“Mircea cel Batran” Naval Academy Scientific Bulletin, Volume XVIII – 2015 – Issue 1
Published by “Mircea cel Batran” Naval Academy Press, Constanta, Romania // The journal is indexed in: PROQUEST SciTech Journals, PROQUEST Engineering Journals, PROQUEST Illustrata: Technology, PROQUEST Technology Journals, PROQUEST Military Collection PROQUEST Advanced Technologies & Aerospace

THE ALGORITHM FOR GENERATING AND COMPUTING THE SUBGRAPH DEFINED BY k GIVEN VERTICES

Paul VASILIU[1]
Tiberiu PAZARA[2]
[1]Lecturer PhD Eng. Department of Electrical Engineering and Electronics, „Mircea cel Batran” Naval Academy, p_vasiliu@yahoo.com
[2]Assistant PhD Department of Electrical Engineering and Electronics, „Mircea cel Batran” Naval Academy

Abstract: *This paper introduces the concept of value for a subgraph with k given vertices and weighted edges. In the following will be described how this subgraph be generated and how the graph’s value is computed. The paper presentsalso a C++ written program that implements the mentioned algorithm. Furthermore, we will present an example of how this program can be used and integrated.*
Keywords*: graph, subgraph, compute, algorithm, program*

INTRODUCTION

For graph theory an important concept is that of a subgraph of a finite graph, directed or undirected. If the number of vertices of a graph is too large, building the subgraph with k vertices can be done only through automation and by using a program. In this paper we will present how a subgraph with k vertices can be built from the vertices of a given graph. The paper includes a C++ program that has as input a graph, k vertices and generates the subgraph with the specified k vertices. Furthermore, the paper will present a scenario of usage for the program. In the final chapter, we will present some future work ideas for this research.

ALGORITHM

In this section we will define the concept of subgraph and we will present how it can be generated.

Let it be $X = \{x_1, x_2, \cdots, x_n\}$ the set of vertices, the application $\Gamma: X \to P(X)$ and the graph $G = (X, \Gamma)$. Let it be $A = (a_{ij})$ with $i = 1,2,\cdots,n$and $j = 1,2,\cdots,n$, $a_{ij} = 1$ if $\Gamma(x_i) = x_j$ and $a_{ij} = 0$ if $\Gamma(x_i) \neq x_j$ the adjacency matrix of the graph $G = (X, \Gamma)$.

Let us consider $X' = \{x_{i_1}, x_{i_2}, \cdots, x_{i_k}\} \subset \{x_1, x_2, \cdots, x_n\}$ a subset not null of vertices of the X set.

Let it be $\Gamma': X' \to P(X')$ the application defined by $\Gamma'\left(x_{i_p}\right) = \Gamma\left(x_{i_p}\right) \cap X'$, for every $x_{i_p} \in X'$. The given pair $G' = (X', \Gamma')$ is called the subgraph of a graph $= (X, \Gamma)$, being defined by the set of vertices X'. The adjacency matrix A', of the subgraph $G' = (X', \Gamma')$, can be obtained from the adjacency matrix A of graph $G = (X, \Gamma)$ keeping the lines and columns with indexes$i_1, i_2, \cdots, i_k$and canceling all lines and columns from Amatrix having indexes $i \notin \{i_1, i_2, \cdots, i_k\}$.

IMPLEMENTATION

The algorithm is presented below in a program implemented in C++.

The program receives as input the name of the file associated with the graph $G = (X, \Gamma)$, the number of vertices and the vertices $x_{i_1}, x_{i_2}, \cdots, x_{i_k}$ of the subgraph that will be generated. The program receives as arguments the name of the file and outputs the number of vertices, the number of edges, the adjacency matrix and the edges of graph $G = (X, \Gamma)$. For the subgraph $G' = (X', \Gamma')$, the adjacency matrix and the edges of the subgraph will be printed.

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#define dim 1000

typedefstruct arc
{
int x; int y; intval;
} edge;

edge mu[dim];   // mu - the array of edges

// Matrix space allocation
// Function for the allocation of the square matrix
// of dimension n+1
// The function returns the address of the matrix or NULL
// The matrix is initialized with 0s
int ** alocmat (int n)
{
inti;
int ** p=(int **) malloc ((n+1)*sizeof (int *));
if ( p != NULL)
for (i=0; i<=n ;i++)
p[i] =(int *) calloc ((n+1),sizeof (int));
return p;
}

// Array allocation with
// n integer elements
int *alocv(int n)
{
int *p;
 p=(int *)malloc(n*sizeof(int));
return p;
}

// Print array
voidprintv(intn,int *v)
{
inti;
printf(" { ");
for(i=0;i<n-1;i++)
printf(" %d , ",v[i]);
printf(" %d ",v[i]);
printf(" } \n");
}

// Function for printing the matrix n x n
voidprintm(int **a,intn,char *c)
{
inti,j;
printf("\n %s \n\n",c);
for(i=1;i<=n;i++)
    {
for(j=1;j<=n;j++)
printf("%2d ",a[i][j]);
printf("\n");
    }
    }

// Function for printing the matrix n x n
voidprintmm(int **a,intn,char *c,intk,int *v)
{
inti,j;
printf("\n %s ",c);
printv(k,v);
for(i=1;i<=n;i++)
    {
for(j=1;j<=n;j++)
printf("%2d ",a[i][j]);
printf("\n");
    }
    }

// Read the array
voidreadv(intn,int *v)
{
inti;
for(i=0;i<n;i++)
 {
printf(" Vertice %d = ",i+1);
scanf("%d",&v[i]);
}
}

// Search value in the array
int search(intk,intn,int *v)
{
inti,found;
found=0;
for(i=0;i<n;i++)
if(v[i]==k)
 {
found=1;
break;
  }
return found;
}

// Read file
intread_n_m(int&n, int&m, char *name)
{
    FILE *f;
if((f=fopen(name,"r"))!= NULL)
    {
fscanf(f,"%d %d",&n,&m);
fclose(f);
return 1;
    }
else
return 0;
}

// Function for printing all edges of the graph
voidprint_edges(int m)
{
inti;
printf("\n Graph G has %d edges \n\n",m);
for(i=1;i<=m;i++)
printf(" Edge %d:\t ( x%d , x%d ) \n",i,mu[i].x,mu[i].y);
printf("\n");
}

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

// Generating the matrix of the subgraph
void gen(intn,int **a,intk,int *v,int **ap)
{
inti,j;
for(i=1;i<=n;i++)
if(search(i,k,v)==1)
for(j=1;j<=n;j++)
if(search(j,k,v)==1)
ap[i][j]=a[i][j];
}
```

```
// Function for printing the edges of the subgraph
voidprint_edges_s(intn,int **ap)
{
inti,j,m;
for(m=1,i=1;i<=n;i++)
for(j=1;j<=n;j++)
if(ap[i][j]==1)
{
mu[m].x=i;
mu[m].y=j;
m++;
}
m--;
printf("\n The subgraph has %d edges \n\n",m);
for(i=1;i<=m;i++)
printf(" Edge %d:\t ( x%d , x%d ) \n",i,mu[i].x,mu[i].y);
printf("\n");
}

// Main function
int main()
{
intn,m,k,**a,**ap,*v;
char name[30];
printf("\n File name ");
fflush(stdin);
gets(name);
if(read_n_m(n,m,name))
 {
printf("\n Number of vertices %d \n",n);
printf(" Number of edges %d \n",m);
getch();
 a=alocmat(n);
if(read_graph(name,a))
 {
printm(a,n,"Adjacency matrix of the graph \n");
print_edges(m);
printf(" Number of vertices from subgraph k = ");
scanf("%d",&k);
 v=alocv(k);
ap=alocmat(n);
readv(k,v);
gen(n,a,k,v,ap);
printmm(ap,n,"Adjacency matrix of the subgraph ",k,v);
print_edges_s(n,ap);
getch();
}}}
```

EXAMPLE

Let us consider the graph $G = (X, \Gamma)$directed definiedby the set $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$and the application $\Gamma: X \to P(X)$definied through the following equalities: $\Gamma(x_1) = \{x_1, x_3, x_5\}$, $\Gamma(x_2) = \{x_1, x_4, x_5, x_6\}$, $\Gamma(x_3) = \{x_2, x_4, x_5, x_6\}$, $\Gamma(x_4) = \{x_2, x_3, x_6\}$, $\Gamma(x_5) = \{x_1, x_3, x_4, x_6\}$, $\Gamma(x_6) = \{x_1, x_2, x_3, x_4, x_6\}$. The adjacency matrix of the graph $G = (X, \Gamma)$ is:

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Let us consider the subgraph $G' = (X', \Gamma')$ defined by the subset $X' = \{x_2, x_4, x_5\}$. .. We will compute the adjacency matrix of the subgraph $G' = (X', \Gamma')$. We compute the following:

$\Gamma'(x_2) = \Gamma(x_2) \cap X' = \{x_1, x_4, x_5, x_6\} \cap \{x_2, x_4, x_5\}$=$\{x_4, x_5\}$. $\Gamma'(x_4) = \Gamma(x_4) \cap X' = \{x_2, x_3, x_6\} \cap \{x_2, x_4, x_5\} = \{x_2\}$.
$\Gamma'(x_5) = \Gamma(x_5) \cap X' = \{x_1, x_3, x_4, x_6\} \cap \{x_2, x_4, x_5\}$ =$\{x_4\}$.

The adjacency matrix A' of the subgraph $G' = (X', \Gamma')$ is obtained from the adjacency matrix A of graph $G = (X, \Gamma)$ by canceling all elements from lines and columns 1, 4, 5. We get the matrix :

$$A' = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The program receives as input a text file with the name graph.txt. The file has on the first line the number n of vertices and the number m of edges separated with spaces. The next m lines will contain the edges of the graph. For the graph presented in the example, the input file graph.txt is displayed below:

```
6 23
1 1
1 3
1 5
2 1
2 4
2 5
2 6
3 2
3 4
3 5
3 6
4 2
4 3
4 6
5 1
5 3
5 4
5 6
6 1
6 2
6 3
6 4
6 6
```

An example of running this program is:

```
File name graph.txt
Number of vertices 6
Number of edges 23

Adjacency matrix of the graph
1 0 1 0 1 0
1 0 0 1 1 1
0 1 0 1 1 1
0 1 1 0 0 1
1 0 1 1 0 1
1 1 1 1 0 1

Graph G has 23 edges

Edge 1:     ( x1 , x1 )
Edge 2:     ( x1 , x3 )
Edge 3:     ( x1 , x5 )
Edge 4:     ( x2 , x1 )
Edge 5:     ( x2 , x4 )
Edge 6:     ( x2 , x5 )
Edge 7:     ( x2 , x6 )
Edge 8:     ( x3 , x2 )
Edge 9:     ( x3 , x4 )
Edge 10:    ( x3 , x5 )
Edge 11:    ( x3 , x6 )
Edge 12:    ( x4 , x2 )
Edge 13:    ( x4 , x3 )
Edge 14:    ( x4 , x6 )
Edge 15:    ( x5 , x1 )
Edge 16:    ( x5 , x3 )
Edge 17:    ( x5 , x4 )
Edge 18:    ( x5 , x6 )
Edge 19:    ( x6 , x1 )
Edge 20:    ( x6 , x2 )
Edge 21:    ( x6 , x3 )
Edge 22:    ( x6 , x4 )
Edge 23:    ( x6 , x6 )

Number of vertices from subgraph k = 3

Vertix 1 = 2
Vertix 2 = 4
Vertix 3 = 5

Adjacency matrix of the subgraph  { 2 , 4 , 5 }

0 0 0 0 0 0
0 0 0 1 1 0
0 0 0 0 0 0
0 1 0 0 0 0
0 0 0 1 0 0
0 0 0 0 0 0

The subgraph has 4 edges

Edge 1:     ( x2 , x4 )
Edge 2:     ( x2 , x5 )
Edge 3:     ( x4 , x2 )
Edge 4:     ( x5 , x4 )
```

This example of execution of the program underlines the correctness of the theoretical and practical results obtained.

Conclusions

In this paper we presented an algorithm and a program written in C++ language for generating the subgraph composed of the k vertices of a given graph.

Among the multiple future direction for this research we will present some of them:

- finding all the subgraph with k given vertices of a graph
- finding all the subgraphs with k given vertices and their values
- finding the subgraphs with minimum value
- finding the subgraphs with maximum value

Bibliography

[1] Blăjină A., Produse software aplicate în programarea matematică şi teoria jocurilor, Ed. Albastra, Cluj-Napoca 2006
[2] Gorunescu F., Prodan A., Modelare stochastică şi simulare, Ed. Albastra, Cluj-Napoca 2001
[3] Marin Ghe., Popoviciu I., Vasiliu P., Lupei T., Chiru C., Cercetări operaţionale – Programare liniară, Ed. ANMB, Constanţa 2002
[4] Marin Ghe., Popoviciu I., Vasiliu P., Caţă M., Cercetări operaţionale. Grafuri. Probleme de coordonare. Teoria aşteptării. Programare dinamică. Teoria deciziei. Ed. ANMB, Constanţa 2003
[5] Vasiliu P., Modelare şi simulare, Ed. ANMB, Constanţa 2013
[6] Văduva I., Modele de simulare cu calculatorul, Editura Tehnică, Bucureşti, 1977