Term_Work: Social Graphs

1. 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.

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<conio.h>
#define initial 1
#define waiting 2
#define visited1 3
#define MAX 30
void socialfollowers();
void DFS(int);
void BFS(int);
void socialfriends();
int queue[MAX], front = -1,rear = -1;
void insert_queue(int);
int delete_queue();
int isEmpty_queue();
int g[MAX][MAX]={0},n;
int state[MAX];
int visited[MAX]={0};
int main()
{
	int ch,i;
	printf("\n 1) social followers...");
	printf("\n 2) social friends...");
	do
	{
		printf("\n enter your choice=");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1:
			socialfollowers();
			printf("\nDFS traversal...");
			DFS(1);
			printf("\nBFS traversal...");
			BFS(1);
			break;
```

```
                case 2:
                socialfriends();
                printf("\nDFS traversal...");
                DFS(1);
                printf("\nBFS traversal...");
                BFS(1);
                break;
                default:
                printf("\n enter valid choice");
                break;
        }
    }while(ch==0);
}
void socialfollowers()
{
    int i,j,count1,count2[30],count,a[30],b[30];
    float ratio[30];
    printf("\n enter number of vertex=");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
                visited[i]=0;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            printf("\n enter g[%d][%d]=",i,j);
            scanf("%d",&g[i][j]);
        }
    }
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            printf("%d \t",g[i][j]);
        }
```

```
        printf("\n");
}
for(i=1;i<=n;i++)
{
        count1=0;
        a[i]=0;
        for(j=1;j<=n;j++)
        {
                if(g[i][j]==1)
                {
                        count1++;
                        a[i]++;
                        printf("\n[%d] follows [%d]",i,j);

                }
        }
}
for(i=1;i<=n;i++)
{
        count=0;
        b[i]=0;
        for(j=1;j<=n;j++)
        {
                if(g[j][i]==1)
                {
                        count++;
                        b[i]++;
                        printf("\n [%d] followed by [%d]",i,j);

                }

        }
}

for(i=1;i<=n;i++)
```

```
        {
                ratio[i]=(float)b[i]/(float)a[i];
                // printf("\n popularity ratio of %d=%f",i,ratio[i]);
                 if(ratio[i]>=2)
                 {
                        printf("\n %d is popular =%f",i,ratio[i]);
                 }
                 else if(ratio[i]==0)
                 {
                        ratio[i]=0;
                        printf("\n %d is not popular =%f",i,ratio[i]);

                 }
                 else
                 {
                        printf("\n %d is not popular =%f",i,ratio[i]);
                 }
        }
}
void DFS(int i)
{
        int j;
    printf("\t%d",i);
        visited[i]=1;
        for(j=1;j<=n;j++)
        if(!visited[j]&&g[i][j]!=0)
                DFS(j);
}

void BFS(int v)
{
    int i;
    insert_queue(v);
    state[v] = 1;
    printf("%d ",v);
```

```
    while(!isEmpty_queue())
    {
        v = delete_queue( );

        for(i=1; i<=n; i++)
        {
            if(g[v][i] != 0 && state[i] == 0)
            {
                insert_queue(i);
                state[i] = waiting;
                    printf("%d ",i);
            }
        }
    }
    printf("\n");
}
int isEmpty_queue()
{
    if(front == -1 || front > rear)
        return 1;
    else
        return 0;
}
void insert_queue(int n)
{
    if(rear == MAX-1)
        printf("Queue Overflow\n");
    else
    {
        if(front == -1)
            front = 0;
        rear = rear+1;
        queue[rear] = n;
    }
}
```

```
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;
}
int FOF(int s)
{
    int friend[n],c=0,p=0,i,j,visit[50]={0};
    for(i=1;i<=n;i++)
    {
        visit[i]=0;
    }
    visit[s]=1;
    for(i=1;i<=n;i++)
    {
            if(g[s][i]!=0)
            {
                friend[p]=i;
                p++;
            }
    }
    for(i=1;i<p;i++)
    {
        for(j=1;j<n;j++)
        {
            if(g[friend[i]][j] != 0 && visit[j]==0)
            {
                visit[j]=1;
```

```
                c++;
            }
        }
    }
    return c;
}
void socialfriends()
{
    int max=0,count,count1,i,j,count2[20];
    printf("\n enter number of vertex=");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            printf("\n enter g[%d][%d]=",i,j);
            scanf("%d",&g[i][j]);
        }
    }
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            printf("%d \t",g[i][j]);
        }
        printf("\n");
    }
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=n;j++)
        {
            if(g[i][j]==1)
            {
                printf("\n g[%d]->g[%d]",i,j);
            }
```

```
        }
}
for(i=1;i<=n;i++)
{
        count=0,count1=0;
        for(j=1;j<=n;j++)
        {
                if(g[j][i]==1)
                {
                        count++;
                }
        }
        for(j=1;j<=n;j++)
        {
                if(g[i][j]==1)
                {
                        count1++;
                }
        }
}

for(i=1;i<=n;i++)
{
        count1=0;
        count2[i]=0;
        for(j=1;j<=n;j++)
        {
                if(g[i][j]!=0)
                {
                  if(max<g[i][j])
                       max=g[i][j];
                        count1++;
                        count2[i]++;
                }
```

```
            }
        printf("\n friend of %d has %d friends    FOF= %d Oldest
Friend=%d",i,count2[i],FOF(i),max);

    }
}
```