MINIMUM SPANNING TREE ALGORITHM

Vikas.C.S
SASTRA University
F-9, P.S.G Doctor's quarters,
Peelamedu, Coimbatore. 641004

ABSTRACT
An algorithm for minimum spanning tree[1] is discussed here. Apart from the traditional Kruskal's[2] and Prim's[3] algorithm for finding the minimum spanning tree, yet another algorithm for the same purpose is described here. Initially we form a forest and then we convert the forest into the minimum spanning tree

Categories and Subject Descriptors
Algorithm C.4 (Data Structures)

General Terms
1. Algorithm.

Keywords
Graph: It is a collection of nodes. It mainly consists of two elements –vertices and edges.

Vertex: It is simply drawn as a node or dot.

Edge: It is a line connecting two vertices.

Degree: It is the total number of edges incident on a vertex.

1. INTRODUCTION
The minimum spanning tree algorithm finds application in many areas like, proving optimal solution of greedy algorithm[4], approximate solution to minimum spanning tree problem[5], defining clusters in a dataset etc.

There have been two well known algorithms in the past for finding the minimum spanning tree. In this paper, I present yet another algorithm for solving the same problem. The "Vicky Algorithm", works by initial forest formation and converting the forest into the minimum spanning tree.

2. Algorithm description:
Primarily every node in the graph is considered individually and the shortest edge from that node is established. After all the minimum edges from all the nodes are established, we go verifying the output. If the best case is taken into account, the output of this step would be the minimum spanning tree itself. If, that is not the case, as we traverse from a start node to the last, we will encounter a break in the traversal. Thus all nodes will not be visited while traversal. And the outcome is the forest which is formed. Hence, to convert the forest so formed by the previous step of the algorithm, at the point of the break the next shortest edge is established. Then the same procedure from the traversal step is continued again till all the nodes are visited.

3. Algorithm:

```
Algorithm Vicky(U)
    Begin
                    /*
                    G is the given undirected graph with cost as cost adjacency matrix 'C'
                    N is the number of vertices
                    U is the chosen source vertex for each iteration
                    X is an element in the array
                    A[] is the array
                    V is the destination vertex
                    */
    For U= 1 to N
        Visited[U]=0
        Let <U,V> be the lightest edge between a chosen source vertex and any given destination vertex in graph 'G'
        It should be included in the forest
    End for
        U=1; i=1; count =1;
        Visited[U]=1;
        Push U into stack;
        Traverse(U);
        While count <N do
        If stack != Empty
                POP stack;
                Store POPed value in A[i];
                i++;
                U=Top(stack);
                Traverse(U);
        End if
        If stack=Empty
                Choose <X,V'>
                /*
            <X,V'> is the next shortest path
                */
```

```
            Visited [V']=1;
            U=V';
            Count++;
            Traverse(U);
            End if
Return

Traverse (U)
begin
If <U,V> exists  && V is not visited
            PUSH  'V' into stack;
            Visited[V]=1;
            U=V;
            Count++;
            Traverse(U);
End if
End
```

3.1 Time complexity:

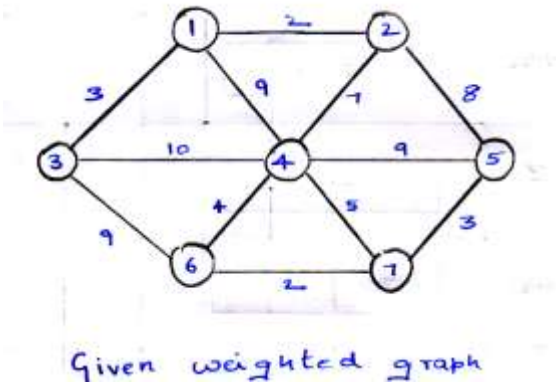
The calculated time complexity for the Vicky algorithm is $O(E^2 \log V)$ where E is the number of edges and V is the number of vertices.

Although the time complexity is the same as that of Kruskal's[6] and Prim's[7] algorithm, this approach has certain advantages over the other its predecessors.

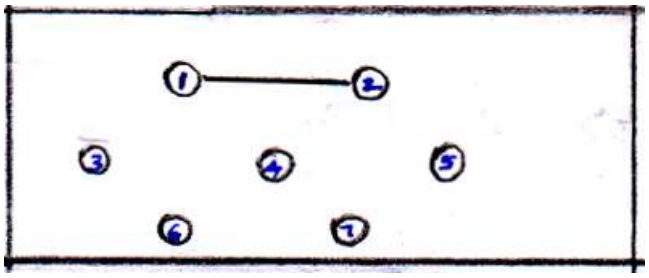
3.2 Advantages:

- This algorithm also takes the same running time as that of the previous two algorithms.
- The additional feature of VICKY algorithm is that, there no formation of loop. Hence running a check for occurrence of closed loop is completely avoided.

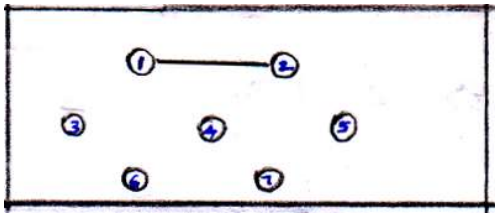
3.3 Test trace:



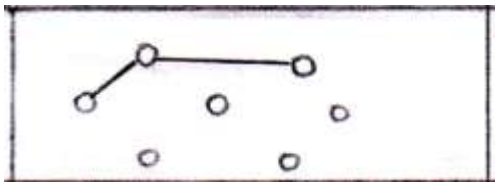
Step 1:



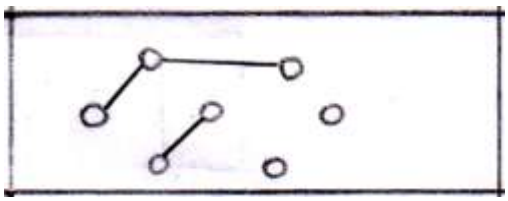
Step 2:



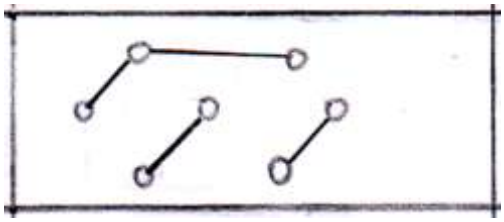
Step 3:



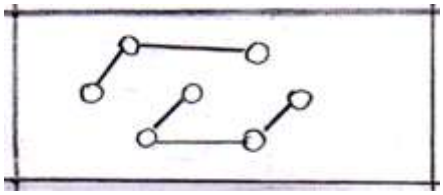
Step 4:



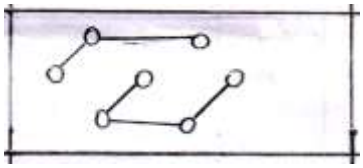
Step 5:



Step 6:

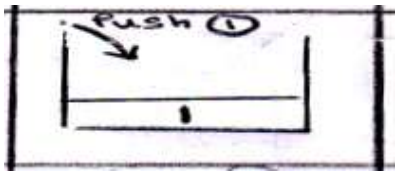


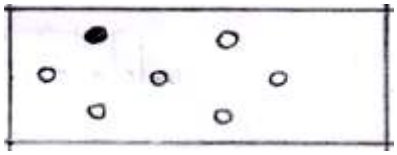
Step 7:



Step 8:

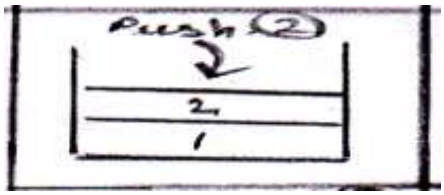
stack

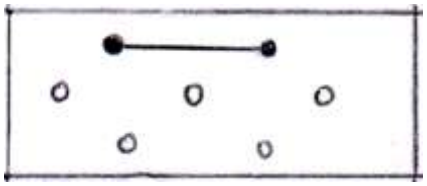




Step 9:

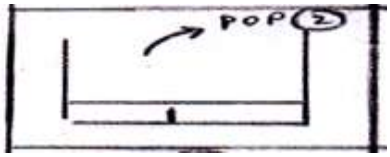
stack

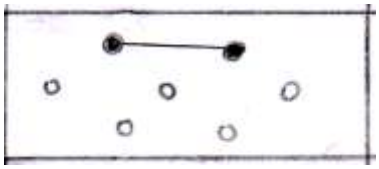




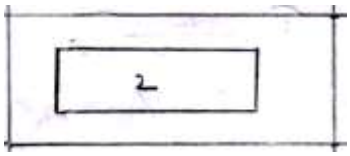
Step 10:

stack



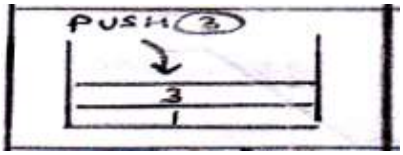


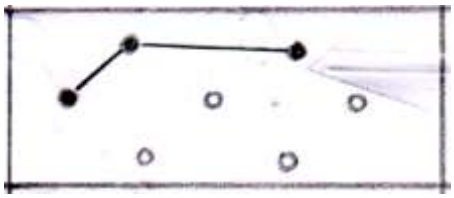
Array



Step 11:

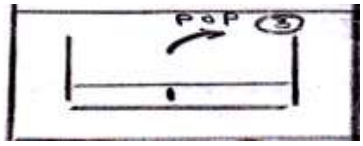
stack

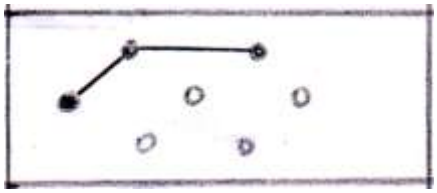




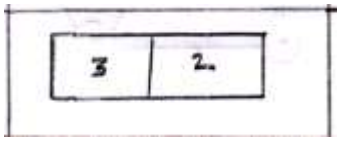
Step 12:

Stack



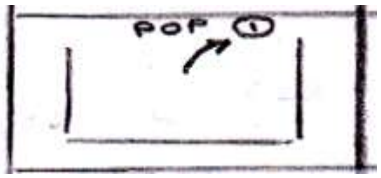


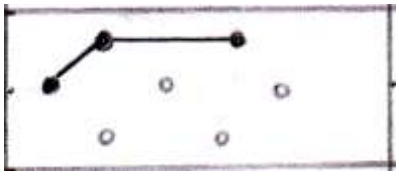
Array



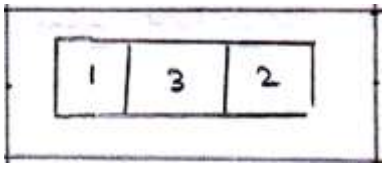
Step 13:

Stack



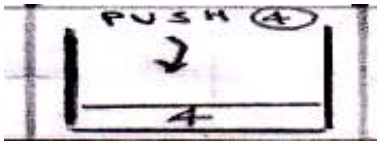


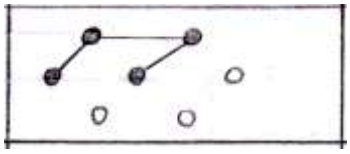
Array



Step 14:

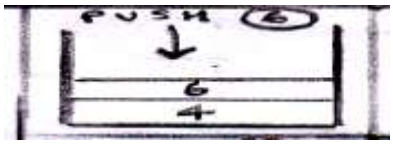
stack

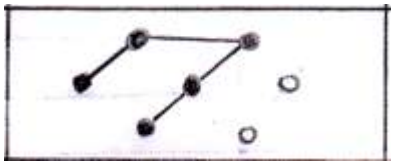




Step 15:

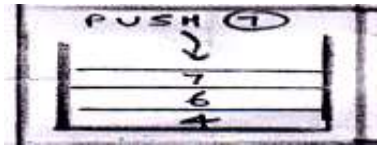
Stack

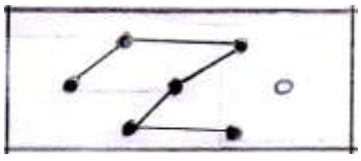




Step 16:

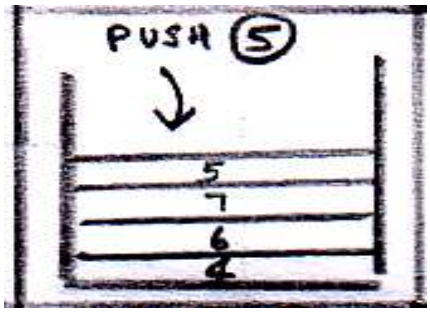
Stack

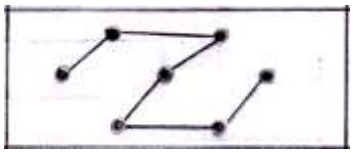




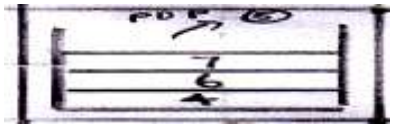
Step 17:

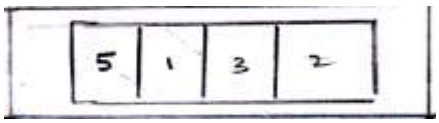
Stack



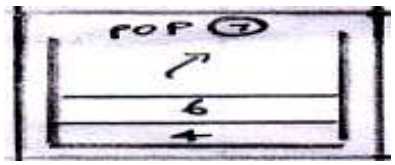


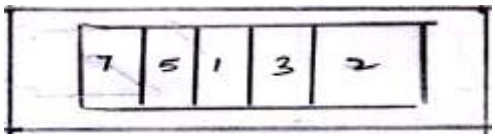
Step 18:



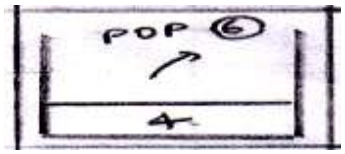


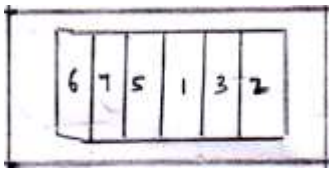
Step 19:



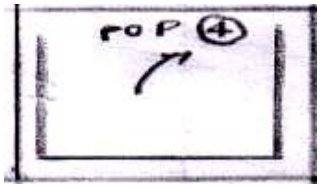


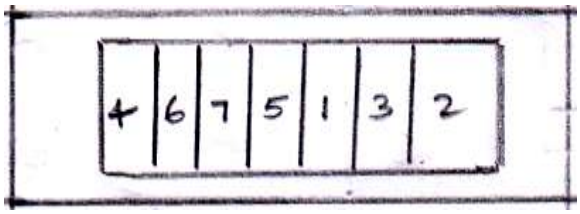
Step 20:



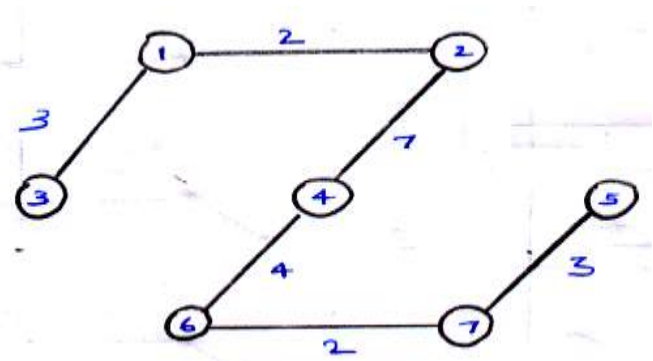


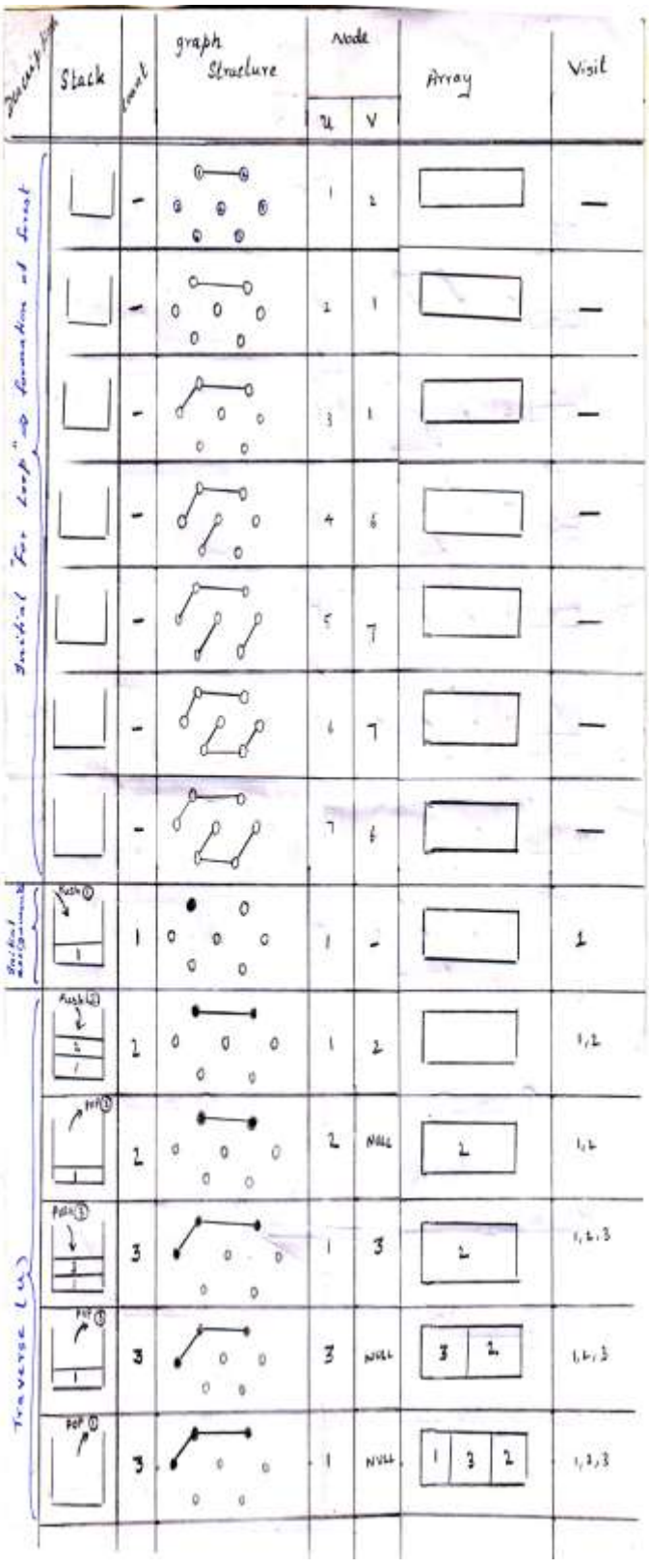
Step 21:

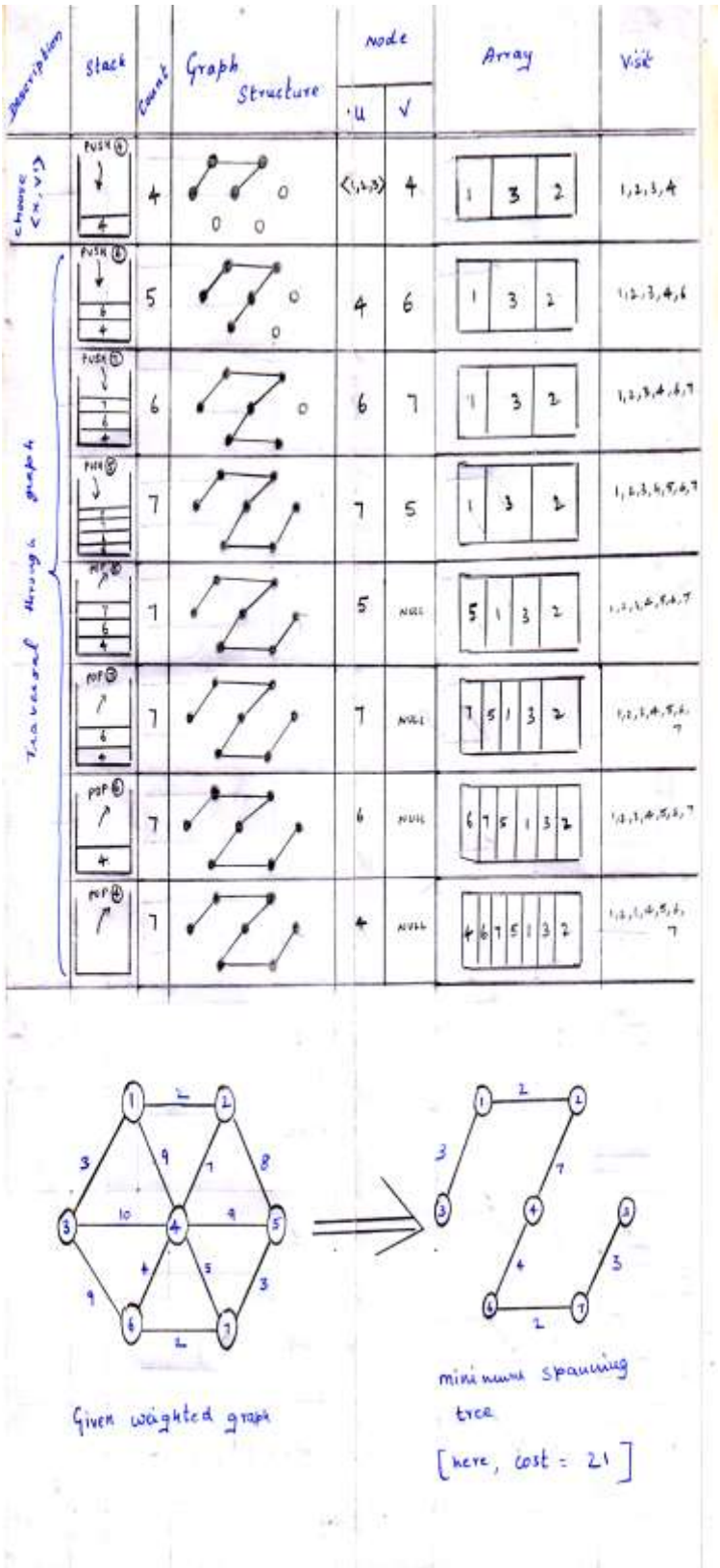




Final output:




Description
Stack
count
graph Structure
Node
u
v
Array
Visit
Initial for Loop" to formation of first
Initial assignment
Traverse (u)


Description
Stack
count
Graph Structure
Node
u
v
Array
Visit
choose (u, v)
Traversal through graph
Given weighted graph
minimum spanning tree
[here, cost = 21]

4. ACKNOWLEDGMENTS

Firstly I would like to thank God for giving me an opportunity to prove myself.

Secondly I would like to thank my FATHER, Mentor,

Dr. C.R.Srinivas (Professor and HOD of Dept. of Dermatology) for all the support he provided to me.

Next I would like to thank my guide, **Shri. R.Radhakrishnan (**Lecturer, SRC, SASTRA DEEMED UNIVERSITY)

LAST but not the least I would like to thank my friends, teachers and family members for their encouragement and moral support.

5. REFERENCES

[1] Thomas.H.Corman, Charles.E.Leisserson, Ronald.L.Rivest & Clifford Stein, "Introduction To Algorithms" Second Edition, Page 561.

[2] Thomas.H.Corman, Charles.E.Leisserson, Ronald.L.Rivest & Clifford Stein, "Introduction To Algorithms" Second Edition, Page 567.

[3] Thomas.H.Corman, Charles.E.Leisserson, Ronald.L.Rivest & Clifford Stein, "Introduction To Algorithms" Second Edition, Page 577.

[4] http://en.wikipedia.org/wiki/Greedy_algorithm

[5] http://en.wikipedia.org/wiki/Travelling_salesman_problem

[6] http://en.wikipedia.org/wiki/Kruskal's_algorithm

[7] http://en.wikipedia.org/wiki/Prim%27s_algorithm