MINAR International Journal of Applied Sciences and Technology

Article type : **Research Article**

Date Received : **22/07/2020**
Date Accepted : **03/08/2020**
Date published : **01/09/2020**
: **www.minarjournal.com**

DESIGN AND IMPLEMENTATION OF KRUSKAL'S MINIMUM SPANNING TREE ALGORITHM IN C++

Nadia Moqbel Hassan ALZUBAYDI[1]

Abstract

The traditional algorithms (Prim) or (Kruskal) are able to obtain A minimum spanning tree (MST) in undirected graph easily. But many algorithms have been proposed for the purpose of obtaining spanning trees in undirected graph, these algorithms are considering the complexities of time and space. Some algorithms are generating spanning trees by using some basic cuts or circuits. In this process, the tree's cost is not considered. In this paper we will describe an algorithm for generating spanning trees in a graph of increasing cost, so we will get many possibilities, such as determining the k-th lowest spanning tree. The lowest spanning tree meets some additional constraints that may be found. We will discuss Murty's algorithm in this paper, which can find all solutions to assignment problem of increasing cost, and also discuss complexities of time and space.

Keywords: Performance Analysis, Computational Complexity, Graph, Minimum Spanning Trees.

[1] Dr., Mustansiriya University, Iraq, nadiahasan@uomustansiriyah.edu.iq

I Introduction

Undirected graph (G) can be characterized as a group of (N,R), while (N) as a gathering of Nodes and (R) as a gathering of rims. each two nodes are associated by rims , i.e.$R = \{(u,n)|u,n \in N\}$. The undirected weighted chart is containing a component of weighting: $w: R \to \Re$, that allocates out a weight for rims. At that point weight of rim can be likewise called as cost or distance.[1] The tree is created from sub-chart through G that has no circuits. Therefore, there is just one course from every node to another. in this way, the crossing tree is made out of all nodes. Also the (MST) of undirected, weighted chart (G) is the tree that having the insignificant entirety of rims ' loads. There are numerous calculations that can locate a (MST) (M, for example, (Kruskal) and (Prim) calculations[2].

Kruskal's calculation: The accompanying advance must be rehashed unto the gathering M arrives at x-1 rims (initially M is unfilled). at that point needed to add the briefest rims to M, with thinking about that the rim doesn't draw up a circuit or loop with different rims in M.

Prim's calculation: The accompanying advance must be rehashed unto the gathering M arrives at x-1 rims (initially M is emptied).[3] The most limited rim between a node in M and a node outside M ought to be Added to M (initially pick an rim with least length).

The two above calculations are diverse in the criticalness, in light of the fact that (Prim) calculation develops a tree unto it turns into the MST, while (Kruskal) calculation develops a gathering of trees unto this gathering is decreased to one tree as MST[4].

The spanning trees can be shown to by an assortment of x-1 rims. Furthermore, the rim can be shown to by untidy pair of nodes.(see Equation 1).

$$s = \{(a_1, b_1), \dots, (a_{n-1}, b_{n-1})\} \quad \text{.....(1)}$$

The character, (A) is symbolizes to all spreading over trees in the diagram G.

There are numerous calculations intended to create all spreading over trees in a diagram, for example, (Shioura and Tamura, 1995, Minty, 1965; Matsui, 1993; [5] Gabow& Myers, 1978; [6] Read &Tarjan, 1975; Kapoor& Ramesh, 1995; [7] Matsui, 1997, Kapoor& Ramesh, 2000;). The complexities of excellent reality are the significant worries of these calculations. numerous calculations are spanning trees by utilizing some basic cuts or circuit, yet nobody considers the expense of tree during creating spanning trees.

Some calculations can produce all ST without weights , for example, (Read& Tarjan, 1975, Minty, 1965;),[8] so categoriation the trees relying upon expanding weight, in the wake of creating them. As this procedure can creates countless trees (for the most part in complete diagrams), along these lines this decision is prohibited for pragmatic purposes[9].

II Creating Spanning Trees arranged by Increasing Cost

Assuming that $c(s_i)$, is the cost allocated to ST, s_i and i is to the position of s_i, at that point If all spanning trees are arranged agreeing expanding cost[10].

Subsequently take on the take on that $c(s_i) \leq c(s_j)$ on the off chance that i < j. The arrangement s_1, s_2, ... are to the arranging of spanning trees arranged by expanding cost.

By the following equation(2):

$$P = \{s: (i_1, j_1) \in s; \dots; (i_r, j_r) \in s; (m_1 p_1) \notin s; \dots, (m_l p_l) \notin s\} \quad \text{.....(2)}$$

know the a portion (P) is characterized as non-void sub-bunch from all spanning trees (H) in the diagram (G). That is, it can be said that P is the gathering of ST that contain all of included rims [symbolized as (i_1, j_1), … , (i_r, j_r)], and not contain any of avoided rims [symbolized as (m_1, p_1), … , (m_l, p_l)). Open rims are known as neither included nor excluded rims in segment. Will be show to the segment P as in the following equation (3):

$$P = \{(i_1, j_1), \dots, (i_r . j_r); \overline{(m_1, p_1)}, \dots ., \overline{(m_l, p_l)}\} \quad \text{.....(3)}$$

The strip above shows that (m_1, p_1), … , (m_l, p_l) are prohibited edges. These rims, lead to make a few allotments not have any ST. This is known as the situation (w) in the diagram G, when avoided rims are expelled or separated from the segment. So we designate the segments that are not containing any crossing trees as (vacant allotments)[11].
It ought to be referenced that (H) which shows to the gathering of all spanning trees, is likewise a segment, however the entirety of its edges are open [12]. Minimum Spanning Tree is characterized as a component of P which is made out of a spanning tree with insignificant expense and contains every included edge as it were. since each spanning tree is containing the rims (i_1, j_1), … , (i_r, j_r), in this manner the base spanning tree can be found via looking through x-r-1 from (open rims). So as to guarantee that every single wanted rim are incorporated into a base ST, they can be included prior every single other rim. Likewise to ensure that nobody from rejected rims is in minimum spanning tree, they can be briefly allocated to boundless expense.

The strategy for framing partitions, guarantees that the sets of included rims isn't containing any loops or circuits. At that point Kruskal's calculation can be begun from this partial ST and keep including rims.
Since the sets of included rims may not create a tree, at that point (Prim) calculation ought to be balanced by the accompanying strategy:
Include the most limited rim between a nodes inside (M) and other nodes, to M, under specification that no circuit is structure with edges of M[13]. The balanced calculation permits edges to interface two disengaged parts in spanning tree, and doesn't permit framing circuits in M.
s(P) demonstrates to the MST in partition P. What's more, c[s(P)] is its expense.
Part P by the MST is the topic of segmentation is the core of the algorithm suggested in this paper[14]. When minimum spanning tree defines a section, that section can be divided into a group of resulting sections in a way that includes notifications such as the empty group is resulted through intersection of two partitions, the minimum spanning tree of main partition is not a component of any resulted partitions, in addition to the major partition is equal to the unifying of resulted partitions after subtracting its minimum spanning tree.
For more explanation, can follow the rule below: Assume that the MST in P is (see Equation 4):

$$s(P) = \{(i_1, j_1), (i_r, j_r), (t_1, n_1), \dots, (t_{x-r-1}, n_{x-r-1})\} \quad \text{.....(4)}$$

Whereas $(t_1,n_1),\dots,(t_{x-r-1},n_{x-r-1})$ are dissimilar from $(m_1,p_1),\dots,(m_l, p_l)$. So P can be defined as the unification of single group {s(P)} and the partitions $P_1,\dots,P_{x-r-1}$, that are not connected (see Equations 5,6,7…. and 8):, where:

$$P_1=\{(i_1, j_1), \dots, (i_r, j_r), (\overline{m_1, p_1}), \dots, (\overline{m_l, p_l}), (\overline{t_1, n_1})\} \quad \text{.....(5)}$$

$$P_1=\{(i_1, j_1), \dots, (i_r, j_r), (t_1, n_1), (\overline{m_1, p_1}), \dots, (\overline{m_l, p_l}), .. (\overline{t_2, n_2})\} \quad \text{.....(6)}$$

$$P_1=\{(i_1, j_1), \dots, (i_r, j_r), (t_1, n_1), (t_2, n_2), (\overline{m_1, p_1}), \dots, (\overline{m_l, p_l}), .. (\overline{t_3, n_3})\} \quad \text{.....(7)}$$

…

$$P_{n-r-1} = \{(i_1, j_1), \dots, (i_r, j_r), (t_1, n_1), \dots, (t_{x-r-2}, n_{x-r-2}), (\overline{m_1, p_1}), \dots, (\overline{m_l, p_l}), .., (\overline{t_{x-r-1}, n_{x-r-1}})\} \quad \text{..... (8)}$$

Note can be that the parts P_1, …, P_{x-r-1} are reciprocally disconnected through noting that every ST in P whether includes (t_1, n_1) or not include (when it is an item or part of P_1). If it does, it either includes (t_2, n_2) or does not (when it is an element or part of P_2). Taking up such this and noting Just that (ST) is containing the rims (i_1, j_1), …, (i_r, j_r), (t_1, n_1), …, (t_{x-r-1}, n_{x-r-1}) is s(P), can find that (see Equation 9):

$$P = \{s(P)\} \cup \bigcup_{i=j}^{x-r-1} P_1 \quad \text{.....(9)}$$

Each ST in partitions P_1 to P_{x-r-1} includes $(i_1,j_1),\dots,(i_r,j_r)$ and each ST does not include $(m_1,p_1),\dots,(m_l,p_l)$.
Phase k in the counting operation indicates to the phase when $s_1,\dots,s_k$ are specified . In this phase, the list that includes a group of sections $M_1,\dots,M_e$ with the characteristics below:

1. M_1, ……, M_e are reciprocally disconnected,
2. Every partitions in the list does not contain any spanning trees that was previously produced (s_u, u = 1, …....., k).
3. Unification of each segmentations in the list is the set of each ST that are not created so far.

Through this characteristics (see Equation 10), can say that[15]:

$$H = \cup_{u=1}^{k}\{ s_i\} \cup \cup_{n=1}^{e} M_n \quad \text{.....(10)}$$

Also through determination a list for phase k, note that the smallest ST (k-*th*) is matching to $s(M_d)$, as M_d is each segmentation in the listing that (see Equation 11):

$$c[s(M_d)] = \min_{i=1..e}\{c[s(M_i)] \quad \text{.....(11)}$$

III Design of Algorithm for organizing ST arranged by expanding cost

At the point when the G contains (x) nodes, the calculation proceeds in phases, until creating the littlest ST (k-*th*) in stage k.

In the first stage: Stage 0 is determined as equal to the partition H. then find out an minimum spanning tree for H (or for G). As in the following (see Equation 12):

$$s_1 = \{(i_1, j_1), \dots, (i_{x-1}, j_{x-1})\}. \quad \text{.....(12)}$$

Partitions M_1, …,M_{x-1}, that are defined as follow, will be created by (MST) of partition H(see Equation 13):

$$\begin{gathered} M_1 = \{(i, j)\} \\ M_2 = \{(i, j), (i_2, j_2)\}, \\ M_3 = \{(i, j), (i_2, j_2), (i_3, j_3)\}, \\ \dots\dots \\ M_{x-1} = \{(i_1, j_1), (i_{x-2}, j_{x-2}), \overline{(i_{x-1}, j_{x-1})}\} \end{gathered} \quad \text{.....(13)}$$

Then{M_1,…,M_{x-1}} is forming a list for stage1, so the partitions which have no spanning trees, may be taken away from the list.

In the second stage: stage (k), Since the list of stage k_1 is consisted from the partitions L_1,…,L_t, can be count up (MST): (L_1),…,s(L_t), of each part in this listing, as costs increase c[s(L_1)],…,c[s(L_t)] for each spanning tree (ST).

Then, smallest ST (k-th), is the tree which has lowest cost (see Equation 14):

$$s_k = \{s(L_i) | c[s(L_i)] = \min_{j=1..t} c[s(L_j)]\} \quad \text{.....(14)}$$

(L_i) represents the partition that have smallest ST of all ST that are not produced so far. A list for s phase k is resulted by removing (L_i) from stage (k-1) list, and putting it in the partitions that are created through segmentation (L_i) according s(L_i). and removing blank partitions from the list. Also solving joints through choosing one partition randomly from the list and disregard others[16],[5].

.

The final stage will take the example: In this example will be explain the steps of algorithm for sorting all ST for increasing cost.

Take into consideration that the graph (G), has five vertices (H, I, J, K, L). Any ST in (G) will be composed of four rims.

The first step is to define the MST in partition (H). Then (MST) in (G) is equal to s1={(H, I), (I, J), (J, K), (K, L)} and c[s1] = 25 (see Fig. 1).

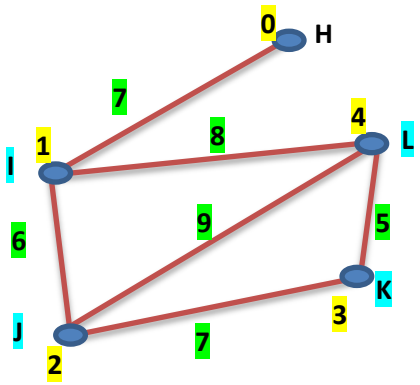


Fig. 1 Example graph (G)

Now, G is subdivided through s_1, in order to obtain four parts, P_1,P_2,P_3 and P_4, form a list for phase 1 (see Equation 15):

$$P_1 = \{(\overline{\mathrm{H,I}})\}$$
$$P_2 = \{(\mathrm{H,I}), (\overline{\mathrm{I,J}})\}$$
$$P_3 = \{(\mathrm{H,I}), (\mathrm{I,J}), (\overline{\mathrm{J,K}})\}$$
$$P_4 = \{(\mathrm{H,I}), (\mathrm{I,J}), (\mathrm{J,K}), (\overline{\mathrm{K,L}})\} \quad \text{.....(15)}$$

Diagrammatically, can represent the partitions See Figure 2 that following illustrate: (a speckled line represents an eliminate (or excluded) rim, and a font that is bold represents an implicated rim).

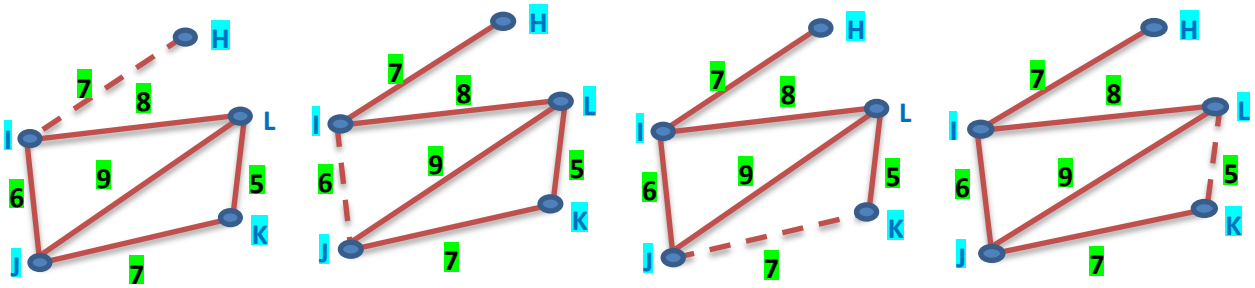


Fig. 2 Partitions P_1,…….,P_4

The next step is to count up (MST) for every part in the list. Because P_1 is disconnected, therefore It does not contain (MST). The MST of P_2 to P_4 are(see Equations 16,17 and 18):

$$s(P_2) = \{(\mathrm{H,I}), (\mathrm{I,L}), (\mathrm{L,K}), (\mathrm{K,J})\} \quad \text{.....(16)}$$

$$s(P_3) = \{(\mathrm{H,I}), (\mathrm{I,J}), (\mathrm{I,L}), (\mathrm{L,K})\} \quad \text{.....(17)}$$

$$s(P_4) = \{(\mathrm{H,I}), (\mathrm{I,J}), (\mathrm{J,K}), (\mathrm{I,L})\} \quad \text{.....(18)}$$

Thus their costs are:- $c[s(P_2)] = 27$, $c[s(P_3)] = 26$, $c[s(P_4)] = 28$

because P_3 contains the (MST) at the lowest cost (see Equation 19):

$$S_2 = s(P_3) = \{(H,I),(I,J),(J,L),(L,K)\} \quad (19)$$

Through subdividing (P_3) according its MST s(P_3) , will be gain the partitions (P_{31}) and (P_{32}) (see Equations 20 and 21):

$$P_{31} = \{(H,I),(I,J),\overline{(J,K)},\overline{(I,L)}\} \quad (20)$$

$$P_{32} = \{(H,I),(I,J),(I,L),\overline{(J,K)},\overline{(L,K)}\} \quad (21)$$

Diagrammatically, these partitions are represented in Figure 3.

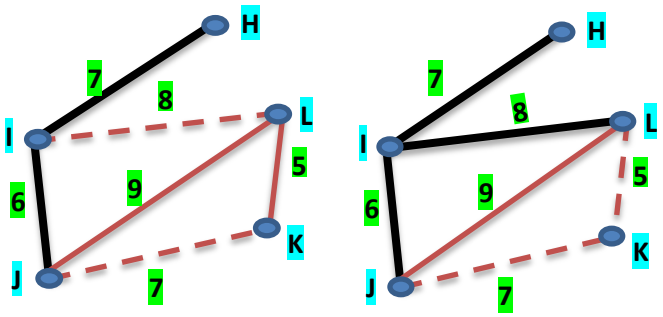


Fig. 3 Partitions P_{31} and P_{32}

The list of stage 2, is composed of {P_2, P_{31}, P_4}. (P_{32}) is removed from the list because it is disconnected. Then the MST for (P_{31}) is (see Equations 20 and 22):

$$s(P_{31}) = \{(H,I),(I,J),(J,L),(L,K)\} \quad (22)$$

At a cost of: $c[s(P_{31})] = 27$

The list for phase 2 holds two parts (P2 and P31) that contain a MST with minimum cost. links such this one can be solved by choosing any partition to be subdivided in next stage. By taking up the same steps, eight ST will be gained with costs range between (25 - 31) [17].

IV Implementation of the proposed algorithm using Computer

So as to play run out this calculation on PC, need to sort the vertexes in the list of present stage in memory. Can be shown to the partition concurring its included and barred edges. So can be represent to the particular diagram utilizing three courses of action, by represent to the head, tail and weight of the edge. successively. Furthermore, the parcel can be partition by two techniques. The first should be possible through assurance of the start and end of included and rejected rims. While the second should be possible through decide sort of each edge in the diagram, on the off chance that it is incorporated, prohibited or open. At that point the rundown of allotments can be acquired by utilizing a related list.
The potential temple of the program that producing every ST arranged by expanding cost is:
Create ST orderly of growing cost using algorithm1:

Inputs: Graph G (V,E) & weight function (w)
Outputs: All spanning trees of Graph, arranged by expanding cost, with implementation of Output File
List = {H}
Calculation of Minimum Spanning tree(MST) of (H)
while Minimum Spanning tree(MST) not equal (∅) do
Obtain partition Ps ∈ List that includes the smallest spanning tree(SST)
Print Minimum Spanning tree(MST) of P_s to Output File
Delete P_s from List
Partition P_s.

After inspection subdividing subroutine adds parts to the listing if they are linked and computing their (MST). A major damage of this method is that either have to hold (MST) from the part in the listing (squandering store) or recalculate it when the part is recovered from the list (squandering time)[18]. So, a major feature of this method is that can be save arranged list of parts rather than of none-arranged one, and it easy to recover smallest partition. A possible program for the subdividing subroutine is:

Pseudo-Code of the PARTITION (P)

Partition($\mathbf{P_1}$) equal Partition($\mathbf{P_2}$) equal Partition($\mathbf{P_2}$)
For every edge in **P do**
if (i) it was not in contained in **P** It has not been deleted from **P** eliminate from **P then**
Make (i) eliminated from $\mathbf{P}_1$;
Make (i) contained in P_2;
Calculate Minimum Spanning tree (MST) (P_1);
if Connected (P_1) **then**
addition $\mathbf{P_1}$ to List;
$\mathbf{P_1}$ equal $\mathbf{P_2}$;

V Complexities with Space and Time

|R| represents the number of rims, |N| represents the number of nodes, and X represents the number of (ST) in specific graph (G). A number of algorithms to generate all (ST) gain good time complexity through output (ST) in a specific arrangement, so can be use short notation format. And can be generate (ST) through interchange one rim from the previous (ST) in obstetrics process [19].
By this method, can be expand the short notation format when the first SP is formed as output, and others are limited to the interchanged twin of rims.

Because no such arrangement like this is gained through this algorithm, therefore need O(X.|N|) area in order to generate all ST. Since each node in the listing can be limited by rotation, the number of ST imposes maximum limit on the number of pars in the list, and the list of parts cannot be bigger than the number of ST, thus includes highest value of X parts. Therefore, the part can be deleted according the situation of its rims (open, included, or excluded). So, the extent of every node is O(|R|). and the complications area in the part listing is O(X.|R|). In most situations, one and only a little part of area at any time.
Depending on the time complexity calculation in ST generation algorithms. Time complexity can be calculated for this algorithm. The way to generate a ST by Kruskals algorithm is O(|R| log |R|).
Following statements, suppose that the partition's list is constantly preserved arranged. In this case, can be retrieve an element from the list in persistent time. Also can put an item into the listing according required O(X) operations, because the top extent of the listing is equal to the maximum number of parts X. therefore I/O activities will be ignored. Can implement many steps in the algorithm in steady time. such as inspection if the part is empty or not (if connected), because this information is obtainable from the (MST) algorithm. The master loop in the algorithm can be implemented (X) times, therefore, the step (PARTITION) can be implemented (X) times. Calculate Minimum Spanning tree (MST) (P1); is O (|R| log |R|) and add to listing is O(X). The procedure has time complexity O(X.|R| log .|R| + X2).
The complexities of “time and space” in our procedure are worse than other procedures. The procedures which were written using mathematicians [Matsui (1993) , Shioura & Tamura (1995 and Gabow & Meyers (1978))], have ability to create each ST of a diagram in O(|N|.|R|) space and O(X.|N| |N| |R|) time. Anyway,

the aim of our algorithm is not creating each ST, but to stop creating when it finds a ST that comply with some additional restrictions. This is leading to create a smallest part of overall number of ST.

VI Uses of Minimum Spanning Tree Problems

Possible applications can be establish in the set of MST problems with other restrictions, the procedure in this paper is to create ST in order of increment cost and inspect if it is complying with additional restrictions. (Murty) algorithm was designed for sorting assignments according increasing cost, and it was used by (Panayiotopoulos, 1982) in comparable manner to create an optimal solution in the issue of (Touring Salesman)[20].

In some issues, may be need to find the largest ST which complies with extra restrictions. So the algorithm can be modified in order to implement this task. For example, the two algorithms[Kruskal's &Prim's] modified to find top ST rather than of the bottom. Also, the procedure of creating ST in order of increasing cost can be modified to create ST in order of decreasing cost[21].

VII Implementation and Experimental Results

In Computer Engineering, a chart is a sort of data structure, explicitly an abstract data type (ADT) that comprises of a lot of nodes and a lot of rims that build up connections (associations) between the nodes. This Section Presents of the implementation and the results of kruskals algorithm in C++ to find Minimum Spanning tree with Code-Blocks, in a given graph, the explanation MST as shown in Figures (4), (5) and (6).

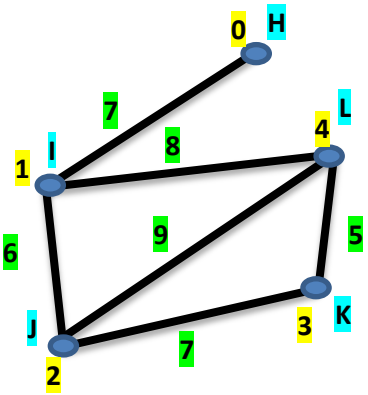


Fig.4 Connecting Weighted Graph

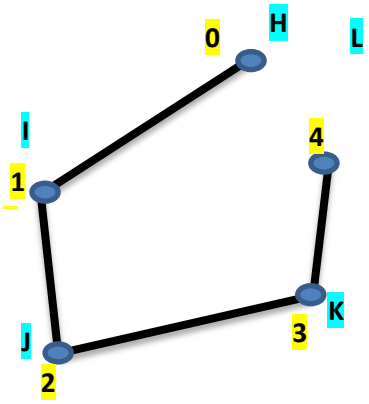
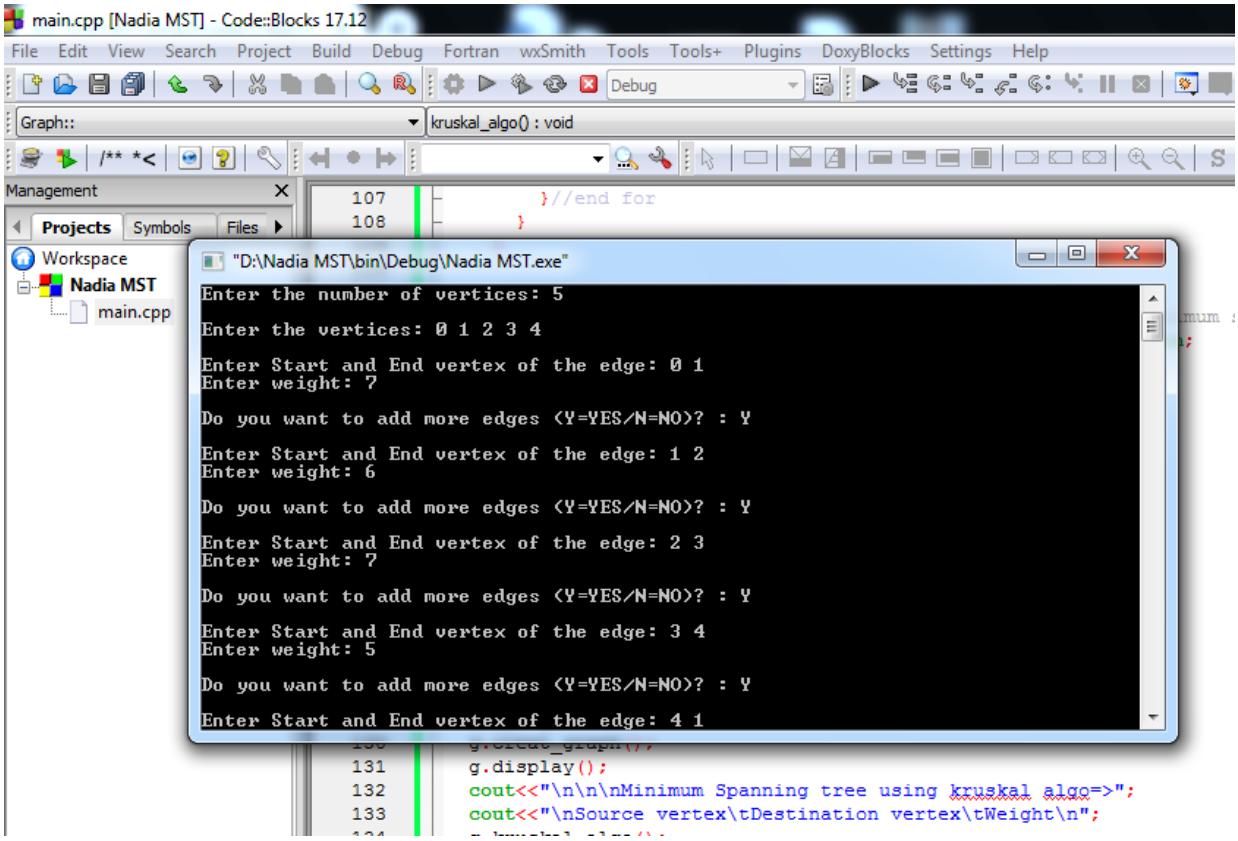


Fig.5 Minimum Spanning Tree



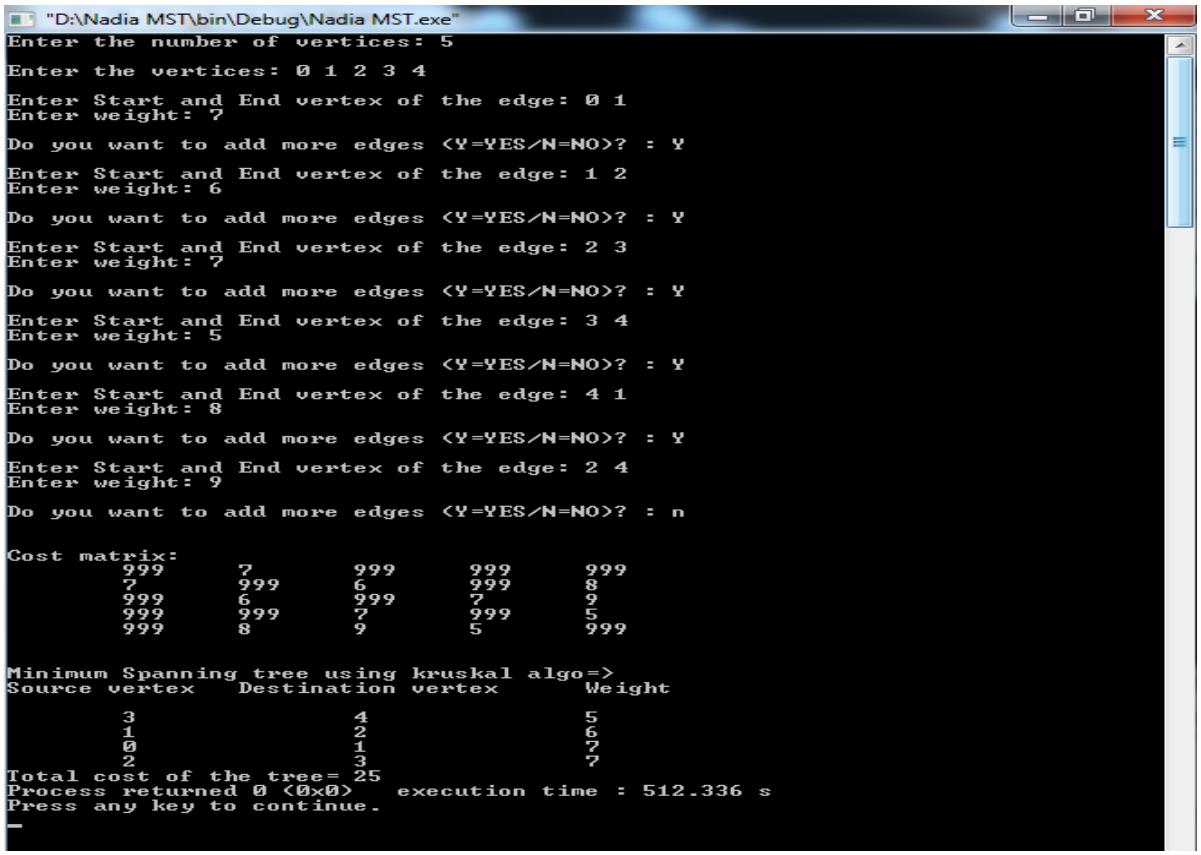


Fig.6 Results of kruskals algorithm in C++ to find Minimum Spanning tree with Code-Blocks

VIII Conclusion

1. After extracting the results a conclusion was made develop an algorithm to arrange all ST in specific diagram in require of increasing cost. so this procedure or algorithm depended on an procedure which has been expanded, by the mathematician (Murty) for arranging tasks in require of increasing cost.
2. Space and time complexities are discussed briefly. And gave some instructions to execute suggested algorithm on a computer.
3. Ultimately, have been explained some possible implementations of the proposed algorithm. Each these implementations can be classified as a MST that shows the most brief arrangement of rims interfacing various nodes. There is consistently one rim not exactly like the rest of nodes in a ST. Also, Kruskal's calculation adds rims to a tree arranged by size. While Prim's calculation adds the closest nodes to a current T.

IX Acknowledgments

The authors would like to thank (Mustansiriyah) University stuff for their support in the present work. (www.uomustansiriyah.edu.iq) Baghdad – Iraq.

X References

Afsana, K. & Afrida, A. & Juthi, S. (2019). A New Algorithmic Approach to Finding Minimum Spanning Tree. Conference Paper. 1-6.

Dahl, G. (1998). The 2-hop spanning tree problem. Operations Research Letters, 23. 21-26.

Sedgewick & Wayne . 2019-12-10 . Data structures and algorithms-Minimum Spanning Trees. Lecture 15a. 1-42.

Diestel, R. (1996). Graph Theory. Springer, New York, xiv + 266 pp.

Gabow, H.N. (1977). Two algorithms for generating weighted spanning trees in order. SIAM Journal on Computing, 6(1), 139-150.

Gabow, H.N. (1978). A good algorithm for smallest spanning trees with a degree constraint. Networks, 8, 201-208.

Nadia, M.H. 5102. Design And Implementation of Shortest Path Algorithm for Network of Roads. Journal of Engineering and Development. Vol.19, No. 06. ISSN 1813-7822. 1-12.

Gabow, H.N. & Myers, E.W. (1978). Finding all spanning trees of directed and undirected graphs. SIAM Journal on Computing, 7, 280-287.

Hall, L. & Magnanti, T. (1992). A polyhedral intersection theorem for capacitated trees. Mathematics of Operations Research, 17, 398-410.

Kenneth, S.& Gerrit, K.(2005) An algorithm to generate all spanning trees of a graph in order of increasing cost

Nadia, M. H. 2017. Analysis and Implementation of the Minimum Route Issues between some Governorates of Iraq using Bellman-Ford Algorithm. Annual Conference on New Trends in Information & Communications Technology Applications-(NTICT'2017). 1-6.

Kapoor, S. & Ramesh, H. (1995). Algorithms for enumerating all spanning trees of undirected and weighted graphs. SIAM Journal on Computing, 24, 247-265.

Kapoor, S. & Ramesh, H. (1997). An algorithm for enumerating all spanning trees of a directed graph. Algorithmica, 27(2), 120-130.

Matsui, T. (1993). An algorithm for finding all the spanning trees in undirected graphs.Technical Report METR 93-08, Dept. of Mathematical Engineering and Information Physics, University of Tokyo, Tokyo.

Matsui, T. (1997). A flexible algorithm for generating all the spanning trees in undirected graphs. Algorithmica, 18(4), 530-543.

Minty, G.J. (1965). A simple algorithm for listing all the trees of a graph. IEEE Transactions on Circuit Theory, CT-12, 120.

Murty, K.G. (1986). An algorithm for ranking all the assignments in order of increasing cost. Operations Research, 16, 682-687.

Panayiotopoulos, J-C. (1982). Probabilistic analysis of solving the assignment problem for the travelling salesman problem. European Journal of Operational Research, 9, 77-82.

Papadimitriou, C. (1978). The complexity of the capacitated tree problem. Networks, 8, 219-234.
Read, R.C. & Tarjan, R.E. (1975). Bounds on backtrack algorithms for listing cycles, paths and spanning trees. Networks, 5(3), 237-252.
Shioura, A. & Tamura, A. (1995). Efficiently scanning all spanning trees of an undirected graph. Journal of the Operations Research Society of Japan, 38(3), 331-344.Pesquisa