Prim’s Algorithm for minimum spanning tree

Chao Yuan[*] and Haofei Zhang

Computer science, Beijing Jiaotong University, Beijing, China

[*]Corresponding author: 19722073@bjtu.edu.cn

Keywords: Spanning tree; Minimum spanning tree; MST; Prim’s Algorithm

Abstract: Prim's algorithm is an algorithm that can search for the minimum spanning tree (MST) in a weighted connectivity diagram. It has been widely used in many fields such as communication, internet and programming. Based on these studies, here we further elaborated the details of Prim's algorithm such as the basic steps of the Prim's algorithm, pseudo-code, theoretical and practical examples, and codes in Java representing the Prim's algorithm to help others understand it. We are also studied the ways of improvements and optimizations to the Prim's algorithm and he deeper understanding and development of Prim's algorithms can optimize technology in many areas.

1. Use Prim’s Algorithm to generate minimum spanning tree

A minimum spanning tree can be generated by many ways, there are two main ways that widely used. The Prim’s Algorithm is one of them.

1.1 Introduction

The algorithm was first discovered by the Czech mathematician Vojtˇech Jarník in 1930, who published a paper in an unknown Czech journal. When Robert prim rediscovered the algorithm in 1957, it became widely known. Therefore, it is called prim algorithm (sometimes called Prim-Jarník algorithm). [1]
It is a procedure for producing a MST in a weighted graph that successively adds edges with minimal weight among all edges incident to a vertex already in the tree so that no edge produces a simple circuit when it is added.

1.2 Basic steps

a. Select the edge with the minimum weight in the graph and add it to the spanning tree to be established.

b. Then, the minimum weight edge associated with the vertex in the established tree is added to the tree, and the edge is required to be added without forming a simple loop with the edge in the tree.

c. When the tree has been added to have n-1 edges, the execution of the algorithm is terminated.

1.3 Pseudo code

```
    Procedure: Prim (G: undirected connected weight graph with n vertices)
T: = minimum- edge weight
    for i: =1 to n−2
    e: = edge of minimum weight incident to a vertex in T and not forming
    simple circuit in T if added to T
    T: = T with e added
    return T {T is a MST}
```

1.4 Theoretical Example

Find the MST

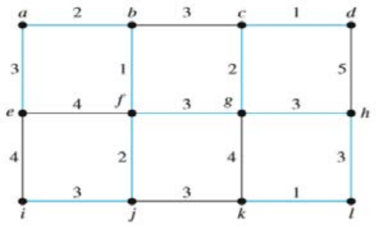


Figure 1. An example of minimum tree

Table 1. The way to find a minimum tree in Figure 1

Choice	Edge	Weight
1	{b, f}	1
2	{a, b}	2
3	{f, j}	2
4	{a, e}	3
5	{i, j}	3
6	{f, g}	3
7	{c, g}	2
8	{c, d}	1
9	{g, h}	3
10	{h, l}	3
11	{k, l}	1

Prim's algorithm is realized by selecting the initial edge of the minimum weight, and then adding the minimum weight edge which is related to a vertex in the tree and does not constitute a simple circuit.

So first choose the minimum weight edge {b, f}, and then add the other edges with minimum weight which are incident to a vertex of the chosen edge (As the Table 1 shows). And the blue edges make up the MST.

1.5 Practical Example

The main purpose of laying optical cables between n cities is to enable any two cities to communicate, but the cost of laying optical cables is very high, and the cost of laying optical cables between cities is different, so the other goal is to minimize the total cost of laying optical cables. This requires finding the weighted MST.

This is a weighted graph showing monthly lease costs for lines in a computer network.

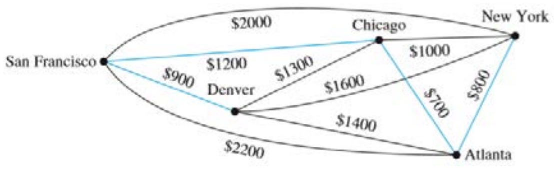


Figure 2. The weight graph of cities

Find the minimum cost by finding a MST whose sum of the weights of the edges of the tree is minimized.

Table 2. The way to find a minimum tree in Figure 4

Choice	Edge	Cost
1	{Chicago, Atlanta}	$700
2	{Atlanta, New York}	$800
3	{Chicago, San Francisco}	$1200
4	{San Francisco, Denver}	$900

As the blue lines in Figure2 and Table2 show, the minimum cost is $3600. [1]

1.6 Understanding of Prim’s Algorithm

Example:

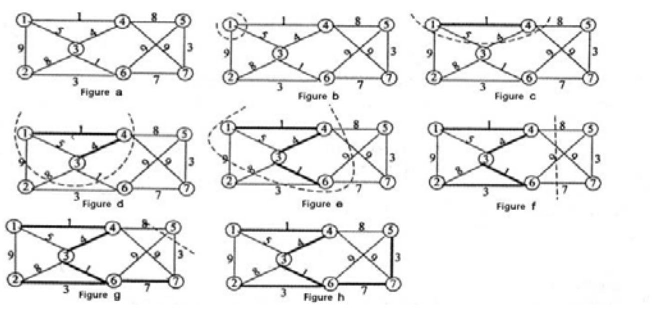


Figure 3. The process of Prim’s Algorithm

Let V be the set of all the vertices in a graph. the vertices in the dotted line belongs to set U, and the vertices outside the dotted line belongs to set V-U.

Procedures:

a. Firstly, all the vertices are in the set V-U. Then, select a vertex randomly to move it from set V-U to set U. In this example, choose vertex 1. So, U = {1}, V-U = {2, 3, 4, 5, 6, 7}, as shown in Figure b.

b. Secondly, because the edge (1,4) has the least weight among all the edges connected with vertex 1, so, move vertex 4 from V-U to U, and move 4 to the inside of the dotted line, then U = {1, 4}, V-U = {2, 3, 5, 6, 7}, as shown in Figure c.

c. Thirdly, the candidate vertices which have edge between set U and set V-U are 2,3,5,7. Because all the edges which one of the ends are in set U and the edges of the other end in set V-U only the edge (4, 3) has the minimum weight. So, move 3 from V-U to U, then U = {1, 3, 4}, V-U = {2, 5, 6, 7}, as shown in Figure d.

d. Fourthly, the candidate vertices are 2, 5, 6, 7. Since all the edges which one of the ends are in set U and the edges of the other end in set V-U, only the edge (4, 3) has the minimum weight. So, move 6 from V-U to U, then U = {1, 3, 4, 6}, V-U = {2, 5, 7}, as shown in Figure e.

e. Fifthly, the candidate vertices are 2,5,7. Because all the edges which one of the ends are in set U and the edges of the other end in set V-U, only the edge (6, 2) has the minimum weight., so, move 2 to U from V-U, then U = {1, 2, 3, 4, 6}, V-U = {5, 7}, as shown in Figure f.

f. Sixthly, only 5 and 7 are candidate vertices. Since all the edges which one of the ends are in set U and the edges of the other end in set V-U, only the edge (6, 7) has the minimum weight, so move 7 from V-U to U, then U = {1, 2, 3, 4, 6, 7}, V-U = {5}, as shown in Figure g.

g. Finally, the last edge (7, 5) has the minimum weight, and then moves 5 from V-U to U. now, all the vertices are in the set U, and get a MST, as shown in Figure h.[3]

1.7 Improvement and optimization of Prim’s Algorithm

Idea 1: Optimizing connected graph

If there is a unique shortest edge in the cut set, it must be included in every spanning tree, then take it as a fixed edge.

By using the idea of ‘Breaking circle method ‘, Remove the only longest edge in the basic loop, and keep the only shortest edge in the basic cut set. Repeat this process, and finally get the MST. On the basis of the optimization of the connected weight graph, can find all the MSTs, and the calculation amount will be greatly reduced, and the efficiency of the algorithm will be greatly improved.

Idea 2: The way mentioned in 1.6

In this way, do not need to find the minimum edge among all the edge of a connected weight graph, just pick any vertex in the graph and choose its minimum edge with other vertices. And this will save a lot of time of finding the minimum edge, and simplify the algorithm. [2]

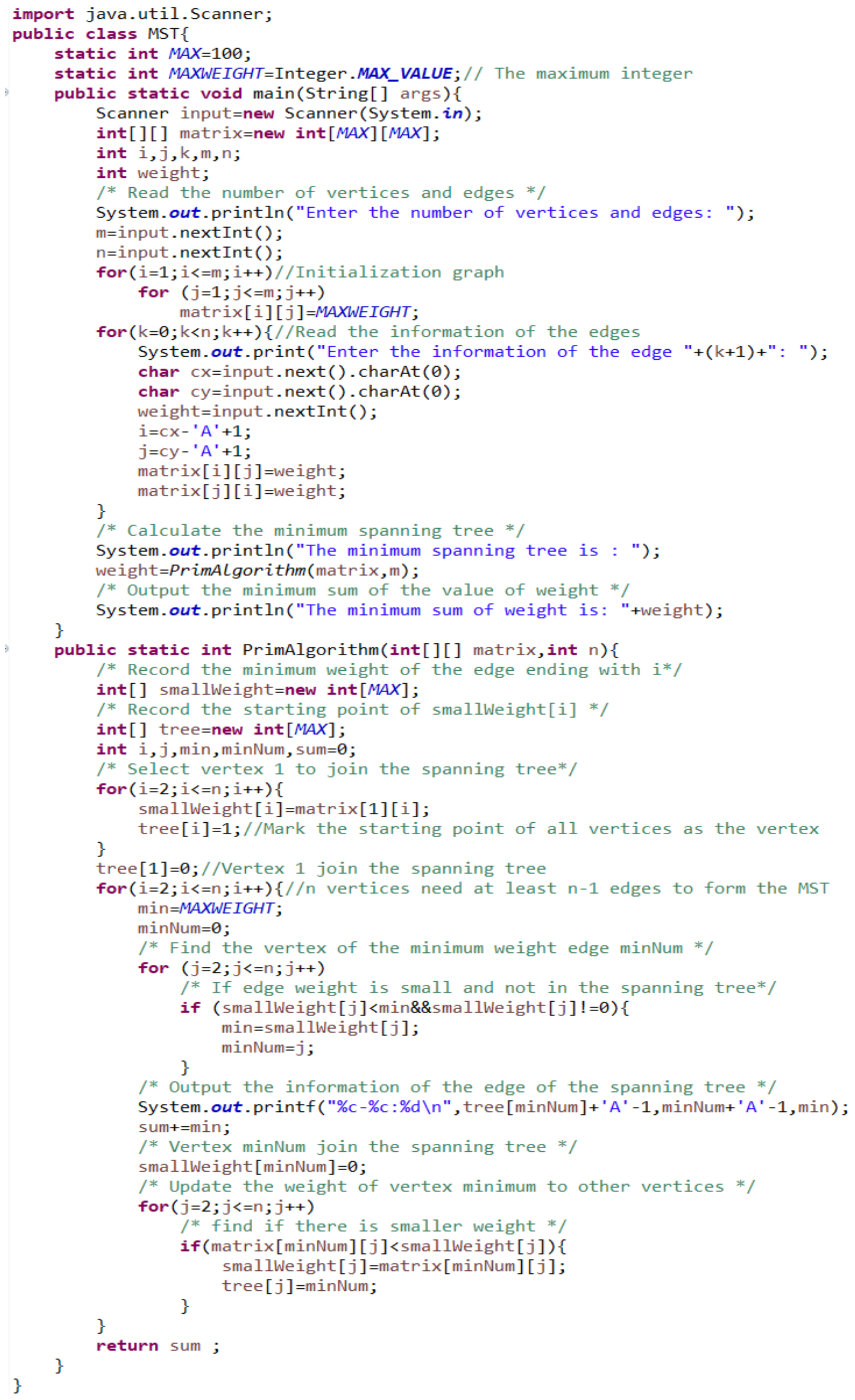
Idea 3: Dynamic optimization to avoid repeated access

On the one hand, the adjacency multiple bidirectional linked list is used as the storage structure, and then a constant time quantity is defined to mark that when an edge is added to U, the edge is deleted from the linked list, which ensures that the added edge will not be accessed again. On the other hand, the auxiliary array matrix of prim algorithm is improved to minimize the comparison times of finding the minimum weight edge in the current u to V-U set, that is, the operation of finding the minimum weight of matrix is not performed. The specific operation is to establish a two-way linked list VU, which consists of non-zero items in the original ‘smallWeight’. Its main function is to store the x edges with the minimum weight from the U set to the V-U set, where x ≤| V-U | (the length of the linked list is equal to x), so as to avoid judging the non-zero items in ‘smallWeight’.

1.8 Implementation of Prim’s Algorithm (JAVA)

When the weighted graph is very complex, it is very difficult for us to calculate the results with the human brain. At this time, use the idea of programming to let computers help us find the MST.

The relevant JAVA source code is shown below.



```java
import java.util.Scanner;
public class MST{
    static int MAX=100;
    static int MAXWEIGHT=Integer.MAX_VALUE;// The maximum integer
    public static void main(String[] args){
        Scanner input=new Scanner(System.in);
        int[][] matrix=new int[MAX][MAX];
        int i,j,k,m,n;
        int weight;
        /* Read the number of vertices and edges */
        System.out.println("Enter the number of vertices and edges: ");
        m=input.nextInt();
        n=input.nextInt();
        for(i=1;i<=m;i++)//Initialization graph
            for (j=1;j<=m;j++)
                matrix[i][j]=MAXWEIGHT;
        for(k=0;k<n;k++){//Read the information of the edges
            System.out.print("Enter the information of the edge "+(k+1)+": ");
            char cx=input.next().charAt(0);
            char cy=input.next().charAt(0);
            weight=input.nextInt();
            i=cx-'A'+1;
            j=cy-'A'+1;
            matrix[i][j]=weight;
            matrix[j][i]=weight;
        }
        /* Calculate the minimum spanning tree */
        System.out.println("The minimum spanning tree is : ");
        weight=PrimAlgorithm(matrix,m);
        /* Output the minimum sum of the value of weight */
        System.out.println("The minimum sum of weight is: "+weight);
    }
    public static int PrimAlgorithm(int[][] matrix,int n){
        /* Record the minimum weight of the edge ending with i*/
        int[] smallWeight=new int[MAX];
        /* Record the starting point of smallWeight[i] */
        int[] tree=new int[MAX];
        int i,j,min,minNum,sum=0;
        /* Select vertex 1 to join the spanning tree*/
        for(i=2;i<=n;i++){
            smallWeight[i]=matrix[1][i];
            tree[i]=1;//Mark the starting point of all vertices as the vertex
        }
        tree[1]=0;//Vertex 1 join the spanning tree
        for(i=2;i<=n;i++){//n vertices need at least n-1 edges to form the MST
            min=MAXWEIGHT;
            minNum=0;
            /* Find the vertex of the minimum weight edge minNum */
            for (j=2;j<=n;j++)
                /* If edge weight is small and not in the spanning tree*/
                if (smallWeight[j]<min&&smallWeight[j]!=0){
                    min=smallWeight[j];
                    minNum=j;
                }
            /* Output the information of the edge of the spanning tree */
            System.out.printf("%c-%c:%d\n",tree[minNum]+'A'-1,minNum+'A'-1,min);
            sum+=min;
            /* Vertex minNum join the spanning tree */
            smallWeight[minNum]=0;
            /* Update the weight of vertex minimum to other vertices */
            for(j=2;j<=n;j++)
                /* find if there is smaller weight */
                if(matrix[minNum][j]<smallWeight[j]){
                    smallWeight[j]=matrix[minNum][j];
                    tree[j]=minNum;
                }
        }
        return sum ;
    }
}
```

Figure 4. Source code

1.9 Example of the program

Example:
Find the MST of Figure 5.

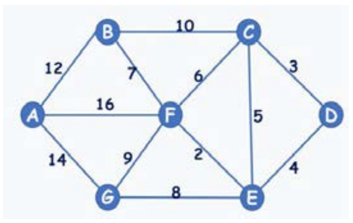
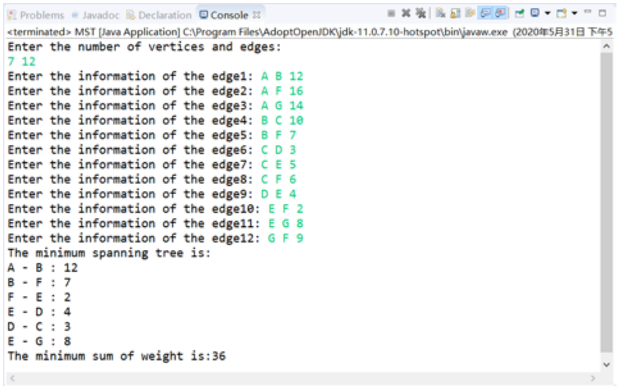


Figure 5. Program example

As the screenshot shows:



```
<terminated> MST [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.7.10-hotspot\bin\javaw.exe (2020年5月31日 下午5
Enter the number of vertices and edges:
7 12
Enter the information of the edge1: A B 12
Enter the information of the edge2: A F 16
Enter the information of the edge3: A G 14
Enter the information of the edge4: B C 10
Enter the information of the edge5: B F 7
Enter the information of the edge6: C D 3
Enter the information of the edge7: C E 5
Enter the information of the edge8: C F 6
Enter the information of the edge9: D E 4
Enter the information of the edge10: E F 2
Enter the information of the edge11: E G 8
Enter the information of the edge12: G F 9
The minimum spanning tree is:
A - B : 12
B - F : 7
F - E : 2
E - D : 4
D - C : 3
E - G : 8
The minimum sum of weight is:36
```

Figure 6. Output screenshot

References

[1] Rosen, K., (2012), Discrete Mathematics and its Applications, McGraw-Hill.

[2] Chen, B., and X., (2013), MST Dynamic Demonstration System Implementation, Applied Mechanics and Materials.

[3] Martel, C., (2002), The expected complexity of Prim's minimum spanning tree algorithm, Information Processing Letters, 81(4), p197-201.