Avrupa Bilim ve Teknoloji Dergisi
Sayı 14, S. 70-76, Aralık 2018

Araştırma Makalesi

www.ejosat.com ISSN:2148-2683

European Journal of Science and Technology
No. 14, pp. 70-76, December 2018

Research Article

Negligence Minimum Spanning Tree Algorithm

Jumana H.S. Alkhalissi[1], Ayla Saylı[1*]

[1] Yildiz Technical University, Faculty of Chemical and Metallurgical Engineering, Department of Mathematical Engineering, Istanbul, Turkey

(First received 30 January 2018 and in final form 23 October 2018)

(DOI: 10.31590/ejosat.386716)

Abstract

In the life, we always aim to do anything with the less cost considering time and distance. In graph theory, finding a minimum weight (cost or path) is a well-known problem. A minimum spanning tree is one of the methods brought for this purpose. In this work, we consider a negligence algorithm to find a minimum spanning tree in another way. We report a comparison between our algorithm and Kruskal's MST algorithm. We implemented some examples of the graphs to get the results in less time and more effectively.

Keywords: Minimum Spanning Tree, Kruskal's, Prim's and Reverse Delete Algorithm, Running Time.

1. Introduction

A special type of tree that connected between edges to minimize the lengths or "weights can be called as a Minimum Spanning Tree (MST) graph. It is useful for many applications like taxonomy, cluster analysis, data dissemination, routing and coordinator election etc. The aim of the MST construction is used to minimize the weight (cost) in objects of a graph. For instance, a salesman who needs to travel between different places with less distance or less time. There are different types of algorithms for the MST such as Kruskal's, Prim's, Reverse Delete, Borøuvka's algorithms. Where V. Loncar and others studied serial variants of Prim's and Kruskal's algorithms and presented their parallelization targeting message passing parallel machines with distributed memory [1]. S. Dutta and others developed a new GIS tool using Prim's algorithm to construct the MST of a connected, undirected and weighted road network [2]. While S. Mohanram and T. D. Sudhakar used the reverse delete algorithm to find the optimal path of a power flow for a given network [3]. There are a lot of applications of the MST studied by different researchers depending on the specific field that he/she wants to employ them. We consider some of these applications as the following list [4]:

• In approximation algorithms, the MST represents an essential part of these algorithms as Steiner tree and the Christofede's algorithm for the traveling salesman.

• In image processing, the arrangement of the image cells can be formed in the MST graph.

• In single-linkage clustering, the MST represents the basis for single-linkage clustering. At each step, two clusters are separated by the shortest distance which is combined.

• In road or telephone networks, places/houses are in the need of a connection path with the minimum length of road/wire possible.

In this work, we shed light on the MST and some of MSTs algorithms and then we propose our new algorithm called as "A Negligence Minimum Spanning Tree Algorithm (NMST)". In addition, we implement one of a classical algorithm to find the MST of a graph which is Kruskal's algorithm. Our goal is to develop this algorithm to compare the performance of our NMST algorithm with minimized costs. The following section is given to explain what the minimum spanning tree is and how it can be constructed.

In graph theory, a connected graph G with n vertices and $n-1$ edges, and not having any loop among the vertices that can be called as a tree. A tree consisting of all vertices of the graph is called a spanning tree. Thus the MST of a graph in Figure 1 is a spanning tree containing all vertices of the graph with the minimum sum of the weights of all edges [2].

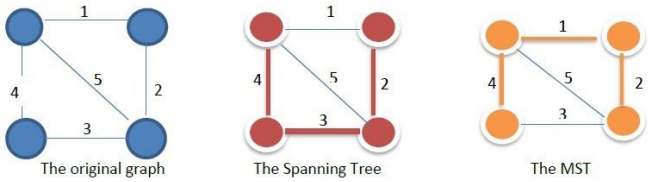


Figure 1. Minimum Spanning Tree

2. Kruskal's, Prim's and Reverse Delete Algorithms

In this section, two most popular algorithms for MSTs will be given in detail, which are Kruskal's and Prim's algorithms.

[1] Corresponding Author: Yildiz Technical University, Faculty of Chemical and Metallurgical Engineering, Department of Mathematical Engineering, Istanbul, Turkey, sayli@yildiz.edu.tr

2.1. Kruskal's Algorithm

Kruskal discovered this algorithm in 1956 [5]. The procedures of this algorithm start with a sorted list of edges and add each edge to the tree if it does not form a cycle. Since we examine the edges in order from lightest "in value of lenghts or weights"to heaviest, any edge we examine is safe if and only if its endpoints are in different components of the partial MST. To be clear about its procedures, an example is given in Figure 2.

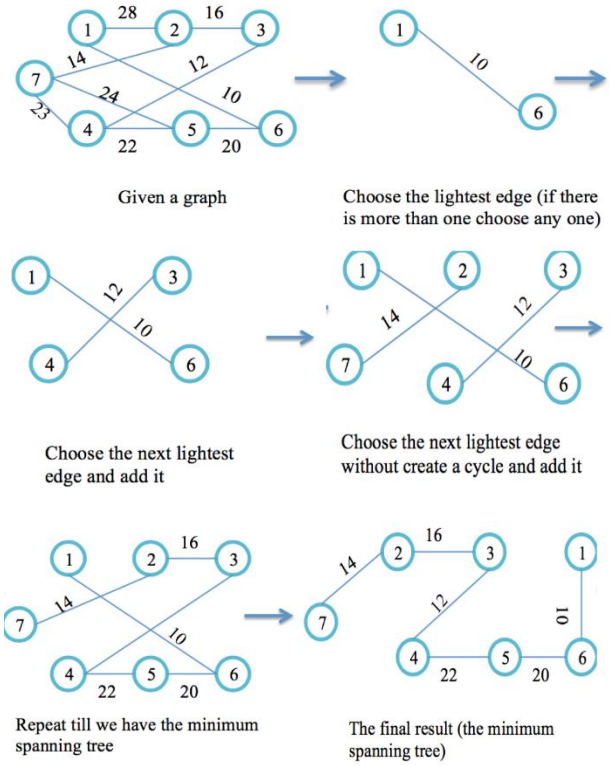


Figure 2. Procedures of Kruskal's Algorithm

2.2. Prim's Algorithm

Rebort Prim developed Prim's algorithm in 1957 [5]. The procedures of this algorithm start from an arbitrary vertex in the graph as a single partial MST, at each step add an edge of the light (lowest) weight connecting the vertex nearest to but not already in the current partial minimum spanning tree with a condition that the properties of a tree should be passing through all vertices (n) of the graph with (n−1) edges. Figure 3 represents the procedures of Prim's algorithm.

2.3. Reverse Delete Algorithm

This is known as a Greedy Algorithm. It is the reverse of Kruskal's algorithm to find a minimum spanning tree.

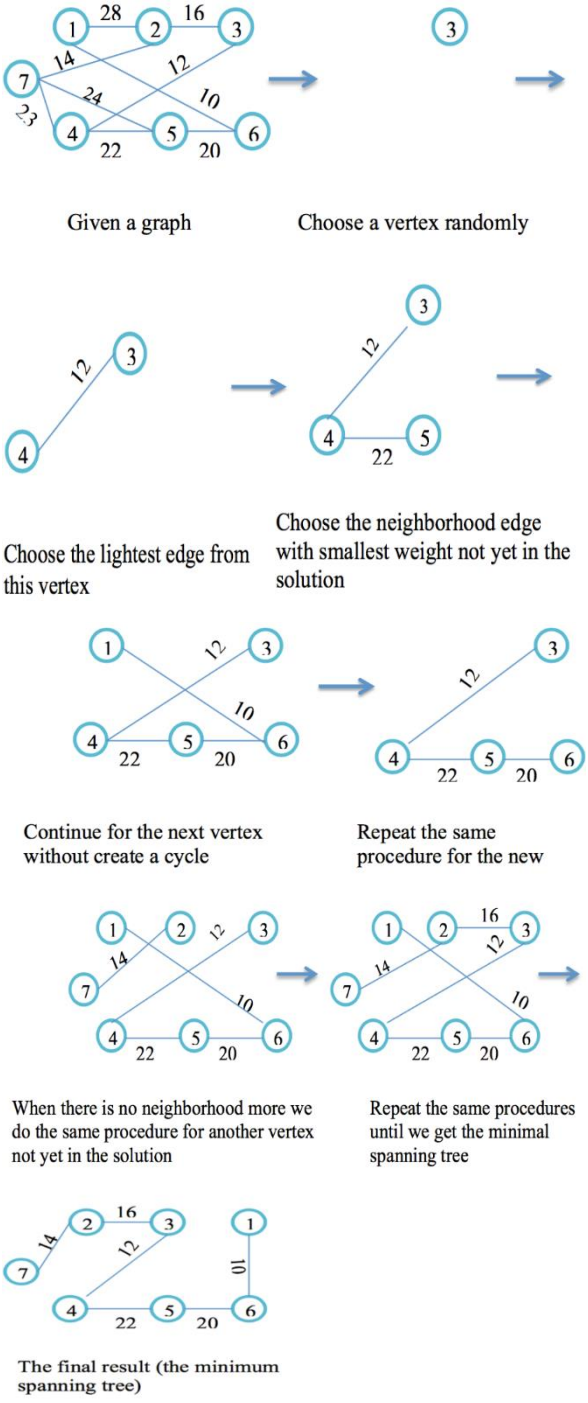


Figure 3. Procedures of Prim's Algorithm

The algorithm works as follows:

Step 1: Start with graph G, which contains a list of edges E.

Step 2: Go through E in decreasing order of edge weights.

Step 3: Check if deleting current edge will further disconnect graph.

Step 5: If G is not further disconnected, delete the edge.

We can see its procedures of the same example in Figure (4).

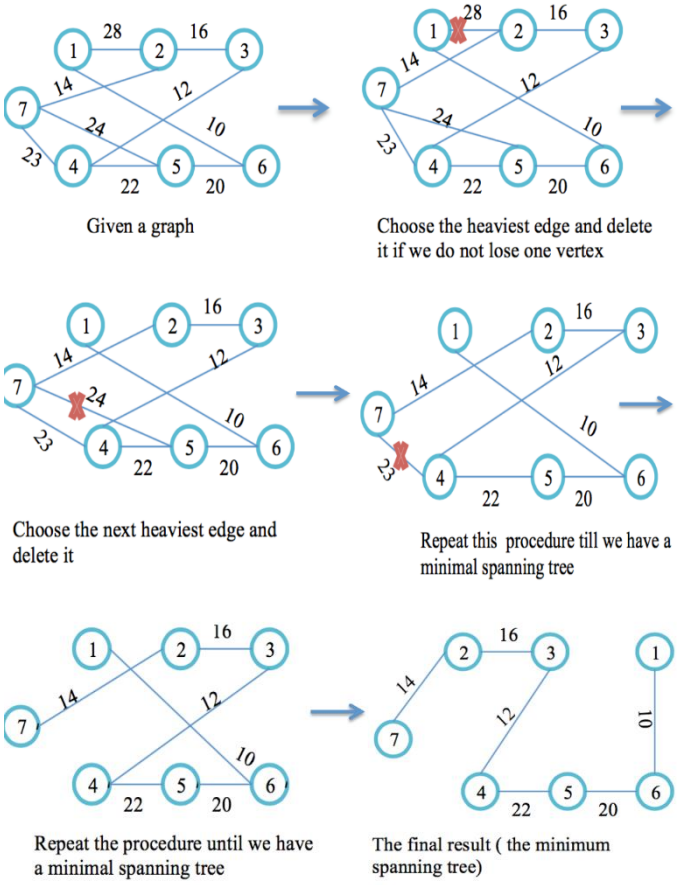


Figure 4. Procedures of Reverse Delete Algorithm

Now, we will introduce our NMST algorithm in Section 4.

3. Negligence Minimum Spanning Tree Algorithm

The procedures of the NMST algorithm can be described as follows:

Step 1: Sort of max weight to minimum weight (ordered the graph decreasingly).

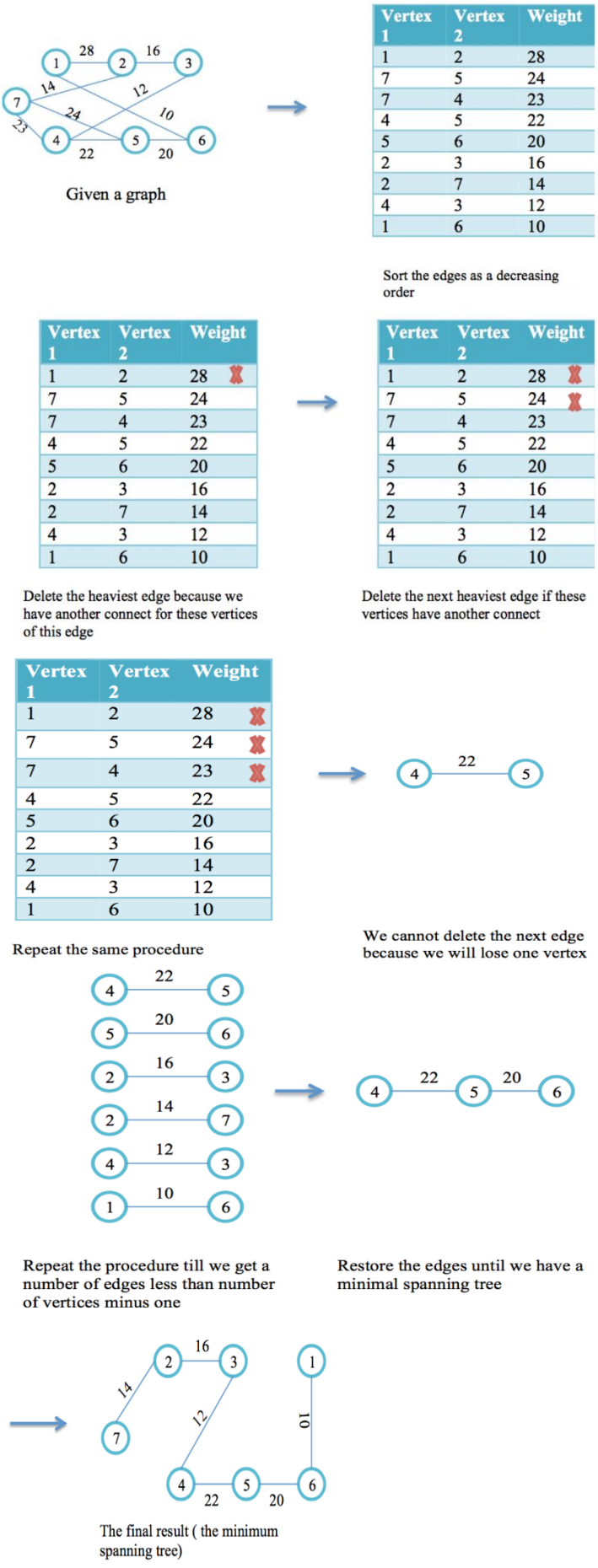
Step 2: Make an array with the number of vertices to record the last visit to each vertex.

Step 3: Remove the heaviest weighted edge if this edge makes no loss of one of the vertices.

Step 4: Repeat the removing producer till reaching the condition that will be sure the number of edges less than of vertices minus one.

Step 5: Resort the reminder edges.

Step 6: Build the MST.



Vertex 1	Vertex 2	Weight
1	2	28
7	5	24
7	4	23
4	5	22
5	6	20
2	3	16
2	7	14
4	3	12
1	6	10

Figure 5. Procedures of NMST Algorithm

Figure 5 shows the NMST algorithm procedures for the same given example. In mathematical terms, let G represents the given

graph with n vertices and E edges. Ordered E in decreasing order, put the vertices in a sequence set. Delete the edge e which is the edge with the biggest weight if this edge does not affect the number of vertices. Continue of deleting the next bigger weight till we satisfy the condition number of E equal to (n − 1). The reminder edges connected to be minimum spanning tree.

The difference between the reverse delete algorithm and the NMST is that the reverse delete algorithm starts with the original graph and eliminates the expensive weights that did not make the graph disconnected, while the NMST starts with the sort of the weight by descending order, then creates another graph which contains the result of edges with the less weight. Also, the NMST has two sorts: one at the beginning and the other at the end is resorting the result to generate the MST which is not like the reverse delete algorithm.

In the other hand, the similarities between the Kruskal's algorithm and the NMST are sorting the edges, choosing the wanted edges and building the MST. While the difference between the Kruskal's algorithm and the NMST is the way of sorting. Kruskal's algorithm is using sorting increasingly and the NMST algorithm is using sorting decreasingly and also the NMST algorithm has two sortings one at the beginning and one at the end which is not like Kruskal's algorithm.

4. MATLAB Application of Kruskal' and NMST Algorithms

In MATLAB (version R2015b), we have implemented our application for the algorithms and its interface is given in Figure 6-10. From the interface, the first procedure is to define the graph, then select one of the algorithms (Kruskal's or NMST), after that the procedures of the chosen algorithms are executed up to the final MST is reached.

In order to show the procedures, we implemented two more examples using Kruskal's and NMST algorithms and executed them in our application to estimate the execution times and the final MSTs.

Example 4.1: At the beginning, we implemented the Kruskal's algorithm on the graph with 7 vertices and 9 edges. Figure 6 shows how we select the data for Example 4.1, which is inserted before and click draw to appear the original graph.

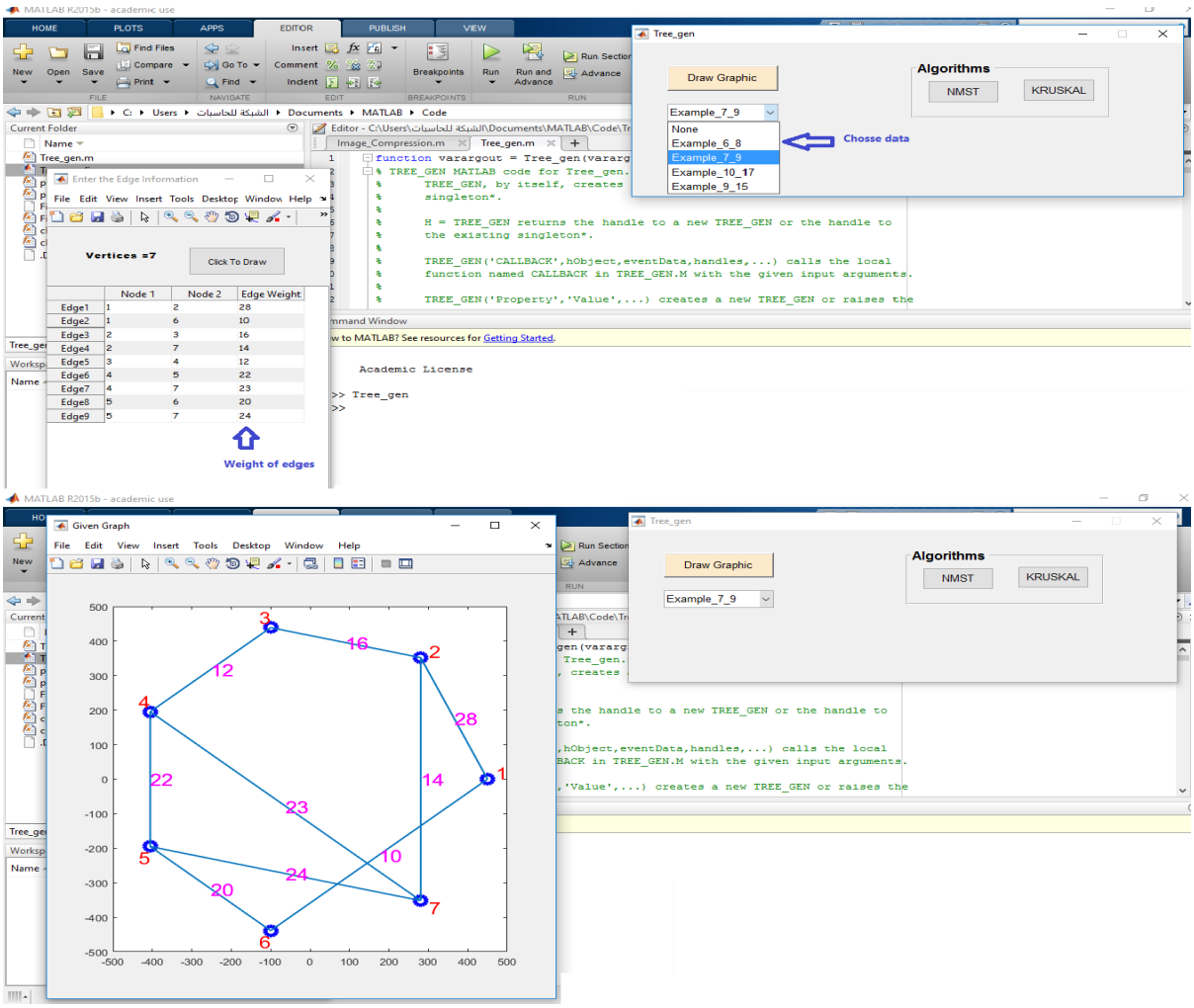


Figure 6. Procedure 1 for Example 4.1's Implementation

In Figure 7, we choose Kruskal's algorithm to find the MST and shows the final MST.

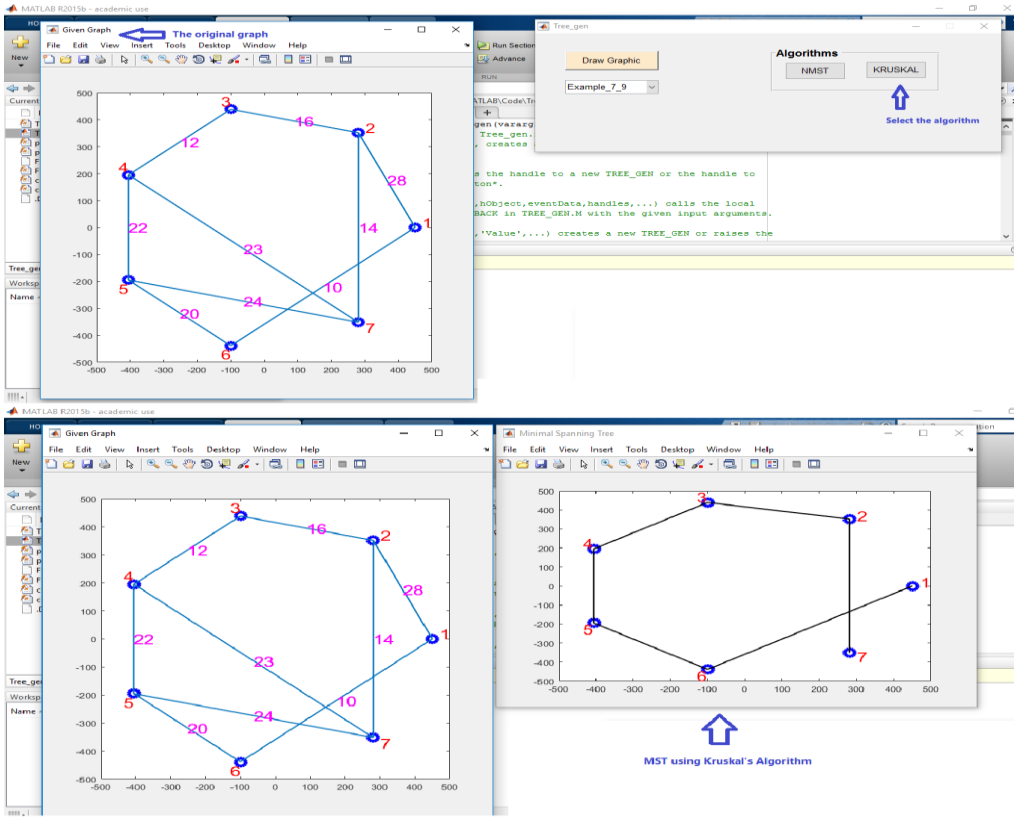


Figure 7. Procedure 2 for Example 4.1's Implementation

In Figure 8, procedure 3 of Example 4.1's implementation is given to select the NMST algorithm to get the MST.

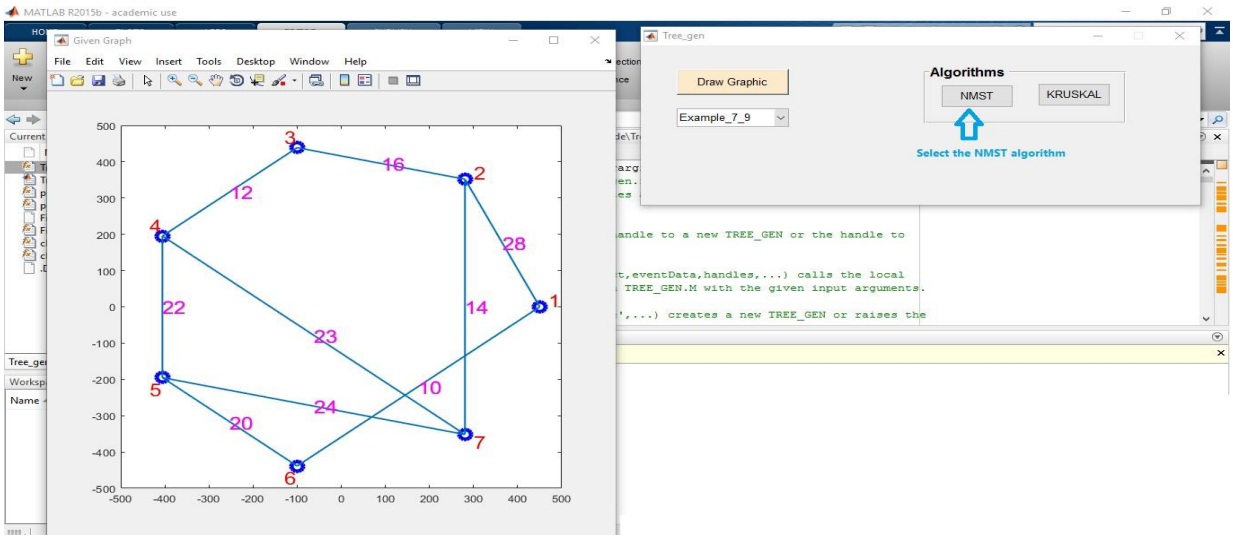


Figure 8. Procedure 3 for Example 4.1's Implementation

The final result is given in Figure 9.

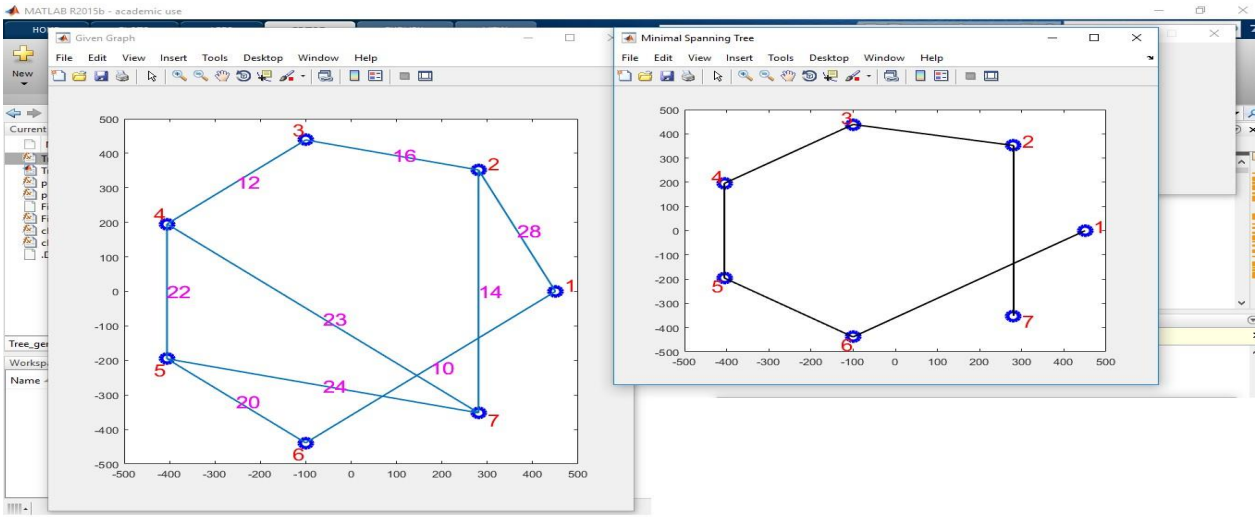


Figure 9. Procedure 4 for Example 4.1's Implementation

Example 4.2 In this example, we will consider a graph with 10 vertices and 17 edges. The original graphs and the final MSTs are given in Figure 10.

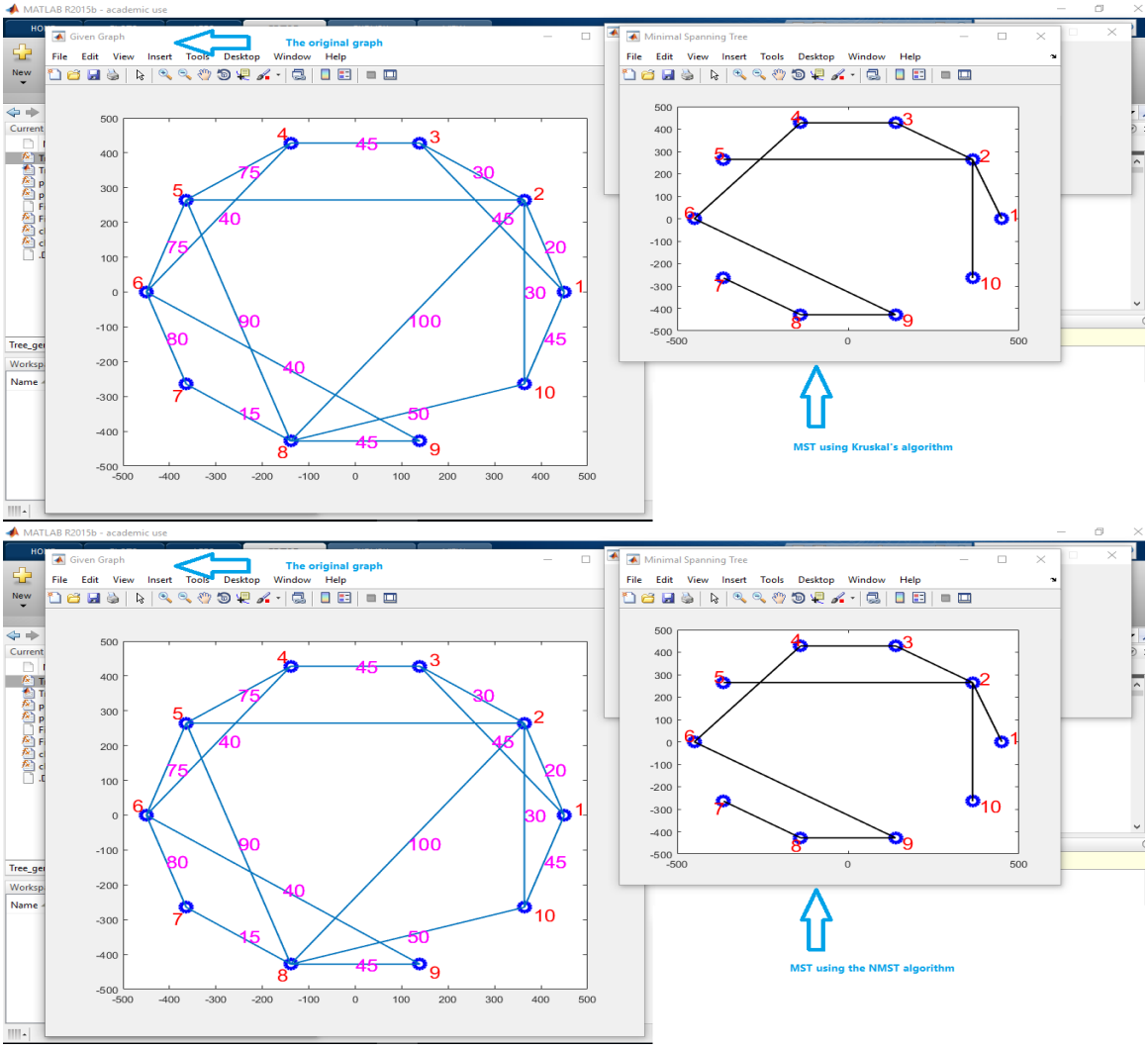


Figure 10. MSTs using Kruskal's and the NMST algorithms of Example 4.2's Implementations

5. Experimental Results

To make the above analysis more clearly, we show the execution time of Kruskal's and the NMST algorithms for Example 4.1 and 4.2 in Table 1.

Table 1. Execution Times of Kruskal and NMST Algorithms for Example 4.1 and 4.2

Number Of Examples	**Kruskal's Sorting Time**	**NMST's Sorting Time**	**Kruskal's Generating Time of MST**	**NMST's Generating Time of MST**
Example 4.1	0.00068	0.00090	0.00040	0.00017
Example 4.2	0.00186	0.00287	0.00056	0.00026

From Table 1, we can see that the NMST takes more time in sorting than Kruskal's algorithm, however, the time of generating the MST of the graph is less than Kruskal's algorithm. As a result, in total, it can be pointed out that the NMST is faster than Kruskal's algorithm to find the MST for a graph. For more spatial cases and more complicated graphs, respectively this difference between two algorithms will increase and we will study these cases in the future.

6. Conclusion

We can analyze the complexity of the Kruskal's and NMST algorithms assuming that V represents a vertices and E represents an edge. In order the Kruskal's algorithm to run faster, we can sort the edges applying Counting Sort. The time detail can be underlined as follows:

- The line of loading the matrix requires O (1) time.
- The lines of making a visiting list require O (V) time.
- The lines of sorting the matrix require O (|V| + |E|) time.
- The lines of adding the small value to the new matrix require O (|E| α |V|) time.

(This is the difference between the NMST algorithm).

The line of return or show values requires O (1) time.

So if we use 'Counting Sort' in order to solve the edges, the time complexity of Kruskal will be O (|v| + |E| log |v|). The same producer in above, only the difference is that O ((|V − 1|) α |V|) + Time of Tree Construction is O (V− 1). The time complexity of NMST will be O ((2v − 1) + log (2v − 1)).

References

[1] S. Skrbic, V. Loncar and A. Balaz. Distributed Memory Parallel Algorithms for Minimum Spanning Trees. The World Congress on Engineering, 2013.

[2] D. Patraa, S. Duttaa, H. Sankar, P.A. Verma, Development of GIS tool for the solution of minimum spanning tree problem using Prim's algorithm. The International Archives of the Photogrammetry, Remote Sensing, and 143 Spatial Information Sciences, 2014, pp. 9–12.

[3] S. Mohanram and T. D. Sudhakar. Power System Restoration using Reverse Delete Algorithm Implemented in FPGA. Dr. M.G.R. University, Maduravoyal, Chennai, Tamil Nadu, India, 2011, pp. 373-378.

[4] S. I. Ramaswamy and R. Patki. Distributed Minimum Spanning Trees, 2015.

[5] J. Kleinberg and E. Tardos. Greedy Algorithms. In Algorithm Design; Goldstein, M.; Suarez-Rivas, M., Eds.; Pearson-Addison Wesley: London, 2005; pp. 115–209.