Istanbul Commerce University Journal of Science

İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, 16(32), Güz 2017

http://dergipark.gov.tr/ticaretfbd

Research Article

THE NEW ALGORITHM INVOLVING MINIMUM SPANNING TREE FOR COMPUTER NETWORKS IN A GROWING COMPANY

Murat KİRİŞCİ*[1] Deniz ÖNCEL[2]

[1]Department of Mathematical Education, Istanbul University, 34116 Istanbul, Turkey
E-mail: mkirisci@hotmail.com,
[2]Ataşehir, Istanbul, Turkey
E-mail: denizoncel844@gmail.com

Abstract

The aim of this article is to present a new algorithm based on minimum spanning trees. Minimum Spanning Trees have long been used in data mining, pattern recognition and machine learning. However, it is difficult to apply traditional minimum spanning tree algorithms to a large dataset since the time complexity of the algorithms is quadratic. The given algorithm is designed to reduce this difficulty. This application has reduced the cost.

Keywords: Graph theory, minimum spanning tree, Prim's algorithm.

Araştırma Makalesi

BÜYÜYEN BİR ŞİRKETTE BİLGİSAYAR AĞLARI İÇİN MINIMUM SPANNING TREES İÇEREN YENİ BİR ALGORİTMA

Öz

Bu makalenin amacı, Minimum Spanning Trees'ye dayalı yeni bir algoritma sunmaktır. Minimum Spanning Trees, veri madenciliği, model tanıma ve makine öğrenmede uzun süredir kullanılmaktadır. Bununla birlikte, geleneksel Minimum Spanning Trees algoritmalarını büyük bir veri kümesine uygulamak zordur. Çünkü algoritmaların zaman karmaşıklığı ikinci derecedir. Verilen algoritma bu zorluğu azaltmak için tasarlanmıştır. Bu uygulama maliyeti düşürmektedir.

Anahtar Kelimeler: Grafik Teori, Minimum Spanning Tree, Prim Algoritması.

Received / Geliş tarihi: 14/09/2017 Accepted / Kabul tarihi: 07/10/2017
*Corresponding Author/ Sorumlu Yazar : mkirisci@hotmail.com

1. INTRODUCTION

A minimum spanning tree (MST) is a spanning tree of an undirected and weighted graph such that the sum of the weights is minimized. Numerous applications have been published on the MST based on a undirected graph (Prim 1957, Kruskal 1956). As the intrinsic structure of a dataset can be roughly estimated, the MST has been broadly applied in image segmentation, cluster analysis, classification, manifold learning, density estimation, diversity estimation, and some applications of the variant problems in the area.

The MST problem, is mostly regarded as a cornerstone of Combinatorial Optimization. In the 1950s, it is commonly known that Kruskal (1956) and Prim (1957) for the first time produced algorithms on a spanning tree of a minimum length in a weighted connected graph. However, the earliest algorithms on the topic were presented by Boruvka (1926).

Trees and spanning trees represent a very fundamental and important graph structure for combinatorial optimization. Spanning trees serve as building blocks when designing telecommunications and electric power networks. Definitions of tree and spanning tree are as follows:

Given a general connected undirected graph $G=(V,E)$, a set tree in *G* is connected subgraph $T=(V',E')$ containing no cycles If $V'=V$ then *T* is a spanning tree for the graph *G*.

Due to the definitions, trees are made of one piece of line/graph and if the tree has the same number of vertices, then the one pieced has the minimum number of edges.

We can give the MST problem as follows:

Given a finite set *V* and real weight function ω on pairs of elements of *V*, find a tree *(V,T)* of minimal weight $\omega(T)=\sum[\omega(x,y):\{x,y\}\in T]$

Example 1.1. (Nesetril et.al. 2001) Let *V* be a subspace of a metric space and weighted function be a distance function. Then, a solution *T* presents the shortest network connecting all points of *V*.

Example 1.2. In the Figure 1, the set $V/(G)=\{1,2,3,4\}$ denotes the set of points. Then, $E(G)=\{(1,2),(1,3),(2,4)\}$ becomes the set of edges.

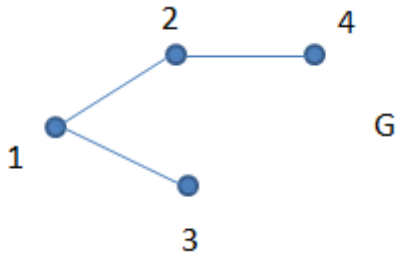


Figure 1.

Another formulation can be given as follows:

Given a undirected graph $G=(V,E)$ with real weights assigned to its edges. Find a spanning tree *(V,T)* of, $G,T\subseteq E$ with the minimal weight $\omega(T)$.

The MST problem has been solved in 1926 by Boruvka (1926a, 1926b). Boruvka (1926b) has defined this problem as follows:

In the space, we consider *n* points. The mutual distances between these *n* points are assumed to be different. The problem is to join them through the net in such a way that;

i. Any two points are joined to each other either directly or by means of some other points,

ii. The total length of the net would be the smallest.

The Boruvka's algorithm begins by first examining each vertex and adding the cheapest edge from that vertex to

another in the graph, without regard to already added edges, and continues joining these groupings in a like manner until a tree spanning all vertices is completed.

Boruvka's algorithm can be given as follows:

Input: A connected graph G whose edges have distinct weights
Initialize a forest T to be a set of one-vertex trees, one for each vertex of the graph.
While T has more than one component:
For each component C of T.
Begin with an empty set of edges S
For each vertex v in C:
Find the cheapest edge from v to a vertex outside of C, and add it to S
Add the cheapest edge in S to T
Combine trees connected by edges to form bigger components
Output: T is the minimum spanning tree of G.

Then, similar algorithms have been constructed by many mathematicians. One of the most typical examples of these algorithms is the Prim's algorithm (Prim, 1957). Firstly, it arbitrarily selects a vertex as a tree, and then repeatedly adds the shortest edge that connects a new vertex to the tree, until all the vertices are included.

In Figure 2, a graph and its MST structure can be seen.

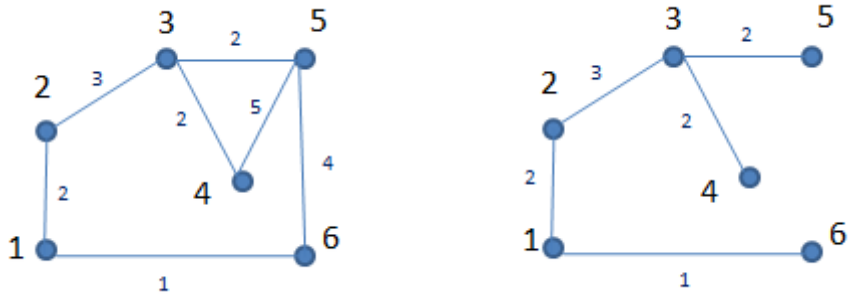


Figure 2.

The purpose of this work is to develop a new algorithm for MST. The new algorithm is implemented on an expanding network of computers.

2. ALGORITHMS

There are various algorithms developed to find the MST of a graph. It is well known that a graph can have more than one MST. One of the best known of these is the Prim algorithm.

It starts with an empty spanning tree. The idea is to maintain two sets of vertices. The first set contains the vertices already included in the MST, the other set contains the vertices not yet included. At every step, it considers all the edges that connect the two sets, and picks the minimum weight edge from these edges. After picking the edge, it moves the other endpoint of the edge to the set containing MST.

The idea behind Prim's algorithm is simple, a spanning tree means all vertices must be connected. So the two disjoint subsets (discussed above) of vertices must be connected to make a Spanning Tree.

Algorithm 1(Prim's Algorithm):

let T *be a single vertex x*
*while (*T *has fewer than n vertices)*
{
find the smallest edge connecting T *to* G-T
add it to T
}

or it can be given as follows:

Step 1. *First begin with any vertex in the graph.*

Step 2. *Of all of the edges incident to this vertex, select the edge with the smallest weight.*

Step 3. *Repeat step 2 using the edges incident with the new vertex and that aren't already drawn.*

Step 4. *Repeat until a spanning tree is created.*

It can also give the Prim's algorithm as follows:

The following codes are given by C++ program for Prim's Minimum Spanning Tree (MST) algorithm (http://scanftree.com/Data_Structure/prim\%27s-algorithm):

```
#include<stdio.h>
#include<conio.h>
int a,b,u,v,n,i,j,ne=1;
int visited[10]={0},min,mincost=0,cost[10][10];
void main()
{
   clrscr();
   printf("\nEnter the number of nodes:");
   scanf("%d",&n);
   printf("\nEnter the adjacency matrix:\n");
   for(i=1;i<=n;i++)
   for(j=1;j<=n;j++)
   {
      scanf("%d",&cost[i][j]);
      if(cost[i][j]==0)
      cost[i][j]=999;
    }
   visited[1]=1;
   printf("\n");
   while(ne < n)
{
      for(i=1,min=999;i<=n;i++)
      for(j=1;j<=n;j++)
      if(cost[i][j]< min)
      if(visited[i]!=0)
{
      min=cost[i][j];
      a=u=i;
      b=v=j;
}
      if(visited[u]==0 || visited[v]==0)
{
      printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);
      mincost+=min;
      visited[b]=1;
}
      cost[a][b]=cost[b][a]=999;
}
      printf("\n Minimun cost=%d",mincost);
      getch();
```



Figure 3: The output of program which is given by C++

We will find the MST that includes edges $\{e_1,e_2,\ldots, e_k\}$ of the graph *G*. Thus, we can solve this generalized minimum spanning problem by using the new algorithm as follows:

Algorithm 2:

Step 1. *Create the MST* T *of a graph using the Prim's algorithm. Choose one of the edges* $\{e_1,e_2,\ldots, e_k\}$.

Step 2. *If the edges* $\{e_1,e_2,\ldots, e_k\}$ *belong to* T, *then* T *is the MST. If the edges* $\{e_1,e_2,\ldots, e_k\}$ *doesn't belong to* T, *create a set* S *of edges that doesn't belong to* T.

Step 3. *Choose an edge from the set* S *and add to* T. *In this case, the cycle occurs in* T. *Choose the longest edge outside the edges* $\{e_1,e_2,\ldots, e_k\}$ *in this cycle and remove the longest edge from* T.

Step 4. *Remove the latest edge added to* T *from the set* S.

Step 5. *Repeat from Step 3, until* S *is an empty set.*

Theorem 2.1. Let $G=(V(G),E(G))$ be a graph and $\{e_1,e_2,\ldots,e_k\}\subset E(G)$. Apply Algorithm 2 to graph *G* according to the given edge lengths. The resulting graph is the MST containing edges $\{e_1,e_2,\ldots, e_k\}$ of *G*.

Proof. Firstly, when Algorithm 2 is applied, it should be shown that the resulting graph is the MST. The graph *T* is the MST because of the *Step 1* of Algorithm 2. In the next steps of the algorithm, *T* is still the MST. It is well known that when an edge is added to *T*, the cycle is obtained. If an edge is removed, then *T* does not contain cycles. Then, the last graph is still the MST.

Now, it will be shown that the obtaining spanning tree at the end of the algorithm is the minimum spanning tree containing edges $\{e_1,e_2,\ldots, e_k\}$. If the graph *T* does not include the given edges, then one of the non-included edges is added to *T* in each step. Thus, the last obtained graph will include all of the edges $\{e_1,e_2,\ldots, e_k\}$.

Suppose that T' is the MST including the edges $\{e_1,e_2,\ldots, e_k\}$. If $T=T'$, then the proof is complete. Consider that $T\neq T'$. Then, there is at least one edge in the tree *T* and not in the tree T'. Let $e=(u,v)$ be one of these edges. Since the tree T' also contains the edges $\{e_1,e_2,\ldots, e_k\}$, the edge *e* cannot be one of the edges in the set $\{e_1,e_2,\ldots, e_k\}$. Then, the edge *e* should be one of the edges added to the graph while applying the Prim's algorithm. If the edge *e* is removed from *T*, *T* becomes the two-part graph. Let's denote the set of points of one of these pieces by *X*. Therefore, the other part of the graph can be represented by $V(G)\setminus X$ (Figure 4).

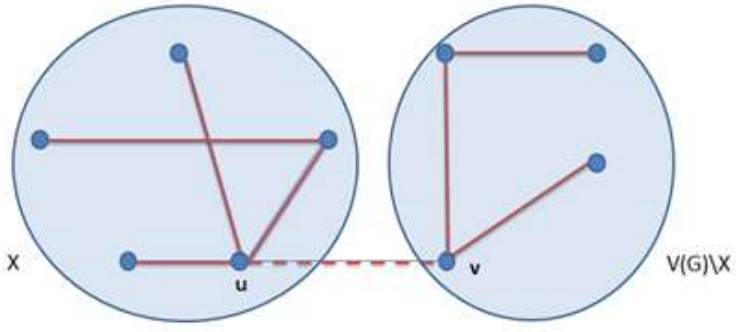


Figure 4: The set *X* and *V(G)\X* for a tree.

Add the edge $e=(u,v)$ to T'. Then, a cycle will occur in the T'. Because there is a path to connect the points u and v in the T'. T' is a one-part graph. Therefore, there is an edge that connects the points of X to the points of $V(G) \setminus X$ in this cycle. Let edge f be one of these edges. Since the e edge is added while applying the Prim algorithm, the e edge must be one of the shortest edges connecting the points of X to the points of $V(G) \setminus X$. This means that the length of edge e is smaller than the length of edge f or equal. Let edge e be a smaller than edge f and add edge e to the tree T'. In this case, the cycle is obtained. Let get tree T'' by removed the edge f from this cycle. Then, tree T'' becomes shorter than tree \$ T'. This is a contradiction, since tree T' is the MST containing edges $\{e_1,e_2,\ldots, e_k\}$. Thus, the length of edge f must be equal to the length of edge e. So tree T'' becomes the same length as tree T'.

The edges in the tree T (but not in T') can be added to the tree T' without changing its weight. At the end of the process, tree T' becomes tree T. So the length of T equals the length of T'. It can be seen that tree T is the MST containing edges $\{e_1,e_2,\ldots, e_k\}$.

3. AN APPLICATION

In this section, we will apply this algorithm to the graph shown in Figure 5 to see how the new algorithm works.

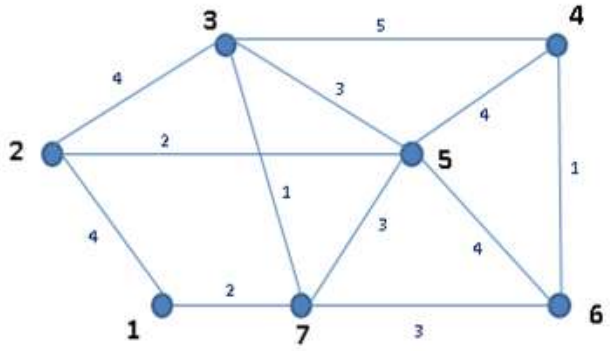


Figure 5.

Consider the edges {(1,7), (4,6), (5,4), (6,7)}. It is requested to find an MST containing these edges. According to the weights MST should also contain edges (7,3) and (5,2). However, the MST is still unique (Figure 6).

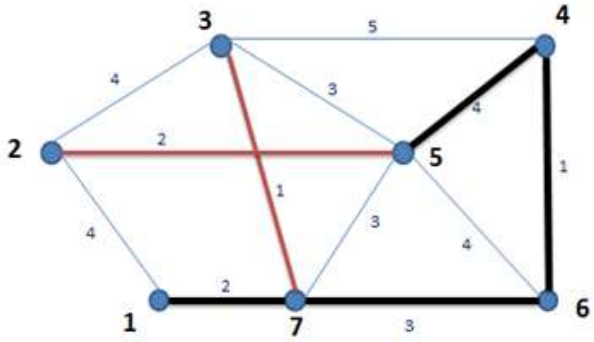


Figure 6.

Denote the tree T in Figure 6. Apply Algorithm 2 to tree T. Because of the first step of Algorithm 2, the Prim algorithm is applied to Figure 5 and an MST is found. Primarily, the tree $T = (V(T), E(T))$ is generated, where $V(T) = \{\ \}$ and $E(T) = \{\ \}$ A random point is selected from the graph G. Assume that the point *1* is selected. Add point *1* to the tree T. Therefore, $V(T) = \{1\}$ and $E(T) = \{\ \}$

In the first step, the edges connecting point *1* to other points of graph G are investigated and the shortest of these points is selected. The shortest edge is (1,7), since its length is *2* units. The edge (1,7) and the point *7* are added to the tree T. Then, $V(T) = \{1,7\}$, $E(T) = \{(1,7)\}$ (see Figure 7).

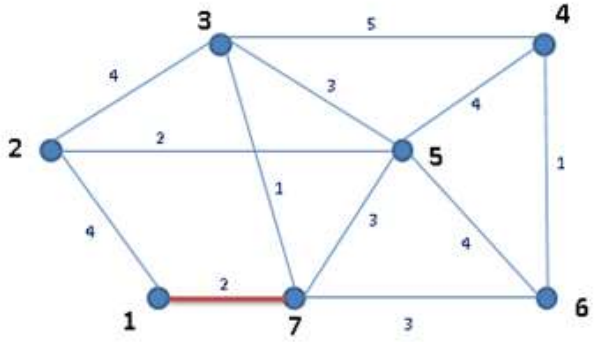


Figure 7.

In the second step, the edges connecting points of the set {1,7} to other points of the graph *G* are examined and the shortest of these points is selected. The edges are {(1,2), (7,3), (7,5), (7,6)}. The shortest edge is (7,3). The edge (7,3) and the point *3* are added to the tree *T*. Then, $V(T) = \{1,7,3\}$, $E(T) = \{(1,7),(7,3)\}$ (see Figure 8).

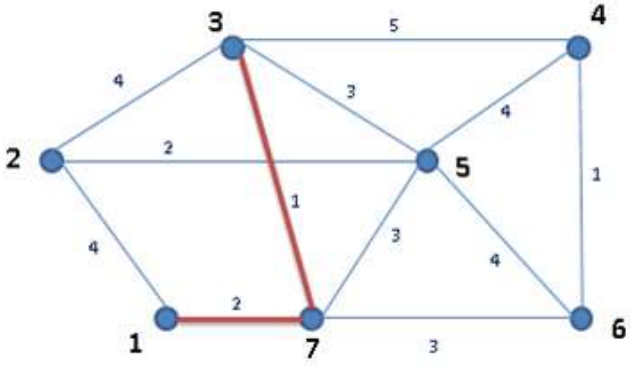


Figure 8.

For the third step, the edges connecting points of the set {1,7,3} to other points of the graph *G* are examined and the shortest of these points is selected. The edges are {(1,2), (3,2), (3,4), (3,5), (7,5), (7,6)}. The shortest edges are (3,5), (7,5), (7,6). Choose the edge (7,6). The edge (7,6) and the point *6* are added to the tree *T*. Then, *V(T)*={1,7,3,6}, E(T)={(1,7), (7,3), (7,6)} (see Figure 9).

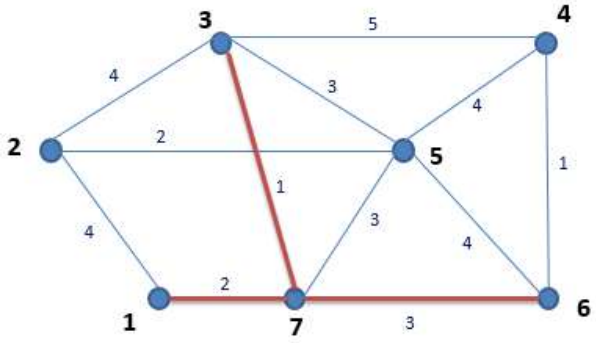


Figure 9.

In the fourth step, the edges connecting points of the set {1,7,3,6} to other points of the graph *G* are examined and the shortest of these points is selected. The edges are {(1,2), (3,2), (3,4), (3,5), (6,4), (6,5), (7,5)}. The shortest edge is (6,4). The edge (6,4) and the point *4* are added to the tree *T*. Then $V(T) = \{(1,73,6,4)\}$, (see $E(T) = \{(1,7),(7,3),(7,6),(6,4)\}$ Figure 10).

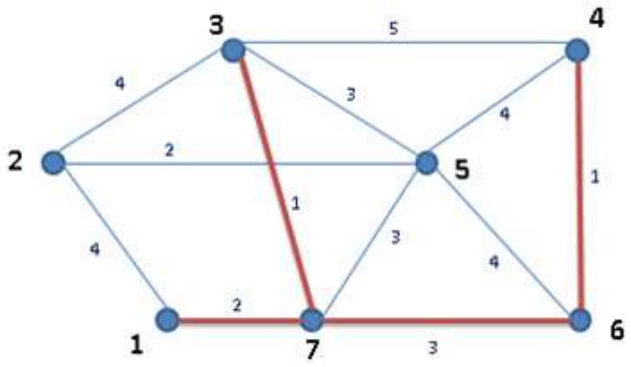


Figure 10.

In the fifth step, the edges connecting points of the set {1,7,3,6,4} to other points of the graph *G* are examined and the shortest of these points is selected. The edges are {(1,2), (3,2), (3,4), (3,5), (4,5), (6,5), (7,5)}. The shortest edges are (3,5), (7,5). Choose the edge (7,5). The edge (7,5) and the point *5* are added to the tree *T*. Then, $V(T) = \{1,7,3,6,4,5\}$, $E(T) = \{(1,7),(7,3),(7,6),(6,4),(7,5)\}$ (Figure 11).

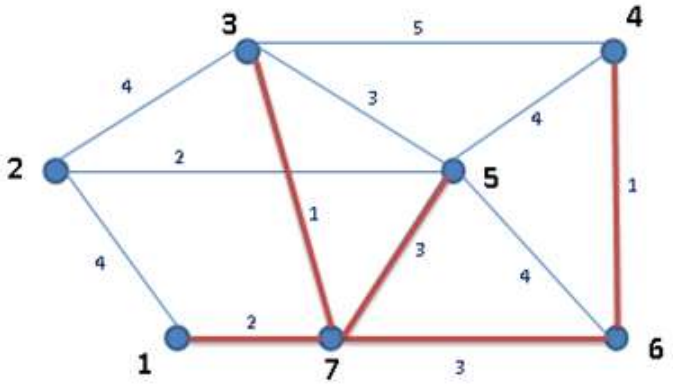


Figure 11.

For the step 6, the edges connecting points of the set {1,7,3,6,4,5} to other points of the graph *G* are examined and the shortest of these points is selected. These are {(1,2), (3,2), (3,4), (3,5), (4,5), (5,2), (6,5)}. The shortest edge is (5,2). The edge (5,2) and the point *2* are added to the tree *T*. Then, $V(T) = \{1,7,3,6,4,5,2\}$, $E(T) = \{(1,7),(7,3),(7,6),(6,4),(7,5),(5,2)\}$ (see Figure 12).

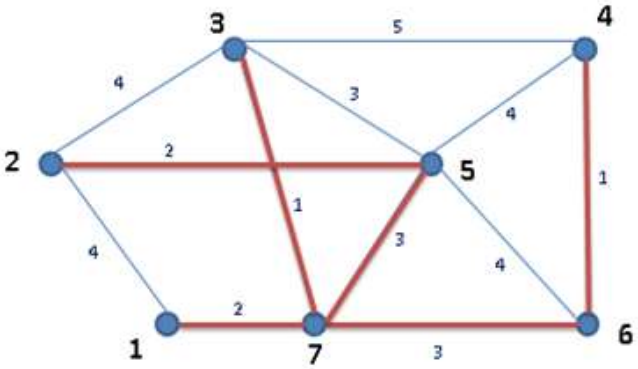


Figure 12.

At this point in the process, the points number of *T* equal to the points number of *G*. Thus, the Prim's algorithm will end up. Obtained graph at the end of the process is an MST of *G*.

Let's start with the second step of the algorithm. It is checked whether the tree *T* contains edges {(1,7), (4,6), (5,4), (6,7)}. The edge (5,4) is not found in the tree *T*. Then, S={(5,4)}. According to the third step of the algorithm, an edge is selected from the set *S*. This is the edge (5,4). The edge (5,4) is added to the tree *T* (Figure 13). In this case, the points *4*, *5*, *7* and *6* generate a cycle. In this cycle, find the longest edge except for the edges

{(1,7), (4,6), (5,4), (6,7)}. This is the edge (5,7), since its length is *3* units. The edge (5,7) is deleted from tree *T*.

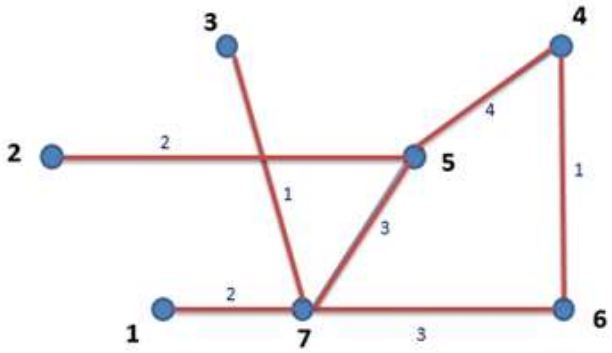


Figure 13.

In the last step of the algorithm, the edge (5,4) is removed from the set *S*. The set *S* becomes an empty set and the algorithm ends (Figure 14).

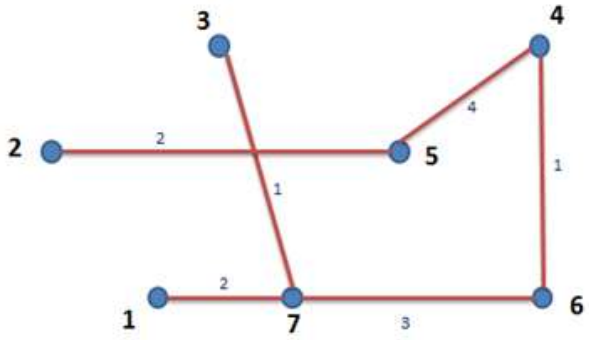


Figure 14.

4. CONCLUSION

In this work, a minimum spanning tree is studied and a new algorithm is given. The new algorithm uses the Prim algorithm as the first step. The given algorithm is applied on a sample. The MST has been generated for a company's computer systems. The MST generated by the new algorithm has a significant contribution to reducing the cost.

In this problem, the MST can be easily found because the number of points and edges are small. It is obvious that it will be difficult to find the MST when the number of points and edges increases. The new algorithm will solve the problem easily and quickly even if the number of points and edges increases.

REFERENCES

Boruvka, O., (1926a) O jistem problem minimalnim(About a certain minimal problem), Prace mor. Prirodove d. spol. V Brne III(3): 37-58.

Boruvka, O., (1926b) Prispevek k reseni otazky ekonomicke stavby elekrovodnich siti(Contribution to the solution of a problem of economical construction of electrical networks), Elektrnicky obzor, 15: 153-154.

Nesteril, J., Mikova, E., Nesetrilova, H., (2001) Otkar Boruvka on minimum spanning tree problem Translation of the both the 1926 papers, comments, history, Discrete Math., 233(1-3): 3-36

Prim, R.C., (1957) Shortest connection networks and some generalizations, Bell Syst. Tech. J., 36: 567-574.

Krishnamoorthy, M. Ernst, A.T., Sharaiha, Y.M., (2001) Comparison of algorithms for the degree constrained minimum spanning tree, Journal of Heuristics, 7: 587-611.

http://scanftree.com/Data_Structure/prim\%27s-algorithm.