© 2018 IJSRST |Volume 4 | Issue 10| Print ISSN: 2395-6011 | Online ISSN: 2395-602X
Themed Section: Science and Technology

Review of a Connected Graph Using Prim's Algorithm to Find a Minimum Spanning Tree

G. Kousika[1], Naveen. L[2]

[1]MSc Mathematics, Department of Mathematics, Dr SNS Rajalakshmi Collage of Arts and Science, Coimbatore, Tamil Nadu, India

[2]MSc, MPhil, Assistant Professor, Department of Mathematics, Dr SNS Rajalakshmi Collage of Arts and Science, Coimbatore, Tamil Nadu, India

ABSTRACT

The spanning tree T of an undirected graph G is a sub graph that is a tree, which Includes all of the vertices of G, with minimum possible number of edges .A minimum spanning tree is a subset of the edges of a connected edge weight undirected graph that connects all the vertices together without any cycles and with the minimum possible total edge weight. This paper highlights the concepts of minimum spanning tree using Prim's algorithm and also includes a graphical representation of the algorithm.

Keywords: Connected Graph, Minimum Spanning Tree, Prim's algorithm, Kruskal's algorithm, travelling sales man problem.

I. INTRODUCTION

A graph is connected when there is a path between every pair of vertices. In a connected graph, there are no unreachable vertices. A spanning sub graph H of a connected graph G such that H is a tree is called a spanning tree of G .The spanning tree of G whose weight is minimum among all spanning trees of G. Such a spanning tree is called a minimum spanning tree. Greedy algorithm using to implementing the minimum spanning tree,

(1) Prim's algorithm
(2) Kruskal's algorithm

These two algorithms are considered as Greedy algorithm. The Kruskal's algorithm firstly processed the edges according to their weights from lowest to largest values. It takes the lowest valued edge and added to the minimum spanning tree .This process continuous until all the vertices are visited. Prim's algorithm find a minimum spanning tree for a weighted undirected graph other well-known algorithms for this problem includes kruskal's algorithm.

II. PRIM'S ALGORITHM

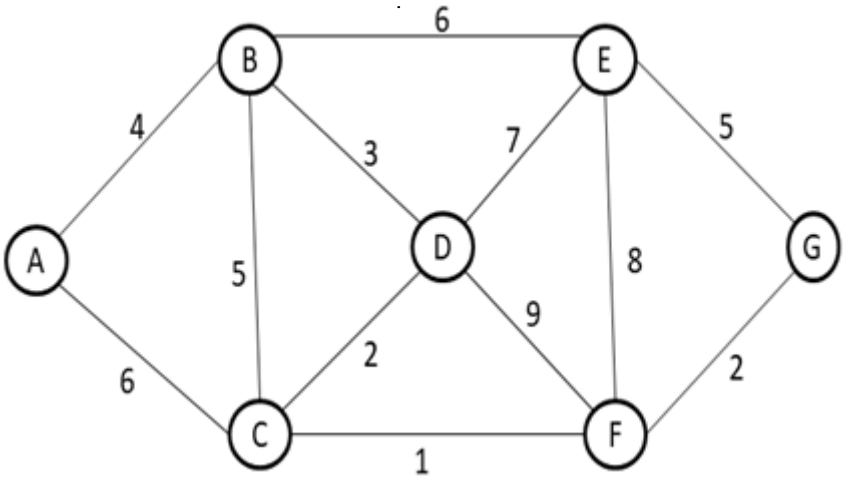
Prim's algorithm to find a minimum spanning tree in the connected graph.

Prim's algorithm use to find the minimum spanning tree as follows:

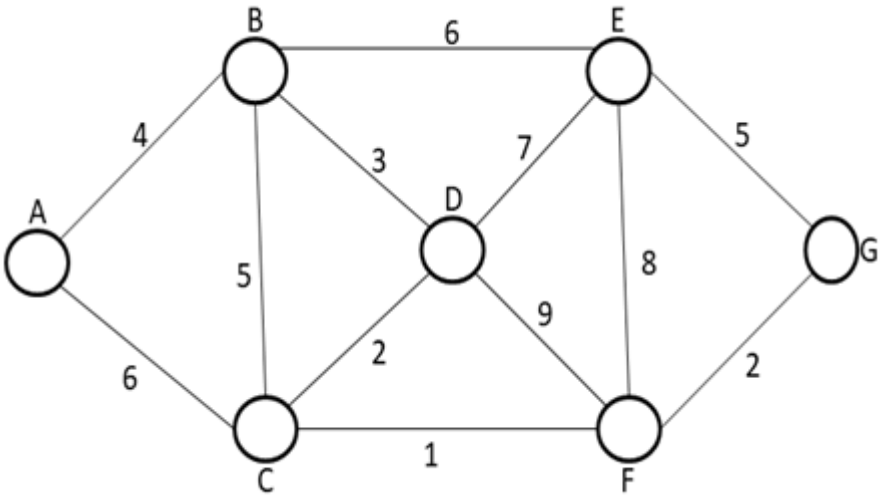
1. Removing a loop in a graph.
2. Choose any node and labeling the nodes with a distance of 0 and other nodes are ∞.The chosen node is treated as current node and considered as visited , also other nodes considered as unvisited.
3. Identify all the unvisited nodes, were presently connected to the current nodes. Calculate the distance from the unvisited nodes to the current nodes.

4. Label each of the vertices with their corresponding weight to the current nodes
5. Mark the current node as visited by coloring it.
6. By all the unvisited nodes, find out the node which has minimum weight to the current node. Consider this node is visited and make it is a current working node.
7. Steps 3, 4 and 5 repeat until all the nodes are visited.
8. Finally, we get the minimum spanning tree.

Calculating the minimum spanning tree using the prim's algorithm as following graph:



STEP1: Redraw the following diagram.

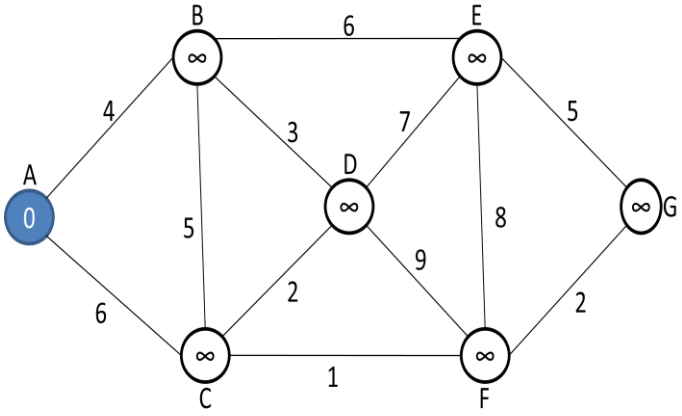


Consider the node A. The node A will be treated as visited.

Table 1

NODES	A	B	C	D	E	F	G
DISTANCE d[v]	0	∞	∞	∞	∞	∞	∞
DISTANCE FROM Ψ [V]	A						

STATUS	visited	unvisited	unvisited	unvisited	unvisited	unvisited	unvisited



STEP2: Calculate the unvisited nodes of current node A. The unvisited nodes are B and C.

Distance (B)>distance (A, B) =>∞>4

Relabel is required in node B.

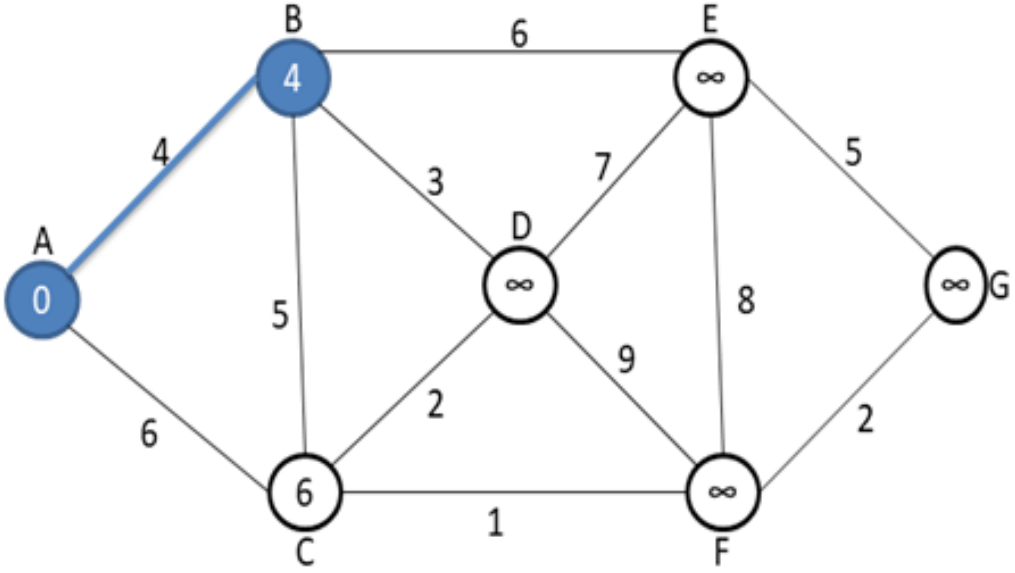
Distance (C)>distance (A, C) =>∞>6

Relabel is required in node C.

Table 2

NODES	A	B	C	D	E	F	G
DISTANCE d[v]	0	4	6	∞	∞	∞	∞
DISTANCE FROM Ψ[V]	A	A	A				
STATUS	visited	unvisited	unvisited	unvisited	unvisited	unvisited	unvisited

The unvisited node B is the smallest distance from node A. So the current working node is B.



STEP3: Calculate the unvisited nodes of current node B. The unvisited nodes are C, D and E

Distance(C)>distance of (B, C) => 6 >5

Relabel is required in node C.

Distance (D)>distance of (B, D) => ∞ >3

Relabel is required in node D.

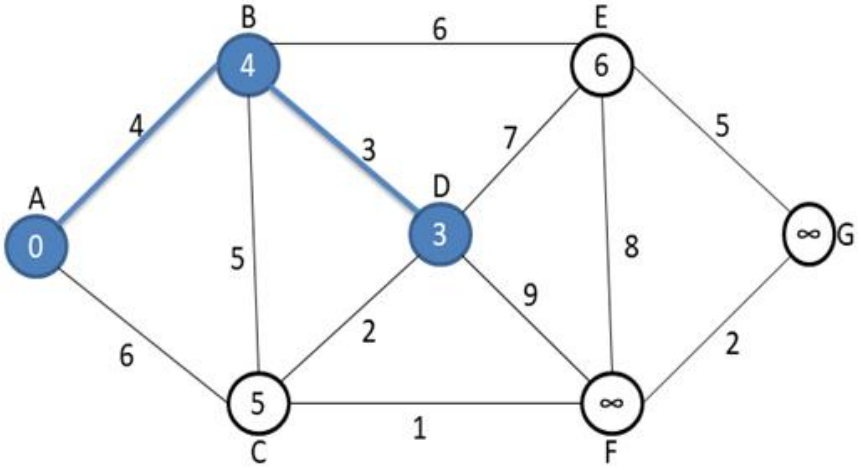
Distance (E)>distance of (B, E) => ∞ >6

Relabel is required in node E.

Table 3

NODES	A	B	C	D	E	F	G
DISTANCE d[v]	0	4	5	3	6	∞	∞
DISTANCE FROM Ψ [V]	A	A	B	B	B		
STATUS	visited	visited	unvisited	unvisited	unvisited	unvisited	unvisited

The unvisited node D is the smallest distance from node B. So the current working node is D.



STEP4: Calculate the unvisited nodes of current node D. The unvisited nodes are C, E and F.

Distance (C)>distance of (D, C) => 5>2

Relabel is required in node C.

Distance (E)<distance of (D, E) => 6<7

Relabel is not required.

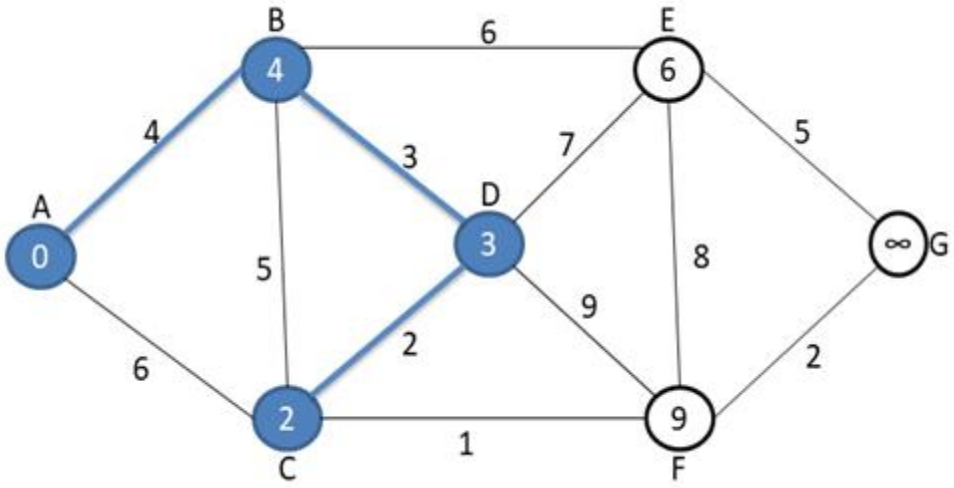
Distance (F)>distance of (D, F) => ∞ >9

Relabel is required in node E.

Table 4

NODES	A	B	C	D	E	F	G
DISTANCE d[v]	0	4	2	3	6	9	∞
DISTANCE FROM Ψ [V]	A	A	D	B	D	D	
STATUS	visited	visited	unvisited	visited	unvisited	unvisited	unvisited

The unvisited node C is the smallest distance from node D. So the current working node is C.



STEP5: Calculate the unvisited nodes of current node C. The unvisited nodes are F.

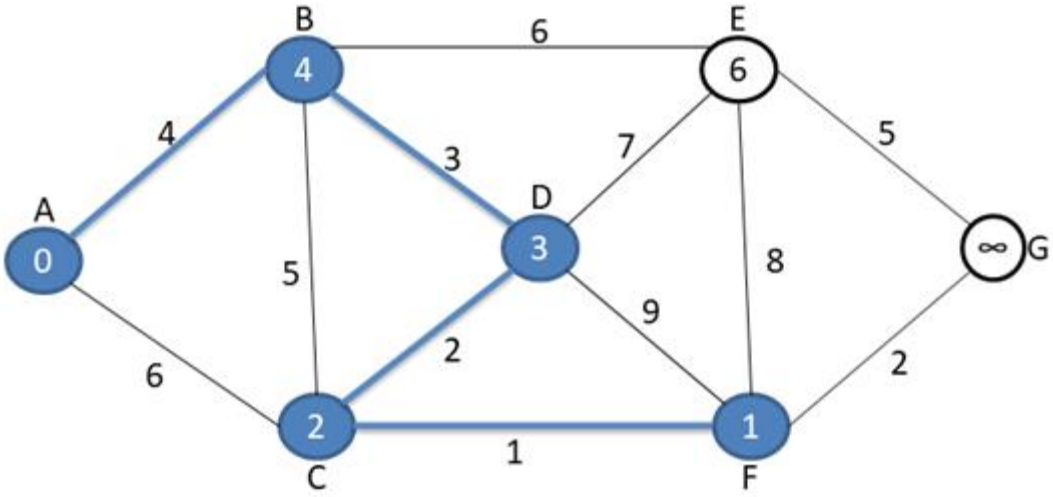
Distance (F)>distance of (C, F) => 9>1

Relabeling is required in node F.

Table 5

NODES	A	B	C	D	E	F	G
DISTANCE d[v]	0	4	2	3	6	1	∞
DISTANCE FROM Ψ [V]	A	A	D	B	D	C	
STATUS	visited	visited	visited	visited	unvisited	unvisited	unvisited

The unvisited node F is the smallest distance from node C. So the current working node is F.



Step6: Calculate the unvisited nodes of current node F. The unvisited nodes are G and E.

Distance (G)>distance of (F, G) =>∞>2

Relabel is required in node G.

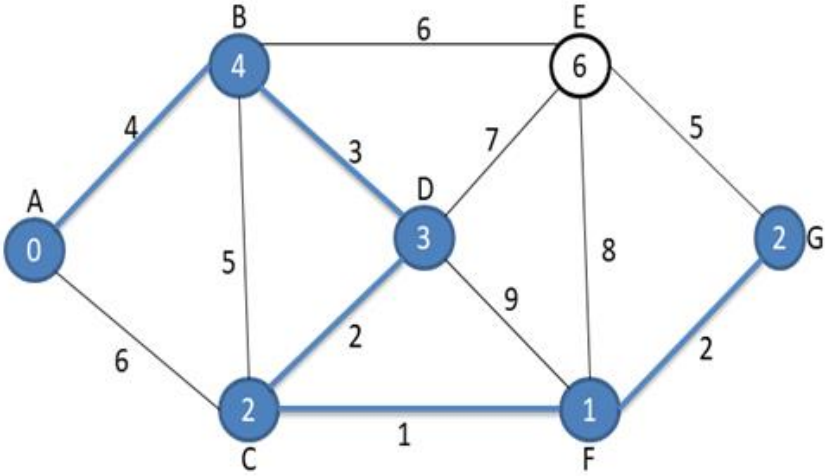
Distance (E)<distance of (F, E) =>6<8

Relabel is not required.

Table 6

NODES	A	B	C	D	E	F	G
DISTANCE d[v]	0	4	2	3	6	1	2
DISTANCE FROM Ψ[V]	A	A	D	B	D	C	F
STATUS	visited	visited	visited	visited	unvisited	visited	unvisited

The unvisited node G is the smallest distance from node F. So the current working node is G.



Step7: Calculate the unvisited nodes of current node G. The unvisited nodes are E.

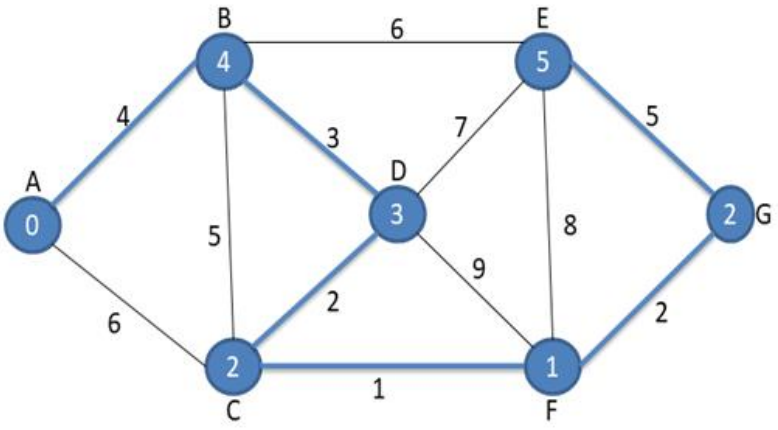
Distance (E)>distance of (G, E) => 6>5

Relabel is required in node E.

Table 7

NODES	A	B	C	D	E	F	G
DISTANCE d[v]	0	4	2	3	5	1	2
DISTANCE FROM Ψ [V]	A	A	D	B	G	C	F
STATUS	visited	visited	visited	visited	unvisited	visited	visited

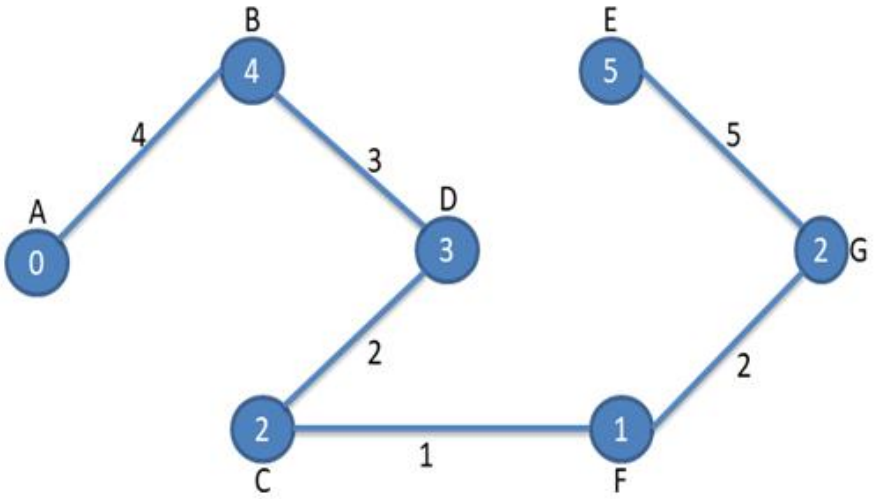
The only unvisited node E .So G has only visited to the node E.



The final table is:

Table 8

NODES	A	B	C	D	E	F	G
DISTANCE d[v]	0	4	2	3	5	1	2
DISTANCE FROM Ψ [V]	A	A	D	B	G	C	F
STATUS	visited	visited	visited	visited	visited	visited	visited



The weight of minimum spanning tree will be:

=(A,B)+(B,D)+(D,C)+(C,F)+(F,G)+(G,E) =4+3+2+1+2+5 =17.

III. CONCLUSION

This paper concludes that, by using a connected graph. We have applied a Prim's algorithm, for constructing the minimum spanning tree with a minimum possible total edge weights.

IV. REFERENCES

[1]. Review and analysis of minimum spanning tree using prim's algorithm. Jogamohan Medak Assistant Professor, North Lakhimpur College (Autonomous), North lakhimpur, Assam – India.

[2]. Minimum spanning tree. Available at https://en.wikipedia.org/wiki/Minimum_spanning_tree.

[3]. Prims Algorithm: Greedy Technique. Available at https://www.youtube.com/watch?v=kuNfKbyOnDs&t=529s.

[4]. Kairanbay, M., Jani, H. M. (2013). A Review and Evaluations of Shortest Path Algorithms. International journal of scientific & technology research volume 2, issue 6.