JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Study of Minimum Cost Spanning Tree Implementation on TSP based on complexities of Algorithm

Dillip Narayan Sahu[1]

[1]Lecturer, Department of MCA, School of Computer Science, GM University, Sambalpur, India

Abstract: A spanning tree is a sub-graph of a graph G which contains all vertices and contains no circles. A graph can have multiple spanning trees. We can also assign some weight/value/cost to each edge of the graph. The minimum cost spanning tree will be the lowest cost spanning tree among different spanning trees of the graph G. This paper presents a fast and efficient algorithm for the construction of minimum cost spanning trees (MCST) based on the concept of Travelling Salesman problem (TSP) in a connected weighted graphs G. The algorithm incorporates the traditional method to sort the weights / costs associated with the edges in the graph G in linear time. Also the algorithm exploits the fact that every connected graph G with "n" number of nodes with exactly "n-1" number of edges is a tree. The complexity of the algorithm is bounded by Big-O(m log n), where m is the number of edges and n is the number of vertices of the graph G. Our objective is to find the minimum cost spanning tree in a graph G and use that concept on a traveling salesman problem (TSP) based on complexities of algorithm.

Keywords: **Complexity, MCST, Simple Graph, TSP, Weighted Graph.**

1. Introduction

A graph (simple graph) G is nothing but collection/set of vertices and edges, where V={ set of vertices v1, v2, v3, ...some vi, ...vn} and E={ set of edges e1, e2, e3, ... some ei, ... en}. Mathematically graph G={V,E}[1][2][3].

A Tree is may be considered as a subgraph of the graph G which holds two properties, i.e. (i) all the vertices are connected and (ii) there should not contains any cycle/circuit or self loop[4][5]. In this paper our objective is to implement the travelling salesman problem (TSP) based on the concept of minimum cost spanning tree (MCST) with the condition based on some complexities of the algorithm.

2. Basic Definition and Applications

2.1 Spanning Tree

A Tree T is said to be a spanning tree of a connected graph G if the tree T is a subgraph of the graph G and the tree T contains all the vertices of the graph G[6].

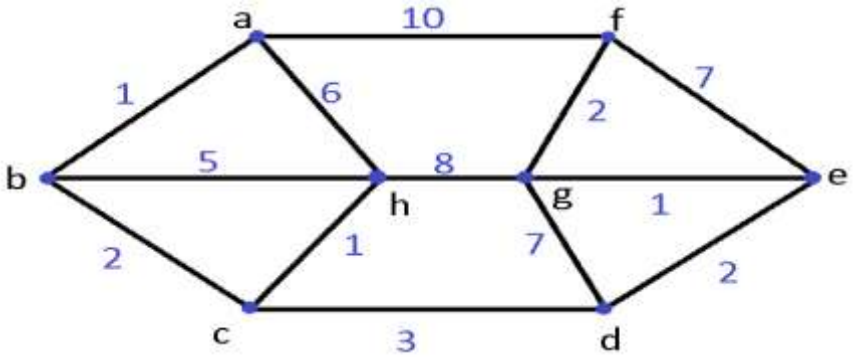

Fig. 1. Graph G

2.2 Minimum Cost Spanning Tree

A Minimum Cost Spanning Tree (MCST) is a spanning tree which has the lowest path cost among all the spanning trees of the original graph G[7]. "**Fig. 2**" shows the minimum spanning tree from the graph G in the figure1. We may have multiple spanning trees from a graph but the optimal one should be the lowest path cost from all the resultant spanning trees from the graph G.

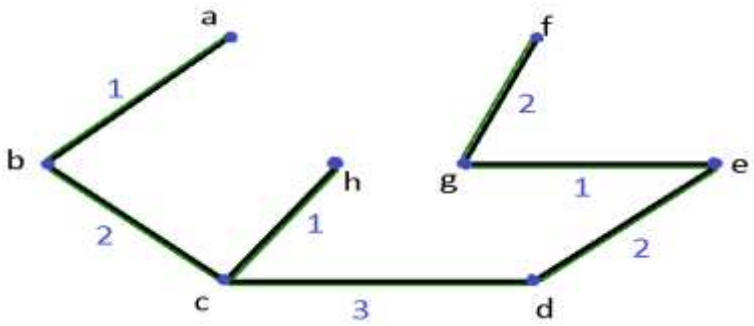

Fig. 2. Spanning Tree

2.3 Travelling Salesman Problem

The Travelling Salesman Problem is all about a salesman and a set of cities. The objective of the TSP is to cover all the cities with minimum cost, so to minimize the total travel cost. Minimum Cost Spanning Tree (MCST) is a spanning tree which has the lowest path cost among all the spanning trees of the original graph G[8].

Application:

Network Analysis:

The standard application of a spanning tree will be used in a network of computers may be organized in a clustered form. Now if we want to reduce the cost or want a minimal cost then we may use the MST from the network graph[9].

Another application could be, suppose we have a group of cities interconnected through different channels/paths (necessary through some bridges), then to reduce the path cost, we may consider the concept of MCST from the city graph to get the optimal solution. Figure3 depicts a simple network graph which is a best example to demonstrate the case of a travelling salesman problem which includes different cities and associated with path cost i.e. weight from one city to another. As in the case of Minimum Spanning Tree, here also we can implement the concept of lowest path cost to find out the optimal minimum weight network tree to cover all the cities with minimum cost.

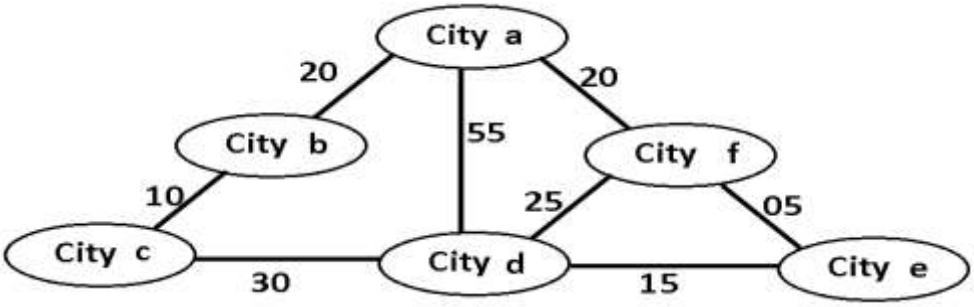

Fig. 3. Network Graph

3. MST – Literature Review

3.1 Applications of MCST-

Network graph - The most important application of MST is network graph including computer network, telephone networks, electrical circuits, TV cable network, road traffic network, islands connection, pipeline network etc[10].

Approximation algorithm for NP-hard problem - this application for MST is used to solve travelling salesman problem (TSP).

Cluster analysis - k means clustering problem can be viewed as finding an MST and deleting the k-1 edges, i.e. most expensive edges.

Clustering, and clustering gene expression data etc

Constructing spanning trees used in computer networks.

Segmentation of Image, feature extraction in computer vision, recognition of handwriting in mathematical expressions and formulation, Circuit design, Geographical areas path finding problem.

3.2 Minimum Cost Spanning Tree (MCST) Algorithm

There are different algorithms to find the Minimum Cost Spanning Tree (MCST) in a given graph G.

1. Prim's Algorithm
2. Kruskal's Algorithm

There are various algorithms to solve minimum cost spanning trees problem. Prim's Algorithm and Kruskal's Algorithm is the greedy approach algorithm which are used to find MCST of an undirected weighted graph G, which minimises the weight and the number of edges[11].

3.2 .1 PRIM'S Algorithm

The prim's algorithm was developed in 1930. The prim's algorithm is the greedy approach algorithm used to find the minimum cost spanning tree MCST for an undirected weighted graph G.

PRIM'S ALGORITHM Steps -

1. Initialise the tree with a single vertex from a given graph G.

2. Choose the shortest outgoing edge from the node (i.e. the single vertex taken in step 1) and add this to the tree T.

3. The edge formed by the two vertices has to be aaded in to the tree T, and we check for the shortest adjacent edge to the node or vertices of tree T in the given graph G and add this to the tree (with the condition that - should not formed any kind of cycle or circuit)

4. Repeat the steps until all the vertices are added to the tree T and should be preserve the properties of the tree T.

Pseudo code of Prim's Algorithm on the input graph G-

Input- A connected undirected weighted graph G

Output- (MCST) minimum cost spanning tree T of the graph G.

Make a queue (Q) for all the vertices (V) of the graph G.

ReachSet = {0} here we can select any node

UnReachSet = {1, 2, ..., N-1}

SpanningTree = {}

while (UnReachSet ≠ empty)
{
Find edge e = (u, v) such that:
1. u ∈ ReachSet
2. v ∈ UnReachSet
3. e_min has smallest cost
Minimum Cost Spanning Tree T = Minimum Cost Spanning Tree T ∪ {e_min}
ReachSet = ReachSet ∪ {u}
UnReachSet = UnReachSet - {v}
}

Adjacency matrix representation of a graph G-

An adjacency matrix is nothing but a square matrix, which is used to represent a finite graph G. The elements of the matrix indicates that whether the pairs of vertices are adjacent to each other or not in the graph G[12].

Let G be a graph with "n" vertices that are assumed to be ordered from v1 (first vertex) to vn (last vertex).

The n x n matrix A,

Where, aij= 1 if there exists a path from vi to vj

aij = 0 otherwise

is called an adjacency matrix.

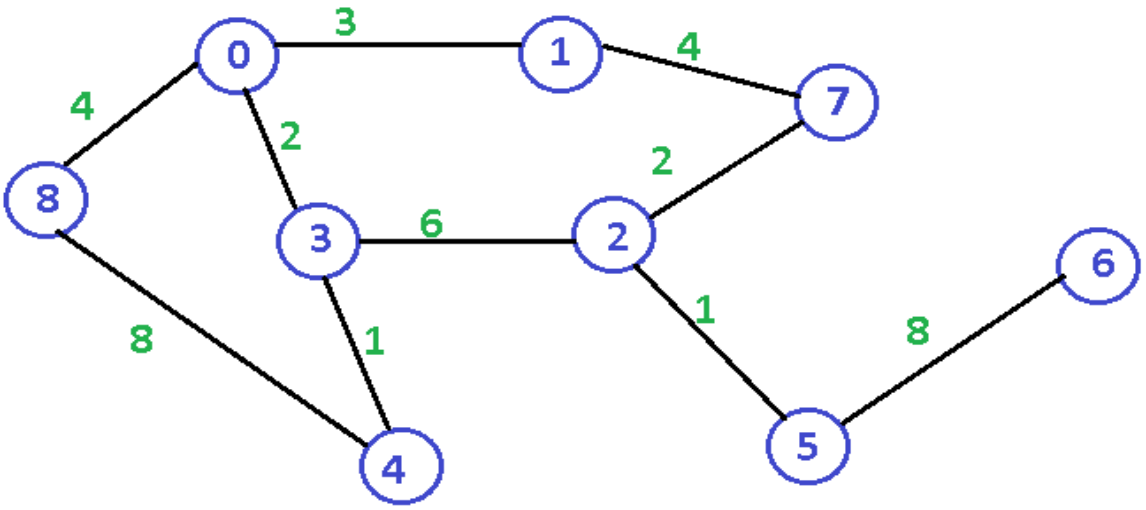

Fig. 4. Graph G

M=

	0	1	2	3	4	5	6	7	8
0	*	3	*	2	*	*	*	*	4
1	3	*	*	*	*	*	*	4	*
2	*	*	*	6	*	1	*	2	*
3	2	*	6	*	1	*	*	*	*
4	*	*	*	1	*	*	*	*	8
5	*	*	1	*	*	*	8	*	*
6	*	*	*	*	*	8	*	*	*
7	*	4	2	*	*	*	*	*	*
8	4	*	*	*	8	*	*	*	*

* = a very large value (infinite)

Fig. 5. Adjacency Matrix of the given graph G in Figure 4

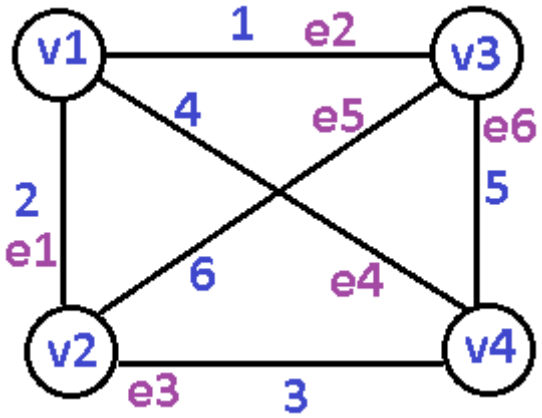

Fig. 6.1. Initial Graph G

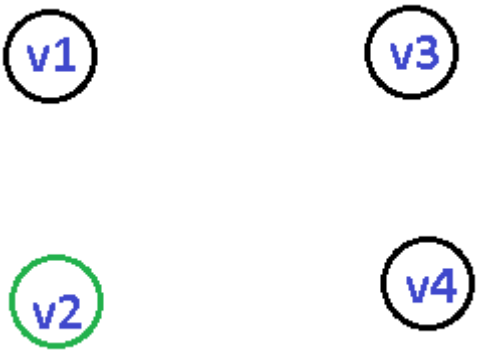

Fig. 6.2. Step 1

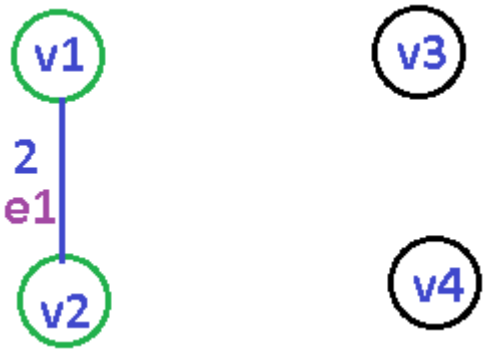

Fig. 6.3. Step 2

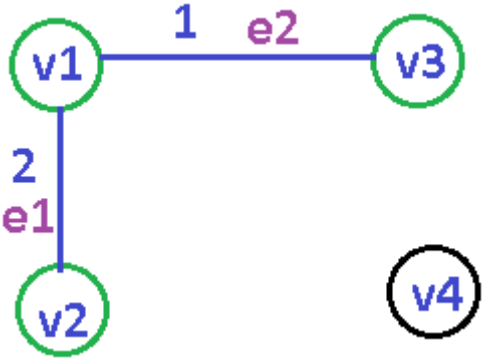

Fig. 6.4. Step 3

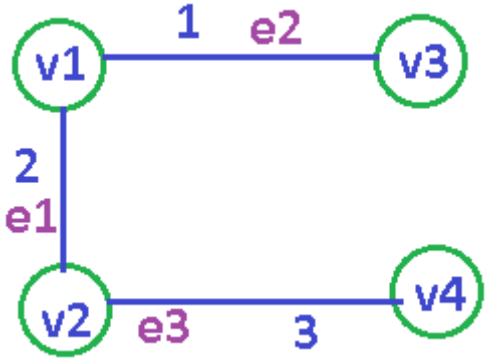

Fig. 6.5. Step 4

Now, Total Cost of MCST- Minimum Cost Spanning tree (from the "**Fig. 6.5**") is

= Cost of (e1) + Cost of (e2)+ Cost of (e3)

=1+2+3

=6 (which is the lowest cost among all others spanning trees of the Initial graph G)

3.2 .2 Kruskal's algorithm

This algorithm uses the greedy approach for finding a minimum cost spanning tree.

This algorithm takes every node as an independent tree and then connects one with another only if the edge has the lowest weight/cost compared to all other weights in the graph G.

KRUSKAL'S ALGORITHM Steps

1. Sort the edges of the graph with respect to their weights or cost associated with the edges.

2. Start adding edges to the minimum spanning tree from the edge with the smallest weight until the Minimum spanning tree cover all the vertices of the graph G.

3. Only add those edges which don't form a circuit or cycle.

Input- A connected undirected weighted graph G

Output- (MCST) minimum cost spanning tree T of the graph G.

Pseudo code of Kruskal's Algorithm on the input graph G-

MCST- KRUSKAL(Graph G):-

Initially Spanning Tree T := Empty

For every node n ∈ G.N

CreateSet(n)

For every path (m, n) ∈ G.E arranged by increasing weights (m,n):
if NewSet(m) < > NewSet(n):

Spanning Tree T = T U {(m, n)}

UNION (m, n)

return Spanning-Tree-T

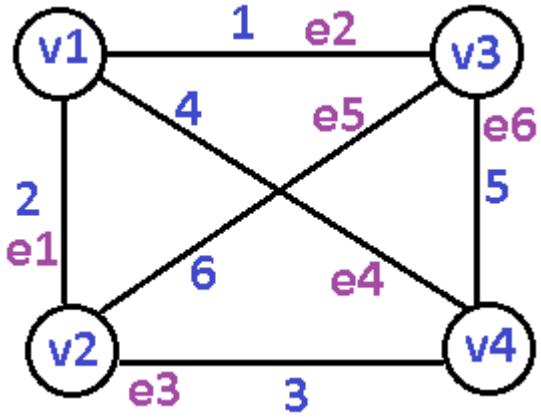

Fig. 7.1. Initial Graph G

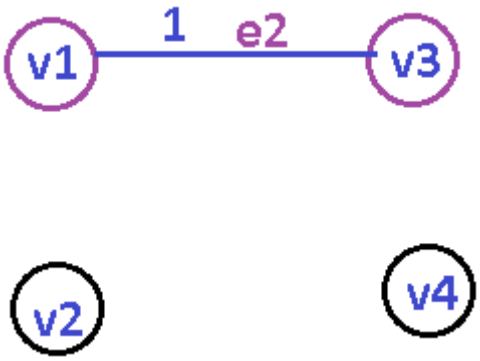

Fig. 7.2. Step 1

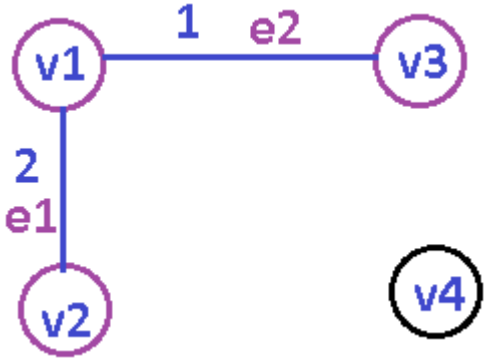

Fig. 7.3. Step 2

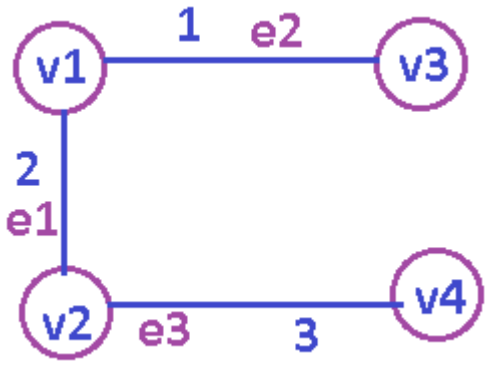

Fig. 7.4. Step 3

Now, Total Cost of MCST- Minimum Cost Spanning tree (from the "**Fig. 7.4**") is

= Cost of (e1)+ Cost of (e2)+ Cost of (e3)

=1+2+3

=6 (which is the lowest cost among all others spanning trees of the Initial graph G)

4.Implementation of MCST on TSP

Travelling Salesman Problem (TSP) is all about a sales person and different cities where the objective of the sales person is to cover/visit all the cities with minimum cost but the problem is that, if that sales person visits all the cities and reach the starting node from where he started to visit, then that create a circuit which makes the entire solution to a wrong state because here we have to discard those edges in the resultant Spanning Tree addition of which makes or forms any kind of circle or circuit[13].

Let G=(V,E) be an undirected weighted connected graph with some set of vertices V and some set of edges E and assigned with some weight or path cost (w).

V={v1,v2,v3,v4, ...}

E={e1,e2,e3,e4,.....}
w1 ∈ v1-v2, w2 ∈ v2-v3, w3 ∈ v3-v4, w4 ∈ v4-v1, w5 ∈ v1-v5, w6 ∈ v2-v5, w7 ∈ v3-v5, w8 ∈ v4-v5

Algorithm:

Input: The Travelling Salesman Problem for an Undirected Weighted Graph G

Output: Minimum Cost Spanning Tree T of Graph G

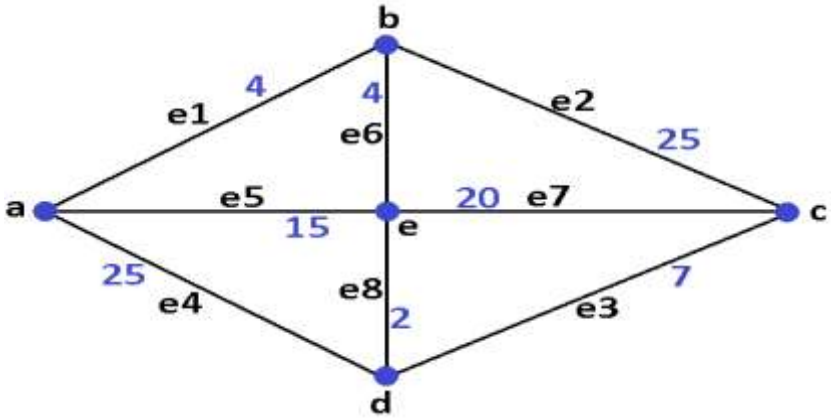

Fig. 8. Graph G

"**Fig. 8**" is an example of simple connected weighted graph G, from which we will try to find the optimal spanning tree which will be the lowest path cost among all the spanning trees.

Step 1: Start

Step 2: Repeat step 3 to 4 until the salesman covers all the vertices atleast once.

Step 3: Starts with vertex 'vi ∈ V ', find the minimum edge adjacent with the vertex 'vi'.

[Assume that the Initial vertex a = vi]

Step 4: Then by taking the finding edge (in step 3), again search for the minimum cost edge adjacent to the already finding edge.

If the corresponding edge forms a cycle then discard it

else

add to the Tree T

Step 5: Construct the Tree T which will be the Minimum Cost Spanning Tree T with the lowest path cost for the Travelling Salesman Problem for the given Graph G.

Step 6: Measure/Calculate Time Complexitiy from the recurrence T(n) = 2T(n/2) + f(n)

[Note : here (n/2) refers to the lower bound of n/2, where n is the input size]

and also space complexity on the resultant tree T. (based on the number of vertices and edges).

If the complexity is satisfactory enough Then produce the output else display the worst case result.

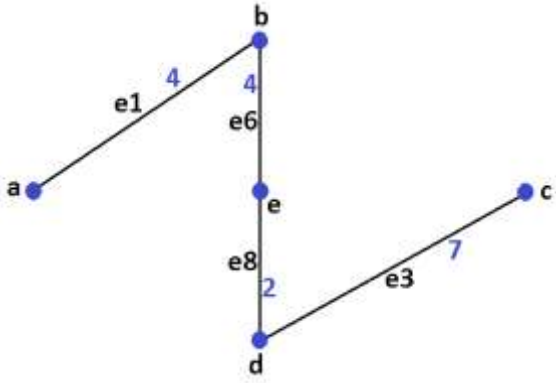

Fig. 9. Minimum Cost Spanning Tree (Final Stage)

To know the value of Time Complexity T(n), we have to calculate sum of vertices and edges of the resultant Spanning Tree (by using Masters Theorem)

Θ(IVI + IEI)

=Θ(I5I + I4I)

=Θ(9)

≅ Θ(n)

Space Complexity = O(V+E) i.e. maximum number of vertices with edges allocated/occupied in the memory location ≅O(n).
"**Fig. 9**" shows the optimal spanning tree with total minimum path cost= w(e1)+w(e6)+w(e8)+w(e7)=4+4+2+7=17 (i.e. lowest path cost among all the spanning trees that can be generated from the graph G show in the figure-4).

5. Results and Findings

As we can see that we can implement the concept of Minimum Cost Spanning Forest/Tree (MCST) based on the concept of Travelling Salesman Problem (TSP) by assuming that the resultant spanning tree should not contain any cycle and satisfies the minimum necessary and sufficient condition of a spanning tree T.

The time complexity of Prim's algorithm is Big-O(VlogV + ElogV) = O(ElogV)

The time complexity of Kruskal's algorithm is Big-O(VlogV + ElogV) = O(ElogV)

But, Prim's algorithm can be improved using Fibonacci Heaps to Big-O(E + logV)

Now the question is, what will be the complexity of an efficient algorithm of finding the minimum cost spanning forest/tree?

so, in case of an efficient algorithm like Kruskal's method, the time complexity T(n) = Big-O(|E| log |E|), as it uses a min-heap of the graph edges and adds the currently least cost edge to the spanning tree T as long as adding the edge does not create a cycle/circuit and there are lesser than |V| edges in the spanning tree T.

The space complexity will be calculated as -

At each and every stage maintain the set T of nodes in the tree, and for each node not in the tree maintain the shortest/least distance from that node to any node in the tree.

At the start T is node 0 and for each other node you compute the least distance from node 0 to that node.

At each stage, we can pick the node not in the tree with least distance, then compute the distance from that node to all other nodes not in the tree. If that is less than their current least/low cost distance update that least/low cost distance.

Storage cost is Big-O(n) to maintain T and Big-O(n) to maintain for each node not in the tree a note of the least distance from the tree to that node.

6. Conclusion

This paper presented a simple study to find Minimum Cost Spanning Tree over a Travelling Salesman Problem of an undirected weighted graph G based on complexities of efficient Algorithm. As there are different algorithms to represent Minimum Cost Spanning Tree , algorithm presented here is based on Minimum Cost Spanning Tree on Travelling Salesman Problem. In future, we shall concentrate to solve different cases of complexities with different real world problem on Minimum Cost Spanning Trees.

References

1. Journal Article

[1] S. G. Shirinivas, S. Vetrivel, and M. Elango,(2010), “Applications of Graph Theory in computer Science review”, Int. J. of Engg. Sci. and Tech., 2(9) 4610-4621. Article (CrossRef Link)

[2] N. Deo, (2000), “Graph Theory with Applications to Engineering and Computer Science”, Prentice Hall Inc., New Jersey. Article (CrossRef Link)

[3] R. Kalaivani, D. vijayalakshmir, “Minimum Cost Spanning Tree using Matrix Algorithm”, International Journal of Scientific and Research Publications, vol. 4, september 2014 Article (CrossRef Link)

[4] J. Dutta, S.C. Pal, A. Mandal, “A New Efficient Technique to Construct a Minimum Spanning Tree”, International Journal of Advanced Research in Computer Science and Software Engineering, oct, 2012. Article (CrossRef Link)

[5] R. Cambos, M. Ricardo, "A fast algorithm for computing minimum routing cost spanning trees Computer Networks", 52 (2008), pp. 3229-3247. Article (CrossRef Link)

[6] G. Zhou, Z. Cao, J. Cao, Z. Meng, "A genetic algorithm approach on capacitated minimum spanning tree problem", International Conference on Computational Intelligence and Security (2006), pp. 215-218. Article (CrossRef Link)

[7] I. A. Sheikh, Shahid-ul-Islam, “Minimum Spanning Tree Algorithms and Techniques”, IJCIRAS | ISSN (O) - 2581-5334 January 2018 | Vol. 1 Issue. 8 Article (CrossRef Link)

[8] J. B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem”, Proceedings of the American Mathematical Society, vol. 7, no. 1, p. 48, 1956. Article (CrossRef Link)

[9] R. C. Prim, "Shortest connection networks and some generalizations", Bell System Technical Journal, vol. 36, no. 6, pp. 1389–1401, 1957. Article (CrossRef Link)
[10] R. Ahuja, K. Mehlhorn, J. Orlin, R. Tarjan, "Faster Algorithms for the Shortest Path Problem", J. ACM 1990, 37, 213–223. Article (CrossRef Link)
[11] D. Johnson, "Efficient Algorithms for Shortest Paths in Sparse Networks", J. ACM 1977, 24, 1–13. Article (CrossRef Link)
[12] A. Datta, M. Hossain, M. Kaykobad, "An improved MST algorithm for ranking players of a round robin tournament", Int. J. Comput. Math. 85, 1–7 (2007) Article (CrossRef Link)
[13] A. Shioura, A. Tamura, "Efficiently scanning all spanning trees of an undirected graph", Journal of the Operations Research Society of Japan, 38(3), 331-344. Article (CrossRef Link)
[14] G. J. Minty, "A simple algorithm for listing all the trees of a graph", IEEE Transactions on Circuit Theory, CT-12, 120. Article (CrossRef Link)
[15] T. Matsui, "A flexible algorithm for generating all the spanning trees in undirected graphs", Algorithmica, 18(4), 530-543. Article (CrossRef Link)
[16] S. Kapoor, H. Ramesh, "An algorithm for enumerating all spanning trees of a directed graph", Algorithmica, 27(2), 120-130. Article (CrossRef Link)
[17] S. Kapoor, H. Ramesh, "Algorithms for enumerating all spanning trees of undirected and weighted graphs", SIAM Journal on Computing, 24, 247-265. Article (CrossRef Link)
[18] S. Kapoor, H. Ramesh, "Algorithms for enumerating all spanning trees of undirected and weighted graphs", SIAM Journal on Computing, 24, 247-265. Article (CrossRef Link)
[19] H. N. Gabow, E. W. Myers, "Finding all spanning trees of directed and undirected graphs", SIAM Journal on Computing, 7, 280-287. Article (CrossRef Link)
[20] H. N. Gabow, "Two algorithms for generating weighted spanning trees in order", SIAM Journal on Computing, 6(1), 139-150. Article (CrossRef Link)

2. Book

[21] R. Balakrishnan, K. Ranganathan, (2012), "A text book of Graph Theory", Second Edition, Springer, New York. [Online].Available: https://gasckondotty.ac.in/assets/images/slider/7ec1f302b7e80c2ed413beb2d77ccfa5.pdf
[22] J. R. Wilson, "Introduction to Graph Theory", Prentice Hall Inc., 1996. [Online]. Available: https://www.maths.ed.ac.uk/~v1ranick/papers/wilsongraph.pdf

Mr. Dillip Narayan Sahu received his B.Sc. degree in physics, Master of Computer Application degree, M.Tech. degree in Computer Science from Sambalpur University, India, M-Phil in Computer Science from MATS University, India, and continuing Ph.D. in Computer Science. His research area includes Artificial Intelligence, Machine Learning, Analysis and Design of Algorithms. He has published many papers in National/International Conferences, Seminars, and Journals. Presently He is working as a Lecturer in the Dept. of MCA, Gangadhar Meher University, India.