

**Data Structures and algorithms (CS09203)**

**Lab Report**

Name: Wajih-ul-hassan

Registration #: SEU-F16-151

Lab Report #: 12

Dated: 27-06-2018

Submitted To: Mr. Usman Ahmed

The University of Lahore, Islamabad Campus

Department of Computer Science & Information Technology

# Experiment # 12 prim’s algorithml

**Objective**

The objective of this session is to show the representation of trees using C++.

**Software Tool**

1. Code Blocks with GCC compiler.

# Theory

Prim’s algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. The algorithm operates by building this tree one vertex at a time, from an arbitrary starting vertex, at each step adding the cheapest possible connection from the tree to another vertex.

# Task

**2.1 Task 1**

Impement Prim’s algorithm.

## Procedure: Task 1

// A C / C++ program for Prim ’ s Minimum Spanning Tree (MST) algorithm .

// The program is for adjacency matrix representation of the graph #include *<*stdio .h*>*

#include *<*limits .h*>* using namespace std ;

// Number of vertices in the graph

#define V 8

// A utility function to find the vertex with minimum key value , from

// the set of vertices not yet included in MST int minKey( int key [] , bool mstSet [ ] )

{

// I n i t i a l i z e min value int min = INT MAX, min index ;

for ( int v = 0; v *<* V; v++)

i f (mstSet [ v ] == false && key [ v ] *<* min) min = key [ v ] , min index = v ;

return min index ;

}

// A utility function to print the constructed MST stored in parent [ ] int printMST( int parent [] , int n, int graph [V] [V])

{

printf (”Edge Weight\n”);

for ( int i = 1; i *<* V; i++) printf(”%d − %d %d \n” , parent [ i ] , i , graph [ i ] [ parent [ i ] ] ) ;

}

// Function to construct and print MST for a graph represented using adjace

// matrix representation void primMST( int graph [V] [V])

{

int parent [V] ; // Array to store constructed MST

int key [V] ; // Key values used to pick minimum weight edge in cut bool mstSet [V] ; // To represent set of vertices not yet included in M

// I n i t i a l i z e all keys as INFINITE for ( int i = 0; i *<* V; i++) key [ i ] = INT MAX, mstSet [ i ] = false ;

// Always include f i r s t 1st vertex in MST.

key [0] = 0; // Make key 0 so that this vertex is picked as f i r s t parent [0] = −1; // First node is always root of MST

// The MST will have V vertices for ( int count = 0; count *<* V−1; count++)

{

// Pick the minimum key vertex from the set of vertices

// not yet included in MST int u = minKey(key , mstSet );

// Add the picked vertex to the MST Set mstSet [u] = true ;

// Update key value and parent index of the adjacent vertices of

// the picked vertex . Consider only those vertices which are not ye

// included in MST for ( int v = 0; v *<* V; v++)

// graph [u ] [ v ] is non zero only for adjacent vertices of m // mstSet [ v ] is false for vertices not yet included in MST

v

// Update the key only i f graph [u ] [ v ] is smaller than key [ v ] i f (graph [u ] [ v ] && mstSet [ v ] == false && graph [u ] [ v ] *<*

key [ v ])

parent [ v ] = u, key [ v ] = graph [u ] [ v ] ;

}

// print the constructed MST printMST( parent , V, graph );

}

// driver program to test above function int main()

{

int graph [V] [V] = {{1 , 8 , 0 , 0 , 0 ,10 ,0 ,5} ,

{8 , 0 , 4 , 0 , 4 ,4 ,0 ,4} ,

{0 , 4 , 0 , 3 , 0 ,3 ,0 ,0} ,

{0 , 0 , 3 , 0 , 1 ,6 ,2 ,0} ,

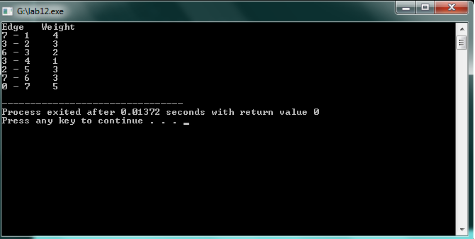


Figure 1: output

{0 , 4 , 0 , 1 , 0 ,0 ,3 ,0} ,

{10 , 4 , 3 , 6 , 0 ,0 ,0 ,0} ,

{0 , 0 , 0 , 2 , 3 ,0 ,0 ,3} ,

{5 , 4 , 0 , 0 , 0 ,0 ,3 ,0} ,

};

// Print the solution primMST(graph );

return 0;

}