Laboratory Note

Paper: Data structure Laboratory

Subject: MCA

Semester: II

Department of Computer Applications

Sikkim University

[April, 2020 - August, 2021]

Name of Student: Ram Babu Ray

Roll No: 20MCA013

Student's Signature:

Course Faculty: Dr. SWARUP ROY

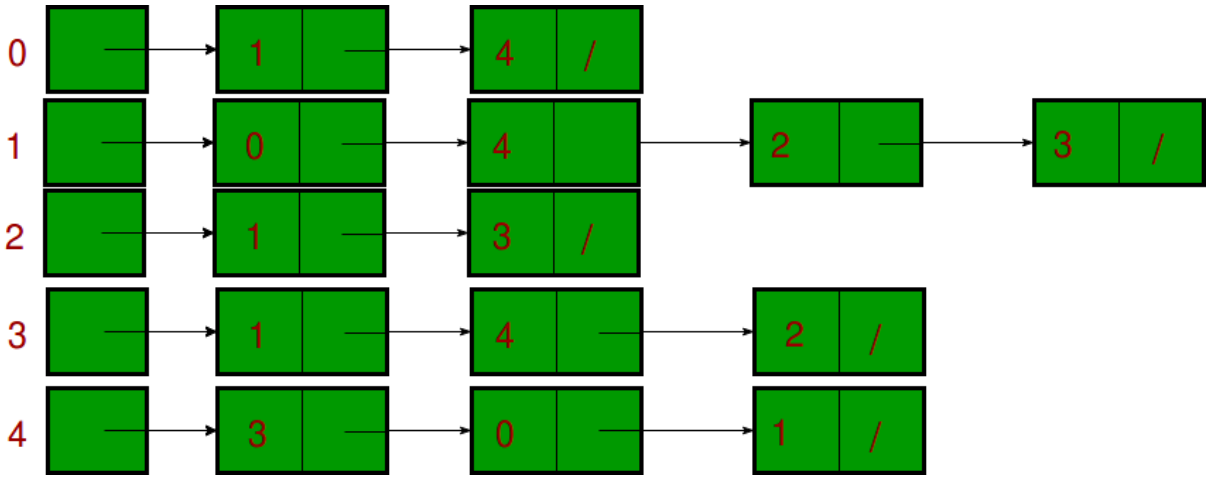
Designation: Head of the Department of Computer Applications, Sikkim University

AIM: Given a list of friend relationship among a pair of friends. Find out the person who is having highest popularity among them. A person with highest link with others is the most popular. Example: A – B, B – C, C- F, C-E, D-C, D-B. Here, C is related with 4 and hence the most popular. Use adjacency list for your implementation.

Introduction:

Adjacency list:

An array of lists is used. The size of the array is equal to the number of vertices. Let the array be an array[]. An entry array[i] represents the list of vertices adjacent to the ***i***th vertex. This representation can also be used to represent a weighted graph. The weights of edges can be represented as lists of pairs. Following is the adjacency list representation of the above graph.



Find the Point on a Circle Given an Angle and the Radius

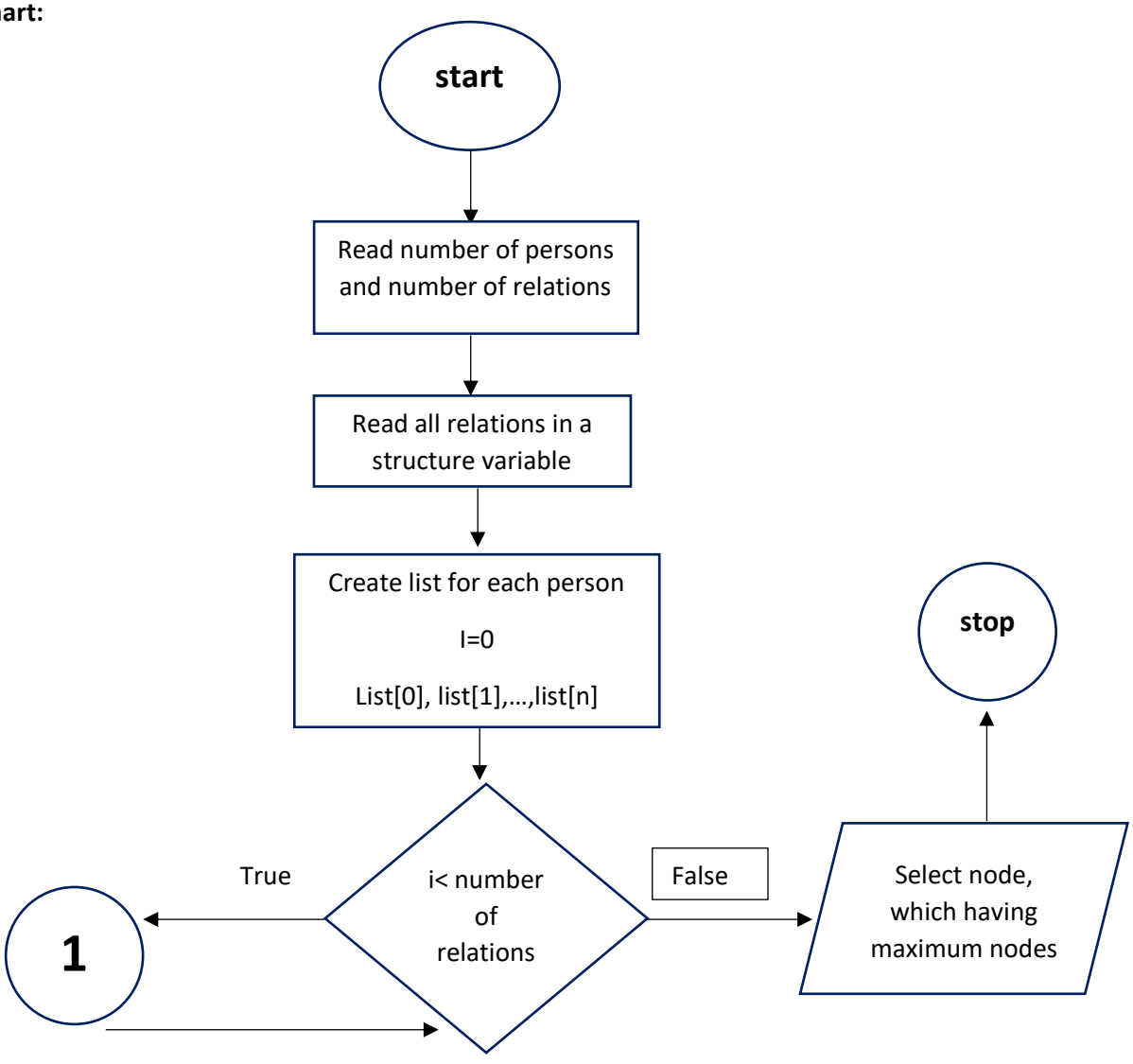
X=radius*cos(degree).

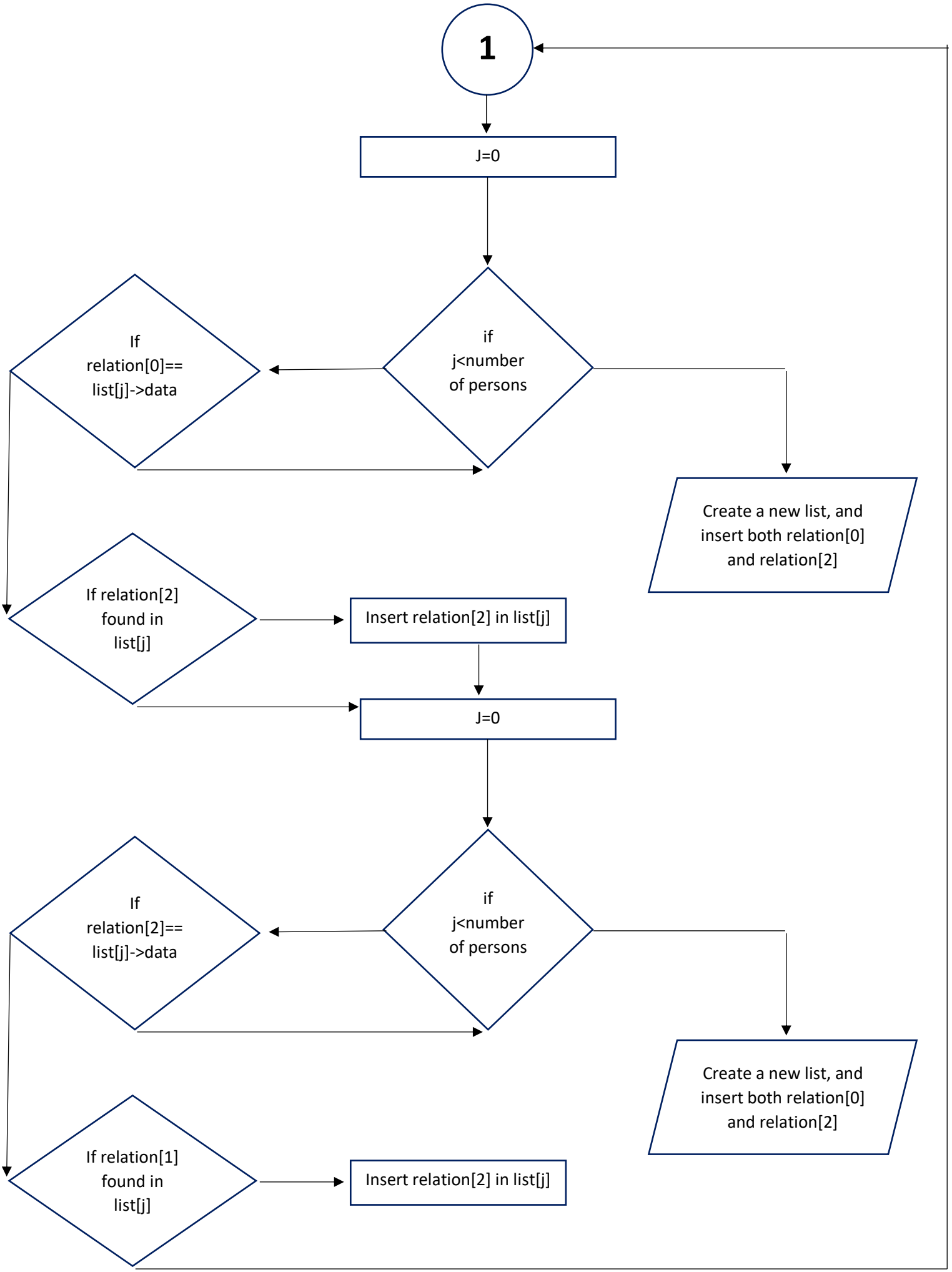
Y=radius*sin(degree).

Algorithm:

1. Read number of person's and number of relations.
2. Read relations using structure variable "data" in variable relation.
3. Call find popular function.
4. Loop for each relation.
5. If relation[0] is found at data part of any root node of list, select that list.
   5.1 if relation[2] is found in selected list, ignore that value.
   5.2 Else insert that value at selected list.
6. Else, create new list and insert both, relation[0] and relation[2] values in newly created list.
7. If relation[2] is found at data part of any root node of list, select that list.
   7.1 if relation[0] is found in selected list, ignore that value.
   7.2 Else insert that value at selected list.
8. Else, create new list and insert both, relation[0] and relation[2] values in newly created list.
9. Select that list which having maximum number of nodes. Root node of that list will be our most popular person.

Flowchart:




1
J=0
if
j<number
of persons
If
relation[0]==
list[j]->data
Create a new list, and
insert both relation[0]
and relation[2]
If relation[2]
found in
list[j]
Insert relation[2] in list[j]
J=0
if
j<number
of persons
If
relation[2]==
list[j]->data
Create a new list, and
insert both relation[0]
and relation[2]
If relation[1]
found in
list[j]
Insert relation[2] in list[j]

Algorithm for graphical display of person's and relations between them.

1. read size of graph from user.
2. Degree_backup=Degree= total number of persons/360.
3. midx= middle value of horizontal screen
4. midy= middle value of vertical screen .
5. Loop from i=0 to number of persons.
   - **5.1** x-axis=radius*cos(degree)
   - **5.2** y-axis=radius*sin(degree)
   - **5.3** draw circle at (x,y).
   - **5.4** print data part of "head" of list[i] at (x+midx, y+midy).
   - **5.5** store values of data part of "head", x+midx and y+midy, in a matrix.
   - **5.6** Degree=degree+degree_backup.
6. Loop from i=0 to number of persons.
   - **6.1** cur=list[i].
   - **6.2** search data part of list[i] in matrix.
     - **6.2.1** x1=x value from matrix.
     - **6.2.2** Y1= y value from matrix.
   - **6.3** cur=cur->next.
   - **6.4** While cur!=NULL
     - **6.4.1** search x & y value for cur-> data from matrix and store in x2 and y2.
     - **6.4.2** Draw line from x1,y1 to x2,y2.

Program:

```
/*************************************************************************
```

AIM: Given a list of friend relationship among a pair of friends. Find out the person

who is having highest popularity among them. A person with highest link with others is the most popular. Example: A – B, B – C, C- F, C-E, D-C, D-B. Here, C is related with 4 and hence the most popular. Use adjacency list for your implementation.
by ram babu ray.

**********/

```
//header files
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>

//structure variable to store relations
typedef struct data
{
   char rel[4];
}data;

// function to read relations from user
void read_relations(data relation[],int relations)
{   int i;
    printf("enter %d relations:\n",relations);
    for(i=0;i<relations;i++)
            {  printf("enter relation %d: ",i);
               scanf("%s",relation[i].rel);
            }
}
//defining list
typedef struct node
{
```

```
   char data;
   struct node *next;
}node;

//function to insert values in list
void insert(node **Top,char data)
{
   node *newnode=NULL,*top=*Top,*p;
   newnode=(node*)malloc(sizeof(node));
   newnode->data=data;
   newnode->next=NULL;
   if(top==NULL)
     top=newnode;
   else
    {
          p=top;
          while(p->next!=NULL)
           {   if(p->data==data)
               { break;}
              p=p->next;
           }
          if(p->data!=data)
          p->next=newnode;
     // printf("inserted after %c",p->data);
    }
   *Top=top;
}
//

//function to display number of relations and display all relatives of given person
int display(node **top)
```

```
{  node *temp=*top;
   int count=1;
   printf("%c->",temp->data);
   temp=temp->next;
   while(temp->next!=NULL)
   {   count++;
       printf("%c, ",temp->data);
       temp=temp->next;
   }
    printf("%c, ",temp->data);
    printf("\nnumber of relations=%d\n",count);
    return count;
}
//function to DRAW GRAPH using graphics
void graph(node *array[],int **matrix,int vertex,int relations)
{
    int midx, midy,gdriver = DETECT, gmode, errorcode,i,j,radius,persons,i2,x,y;
    char a[2];
    float val=3.14/180,sum=0,degree;
    node *top=array[0],*temp1;
 matrix=(int**)malloc(sizeof(int)*vertex);
   for(i=0;i<vertex;i++)
     matrix[i]=(int*)malloc(sizeof(int)*3);
   a[1]=NULL;

   /* initialize graphics and local variables */
   initgraph(&gdriver, &gmode, "C:\\TC\\bgi");

   midx = getmaxx() / 2;
   midy = getmaxy() / 2;
   //setcolor(getmaxcolor());
```

```
printf("enter radius:\n");
scanf("%d",&radius);
persons=vertex;
clrscr();
degree=sum=360/persons;

 circle(midx+radius,midy,20);
 matrix[0][0]=top->data;
 matrix[0][1]=radius+midx;
 matrix[0][2]=midy;

        a[0]=top->data;

        outtextxy(radius+midx,midy,a);

for(i=0;i<persons-1;i++)
 {
  x=(radius)*cos(degree*val);
  y=(radius)*sin(degree*val);
        circle(x+midx,y+midy,20);
        top=array[i+1];
        a[0]=top->data;
        outtextxy(x+midx,y+midy,a);
 matrix[i+1][0]=top->data;
 matrix[i+1][1]=x+midx;
 matrix[i+1][2]=midy+y;
  degree=degree+sum;
 }
 int x1,x2,y1,y2;
   for(i=0;i<vertex;i++)
```

```
   { temp1=array[i];
     for(j=0;j<vertex;j++)
           if(matrix[j][0]==temp1->data)
             {x1=matrix[j][1]; y1=matrix[j][2];}
     while(temp1->next!=NULL)
     {
      for(j=0;j<vertex;j++)
            {if(matrix[j][0]==temp1->next->data)
              {x2=matrix[j][1]; y2=matrix[j][2];
               line(x1,y1,x2,y2);
               break;
              }
            }
           temp1=temp1->next;
     }
   }

  /* clean up */
  getch();
  closegraph();
}

// function to calculate most popular person
void find_popular(data relation[],int relations,int vertex)
{
   node **array,*temp1;
   int **matrix;
   int i,temp=-1,j,flag,*count;
   count=(int*)malloc(sizeof(int)*vertex);
   array=(node**)malloc(sizeof(node)*vertex);
   for(i=0;i<vertex;i++)
```

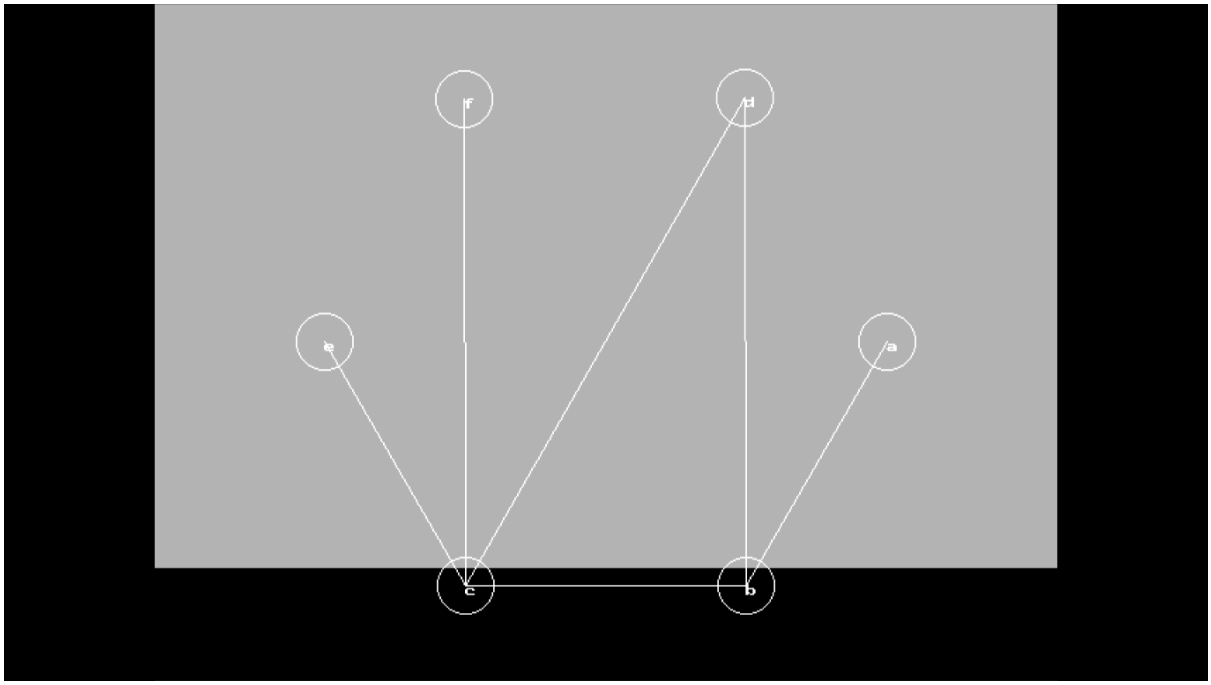
```
  array[i]=NULL;


for(i=0;i<relations;i++)
 {   flag=0;
         for(j=0;j<=temp;j++)
          {
            if(array[j]->data==relation[i].rel[0])
                  {
                     insert(&array[j],relation[i].rel[2]);
                     flag=1;
                     break;
                  }
          }
         if(flag==0)
          {  ++temp;
                insert(&array[temp],relation[i].rel[0]);
                insert(&array[temp],relation[i].rel[2]);}
          flag=0;
          //
         for(j=0;j<=temp;j++)
          {
            if(array[j]->data==relation[i].rel[2])
                  {
                     insert(&array[j],relation[i].rel[0]);
                     flag=1;
                     break;
                  }
          }
         if(flag==0)
          {  ++temp;
                insert(&array[temp],relation[i].rel[2]);
```

```
                    insert(&array[temp],relation[i].rel[0]);}
     }
   //printf("a");
    for(i=0;i<vertex;i++)
     {count[i]=display(&array[i]);printf("\n");}
    int max=count[0],index=0;
    for(i=1;i<vertex;i++)
     if(count[i]>max)
     {max=count[i];index=i;}
    for(i=0;i<vertex;i++)
      {   if(max==count[i])
                { temp1=array[i];
                 printf("most popular value is %c\n",temp1->data); }
      }
   i=0;
   printf("enter 1 to display the graph\n");
   scanf("%d",&i);
   if(i==1)
   graph(array,matrix,vertex,relations);
}

//main function
void main()
{   int relations,vertex;
    data *relation;

    printf("enter number of relations:\n");
    scanf("%d",&relations);

    printf("enter number of vertex:\n");
    scanf("%d",&vertex);
```

```
    relation=(data*)malloc(sizeof(data)*relations);
    read_relations(relation,relations);
    find_popular(relation,relations,vertex);
    getch();
}
```

Output:

Graphical representation of relations:



Link of code: https://onlinegdb.com/NGEZohqbw