Matrix method for determining Minimum Spanning Tree

Prof. Farhan Banu

Assistant Professor
Department of Mathematics,
University of Dhaka, Bangladesh

***Abstract*: This paper is concerned with Minimum Spanning Tree problem, a fundamental problem of Network modeling. Here we have proposed a novel approach to determine minimum spanning tree of an undirected connected network which is also demonstrated with numerical example.**

***Index Terms*: spanning tree, Network model, shortest length.**

I. Introduction

Minimum Spanning Tree (MST) problem: Given a connected graph G which has positive weights on each edge. The target is to find a set of edges with minimum weights that connects all of the vertices. A graph can have a number of different spanning trees. A Minimum Spanning Tree (MST) for a weighted, connected and undirected graph is a spanning tree with 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. The minimum spanning tree may not be unique if the graph has two or more edges with equal weights.
In network modeling, determining Minimum Spanning Tree (MST) is a fundamental problem which has a variety of applications in different sectors such as: Network design: TV /computer cable, telephone, road, Travelling salesman problem, Taxonomy [1], Cluster analysis [2, 3], Circuit design[4] etc.
A number of algorithms are developed for solving MST. Among them some greedy algorithms are mainly used now a days. The very first algorithm [5, 6] for finding a minimum spanning tree was developed in 1926 by Otakar Borvka. After that Prim's algorithm which is mostly used was invented by Vojtch Jarnk in 1930 and rediscovered by Prim in 1957[7]. Kruskal's[8] algorithm and reverse-delete algorithm, which is the reverse of Kruskal's algorithm are also well known though reverse-delete algorithm are usually not in use.
Our approach is inspired by the idea of Prim's algorithm.
The rest of the paper is organized as follows:
In the next section we have discussed the basic idea of our proposed approach. We have explained the methodology in details in the following section. A numerical example is included in section 3. Finally we have concluded our work in section 4.

II. Methodology

The idea of Matrix method for MST is inspired by Prim's Algorithm described as follows:

Initial step: We start with the first node say a. Consider a connected set C whose first entry is the node a and let DC be the disconnected set which contains all other nodes in the network. We can start with any other node as well.

- Consider a matrix whose columns are labelled with the name of nodes and entries are the length of all adjacent nodes to node a in the respective column.
- Determine the minimum of all these distances from node a to all other nodes in the disconnected set . Select the node corresponding to the minimum length as the next node to enter the connected set

General steps: The newly selected node enters the connected set and leaves the disconnected set. We insert a new row to the matrix of connected set described in initial step whose entries are the lengths of all adjacent nodes to this newly selected node in the respective column.
Determine the minimum of all the lengths from the nodes of connected set to all the nodes in the disconnected set. Consider the node corresponding to the minimum length as the next node to enter the connected set. Cross out all the connection (length) between the nodes in the connected set to avoid cycle. At each iteration we include a node to the connected set. We continue the process until we get all the nodes in the connected set.
If in a network there are n nodes then we need $(n-1)$ iteration to complete the procedure.

III. Explanation:

Suppose we have a network of n nodes $1, 2, 3, \dots, n$. We define the disconnected set

$DC = \{1, 2, 3, \dots, n\}$ The distance matrix $D = [d_{ij}]$ where d_{ij} represents the distance from node i to node j is as follows:

	1	2	3	…	n
1	-	d_{12}	d_{13}	…	d_{1n}
2	d_{21}	-	d_{23}	…	d_{2n}
3	d_{31}	d_{32}	-	…	d_{3n}
⋮	⋮	⋮	⋮	⋮	⋮
n	d_{n1}	d_{n2}	d_{n3}	…	-

We start 1st iteration with node 1 and the corresponding table:

	1	2	3		k	…	n
1	-	d_{12}	d_{13}	…	d_{1k}	…	d_{1n}

Table 1: Iteration: 1

Suppose d_{1k} is the smallest length. At the 2nd iteration we include k to the connected set and the table at this stage is shown in Table 2.

	1	2	3	…	k	…	n
1	-	d_{12}	d_{13}	…	$d_{1k}{}^{*}$	…	d_{1n}
k	-	d_{k2}	d_{k3}	…	-	…	d_{1n}

Table 2: Iteration 2

Since 1 is in the connected set so in Table-2 we replaced d_{k1} with a bar $(-)$ to avoid cycle. In general, at any iteration if j is selected to enter the connected set with shortest distance d_{sj} from node s where s is in C and if i is any node in C, then at the next iteration we put bar $(-)$ for all the distance d_{ij} and d_{ji} except d_{sj} as shown in Table 3.

We continue the process until all the nodes of the graph are in the connected set C.

	1	2	3	…	k	…	s	…	j	…	N
1	–	d_{12}	d_{13}	…	$d_{1k}{}^*$	…	d_{1s}	…	–	…	d_{1n}
k	–	d_{k2}	d_{k3}	…	–	…	–	…	–	…	d_{kn}
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
s	–	d_{s2}	d_{s3}	…	–	…	–	…	$d_{sj}{}^*$	…	d_{sn}
j	–	d_{j2}	d_{j3}	…	–	…	–	…	–	…	d_{jn}

Table 3: At some Iteration: Here the connected set is $C = \{1, k, \dots, s, j\}$.

IV. Numerical Experiment

To explain the algorithm with a numerical example, we choose a graph of 6 nodes shown in Figure 1 :

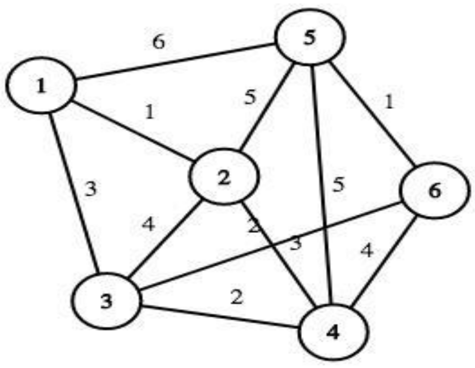

Figure 1: Undirected graph

The distance matrix D for the above graph is given by

Table 4

	1	2	3	4	5	6
1	–	1	3	–	6	–
2	1	–	4	3	5	–
3	3	4	–	2		2
4	–	3	2	–	5	4
5	6	5	–	5	–	1
6	–	–	2	4	1	–

.

Iteration 1 starts with node 1. We can choose any node to start with.

Table 5: Iteration 1: $C = \{1\}$, $DC = \{2, 3, 4, 5, 6\}$.

	1	2	3	4	5	6
1	–	1	3	–	6	–

Here 1 is the smallest length from node 1 to node 2 so we select 2 as the next entering node in the connected set as shown in table 6.

Table 6: Iteration 2: $C = \{1, 2\}$, $DC = \{3, 4, 5, 6\}$.

	1	2	3	4	5	6
1	–	1^*	3	–	6	–
2	–	–	4	3	5	–

At iteration 2, since node 1 is already in the connected set so we replace the length d_{21} with a bar $(-)$ which implies this length will be excluded for further consideration to avoid the generation of cycle. The minimum length from the nodes of connected set to all other nodes of disconnected set is 3 which appears in two cells d_{13} and d_{24}. We can choose any of them randomly and the resulting minimum length of the spanning tree will be same. First we choose the cells d_{24} and continue further iterations. Thus node 4 becomes the next entering node in the connected set. Alternatively we can select d_{13} which will be shown later.

Table 7: Iteration 3: $C = \{1, 2, 4\}$, $DC = \{3, 5, 6\}$.

	1	2	3	4	5	6
1	–	1^*	3	–	6	–
2	–	–	4	3^*	5	–
4	–	–	2	–	5	4

At iteration 3 we cross out the lengths d_{42} to avoid cycles. In general if a node is in the connected set then the column corresponding to that node will contain a single entry (the length for which it was selected to entre the connected set) at the final matrix.

The following iteration selects node 6 to entre the connected set. At the last iteration, node 5 enters the connected set and thus $DC = \{\}$. We get the minimum spanning tree with edges $E = \{(1, 2), (2, 4), (4, 3), (3, 6), (6, 5)\}$ and minimum length $ML = 9$.

Table 8: Iteration 4

$C = \{1, 2, 4, 3\}, DC = \{5, 6\}$

	1	2	3	4	5	6
1	–	1^*	–	–	6	–
2	–	–	–	3^*	5	–
4	–	–	2^*	–	5	4
3	–	–	–	–	–	2

Table 9: Iteration 5

$C = \{1, 2, 4, 3, 6\}, DC = \{5\}$

	1	2	3	4	5	6
1	–	1^*	–	–	6	–
2	–	–	–	3^*	5	–
4	–	–	2^*	–	5	–
3	–	–	–	–	–	2^*
6	–	–	–	–	1	–

Alternatively: if we choose d_{13} as the minimum length and select node 3 in iteration 2 as shown in table 6, we will have Table 10 and the successive iteration will be as follows.

Table 10: Iteration 2: $C = \{1, 2\}$

	1	2	3	4	5	6
1	−	1^*	3	−	6	−
2	−	−	4	3	5	−

Table 11: Iteration 3: $C = \{1, 2, 3\}$

	1	2	3	4	5	6
1	−	1^*	3^*	−	6	−
2	−	−	−	3	5	−
3	−	−	−	2	−	2

Thus we get a different spanning tree with edges $E = \{(1, 2), (1, 3), (3, 6), (6, 5), (3, 4)\}$ and minimum length $ML = 9$. We may have some other alternative minimum span- ning tree since we have several edges with equal lenght. The number of iteration required for determining these MST will be same

Table 12: Iteration 4: $C = \{1, 2, 3, 6\}$

	1	2	3	4	5	6
1	−	1^*	3^*	−	6	−
2	−	−	−	3	5	−
3	−	−	−	2	−	2^*
6	−	−	−	4	1	−

Table 13: Iteration 5: $C = \{1, 2, 3, 6, 5\}$

	1	2	3	4	5	6
1	−	1^*	3^*	−	6	−
2	−	−	−	3	5	−
3	−	−	−	2	−	2^*
6	−	−	−	4	1^*	−
5	−	−	−	5	−	−

V. Conclusion

In network modeling, Minimum Spanning Tree (MST) is a fundamental problem. A number of algorithms are developed for solving MST . Among them Prim's algorithm and Kruskal's algorithm are mainly in use. In this paper we have proposed a new approach which is inspired by the idea of Prim's algorithm. The advantage of our approach is that we do not need to work with graph at each iteration unlike Prim's and Kruskal's algorithms. We first determine the distance matrix and then solve the problem using the distances. Though the number of iteration required for our proposed approach is similar to that of Prim's and Kruskal's but working with matrix is a great advantage over that with graphs.

REFERENCES

[1] P.H.A. Sneath, The Application of Computers to Taxonomy. Journal of General Microbiology. 17 (1): 201226. doi:10.1099/00221287-17-1-201, 1957.

[2] S. Theodoridis and K. Koutroumbas, Pattern Recognition. Elsevier Inc., 2009.

[3] S. Theodoridis and K. Koutroumbas. An Introduction to Pattern Recognition, A MATLAB approach, Elsevier Inc., 2010.

[4] H. Ohlsson, Implementation of low complexity FIR filters using a minimum spanning tree. 12th IEEE Mediterranean Electrotechnical Conference (MELE- CON 2004). 1, 261264. doi:10.1109/MELCON.2004.1346826.

[5] Borvka, Otakar, O Jist´em Probl´emu Minim´aln lm. Pr´ace Moravsk´e Pˇr lrodovdeck´e Spoleˇcnosti III, 3 (1926): 3758.

[6] https://algowiki-project.org/en/Boruvka's algorithm# cite note-1 .

[7] R. C. Prim, Shortest Connection Networks and Some Generalizations, Bell System Technical Journal, 36(6), 13891401, 1957. doi:10.1002/j.1538- 7305.1957.tb01515.x.

[8] J. B. Kruskal, On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem, Proceedings of the American Mathematical Society, 7(1), 1956, 48-50. doi:10.1090/S0002-9939-1956-0078686-7.