

Generation of a Random Simple Graph And Its Graphical Presentation

Krishnendu Basuli*

krishnendu.basuli@gmail.com

Saptershi Naskar*

sapgrin@gmail.com

*Department of Computer Science and Engineering,
University of Calcutta, 92, A. P. C. Road,
Kolkata – 700 009, India

Most of the physical and mathematical problems can be formulated in terms of Graph Theory [1]. Generation of a single spanning tree for a simple, symmetric and connected graph G is well known polynomial time solvable problem [1]. Also there are some intractable problems like *Graph Coloring*, *Vertex Connectivity*, *Isomorphism* etc. in graph theory [2,3]. To solve these problems we need some Soft computing approaches like GA, SA, Fuzzy Set, Rough Set etc. [4,6].

Graphs can be represented in different way, like *Adjacency Matrix*, *Adjacency list*, *Incidence Matrix*, etc. [1,5]. While developing algorithms for solution of some problems related to graph, need to represent the graph in any of the above ways.

In this paper there is a program that actually generates an adjacency matrix of a randomly generated graph and also plotted the graph in the screen to visualize it. We have implemented an existing algorithm for generation of a Random Graph in C Language (Borland C++ compiler).

Analysis of the written functions:

void RandGraph(void);

This function actually generates the *Random Graph of given number of vertices*.

void Disp(void);

This function displays the *Adjacency Matrix* of the generated Graph.

void PlotGraph(void);

This function plots the Graph in the screen.

Source Code:

```
#include<stdio.h>
#include<conio.h>
#include<time.h>
#include<stdlib.h>
#include<graphics.h>
#include<alloc.h>

int gph[20][20];
int nd;

struct cod
{
    int x,y;
};

void RandGraph(void);
void Disp(void);
void PlotGraph(void);
```

```
void main()
{
int gd=0,gm=0;
initgraph(&gm,&gd,"e:\\TC\\bgi");
RandGraph();
Disp();
PlotGraph();
getch();
}

void RandGraph()
{
int i,j,k;
time_t t;
srand((unsigned) time(&t));
printf("\nEnter number of nodes :");
scanf("%d",&nd);
for(i=0;i<=nd-1;i++)
{
    for(j=i;j<=nd-1;j++)
    {
        if(i!=j)
        {
            k=rand()%100;

            if(k>=55)
            {
                gph[i][j]=1;
                gph[j][i]=1;
            }
            else
            {
                gph[i][j]=0;
                gph[j][i]=0;
            }
        }
    }
}
}

void Disp()
{
int i,j;
printf("\nThe Random Graph is :\n\n");

for(i=0;i<=nd-1;i++)
{
    for(j=0;j<=nd-1;j++)
        printf("%d ",gph[i][j]);
    printf("\n");
}
}

void PlotGraph()
{
```

```

int k=0,i,j,rmx,rmy,p=410,q=100,rw,b=10,a=30;
struct cod *xy,*pq;
char ch[4],h=97;
xy=(struct cod *)malloc(sizeof(struct cod)*nd);
pq=xy;
if(nd%3 == 0)
    rw=nd/3-1;
else
    rw=nd/3;
for(i=0;i<=rw;i++)
{
    for(j=0;j<=2;j++)
    {
        xy->x=p;
        xy->y=q+b;
        xy++;
        k++;
        b=-b;
        p=p+50;
    }
    q=q+50;
    p=410+a;
    a=-a;
}
xy=pq;
setcolor(GREEN);
k=0;
for(i=0;i<=rw;i++)
{
    if(i!=rw)
    {
        for(j=0;j<=2;j++)
        {
            circle(xy->x,xy->y,5);
            ch[0]=h;
            ch[1]='\0';
            moveto(xy->x+8,xy->y-8);
            outtext(ch);
            rmx=xy->x;
            rmy=xy->y;
            xy++;
            h++;
        }
    }
    else
    {
        if(nd %3 !=0)
        {
            for(j=0;j<=nd%3;j++)
            {
                circle(xy->x,xy->y,5);
                ch[0]=h;
                ch[1]='\0';
                moveto(xy->x+8,xy->y-8);
                outtext(ch);
            }
        }
    }
}

```

```

    rmx=xy->x;
    rmy=xy->y;
    xy++;
    h++;
    }
    }
    else
    {
    for(j=0;j<=2;j++)
    {
    circle(xy->x,xy->y,5);
    ch[0]=h;
    ch[1]='\0';
    moveto(xy->x+8,xy->y-8);
    outtext(ch);
    rmx=xy->x;
    rmy=xy->y;
    xy++;
    h++;
    }
    }
    }
}

if(nd%3!=0)
{
    setcolor(BLACK);
    circle(rmx,rmy,5);
    moveto(rmx+8,rmy-8);
    outtext(ch);
}

xy=pq;
setcolor(RED);
for(i=0;i<=nd-1;i++)
{
    for(j=i;j<=nd-1;j++)
    {
        if(gph[i][j]==1)
        {
            line((xy+i)->x,(xy+i)->y,(xy+j)->x,(xy+j)->y);
        }
    }
}
}
}

```

Reference:

- [1] Combinatorial Algorithms: Theory and Practice, Reingold, Nievergelt and Deo, PHI, 1977.
- [2] Computer and Intractability – A guide to the Theory of NP Completeness, Garry and Johnson, W.H. Freeman and Company, 1999.
- [3] Computer Architecture and Organization, Hayes J. P., McGraw Hill, 2nd Edition.
- [4] Data mining: Multimedia, Soft Computing and Bio-informatics, Susmita Mitra and Tinku Acharjye, John Willey.
- [5] Fundamental of Computer Algorithms, Horowitz, Sahani and Rajasekharan, Galgotia Publication Pvt. Ltd., 2000.
- [6] Modern Heuristic Technique for Combinatorial Problems, Reeves C.R., John Wiley & Sons, Inc., 1993.