



Department of
Computer Science and Engineering

Title: Implement Prim's Algorithm

Algorithms Lab

CSE 206

Submitted by : Shamim Ahmed

ID: 201902067



Green University of Bangladesh

1 Objective(s)

- To learn Prim's algorithm to find MST of a graph.

2 Problem Analysis

2.1 Prim's Algorithm

Prim's algorithm is a minimum spanning tree algorithm that takes a graph as input and finds the subset of the edges of that graph which

- form a tree that includes every vertex.
- has the minimum sum of weights among all the trees that can be formed from the graph.

2.2 How Prim's algorithm works

It falls under a class of algorithms called greedy algorithms that find the local optimum in the hopes of finding a global optimum. We start from one vertex and keep adding edges with the lowest weight until we reach our goal. The steps for implementing Prim's algorithm are as follows:

- Initialize the minimum spanning tree with a vertex chosen at random.
- Find all the edges that connect the tree to new vertices, find the minimum and add it to the tree. •

Keep repeating step 2 until we get a minimum spanning tree.

2.3 Example of Prim's algorithm

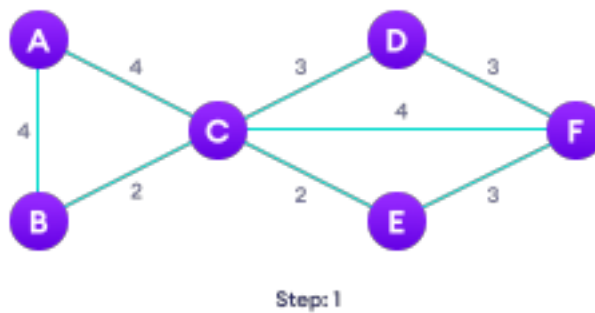


Figure 1: Start with a weighted graph



(a) Choose the edge with the least weight, if there are more than 1, choose any (b) Choose the next shortest edge and add it Figure 2: Step 2 and 3



Step: 5

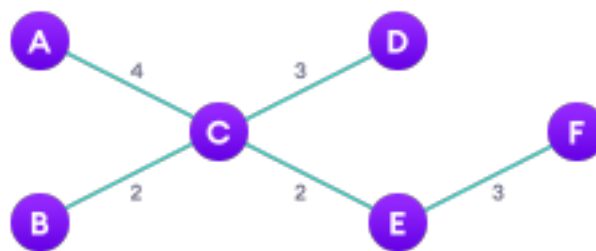


Step: 4

(a) Choose the next shortest edge that doesn't create a cycle and add it

(b) Choose the next shortest edge that doesn't create a cycle and add it

Figure 3: Step 4 and 5



Step: 6

Figure 4: Repeat until you have a spanning tree

3 Algorithm

Algorithm 1: Prim's Algorithm

```

1 T = ∅;
2 U = 1 ;
3 while (U ≠ V) do
4 let (u, v) be the lowest cost edge such that u ∈ U and v ∈ V - U;
5 T = T ∪ (u, v)
6 U = U ∪ v
7 end
  
```

4 Implementation in C++

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
#define V 5           //No of vertices
```

```
int selectMinVertex(vector<int>& value,vector<bool>& setMST)
```

```
{
```

```
    int minimum = INT_MAX;
```

```
    int vertex;
```

```
    for(int i=0;i<V;++i)
```

```
    {
```

```
        if(setMST[i]==false && value[i]<minimum)
```

```
        {
```

```
            vertex = i;
```

```
            minimum = value[i];
```

```
        }
```

```
    }
```

```
    return vertex;
```

```
}
```

```
void findMST(int graph[V][V])
```

```
{
```

```
    int parent[V];
```

```
vector<int> value(V,INT_MAX);
```

```
vector<bool> setMST(V,false);
```

```
parent[0] = -1;
```

```
value[0] = 0;
```

```
for(int i=0;i<V-1;++i)
```

```
{
```

```
    int U = selectMinVertex(value,setMST);
```

```
    setMST[U] = true;
```

```
    for(int j=0;j<V;++j)
```

```
    {
```

```
        if(graph[U][j]!=0 && setMST[j]==false && graph[U][j]<value[j])
```

```
        {
```

```
            value[j] = graph[U][j];
```

```
            parent[j] = U;
```

```
        }
```

```
    }
```

```

    }

    //Print MST

    for(int i=1;i<V;++i)

        cout<<"U->V: "<<parent[i]<<"->"<<i<<" wt = "<<graph[parent[i]][i]<<"\n";

    }


int main()

{

    int graph[V][V] = { { 0, 2, 0, 6, 0 }, { 2, 0, 3, 8, 5 }, { 0, 3, 0, 0, 7 }, { 6, 8, 0, 0, 9 }, { 0, 5, 7, 9, 0 } };

    findMST(graph);

    return 0;

}

```

5 Sample Input/Output (Compilation, Debugging & Testing)

Edge Weight

```

0 - 1 => 2
1 - 2 => 3
0 - 3 => 6
1 - 4 => 5

```

```
prims_algorithm.cpp - Code::Blocks 20.03
File Edit View Search Project Build Debug Tools Plugins Settings Help

Start here 0-1 Knapsack.cpp prims_algorithm.cpp

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 #define V 5 //No of vertices
5
6 int selectMinVertex(vector<int>& value,vector<bool>& setMST)
7 {
8     int minimum = INT_MAX;
9     int vertex;
10    for(int i=0;i<V;++i)
11    {
12        if(setMST[i]==false && value[i]<minimum)
13        {
14            vertex = i;
15            minimum = value[i];
16        }
17    }
18    return vertex;
19 }
20
21 void findMST(int graph[V][V])
22 {
23     int parent[V];
24     vector<int> value(V,INT_MAX);
25     vector<bool> setMST(V,false);
26
27     parent[0] = -1;
28     value[0] = 0;
29
30     for(int i=0;i<V-1;++i)
31     {
32
33         int U = selectMinVertex(value,setMST);
34         setMST[U] = true;
35
36         for(int v=0;v<V;v++)
37         {
38             if(graph[U][v]<value[v] && setMST[v]==false)
39             {
40                 value[v] = graph[U][v];
41             }
42         }
43     }
44 }
45
46 int main()
47 {
48     int graph[V][V];
49     for(int i=0;i<V;i++)
50     {
51         for(int j=0;j<V;j++)
52         {
53             graph[i][j] = 0;
54         }
55     }
56     graph[0][1] = 2;
57     graph[1][2] = 3;
58     graph[0][3] = 6;
59     graph[1][4] = 5;
60
61     findMST(graph);
62
63     return 0;
64 }
```

```
/home/shamim/Desktop/C++/prims_algorithm
+
U->V: 0->1 wt = 2
U->V: 1->2 wt = 3
U->V: 0->3 wt = 6
U->V: 1->4 wt = 5

Process returned 0 (0x0)   execution time : 0.003 s
Press ENTER to continue.
```

7 Lab Task

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
int rep[10000];
```

```
vector<int>va;
```

```
int edg;
```

```
struct edge
```

```
{
```

```
    int a, b , c;
```

```
}arr[100005];
```

```
bool cmp( edge x, edge y )
```

```
{  
    return x.c < y.c;  
}
```

```
void makeset(int n)
```

```
{  
    for(int i=1;i<=n;i++) rep[i]=i;  
}
```

```
int findr( int x )
```

```
{  
    if( rep[x] == x ) return x;  
  
    return rep[x] = findr( rep[x] );  
}
```

```
int unio(int i,int sum)
```

```
{  
    int x,y;  
    x = findr( arr[i].a );  
    y = findr( arr[i].b );  
    if( x != y )  
    {  
        rep[x] = y;  
        va.push_back(i);  
        sum += arr[i].c;  
    }
```



```
    }  
    return sum;  
}
```

```
int unio2(int i,int sum)  
{  
    int x,y;  
    x = findr( arr[i].a );  
    y = findr( arr[i].b );  
    if( x != y )  
    {  
        rep[x] = y;  
        sum += arr[i].c;  
        edg++;  
    }  
    return sum;  
}
```

```
int main()  
{  
    int n , m;  
    cin >> n >> m;  
  
    makeset(n);  
  
    for( int i = 0; i < m; i++ )  
    {  
        int a, b, c ;  
        cin >> a >> b >> c;
```

```
    arr[i].a = a;
    arr[i].b = b;
    arr[i].c = c;
}
```

```
sort( arr, arr+m , cmp );
```

```
int sum=0;
for(int i=0;i<m;i++)
{
    sum=unio(i,sum);
}
```

```
cout << "MST: " << sum << "\n"; //cost
```

```
int sec_best_mst=INT_MAX/3;
```

```
cout<<"All other spanning trees:\n";
```

```
sum=0;
int j;
for(j=0;j<va.size();j++)
{
    makeset(n);
    edg=0;
    for(int i=0;i<m;i++)
    {
        if(i==va[j]) continue;
```

```

        sum=unio2(i,sum);
    }
    if(edg!=n-1)
    {
        sum=0;
        continue;
    }
    cout<<sum<<"\n" ;
    if(sec_best_mst>sum) sec_best_mst = sum;
    sum=0;
}

```

```

cout<<"SEC BEST MST: "<<sec_best_mst<<"\n";

```

```

}

```

```

/*

```

```

6 8

```

```

1 3 4

```

```

1 2 4

```

```

2 3 2

```

```

3 4 3

```

```

3 6 4

```

```

3 5 2

```

```

4 6 3

```

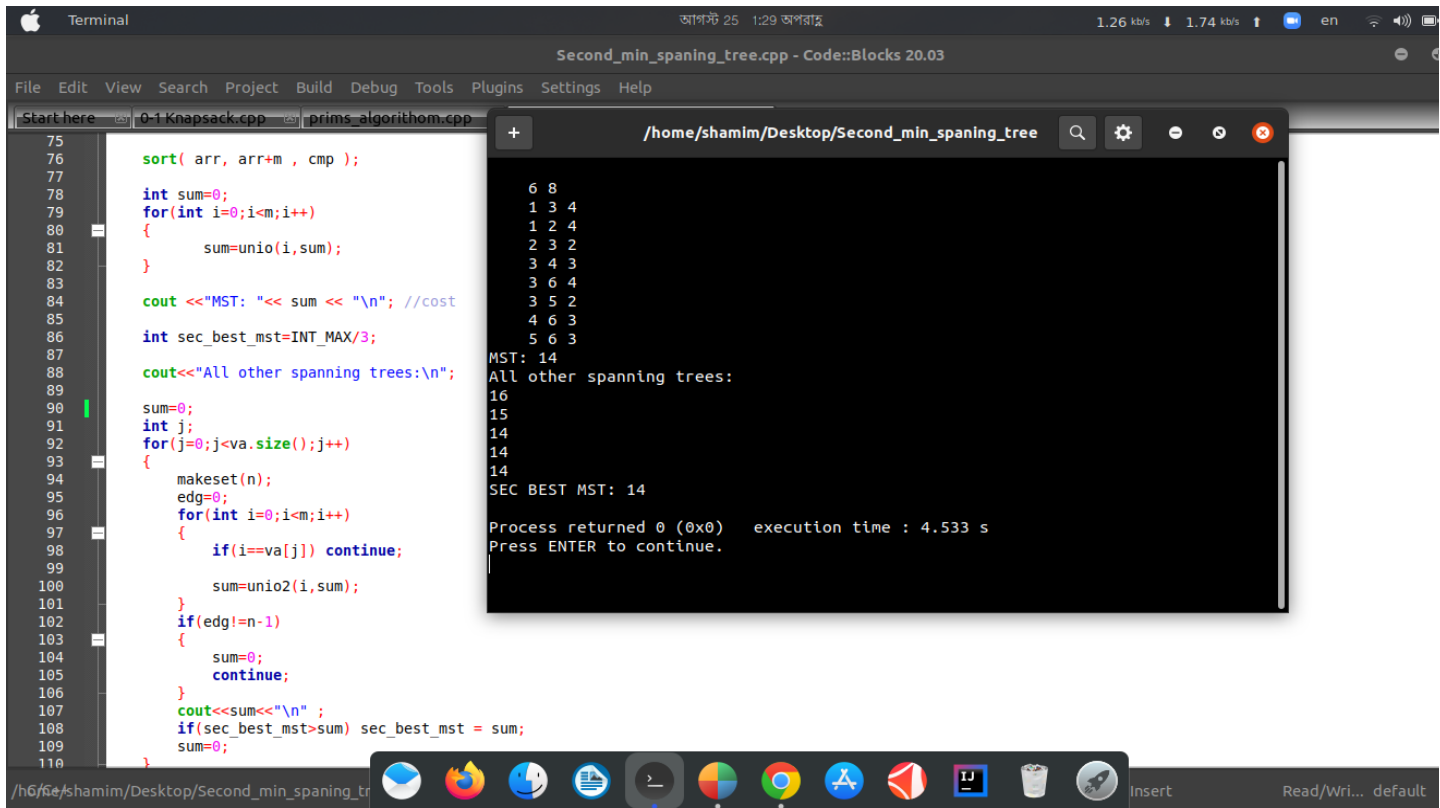
```

5 6 3

```

*/

8 Lab Tasks output



The screenshot shows the Code::Blocks IDE with a C++ project named 'Second_min_spaning_tree.cpp'. The code implements Kruskal's algorithm to find the second minimum spanning tree. The output window displays the following results:

```
6 8
1 3 4
1 2 4
2 3 2
3 4 3
3 6 4
3 5 2
4 6 3
5 6 3
MST: 14
All other spanning trees:
16
15
14
14
14
SEC BEST MST: 14
Process returned 0 (0x0)   execution time : 4.533 s
Press ENTER to continue.
```

9. Discussion

The time complexity for second minimum spanning tree is : $O(V^2)$. Here we use kruskal algorithm . It was much easier to implement kruskar than prims algorithm for second minimum spanning tree.