Journal of Physical Sciences, Vol. 13, 2009, 209-216
ISSN: 0972-8791: www.vidyasagar.ac.in/journal

Generation of Minimal Spanning Tree Based on Analytical Perspective of Degree Sequence

Sanjay Kumar Pal[1], Samar Sen Sarma[2] and Puspita Manna[3]

[1]Department of Computer Science and Applications
NSHM College of Management and Technology
B. L. Saha Road, Kolkata, INDIA
E-mail: pal.sanjaykumar@gmail.com

[2]Department of Computer Science & Engineering
University of Calcutta
Kolkata, INDIA
E-mail: sssarma2001@yahoo.com

[3]Department of OCLAN
Bharat Sanchar Nigam Limited
Champahati Telephone Exchange, West Bengal Circle
E-mail: puspita3@yahoo.com

Received August 17, 2009; accepted October 21, 2009

ABSTRACT

This paper considers generation of Minimal Spanning Trees (MST) of a simple symmetric and connected graph G. In this paper, we propose a new algorithm to find out minimum spanning tree of the graph G based on the degree sequence factor of nodes in graph. The time complexity of the problem is less than $O(N \log |E|)$ compared to the existing algorithms time complexity, $O(|E| \log |E|) + C$ of Kruskal algorithm, which is optimum. The goal is to design an algorithm that is simple, elegant, efficient, easy to understand and applicable in the field of networking design, mobile computing and engineering.

Keywords: Graph, Tree, Sequence, Degree Factor, MST, Algorithm.

1. Introduction

Graph theory is one of the rapidly developing branches of mathematics and finds wide applicability in computer science [8]. Most of the mathematical and scientific problems can be formulated in terms of Graph Theory [3]. It is also applied in social sciences, linguistic, physical sciences, communication engineering and plays an important role in switching theory, artificial intelligence, formal languages, computer graphics, operating systems, compiler writing, information

organization, and retrieval [3, 4, 5]. Graphs, especially trees and binary trees are widely used in the representation of data structure [4, 5, 6].

A Spanning Tree is a tree of a connected graph G, which connect all vertices of the graph. Generation of a single spanning tree for a simple, symmetric and connected graph G is well known polynomial time solvable problem [3, 4]. Enumeration of spanning tree in undirected simple connected graphs is an important issue in many engineering networking design, mobile communication and scientific problems [10, 13]. Applying graph theory easily solves most of the problems in the fields like networking and circuit analysis. In 1981 coauthor Samar Sen Sarma published in his paper an algorithm, one of the most important graph theory problems; generation of all spanning trees of a simple connected graph [1]. The spanning trees generated by this algorithm are all distinct i.e. there is no possibility of generation of duplicate spanning trees, and also prohibit generation of all the non-tree sub-graphs.

Many practical applications, particularly design of electrical circuits, communication networks and transportation networks can be formulated as optimization of minimal spanning tree problem [1, 2, 3, 10, 14]. The goal of optimization of minimal spanning tree is to find a solution that is appropriate for a particular application. When studying diverse problems, one often makes an assumption of general position: for minimal spanning trees, one can infinitesimally perturb the edge weights so that all are distinct; in this way picking out a unique solution. Several algorithms exist for generation of Minimal Spanning Tree. Otakar Boruvka described an algorithm for finding a Minimal Spanning Tree in a graph for which all the edge weights are distinct [11]. In 1957, Computer Scientist C. Prim discovered another algorithm that finds a minimal spanning tree for a connected weighted graph [3]. This algorithm continuously increases the size of a tree starting with a single vertex until it spans over all the vertices. This algorithm was actually discovered in 1930 by mathematician Vojtech Jarnik. Joseph Kruskal described another minimal spanning tree algorithm where total weight of all the edges of the tree is minimized [3, 4]. Edsger Dijkstra in 1959 discovered a minimal spanning tree algorithm that solves the single source-shortest path problem for a directed graph with non-negative edge weights [3, 4, 5, 6].

Several distinct techniques exist for generation of all spanning trees of a graph[1, 2, 13]. In 2007, Authors have discussed an algorithm where trees are generated by examining ${}^{e}C_{n-1}$ sets of edges where e is the number of edges and n is the number of vertices of a simple connected graph eliminating some set of edges which form circuit [2]**.**Here, in this paper we introduce a new algorithm for generation of minimal weight spanning tree of a graph which requires less execution time and memory space compared to the existing algorithm. The algorithm is based on the degree factor of the degree sequence and the weight of edges in the graph G. A sequence $d_1, d_2, d_3, d_4, ..., d_n$ of nonnegative integers is called a degree sequence of given graph G, if the vertices of G can be labeled $v_1, v_2, v_3, v_4, \ldots, v_n$ so that degree $v_i = d_i$; for all i [7]. The sum of the integers $d_1, d_2, d_3, d_4, ..., d_n$ is

equal to 2e, where e is the number of edges in a graph G. For a given graph G, a degree sequence of G can be easily determined [7, 9]. Now the question arises that, given a sequence $\xi = d_1, d_2, d_3, d_4, ..., d_n$ of nonnegative integers, under what conditions does there exist a graph G? A necessary and sufficient condition for a sequence to be graphical was found by Havel and later rediscovered by Hakimi [7, 9, 12]. Based on the above concept here, we introduce a new technique to find out a minimum spanning tree of a graph G considering the degree sequence factor of the nodes. The time complexity of the new algorithm is optimal in comparison to the algorithms of Kruskal and Prim and also optimized the space complexity as well in the new algorithm.

2. Terminology

2.1 Realisation:

A sequence $\xi = d_1, d_2, d_3, d_4, ..., d_n$ of nonnegative integers is said to be graphic sequence if there exists a graph G whose vertices have degree d_i and G is called realization of ξ.

2.2 Spanning Tree: A Spanning Tree S is a tree of a connected graph G, which touches all vertices of the graph. A spanning tree has n vertices and exactly (n-1) edges of a graph G.

2.3 Minimal Spanning Tree: Let *G* be a connected, edge-weighted graph. A *minimal spanning tree* is a subgraph of *G* that satisfies the following properties:

- It is a *tree*, that is, it is connected and has no cycles.
- It is *spanning*, that is, it contains all vertices of *G*.
- It has *minimal total edge-weight* among all possible trees.

2.4 Adjacency Matrix: For a graph G of n vertices and e edges, if, set of vertices, $V(G) = \{v_1, v_2, v_3, \ldots\ldots, v_n\}$ and set of edges $E(G) = \{e_1, e_2, e_3, \ldots\ldots, e_n\}$. The adjacency matrix A, of weighted graph G, is $n \times n$ matrix and it can be represent by $A = [a_{ij}]$, where

$$a_{ij} = \begin{cases} w_{ij} & \text{if there is an edge between } v_i, v_j \in E(G) \text{ and } w_{ij} \text{ is weight of edge} \\ 0 & \text{if there is no edge} \end{cases}$$

2.5 Degree of a vertex: The degree d_i of a vertex v_i in a graph G is the number of edges connected with v_i. In other words, degree d_i is the number of vertices adjacent to the vertex v_i.

2.6 Node Degree Factor: Node degree factor of each node v_i is the ratio between summations of degree of all nodes of graph G and degree of a particular node i.e.

$$Node\ Degree\ Factor = \frac{Summation\ of\ Degree\ of\ nodes\ of\ graph}{\deg ree\ of\ a\ node} = \frac{\sum_{i=1}^{n} d_i}{d(v_i)}$$

3. Minimal Spanning Tree Generation

A tree having n nodes and n-1 edges is spanning tree of a graph. A preferable and efficient algorithm is one that generates trees by selecting only the minimal cost edges of the graph in such a way that it will not produce cycle. The present algorithm is not required to test circuits for the generation of minimum spanning tree. Therefore, the new spanning tree generation algorithm is more efficient in terms of the time complexity and required execution time. In this algorithm, first we calculate degree factor of every node and then identify a node with maximum degree factor. This node will be used to identify a minimum weight edge incident to it. Then the graph G is to be reconstructed by removing the node of higher degree factor and its incident edges. This minimum weight edge to be included in the list of minimum spanning tree (MST), S. This process is to be continued till the (n-1) edges does not include in the MST, S.

Theorem 1: A spanning tree S of a weighted connected graph G is the minimal weight spanning tree if and only if there exist no other spanning tree of G at a distance of one from S whose weight is smaller than that of S.

Proof: Let S_1 be a spanning tree in graph G satisfying the hypothesis of the theorem there is no spanning tree at a distance of one (of G) from S_1 which is smaller than S_1. If S_2 is the smallest spanning tree in G, the weight of S_1 will also be is equal to that of S_2. The spanning tree S_2 is smallest if and only if, it satisfies the hypothesis of the theorem.

Suppose, an edge e in S_2 is selected based on the higher degree factor of node in the graph G but it is not in S_1. Adding e to S_1 forms a fundamental circuit with branches of S_1. Some of the branches in S_1 that form fundamental circuit with e in S_2; each of the branches of S_1 has weight either smaller than or equal to e because S_1 is minimal weight. Amongst all these edges of circuit but not in S_2, at least one, say b, must form fundamental circuit in S_2 containing e. So, b must have same weight as e. Therefore, spanning tree $\overline{S}_1 = (S_1 \cup (e-b))$, obtained from S_1, though one cycle exchange, has same weight as S_1. S_1 has one more edge common with S_2 and it satisfies the condition of theorem.

Theorem 2: An edge e corresponding to the node of highest degree factor in the graph G forms a spanning tree, if it has minimal weight.

Proof: A spanning tree S of a graph G contains all the vertices (exactly once) and n-1 edges, where n is the number of vertices in the graph G. An edge e is to be selected based on the degree factor of the node. The degree factor of the node shows how

many edges are incident to a particular node. The highest degree factor of a node, the number of edges incident to which is minimum with at least one edge whose weight is minimal, is to be included in the minimum spanning tree S.

Theorem 3: The combination of n-1 distinct edges forms spanning tree according to theorem 1, if it is circuitless.

Proof: The n-1 edges combinations of a graph must contain all the vertices of the graph. These combinations obviously either contain a circuit or a spanning tree of the graph. Since, this algorithm prohibits the generation of circuit, therefore, n-1 distinct edge combination form spanning tree.

Theorem 4: A Sequence $D = d_1, d_2, d_3, d_4, ..., d_n$ of nonnegative integers with $d_1 \geq d_2 \geq d_3 \geq d_4 .... \geq d_n$, n $\geq$ 2, $d_1 \geq 1$ is graphical if and only if the sequence $\mathrm{D}' = \mathrm{d}_2 - 1, \mathrm{d}_3 - 1, \mathrm{d}_4 - 1, ..., \mathrm{d}_{\mathrm{d}_1+1} - 1, \mathrm{d}_{\mathrm{d}_1+2}, ..., \mathrm{d}_{\mathrm{n}}$ is graphical [1,2].

Proof: Let D' is a graphical sequence. There exists a graph G' of order $n-1$, such that D' is the degree sequence of G'. Therefore, the vertices of G' can be labeled as $\mathrm{V}_2, \mathrm{V}_3, \ldots, \mathrm{V}_{\mathrm{n}}$; such that

$$\deg(V_i) = \begin{vmatrix} d_i - 1; 2 \leq i \leq d_1 + 1 \\ d_i; d_1 + 2 \leq i \leq 1 \end{vmatrix}$$

A new graph G can be constructed by adding a new vertex V_1 and the d_1 edges $V_1V_i; 2 \leq i \leq d_1 + 1$. Then in G, $\deg(\mathrm{V_i}) = \mathrm{d_i}$ *for* $1 \leq i \leq n$ and so $D = \mathrm{d}_1, \mathrm{d}_2, \mathrm{d}_3, \mathrm{d}_4, \ldots, \mathrm{d}_n$ is graphical.

Conversely, let D be a graphical sequence. Hence there exist graphs of order n with degree sequence D. Among all such graphs let G be one, such that $\mathrm{V(G)} = \{\mathrm{V}_1, \mathrm{V}_2, \mathrm{V}_3, \mathrm{V}_4, \ldots, \mathrm{V}_{\mathrm{n}}\}$; $\deg(\mathrm{V_i}) = \mathrm{d_i}$ for $\mathrm{i} = 1,2,3,\ldots,\mathrm{n}$ and the $\sum d_i =$ even number, the sum of degrees of the vertices adjacent with V_1 is maximum. We show first that V_1 is adjacent to vertices having degrees $\mathrm{d}_2, \mathrm{d}_3, \mathrm{d}_4, \ldots, \mathrm{d}_{\mathrm{d}_1+1}$.

Suppose, to the contrary, that V_1 is not adjacent to vertices having degrees $\mathrm{d}_2, \mathrm{d}_3, \mathrm{d}_4, \ldots, \mathrm{d}_{\mathrm{d}_1+1}$. Then there exist vertices V_r and V_s with $\mathrm{d_r} \succ \mathrm{d_s}$ such that V_1 is adjacent to V_s, but not to V_r. Since, the degree of V_r exceeds that of V_s, there exists a vertex V_t, such that V_t is adjacent to V_r but not to V_s. Removing the degrees V_1V_s and V_rV_t and adding the edges V_1V_r and V_sV_t results in a graph $\overline{G}$

having the same degree sequence as *G*. However, in $\overline{G}$ the sum of the degrees of the vertices adjacent to V_1 is larger than that in *G*, contradicting the choice of *G*. Thus, V_1 is adjacent with vertices having degrees $d_2, d_3, d_4, \ldots, d_{d_1+1}$, and the graph $(G - V_1)$ has degree sequence D', so D' is graphical.

4. Algorithm for Generation of Minimal Spanning Tree

Initially we generate random weighted graph according to the given number of nodes and edge density. The weight matrix of the randomly generated graph is used as input for generation of minimum spanning tree of the graph. The output of the algorithm is minimum weight spanning tree where each node of the graph is represented by the edge number from 0,1,2,……..,n. The weight, w of the edge is stored in the adjacency matrix if there is an edge between the nodes.

Step 1: Generate random weighted graph and corresponding weight matrix according to the given number of nodes and edge density.
Step 2: Calculate Degree Factor of all the nodes of graph G.
Step 3: Select a minimum weight edge $e_i \in E$ of a node $v \in G$ whose degree factor is higher compared to other nodes.
Step 4: Put e_i into MST, S and construct new graph G' removing the highest degree factor node and its incidence edges from G.
Step 5: Find out the degree factor of newly constructed graph G'.
Step 6: Apply iteratively from step 2 to step 5, till (n-1) edges are not selected in MST.
Step 7: Calculate sum of the weight of the edges in the MST, S.
Step 8: Stop.

6. Complexity of the Proposed Algorithm

The Circuit testing is not required in the minimum spanning tree generation algorithm because we have always chosen a node v in the graph G exactly once for the MST, S. The sorting of edges and finding the minimum weight edges neighbors of the constructed tree is not required. The time complexity of new algorithm is less than $O(N \log |E|)$ and it is reduced due to non requirement of checking of circuit in generation of tree. The memory space required to execute the program of new algorithm is n^2 where n is the number of vertices of the graph G.

7. Results and Conclusion

Hardware used to carry out this experiment is Pentium IV computer and 1 GB DDR2 RAM. The program is written in 'C' programming language and Turbo 'C' compiler is used for compilation and execution purpose. The experiment has been performed on several graphs of different types.

The storage requirement of this algorithm is proportional to n^2. The experimental result of the algorithms is given in Table 1 as comparative study.

No. of Nodes	No. of Edges	Execution Time of Algorithm * 100 (in Second)		
		Kruskal	Prim	New Algorithm
3	3	1.70	1.95	1.70
4	4	4.56	4.46	4.54
4	5	4.67	4.72	4.61
5	8	7.36	7.42	7.42
6	12	8.84	8.90	8.67
7	13	18.78	16.75	16.32
7	17	19.12	20.59	18.88
8	25	18.89	18.89	18.44
9	21	24.44	21.75	21.44
9	32	37.35	31.69	30.89
10	18	33.28	32.76	32.14
10	42	38.66	36.80	35.78
11	27	36.80	36.74	35.83
11	50	40.18	37.56	36.47
12	20	36.68	37.14	35.82
12	53	51.30	40.32	39.22
13	31	73.60	51.74	50.98
13	70	79.94	55.03	53.24
14	55	71.94	67.87	65.93
14	82	124.68	120.20	117.32
15	32	87.60	73.68	72.99
15	72	121.65	101.85	99.12
16	48	98.25	74.35	72.67
16	108	123.70	118.65	114.21
17	41	121.40	109.30	106.78
17	55	144.75	124.55	120.27
18	76	146.35	125.85	121.02
19	68	177.95	156.80	151.11
20	95	317.60	231.81	223.73
21	74	343.80	312.60	292.89
22	69	530.6	292.2	280.11
23	126	596.40	336.63	318.32
24	82	561.22	441.02	424.79
25	78	540.34	434.77	408.31
30	30	662.32	616.43	596.33

Table1: Execution time of Kruskal, Prim and New algorithms

REFERENCES

1. A. Rakshit, A. K. Choudhury, S. S. Sarma and R. K. Sen, "An Efficient Tree Generation Algorithm," IETE, vol. 27, pp. 105-109, 1981.
2. Sanjay Kumar Pal and Samar Sen Sarma, "An Efficient All Spanning Trees Generation Algorithm", IJCS, vol. 2, No. 1, pp. 48 – 59, January 2008.
3. N. Deo, "Graph Theory with Application to Engineering and Computer Sciences," PHI, Englewood Cliffs, N. J, 2007.
4. Thomas H. Coremen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, "Introduction to Algorithms", PHI, Second Edition, 2008.
5. Harowitz Sahnai & Rajsekaran, "Fundamentals of Computer Algorithms", Galgotia Publications Pvt. Ltd., 2000.
6. Sanjoy Dasgupta, Christos Papadimitriou, Umesh Vazirani, "Algorithms", Tata McGraw-Hill, First Edition, 2008.
7. Arumugam S. and Ramachandran S., Invitation to Graph Theory, Scitech Publications (INDIA) Pvt. Ltd., Chennai, 2002.
8. R.J. Wilson, "History of Graph Theory", Section 1.3, Handbook of Graph Theory, pp. 29 – 49, 2004.
9. F. A. Muntaner-Batle and M. Rius Font, "A Note on degree Sequence of Graphs with restrictions", http://upcommons.upc.edu/eprints/bitstream/2117/1490/1/sequences.pdf
10. J. A. Bondy and U. S. R. Murty, "Graph Theory with Applications", The Macmillan Press, Great Britain, 1976.
11. http://en.wikipedia.org/wiki/Bor%C5%AFvka%27s_algorithm.
12. http://en.wikipedia.org/wiki/Degree_(graph_theory).
13. Kenneth Sorensen and Gerrit K. Janssens, "An Algorithm to Generate All Spanning Trees of a Graph in Order of Increasing Cost," Pesquisa Operacional, vol. 25, pp. 219- 229, 2005.
14. S. Chatterjee and S. S. Sarma, "In Search of a Versatile and Flexible Graph Mod el to Implement Computer Network", Proceeding of International Conference on Communications, Devices and Intelligent Systems, pp. 635-638, 2004.