"Mircea cel Batran" Naval Academy Scientific Bulletin, Volume XV – 2012 – Issue 2
Published by "Mircea cel Batran" Naval Academy Press, Constanta, Romania

A NEW FAST IMPLEMENTATION OF THE KRUSKAL'S ALGORITHM

Paul VASILIU [1]
[1] Lecturer Engineer, PhD "Mircea cel Bătrân" Naval Academy, Constanța, România

Abstract: *In this paper it is described a new fast implementation in the 'C' programming language of the Kruskal's algorithm to computing the Minimum Spanning Tree (MST) for a given* undirected *connex graph* with edges that hold values*.*
Keywords: *Algorithm, Minimum,* Spanning, Tree

1. INTRODUCTION

Let it $G = (X, U, l)$, $X = \{x_1, x_2, \ldots, x_n\}$ a connex graph (necessary assumption to ensure the existence of at least one tree) undirected with edges that hold values $l(x_i, x_j)$.

A Spanning Tree is a free tree that contains part of the edges of the graph that cover all the vertices of the graph. One edge covers the vertices that it unites.

A connex graph may have multiple spanning trees. If it is encountered a high number of cycles in the graph, than there will be also a high number of spanning trees in this graph. For a connex graph with n vertices there will be spanning trees that have $n-1$ edges.

We are interested in computing the MST for a given graph. In more details, we are looking for linking edges of the graph between all vertices so that we can obtain the minimum sum of the edges' values from the graph.

This problem is found in the computer networking field, where the purpose is to minimize the length of the cable that is used to connect all the nodes that must communicate between them. Moreover, the problem is encountered in the road network design, sewer systems, computer networks.

2. KRUSKAL'S ALGORITHM

The algorithm has been developed in 1956 by Joseph Kruskal. The algorithm is a greedy algorithm in graph theory that finds a MST for a connected weighted 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 (2).

Let it be $G = (X, U, l)$, $X = \{x_1, x_2, \ldots, x_n\}$, a finite graph, undirected, connex having edges with values. The idea of the Kruskal's algorithm is to select at each step the edges with minimum value from the unselected edges. The constraint is that the selected edge does not form a cycle with the edges already included in MST (previously selected).

The condition for the edge (x_i, x_j), not to form a cycle with the other edges previously selected, is that the vertices x_i and x_j to be in distinct connex components.

At the beginning, each vertex forms a connex component. Next, a connex component contains all the vertices covered with edges from MST and the uncovered vertices build other connex components.

An edge that links two vertices from the same connex component will build a cycle with the previously selected edges. This type of edge is not accepted. It is allowed only an edge that links vertices from distinct connex components.

Let it be nrc the number of connex components of the graph. At the beginning, it is considered that nrc = n.

The Kruskal's 1 algorithm for discovering of a MST from G graph can be described as it follows:

```
Reading data
Ascending sorting of the edges' value
Create sorted queue of the edges
nrc=n
repeate
{
        extract the edge with the minimum value from the queue
        if the extracted edges is acceptable
        {
                print nrc
                nrc = nrc -1
                update the connex components
        }
}
until nrc == 1
```

3. ALGORITHM IMPLEMENTATION

Let it be $G = (X, U, l)$, $X = \{x_1, x_2, \ldots, x_n\}$, a finite graph, undirected, connex having edges with values and m edges.

The program has as input a text file with m+1 lines.

On the first line we find n, the number of vertices and m, the number of edges from the G graph separated with a blank space. On each of the following m lines we have the edges and the values from the G graph, separated with one blank space.

The program determines the MST for the G graph.

```
// Kruskal's algorithm
#include <stdio.h>
```

```
#include <stdlib.h>
#include <conio.h>
#include <malloc.h>
#define dim 1000                          // maximum number of vertices from the graph

typedef struct SET
{
int c;                                    //c  the number of elements from the set
int S[dim];                               //S the array with the set elements
} set;

typedef struct EDGE
{
int x; int y; int val;
}edge;

edge mu[dim];                             // mu the edges' array
set C[dim];                               // C the array of the connex components

// Allocate matrix / 2-dimesional array
// Function to allocate the squared matrix / 2-dimensionl array of n+1 dimension
// The function returns the address of the matrix or NULL
int ** alocmat (int n)
{
int i;
int ** p=(int **) malloc ((n+1)*sizeof (int*));
if ( p != NULL)
for (i=0; i<=n ;i++)
p[i] =(int*) malloc ((n+1)*sizeof (int));
return p;
}

//Function for reading the input file
int read(int &n, int &m, char *name)
{
int i;
FILE *f;
if((f=fopen(name,"r")) != NULL)
{
fscanf(f,"%d",&n);
fscanf(f,"%d",&m);
for(i=0;i<=m-1;i++)
fscanf(f,"%d %d %d",&mu[i].x,&mu[i].y,&mu[i].val);
return 1;
}
else
return 0;
}

// Function to create the connex components
void create_connex(int n)
{
int i,j;
j=1;
for(i=0; i<=n-1; i++)
{
C[i].c=1;
C[i].S[0]=j;
j++;
}
}

// Function to determine the connex components that contain the x vertix
int connex(int x,int ncc)
{
int j,i;
for(i=0;i<=ncc-1;i++)
{
for(j=0; j<=C[i].c-1; j++)
if(x==C[i].S[j])
return i;
}
```

```
return -1;
}

// Union of two connex components C[a]=C[a] U C[b]
void UNION(int a,int b,int &ncc)
{ int i;
for(i=0;i<=C[b].c-1;i++)
if(connex(C[b].S[i],ncc)!=a)
{ C[a].S[C[a].c]=C[b].S[i]; C[a].c++; }
for(i=b; i<=ncc-1; i++)
C[i]=C[i+1];
ncc--;
}

// Function to determin the edge with the minimum value
edge min_edge(int &m)
{
int i,min=mu[0].val,poz=0;
edge x=mu[0];
for(i=1; i<=m-1; i++)
if(mu[i].val < min)
{ min=mu[i].val; poz=i; x=mu[i]; }
for(i=poz; i<=m-2; i++)                    // delete edge with minimum value from the graph
mu[i]=mu[i+1];
m--;
return x;
}

// Function to implement the Kruskal's algorithm
void Kruskal(int n, int &m, int ncc, int **mst)
{
edge a;
create_connex(n);
int u,v,i,j,nm=0;
for(i=1;i<=n;i++)
for(j=1;j<=n;j++)
{
if(i!=j)
mst[i][j]=-1;
else
mst[i][j]=0;
}
while (nm < n-1)                           // while there weren't selected n-1 edges cat
{
a=min_edge(m);
u=a.x; v=a.y;
if(connex(u,ncc)!=connex(v,ncc))           // if u and v are in distinct connex components
{
UNION(connex(u,ncc),connex(v,ncc),ncc);
mst[u][v]=a.val;            // adding edge to MST
mst[v][u]=a.val;
nm++;
}
else
continue;
}
}

// Function to display MST
void display(int n,int **mst)
{
int i,j,valmin=0;
printf("\n\n The Minimum Spanning Tree has the edges \n\n");
for(i=1; i<=n; i++)
for(j=i+1; j<=n; j++)
{
if(mst[i][j] > 0)
{
valmin+=mst[i][j];
printf("\n Edge ( x%d , x%d ) with value %d \n", i,j,mst[i][j]);
}
}
```

```
printf("\n\n The minimum value for MST is %d \n",valmin);
}

// n the number of vertices of the graph
// m the current number of edges of the graph
// ncc the current number of connex components of the graph
// mst the spanning tree with minimum value
int main()
{
int n,m,ncc;
int **mst;
char name[30];
printf("\n\n Kruskal's algorithm \n\n");
getch();
printf(" File name ");
gets(name);
if(read(n,m,name))
{
printf("\n The file %s was read \n",name);
printf("\n The graph has %d vertices \n",n);
printf("\n The graph has %d edges \n\n",m);
getch();
mst=alocmat(n);
ncc=n;
Kruskal(n,m,ncc,mst);
display(n,mst);
free(mst);
}
else
printf(" Error reading file %s \n",name);
getch();
}
```

4. AN EXAMPLE

Let it be the finite graph, undirected and connex $G=\left(X,U,l\right)$, $X=\left\{x_1,x_2,x_3,x_4,x_5,x_6\right\}$ having edges with values. The edges and the values are represented in the Table 1.

The finite, undirected and connex graph having edges with values

Edge	Value	Edge	Value
$\left(x_1,x_2\right)$	6	$\left(x_3,x_4\right)$	5
$\left(x_1,x_3\right)$	1	$\left(x_3,x_5\right)$	6
$\left(x_1,x_4\right)$	5	$\left(x_3,x_6\right)$	4
$\left(x_2,x_3\right)$	5	$\left(x_4,x_6\right)$	2
$\left(x_2,x_5\right)$	3	$\left(x_5,x_6\right)$	6

Table 1

By applying the Krukal's algorithm we obtain the spanning tree having the minimum value 15. This can be observed in Table 2.

The MST

Edge	Value	Edge	Value
$\left(x_1,x_3\right)$	1		
		$\left(x_3,x_6\right)$	4
$\left(x_2,x_3\right)$	5	$\left(x_4,x_6\right)$	2
$\left(x_2,x_5\right)$	3		

Table 2

Running the above program we obtain the following results:

```
Kruskal's algorithm
File name graph.txt
The file graph.txt was read

The graph has 6 vertices
The graph has 10 edges
The Minimum Spanning Tree has the edges

Edge ( x1 , x3 ) with value 1
Edge ( x2 , x3 ) with value 5
Edge ( x2 , x5 ) with value 3
Edge ( x3 , x6 ) with value 4
Edge ( x4 , x6 ) with value 2

The minimum value for MST is 15
```

5. CONCLUSION

This implementation of the Kruskal's algorithm can be used for the future fast implementation of the Prim's algorithm to find a MST for a finite graph, undirected, connex having edges with values.

6. REFERENCE

1. Proceedings of the American Mathematical Society, pp. 48–50 in 1956
2. www.wikipedia.org/wiki/Kruskal's_algorithm