Indian Journal of Engineering
International Journal for Engineering
ISSN 2319 – 7757 EISSN 2319 –7765

A study on undirected minimum cost spanning tree

Publication History
Received: 10 July 2015
Accepted: 15 August 2015
Published: 1 September 2015

Citation
Jasmine Priskilla D. A study on undirected minimum cost spanning tree. *Indian Journal of Engineering*, 2015, 12(30), 176-186

A study on undirected minimum cost spanning tree

D. Jasmine Priskilla

Research scholar, Bharathiyar University, priskillajas@yahoo.com

ABSTRACT

Minimum spanning tree can be obtained for connected weighted edges with no negative weight using classical algorithms such as Prim's and Kruskal. This paper presents a comparison study on the classical and the more recent algorithms with different techniques. This paper also contains comparisons of MST algorithm and their advantages and disadvantages.

Keywords – Graph, MST, Tree, Kruskal, Prim, Undirected graph

I. INTRODUCTION

The minimum spanning tree (MST) problem is, given a connected, weighted, and undirected graph G =(V; E; w), to find the tree with minimum total weight spanning all the vertices V. Here w: E -> R is the weight function. The problem is frequently defined in geometric terms, where V is a set of points in d-dimensional space and w corresponds to Euclidean distance. The main distinction between these two settings is the form of the input. In the graph setting the input has size O(m + n) and consists of an enumeration of the n = |V| vertices and m = |E| edges and edge weights. In the geometric setting the input consists of an enumeration of the coordinates of each point (O(dn)space):all $\binom{v}{2}$ edges are implicitly present and their weights implicit in the point coordinates

1.1 Applications of MST

Boruvka invented the MST problem while considering the practical problem of electrifying rural Moravia(present day Czech Republic) with the shortest electrical network. MSTs are used as a starting point for heuristic approximations to the optimal traveling salesman tour and optimal Steiner tree, as well as other network design problems. MSTs are a component in other graph optimization algorithms, notably the single-source shortest path algorithms of Thorup [2] and Pettie–Ramachandran [1]. MSTs are used as a tool for visualizing data that is presumed to have a tree structure; for example, if a matrix contains dissimilarity data for a set of species, the minimum spanning tree of the associated graph will presumably group closely

related species; see [3]. Other modern uses of MSTs include modeling physical systems [5] and image segmentation [4].

1.2 Objective of MST

- To minimize cost of the tree spanning tree for both directed and undirected.
- To minimize load on the network.
- To eliminate the cycle from the graph from the MST.
- To improve the complexity of the MST.

II. MST ALGORITHMS

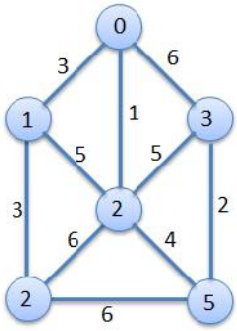
There are various classical algorithms available which describe below.

Kruskal's and Prim's algorithm is a greedy algorithm which used to find a minimum spanning tree for a connected weighted undirected graph. This means when the total weight of all the edges is minimized in the tree, at that time it finds a subset of the edges which forms a tree which includes every vertex.

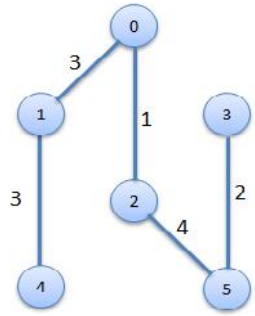
2.1 Kruskal's Algorithm

Kruskal's algorithm is a greedy algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. It finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. This algorithm is directly based on the MST(minimum spanning tree) property.

Example



A Simple Weighted Graph

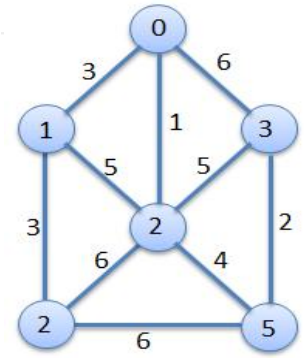


Minimum-Cost Spanning Tree

Kruskal's Algorithm

```
MST-KRUSKAL(G, w)
1.  A ← Ø
2.  for each vertex v V[G]
3.      do MAKE-SET(v)
4.  sort the edges of E into nondecreasing order by weight w
5.  for each edge (u, v) E, taken in nondecreasing order by
                                weight
6.      do if FIND-SET(u) ≠ FIND-SET(v)
7.          then A ← A {(u, v)}
8.              UNION(u, v)
9.  return A
```

Example



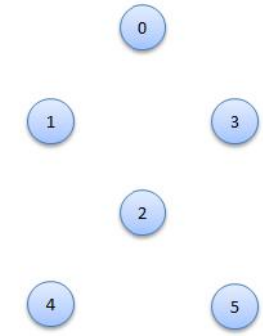
Step1. Edges are sorted in ascending order by weight.

Edge No.	Vertex Pair	Edge Weight
E1	(0,2)	1
E2	(3,5)	2
E3	(0,1)	3
E4	(1,4)	3
E5	(2,5)	4
E6	(1,2)	5
E7	(2,3)	5
E8	(0,3)	6
E9	(2,4)	6

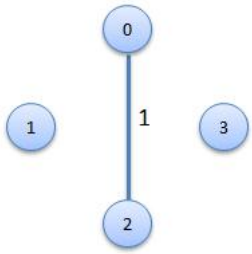
E10	(4,5)	6

Step2. Edges are added in sequence.

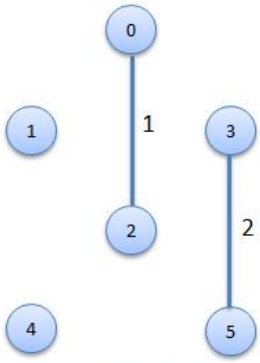
Graph



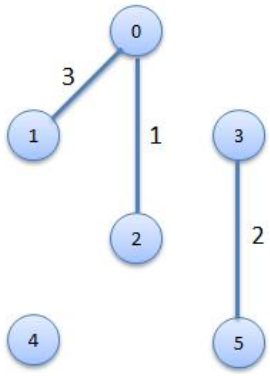
Add Edge E1



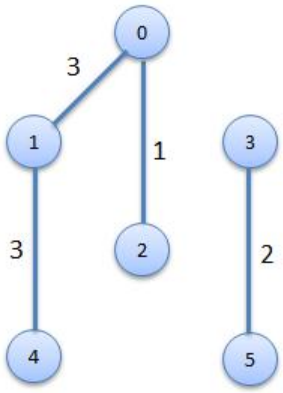
Add Edge E2



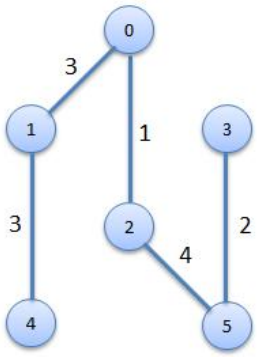
Add Edge E3



Add Edge E4



Add Edge E5



Total Cost = 1+2+3+3+4 = 13

Running Time = O(m log n) (m = edges, n = nodes)

Testing if an edge creates a cycle can be slow unless a complicated data structure called a "union-find" structure is used. It usually only has to check a small fraction of the edges, but in some cases (like if there was a vertex connected to the graph by only one edge and it was the longest edge) it would have to check all the edges. This algorithm works best, of course, if the number of edges is kept to a minimum.

Advantages are:

1) Easy to understand

2) Give good result for large number of vertices and edges.

Disadvantages are:

1) Difficulty of checking whether arcs form cycles makes it slow and hard to program

2) Same weight may increase the complexity.

Prim's Algorithm

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a connected weighted undirected graph. It finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. This algorithm is directly based on the MST(minimum spanning tree) property.

Prim's Algorithm

```
MST-PRIM(G, w, r)
1.  for each u V [G]
2.  do key[u] ← ∞
3.  π[u] ← NIL
4.  key[r] ← 0
5.  Q ← V [G]
6.  while Q ≠ Ø
7.        do u ← EXTRACT-MIN(Q)
8.           for each v Adj[u]
9.                 do if v Q and w(u, v) < key[v]
10.                      then π[v] ← u
11.                           key[v] ← w(u, v)
```

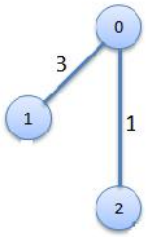
Procedure for finding Minimum Spanning Tree

Step1

No. of Nodes	0	1	2	3	4
Distance	0	3	1	6	∞
Distance From		0	0	0	

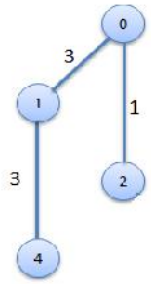
Step2

Distance	0	3	0	5	6	4
Distance From		0		2	2	2
No. of Nodes	0	1	2	3	4	5



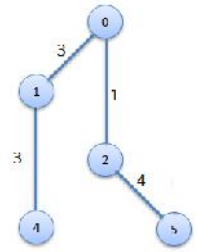
Step3

No. of Nodes	0	1	2	3	4
Distance	0	0	0	5	3
Distance From				2	1



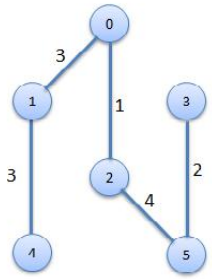
Step4

No. of Nodes	0	1	2	3	4
Distance	0	0	0	5	0
Distance From				2	



Step5

No. of Nodes	0	1	2	3	4
Distance	0	0	0	3	0
Distance From				2	



Minimum Cost = 1+2+3+3+4 = 13

Running Time = O(m + n log n) (m = edges, n = nodes)

If a heap is not used, the run time will be O(n^2) instead of O(m + n log n). However, using a heap complicates the code since you're complicating the data structure. A Fibonacci heap is the best kind of heap to use, but again, it complicates the code.

Unlike Kruskal's, it doesn't need to see the entire graph at once. It can deal with it one piece at a time. It also doesn't need to worry if adding an edge will create a cycle since this algorithm deals primarily with the nodes, and not the edges.

For this algorithm the number of nodes needs to be kept to a minimum in addition to the number of edges. For small graphs, the edges matter more, while for large graphs the number of nodes matters more.

Advantages are:

1) Easy to understand.

2) Root node is selected so clear about the starting node.

Disadvantages are:

1) Time taken to check for smallest weight arc makes it slow for large numbers of nodes.

2) Difficult to program, though it can be programmed in matrix form

3) Same weight may increase the complexity when one of the weights is eliminated in a cycle

III. LITERATURE SURVEY

In this section, lot of research works has been recorded from past few years. They are presented here:

Modified prim's algorithm:

In this algorithm, Instead of choosing randomly, root node is chosen with minimum edge weight. Remaining procedure is same as used by prims. Due to this only minimum weight edges are included. Although complexity remains same as prim's algorithm.

Advantages are:

It gives slightly better performance in case where minimum weight edge is required from the starting phase of minimum spanning tree formation.

Disadvantages are:

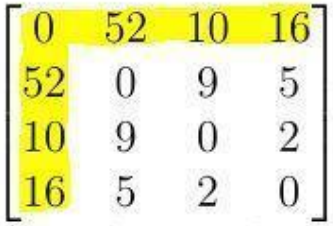
Complexity is remaining same.

Visit, Mark and Construct MST algorithm:

In this algorithm, adjacency matrix is used which help to reduce the step at the time of constructing MST. This method is based on the Kruskal algorithm with modification which used improve the complexity of the MST algorithm for the undirected graph. This method is purely for the undirected graph. So the weight of the 1 to 2 vertices is same for the 2 to 1 vertices. See the below Fig.1. In which edges 1 to 2 contain 52 weights and the weight for the edges 2 to 1 is also 52.So, it is same for undirected graph.

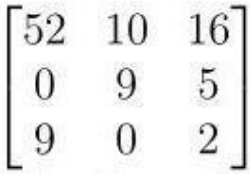
Fig.1 n*n weighted matrix

Now for the vertices 1 to 1, 2 to 2....n to n. there is no weight and if there is a weight then it automatically removed because of generating a cycle. So, it places 0 as infinite. Now if we have n vertices then we have n*n adjacency matrix. So, need to perform n^2 steps, because it will check all the elements from the graph. So, author first removes unused row column from the adjacency matrix. Here first row and last column are never used during the implementation because edges 1 to 2 has the same 2 to 1 and edges 1 to 1 is always 0 or automatically eliminated because of generating cycle. So, adjacency matrix has (n-1) rows and (n-1) columns. See the below adjacency Fig.2 and Fig.3.



$$\begin{bmatrix} 0 & 52 & 10 & 16 \\ 52 & 0 & 9 & 5 \\ 10 & 9 & 0 & 2 \\ 16 & 5 & 2 & 0 \end{bmatrix}$$

Fig.2 Reducing the n*n order matrix to order m*m where m = (n-1)



$$\begin{bmatrix} 52 & 10 & 16 \\ 0 & 9 & 5 \\ 9 & 0 & 2 \end{bmatrix}$$

Fig.3 m*m operational weight Matrix

So, n^2− 2n +1 steps will be performed. So, complexity is O (m^2) where m is (n-1) [7]. This algorithm works in the following two passes. 1) Mark Phase: In which algorithm marks the candidate edge from the graph for the minimum spanning tree. 2) MST Construction Phase: In the second phase, the algorithm constructs the desired minimum spanning tree T including only the marked edges from the upper triangular weight matrix M, which were marked during Marking Pass.

In this algorithm, minimum weight are marked and visited first. Once weight is visited and it doesn't create a cycle then it will be added to the list of minimum spanning tree edges. Otherwise it will be removed and next minimum weight should be taken for the further

procedure. This will happen till n-1 edges get for the minimum spanning tree edges. Once the n-1 edges are get than it will stop algorithm and calculate total cost.

Advantages are:

Complexity is O(n) for the best case.

Disadvantages are:

Complexity is O(n^2) for the best case.

IV. CONCLUSION

This paper presents classical algorithms and advance MST algorithm and it is observed that complexity is very high because of cycle in the graph and the edges with the same weight. It also observed that complexity can be improved using following steps. 1) Marked and visit the maximum weight of the edges. 2) If it creates a cycle then eliminate those edges from the current graph. 3) Above two steps will be repeated till edges = vertices-1. Prim's algorithms span from one node to another while Kruskal's algorithm select the edges in a way that the position of the edge is not based on the last step. In prim's algorithm, graph must be a connected graph while the Kruskal's can function on disconnected graphs too. Prim's algorithm has a time complexity of $O(V^2)$, and Kruskal's time complexity is O(logV).

References

1. Pettie, S., Ramachandran, V.: A shortest path algorithm for real weighted undirected graphs. SIAM J. Comput. **34**(6), 1398–1431 (2005)
2. Thorup, M.: Undirected single-source shortest paths with positive integer weights in linear time. J. ACM **46**(3), 362–394 (1999)
3. Graham, R.L., Hell, P.: On the history of the minimumspanning tree problem. Ann. Hist. Comput. **7**(1), 43–57 (1985)
4. Ion, A., Kropatsch, W.G., Haxhimusa, Y.: Considerations regarding the minimum spanning tree pyramid segmentation method. In: Proc. 11th Workshop Structural, Syntactic, and Statistical Pattern Recognition (SSPR). LNCS, vol. 4109, pp. 182–190. Springer, Berlin (2006)
5. Subramaniam, S., Pope, S.B.: A mixing model for turbulent reactive flows based on Euclidean minimum spanning trees. Combust. Flame **115**(4), 487–514 (1998)
6. http://scanftree.com/Data_Structure/prim's-algorithm
7. http://delab.csd.auth.gr/~manolopo/graph/Lec9_MST.ppt