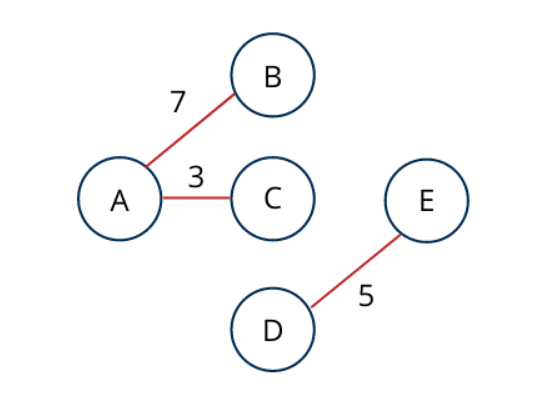
We will add edges to the graph beginning from the one which has the least weight. As well as also we will keep checking this edge should not form any cycle. In cycle will create by adding one edge or any other property that violates the spanning-tree property then we will not include the edge in the graph.



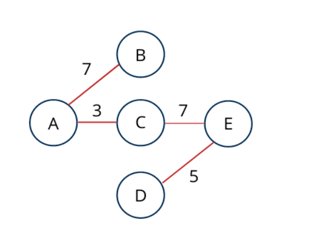
Next we will select Edge DE which have least weight 5 and not violating spanning tree properties.



Next we will select Edge AB which have least weight 7 and not violating spanning tree properties.



Next we will select Edge CE which have least weight 7 and not violating spanning tree properties.



So, Create Minimum spanning tree using Kruskal’s algorithm will look like above.

**Prim’s Algorithm**

The algorithm was developed in 1930 by Jarnik and later rediscovered by computer scientist Robert C. Prim in 1957 and then again rediscovered by Edsger Dijkstra in 1959. Therefore it is also known as the DJP (Dijkstra-Jarnik Problem) algorithm, the Jarnik algorithm, or the Prim–Jarnik algorithm. Using a simple binary heap data structure complexity is O(|E| log |V|) where |E| is the number of edges and |V| is the number of vertices.

* 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.
* It is used for finding the Minimum Spanning Tree (MST) of a given graph and graph must be weighted, connected and undirected.

**Steps involved in Prim’s Algorithm**

Step 1 :

* Randomly choose any vertex.
* The vertex connecting to the edge having least weight is usually selected.

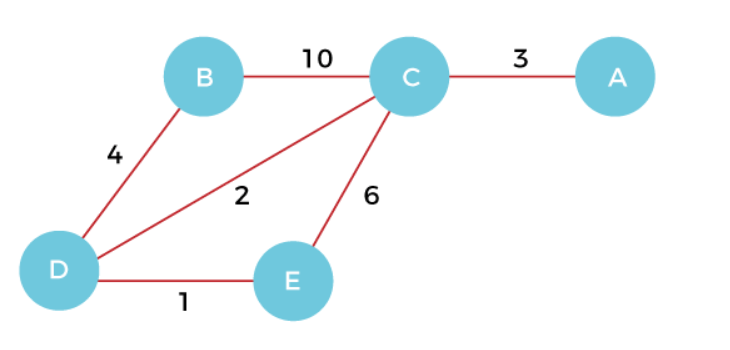
Step 2 : Find all the edges that connect the tree to new vertices.

* Find the least weight edge among those edges and include it in the existing tree.
* If including that edge creates a cycle, then reject that edge and look for the next least weight edge

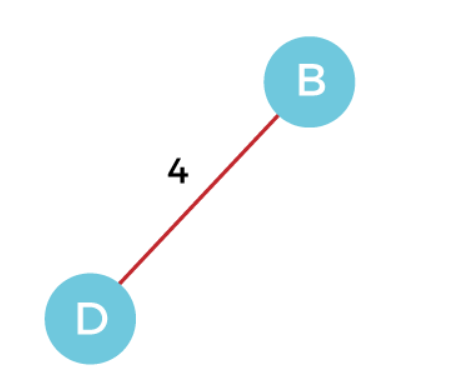
Step 3 : Keep repeating step-02 until all the vertices are included and Minimum Spanning Tree (MST) is obtained.

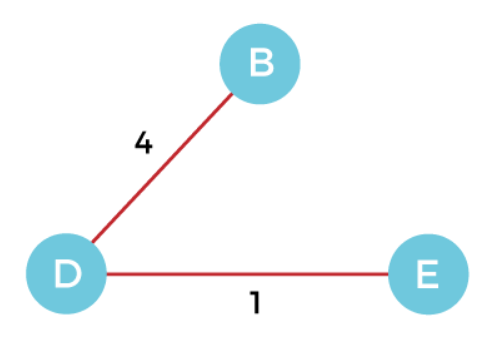
Now, let's see the working of prim's algorithm using an example. It will be easier to understand the prim's algorithm using an example.

**Suppose, a weighted graph is -**

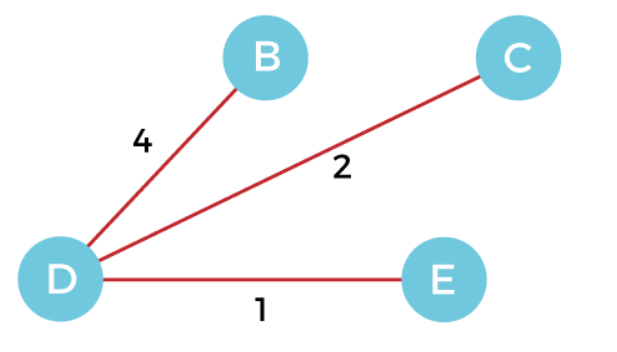


**Step 1 -** First, we have to choose a vertex from the above graph. Let's choose B.

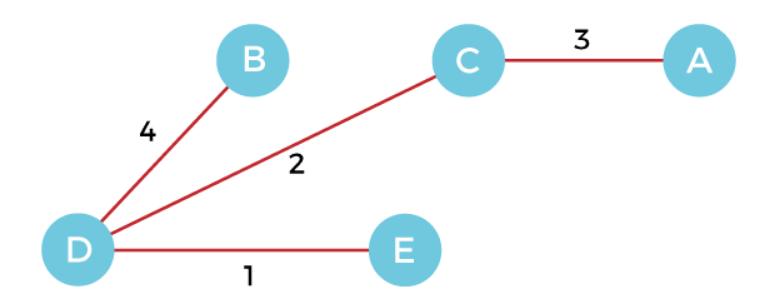
**Step 2 -** Now, we have to choose and add the shortest edge from vertex B. There are two edges from vertex B that are B to C with weight 10 and edge B to D with weight 4. Among the edges, the edge BD has the minimum weight. So, add it to the MST.

**Step 3 -** Now, again, choose the edge with the minimum weight among all the other edges. In this case, the edges DE and CD are such edges. Add them to MST and explore the adjacent of C, i.e., E and A. So, select the edge DE and add it to the MST.

**Step 4 -** Now, select the edge CD, and add it to the MST.

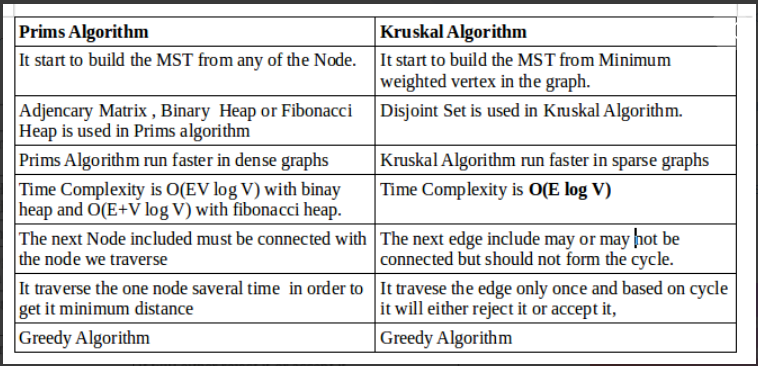


**Step 5 -** Now, choose the edge CA. Here, we cannot select the edge CE as it would create a cycle to the graph. So, choose the edge CA and add it to the MST.



So, the graph produced in step 5 is the minimum spanning tree of the given graph.  
Cost of MST = 4 + 2 + 1 + 3 = 10 units.

**Difference between Prim’s and Kruskal Algorithm:**



**Overview of Clustering:**

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

Clustering has a large no. of applications spread across various domains. Some of the most popular applications of clustering are:

* Recommendation engines
* Market segmentation
* Social network analysis
* Search result grouping

Broadly speaking, clustering can be divided into two subgroups :

* **Hard Clustering:** In hard clustering, each data point either belongs to a cluster completely or not. For example, in the above example each customer is put into one group out of the 10 groups.
* **Soft Clustering**: In soft clustering, instead of putting each data point into a separate cluster, a probability or likelihood of that data point to be in those clusters is assigned. For example, from the above scenario each costumer is assigned a probability to be in either of 10 clusters of the retail store.

**Clustering using Prim's and Kruskal's algorithm**

**Union -Find Data Structures**

A union–find data structure or merge–find set, is a data structure that stores a collection of disjoint (non-overlapping) sets.

To work efficiently Kruskal’s algorithm requires a data structure to store the collection of connected components of a graph and merge then when necessary

More precisely, the data structure has to support the following operations:

find (x) : returns a label to which set x belongs

make\_set **:** function that sets the value of a given vertex to “None” in the dictionary “parent” (it creates a singleton)

union(x, y) : merge the sets containing x and y

Kruskal’s Algorithm with Union-Find

* To avoid searching for an edge of minimum weight, sort the edges in the beginning of the algorithm
* If T is the current set of selected edges, the data structure contains the collection of connected components of
* To check whether edge (v,w) forms a cycle just check if find(v) = find (w)
* Follow the sorted list of edges. Every time for edge (v,w) if it forms a cycle, remove it from the list. If it does not, merge the sets containing v and w, and remove the edge from the list .

**Advantages of Kruskal’s Algorithm:**

1) Easy to understand

2) Give good result for large number of vertices and edges.

**Disadvantages of Kruskal’s Algorithm:**

1) Difficulty of checking whether arcs form cycles makes it slow and hard to program.

2) Same weight may increase the complexity

**Priority Queue And Heap :**

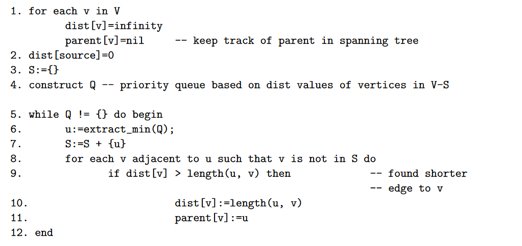
A priority queue is a data structure that supports insertion and delete-  
maximum operations (the latter removes the maximum element from the queue). Sometimes we also want to allow an update operation that changes the value of an element in the queue or a delete operation.

A heap is a binary tree such that each node is greater than or equal to  
its children. (A min-heap would be just the opposite). It is also a balanced binary tree, so the leaves are all on the last one or two levels of the tree.

**Prim's Algorithm implementation :**

Prim’s MST Algorithm uses a distance function to specify the closeness of data objects to establish the weight between them by choosing an arbitrary point to the next adjacent point of minimum weight.

For clustering, an edge inconsistency measure is defined to identify an inconsistent edge to be removed to partition the whole dataset into clusters



**Advantages of Prim’s Algorithm:**

1) Easy to understand.

2) Root node is selected so clear about the starting node.

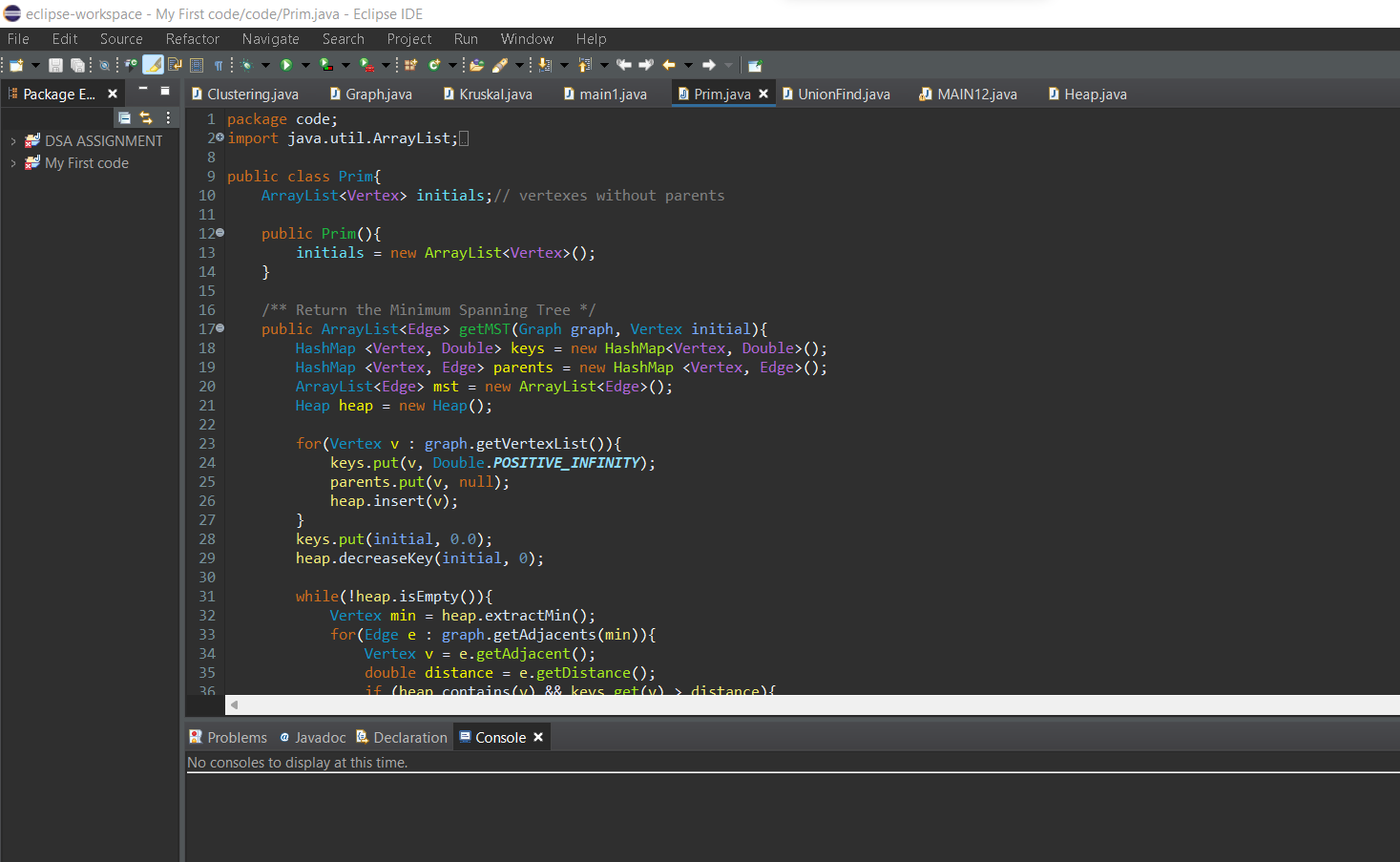
**Disadvantages of Prim’s Algorithm:**

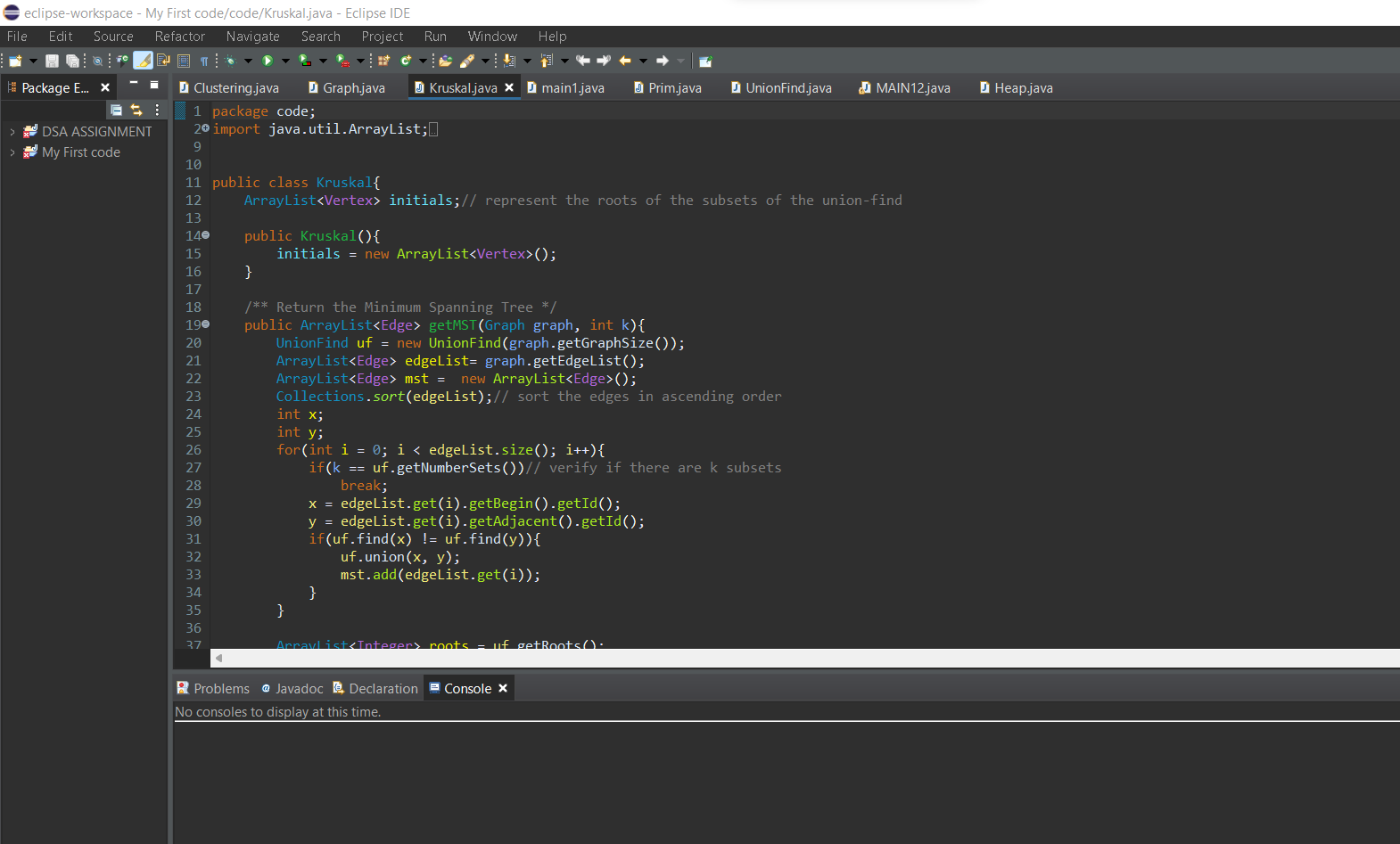
1) Time taken to check for smallest weight arc makes it slow for large numbers of nodes.

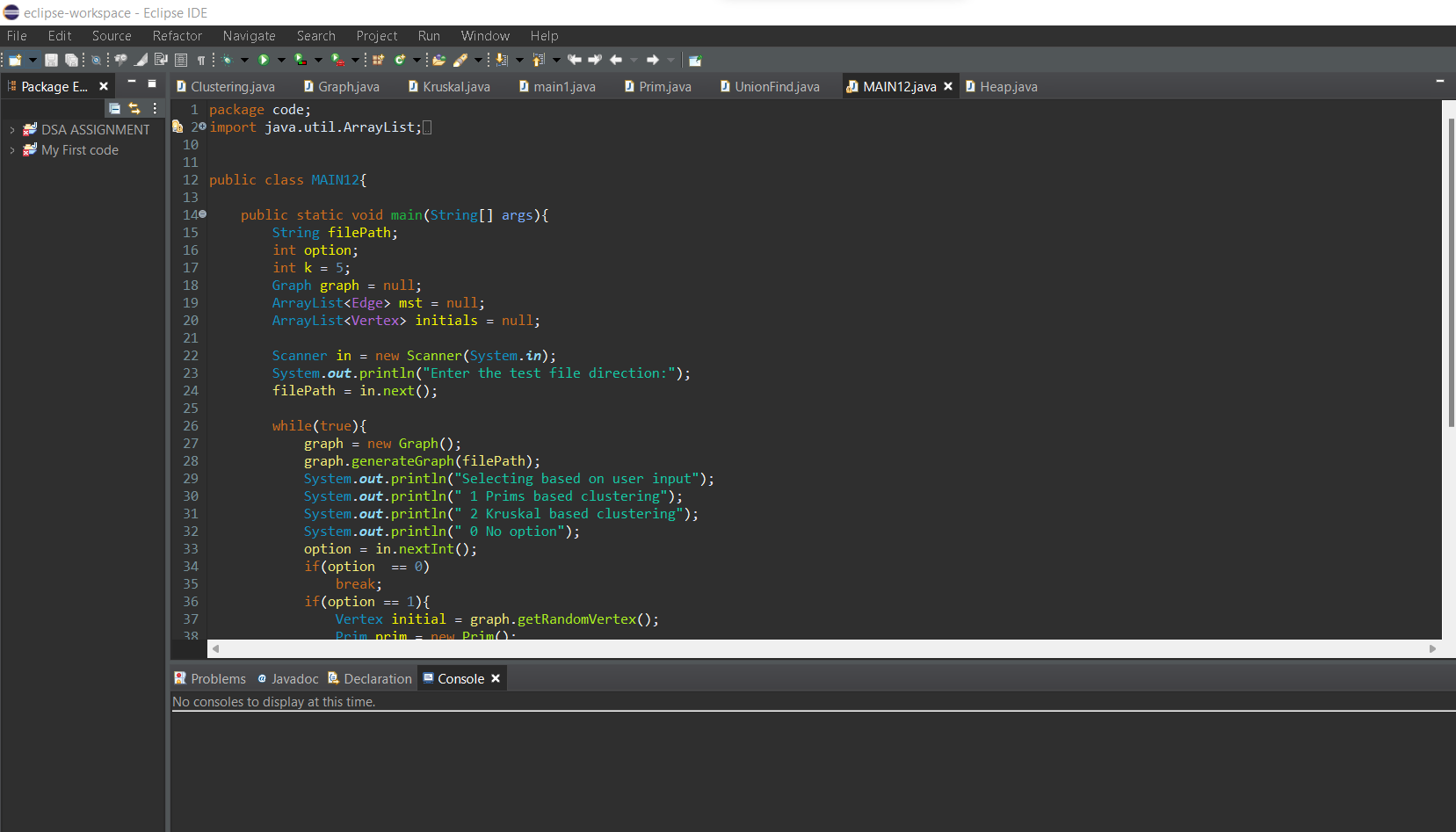
2) Difficult to program, though it can be programmed in matrix form

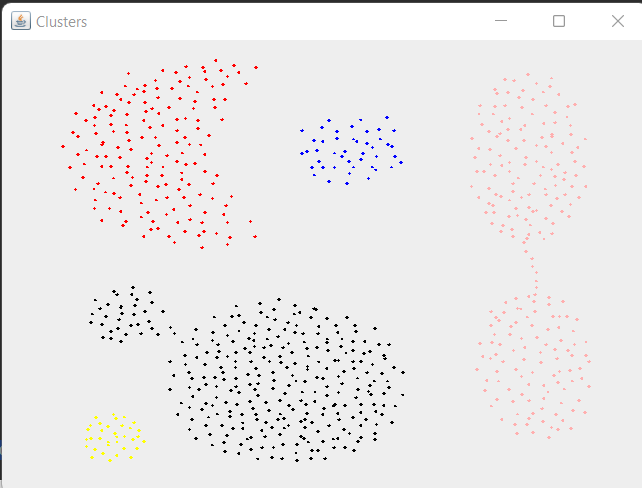
3) Same weight may increase the complexity when one of the weights is eliminated in a cycle

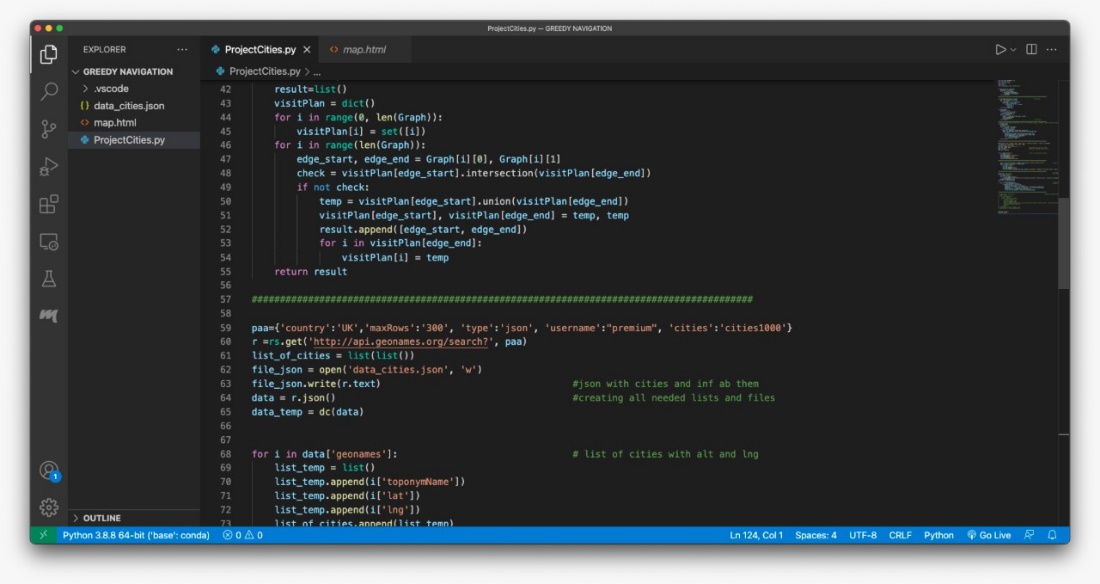
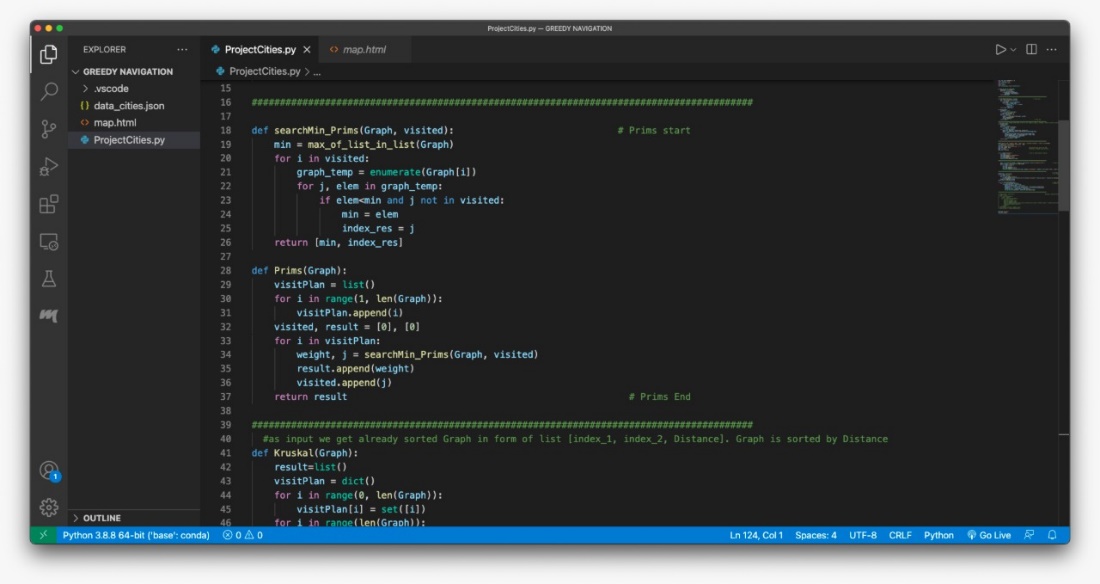
**IMPLEMENTATION**

**PRIMS FUNCTION:**

**KRUSKAL FUNCTION:**

**MAIN CODE:**

**OUTPUT :**

**PYTHON IMPLEMENTATION :**

**OUTPUT:**



**CONCLUSION**

Through this project our team were clearly able to understand concept of clustering a graph and also we gained thorough knowledge on kruskal’s and prim’s algorithm in terms of MST. We were also able to become a bit more familiar with the libraries and syntax of phython and java.

By taking this module, I learnt to read more and think more, and to draw information from a wide range of sources, such as lecture notes, literature, journals, and websites, to support my essential learning. Therefore, we can go beyond the superficial knowledge of clustering and MST.

**REFERENCES**

1. [**https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/**](https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/)
2. [**https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/**](https://www.geeksforgeeks.org/prims-minimum-spanning-tree-mst-greedy-algo-5/)
3. [**https://www.javatpoint.com/prim-algorithm**](https://www.javatpoint.com/prim-algorithm)