DHARAMSINH DESAI UNIVERSITY , NADIAD.

Faculty of management and information Technology

DATABASE MANAGEMENT SYSTEM

TERMWORK SUBMISSION

SUBMITTED BY : Gandhi Zeel Kamleshkumar

SEMESTER : MCA-II

ROLL NO.: MA002 (18MAPOG019)

SUBMITTED TO:

Prof. HIMANSHU PUROHIT

Term_Work: Social Graphs

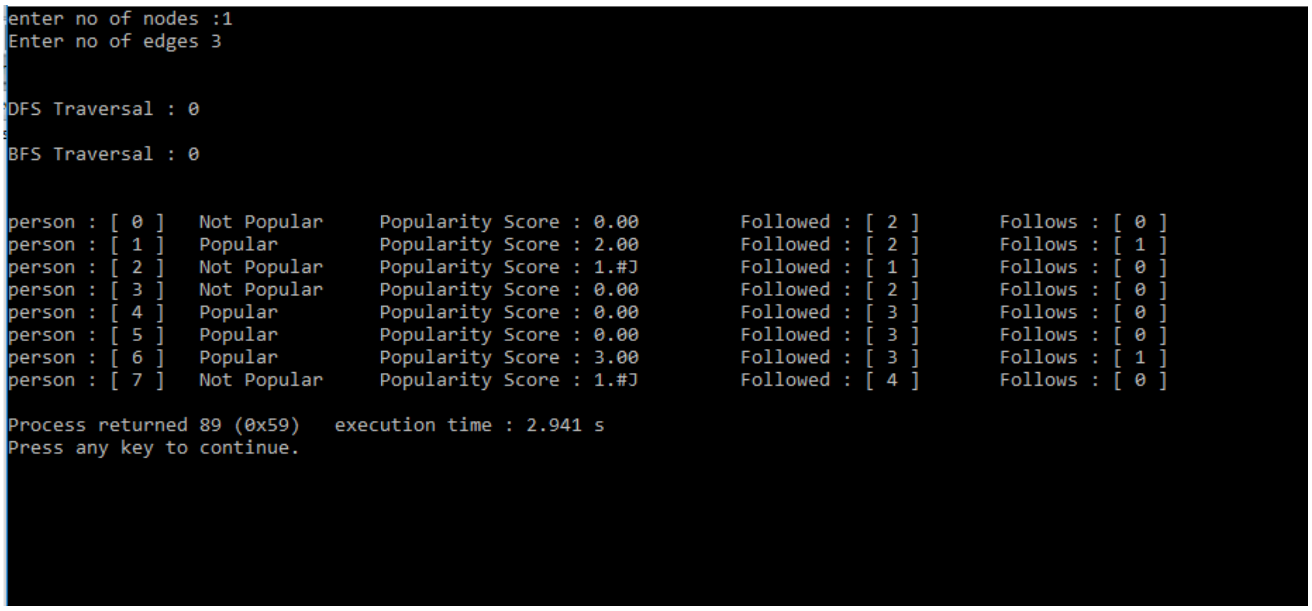
For this program, you will read in two different types of social graphs and

compute some statistics using them.
Social followers
If the graph to be processed is a directed graph, then it represents a small
part of a followers graph. An edge from node *u* to node *v* means that user *u*
"follows" user *v* or alternatively, that user *v* is “followed by” user *u*. For
followers graphs, your goal is to determine which users are popular. A
person (user) P is considered popular if P's popularity ratio (computed as
the number of users who follow P divided by the number of users P
follows) is at least 2. There is a special case. If user P does not follow any
other users, the popularity ratio would be undefined. So, in that case, P is
considered popular if he or she is followed by at least 3 people.

```
#include<stdio.h>
#define max 200
#define initial 1
#define waiting 2
#define visit 3

int state[max];
int queue[max], front = -1,rear = -1;
void insert_queue(int vertex);
int delete_queue();
int isEmpty_queue();

int adj[max][max];
int visited[max];
int n,e;

void add_edge(int u ,int v)
{
   adj[u][v]=1;
}
void read_file()
{
   FILE *in_file;
   int n,m;

   in_file = fopen("tgraph.txt","r");

   if(in_file == NULL)
   {
      printf("error\n");
   }
   while(fscanf(in_file,"%d %d",&n,&m) != EOF)
   {
     add_edge(n,m);
   }
   fclose(in_file);
}

void Display()
{
   float pos;
   int e,i,j,origin,dest;
   for(i=0;i<=7;i++)
   {
      int indeg=0,outdeg=0;
      for(j=0;j<=7;j++)
      {
         if(adj[j][i] == 1)
         {
            indeg++;
         }
      }
      for(j=0;j<n;j++)
      {
         if(adj[i][j] == 1)
```

```
            {
               outdeg++;
            }
        }

        pos=(float)indeg/outdeg;
        if(pos == 2.0)
        {
            printf("person : [ %d ]\t Popular\t Popularity Score : %.2f\t
Followed : [ %d ] \tFollows : [ %d ]\t\n",i,pos,indeg,outdeg);
        }
        else if(pos == 3.0)
            printf("person : [ %d ]\t Popular\t Popularity Score : %.2f\t
Followed : [ %d ] \tFollows : [ %d ]\t\n",i,pos,indeg,outdeg);
        else if(indeg == 2 && outdeg == 0)
        {
            pos=0;
            printf("person : [ %d ]\t Not Popular\t Popularity Score :
%.2f\t Followed : [ %d ] \tFollows : [ %d ]\t\n",i,pos,indeg,outdeg);
        }
        else if(indeg == 3 && outdeg == 0)
        {
            pos=0;
            printf("person : [ %d ]\t Popular\t Popularity Score : %.2f\t
Followed : [ %d ] \tFollows : [ %d ]\t\n",i,pos,indeg,outdeg);
        }
        else
            printf("person : [ %d ]\t Not Popular\t Popularity Score :
%.2f\t Followed : [ %d ] \tFollows : [ %d ]\t\n",i,pos,indeg,outdeg);
    }
}
void DFS(int i)
{
    int j;
    printf("%d ",i);
    visited[i]=1;
    for(j=0;j<n;j++)
    {
       if(!visited[j] && adj[i][j]==1)
       {
         DFS(j);
       }
    }
}
void BFS(int v)
{
        int i;
        insert_queue(v);
        state[v] = waiting;
        while(!isEmpty_queue())
        {
                v = delete_queue( );
                printf("%d ",v);
                state[v] = visit;

                for(i=0; i<n; i++)
                {
                        if(state[i] == 0 && adj[v][i] !=0)
                        {
                                insert_queue(i);
                                state[i] = visit;
                        }
                }
        }
        printf("\n");
}

void insert_queue(int vertex)
{
        if(rear == max-1)
                printf("Queue Overflow\n");
        else
        {
                if(front == -1)
                front = 0;
            rear = rear+1;
            queue[rear] = vertex ;
        }
}

int isEmpty_queue()
{
        if(front == -1 || front > rear)
                return 1;
        else
                return 0;
}

int delete_queue()
{
        int delete_item;
        if(front == -1 || front > rear)
        {
                printf("Queue Underflow\n");
                exit(1);
        }

        delete_item = queue[front];
        front = front+1;
        return delete_item;
}

void main()
{
   read_file();

   int e;
   printf("enter no of nodes :");
   scanf("%d",&n);
   printf("Enter no of edges ");
   scanf("%d",&e);
```

```
    printf("\n\n");

    printf("DFS Traversal : ");
    DFS(0);
    printf("\n\n");
    printf("BFS Traversal : ");
    BFS(0);
    printf("\n\n");
    Display();
}
```



```
enter no of nodes :1
Enter no of edges 3


DFS Traversal : 0

BFS Traversal : 0


person : [ 0 ]   Not Popular     Popularity Score : 0.00        Followed : [ 2 ]       Follows : [ 0 ]
person : [ 1 ]   Popular         Popularity Score : 2.00        Followed : [ 2 ]       Follows : [ 1 ]
person : [ 2 ]   Not Popular     Popularity Score : 1.#J        Followed : [ 1 ]       Follows : [ 0 ]
person : [ 3 ]   Not Popular     Popularity Score : 0.00        Followed : [ 2 ]       Follows : [ 0 ]
person : [ 4 ]   Popular         Popularity Score : 0.00        Followed : [ 3 ]       Follows : [ 0 ]
person : [ 5 ]   Popular         Popularity Score : 0.00        Followed : [ 3 ]       Follows : [ 0 ]
person : [ 6 ]   Popular         Popularity Score : 3.00        Followed : [ 3 ]       Follows : [ 1 ]
person : [ 7 ]   Not Popular     Popularity Score : 1.#J        Followed : [ 4 ]       Follows : [ 0 ]

Process returned 89 (0x59)   execution time : 2.941 s
Press any key to continue.
```

Social friends

If the graph to be processed is an undirected graph, then it represents
a
portion of a social friendship graph. Social platform users agree to be
mutual friends, so an undirected graph is appropriate. In this case
though,
the graph is weighted. The weight on an edge (u, v) represents the
number
of days u and v have been friends (on Facebook). For this type of
graph,
you will need to compute several statistics for each user/node P:

• The number of friends of P
• The number of friends of friends of P
• The longest friendship P has with a friend on social platform

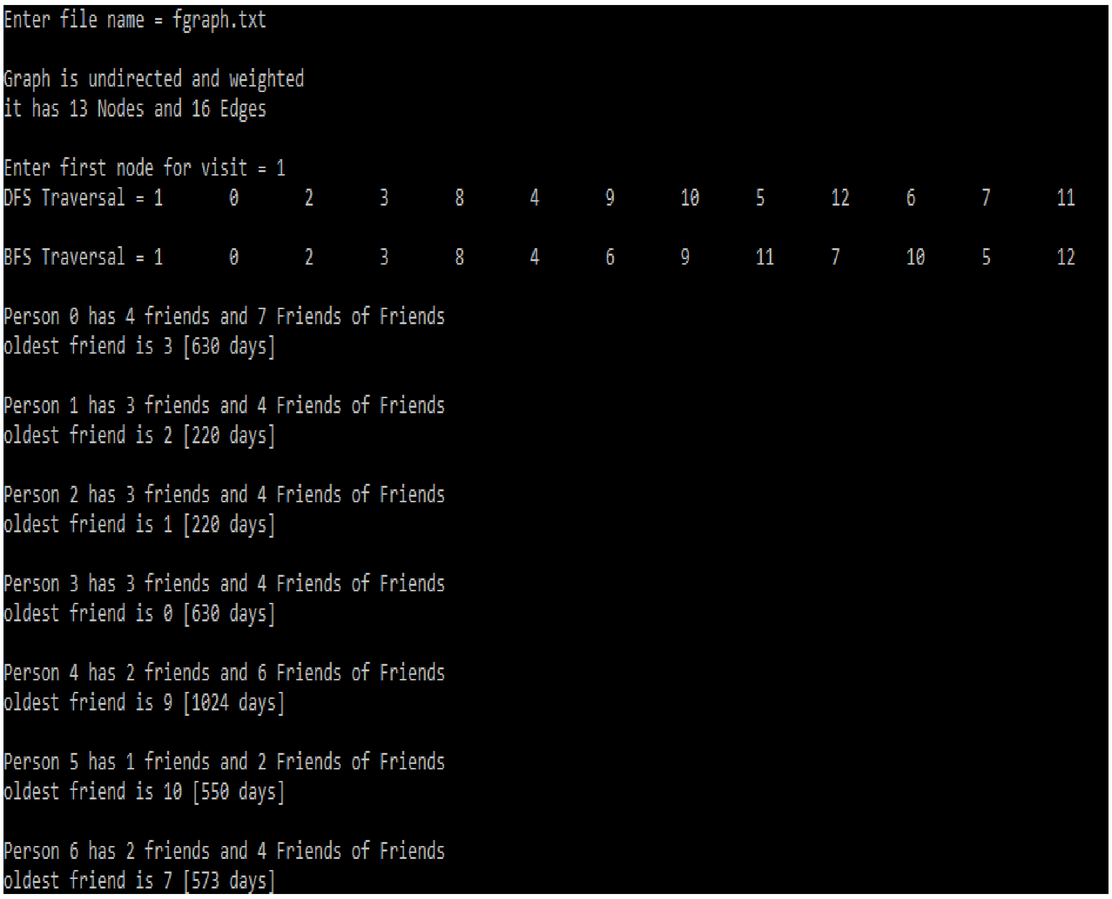
For the friend of friend (FoF) computation, don't count any FoF more
than
once. Do NOT count P as a friend or FoF of himself. Overall Process
Once
you read in and instantiate a graph, you must do graph traversals
(DFS and
BFS) to verify that the graph is constructed correctly.

```
#include<stdio.h>
#include<string.h>
#define N 50
int adj[N][N],visit[N],q[N],fl[N];
int f=0,r=-1;
int n,e;
void dfs(int i)
{
	int j;
	printf("%d\t",i);
	visit[i]=1;
	for(j=0;j<n;j++)
	{
		if(visit[j]==0&&adj[i][j]!=0)
			dfs(j);
	}
}
void bfs(int i)
{
	int j;
	printf("%d\t",i);
	visit[i]=1;
	for(j=0;j<n;j++)
	{
		if(visit[j]==0 && adj[i][j]!=0)
		{
			q[++r]=j;
			visit[j]=1;
		}
	}
	if(f>r)
		return;
	bfs(q[f++]);

}
int find(int c,int count,int tmp[])
{
	int i,j,cnt=0,t;
	for(i=0;i<count;i++)
	{
		for(j=0;j<n;j++)
		{
			if(adj[tmp[i]][j]!=0)
				fl[j]=1;
		}
	}
	fl[c]=0;
	for(i=0;i<n;i++)
	{
		if(fl[i]==1)
			cnt++;
	}
	return cnt;
}
void main()
{
	int i,j,s,d,w,t;
	int a,in[N],out[N],old[N],fr[N],fof[N],tmp[N];
	char file[20];
	float pop;
	FILE *fp;
	printf("Enter file name = ");
	scanf("%s",file);
	fp=fopen(file,"r");
	if(fp==NULL)
	{
		printf("Please enter valid name = ");
		return;
	}
	fscanf(fp,"%d%d",&n,&e);
	if(n==1 && e==0)
	{
		fscanf(fp,"%d",&t);
		printf("\nGraph is directed and unweighted\n");
		fscanf(fp,"%d%d",&n,&e);
		printf("it has %d Nodes and %d Edges\n",n,e);
		for(i=0;i<e;i++)
		{
			fscanf(fp,"%d%d",&s,&d);
			adj[s][d]=1;
		}
		for(i=0;i<n;i++)
```

```
			visit[i]=0;
		printf("\nEnter first node for visit = ");
		scanf("%d",&a);
		printf("DFS Traversal =\t");
		dfs(a);
		printf("\nBFS Traversal =\t");
		for(i=0;i<n;i++)
			visit[i]=0;
		bfs(a);
		for(i=0;i<=n;i++)
		{
			in[i]=out[i]=0;
			for(j=0;j<=n;j++)
			{
				if(adj[i][j]==1)
					out[i]++;
				if(adj[j][i]==1)
					in[i]++;
			}
		}
		for(i=0;i<n;i++)
		{
			if(out[i]==0)
			{
				if(in[i]>=t)
					printf("\n%d is popular\nfollows =
%d\tFollowers = %d\n",i,out[i],in[i]);
				else
					printf("\nPerson %d is not
popular\nfollows = %d\tFollowed by = %d\n",i,out[i],in[i]);
			}
			else
			{
				pop=(float)in[i]/(float)out[i];
				if(pop>=2)
					printf("\n%d is popular, Popularity score
= %.2f\nfollows = %d\tFollowed by = %d\n",i,pop,out[i],in[i]);
```

```
                    else
                              printf("\n%d is not popular, Popularity
score = %.2f\nfollows = %d\tFollowed by = %d\n",i,pop,out[i],in[i]);
                }
          }
          printf("\n");
     }
     if(n==0 && e==1)
     {
          printf("\nGraph is undirected and weighted\n");
          fscanf(fp,"%d%d",&n,&e);
          printf("it has %d Nodes and %d Edges\n",n,e);
          for(i=0;i<e;i++)
          {
               fscanf(fp,"%d%d%d",&s,&d,&w);
               adj[s][d]=w;
               adj[d][s]=w;
          }
          for(i=0;i<n;i++)
               visit[i]=0;
          printf("\nEnter first node for visit = ");
          scanf("%d",&a);
          printf("DFS Traversal =\t");
          dfs(a);
          printf("\nBFS Traversal =\t");
          for(i=0;i<n;i++)
               visit[i]=0;
          bfs(a);
          for(i=0;i<n;i++)
          {
               out[i]=0;old[i]=0;
               fof[i]=0;
               for(j=0;j<n;j++)
               {
                    fl[j]=0;
                    fl[j]=0;visit[j]=0;
                    if(adj[i][j]!=0)
```

```
                        {
                                tmp[out[i]]=j;
                                out[i]++;
                        }
                        if(adj[j][i]>old[i])
                        {
                                old[i]=adj[j][i];
                                fr[i]=j;
                        }
                }
                fof[i]=find(i,out[i],tmp);
        }
        for(i=0;i<n;i++)
                printf("\nPerson %d has %d friends and %d Friends
of Friends\noldest friend is %d [%d
days]\n",i,out[i],fof[i],fr[i],old[i]);
    }
    printf("\n");
}
```



```
Enter file name = fgraph.txt

Graph is undirected and weighted
it has 13 Nodes and 16 Edges

Enter first node for visit = 1
DFS Traversal = 1       0       2       3       8       4       9       10      5       12      6       7       11

BFS Traversal = 1       0       2       3       8       4       6       9       11      7       10      5       12

Person 0 has 4 friends and 7 Friends of Friends
oldest friend is 3 [630 days]

Person 1 has 3 friends and 4 Friends of Friends
oldest friend is 2 [220 days]

Person 2 has 3 friends and 4 Friends of Friends
oldest friend is 1 [220 days]

Person 3 has 3 friends and 4 Friends of Friends
oldest friend is 0 [630 days]

Person 4 has 2 friends and 6 Friends of Friends
oldest friend is 9 [1024 days]

Person 5 has 1 friends and 2 Friends of Friends
oldest friend is 10 [550 days]

Person 6 has 2 friends and 4 Friends of Friends
oldest friend is 7 [573 days]
```