East West University

**Department of Computer Science and Engineering**

**Project report**

**Course : CSE205**

**Section**  : 1

**Semester :** Summer’18

**Course Title :** Discrete Mathematics

**Lab title :** Identify whether two graphs are isomorphic or not.

**Submitted By:**

Snehashis Ghosh Pial

ID: 2017-2-60-163

**Submitted To:**

Maheen Islam  
Assistant Professor  
Department of Computer Science & Engineering

East West University

**Date of Submission:** 09.09.2018

**Introduction**: Isomorphism is a very general concept that appears in several areas of mathematics. The word derives from the Greek iso, meaning “equal’’ and morphosis, meaning “to form’’ or “to shape’’. Formally, an isomorphism is bijective morphism. Informally, an isomorphism is a map that preserves sets and relations among elements.

Two graphs are isomorphic if both have same number of vertices, edges, same number of vertices having eual degree and their sequence is same.

**Procedure:**

User will take two graphs as input. The task of this project is to determine whether those two graphs are isomorphic or not. As we know the definition of **Isomorphism is,** two graphs are Isomorphic if both have same number of vertices, edges & same number of vertices having equal degree.

The program will check the conditions given in the definition of **Isomorphism** and will determine if those two graphs are **Isomorphic** or not.

By using c language this program has done. Many function being used in this program. 2d array has been used in this program. There are some if and else statement which hold the characteristics of the following graphs. User can putting input randomly.

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

#include<ctype.h>

#include<math.h>

#include<string.h>

#define max 100

void generation(int[max][max], int\*, int edges, int a, int check[max]);

void display(int[max][max], int\*, int edges);

int checkdegree(int[max][max], int\*, int[max]);

void sort(int[max], int\*);

int isomorphic(int[max], int[max], int\*);

int neighbourchecking(int A[max][max], int B[max][max]);

int A1[max], B1[max], count1, count2, check[max]={0};

int main(void)

{

int A[max][max], B[max][max], vertex, degA[max], degB[max], degreeA=0, degreeB=0, flag=0, edges1, edges2;

printf("\t\t\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n\tThis is discrete mathematics project\n");

printf("\n\tFinding a graph is isophormic or not");

printf("\n\tRandom Graph generation:\n\tAssume Simple, Symmetric, Undirected, Unweighted Graph\n");

printf("\n\tTwo graphs are Isomorphic if: both have same number of vertices, edges\n\t& same number of vertices having eual degree.\n\n");

printf("\n\tEnter number of vertex of Graph A: ");

scanf("%d", &vertex);

printf("\n\tEnter number of nodes of Graph A: ");

scanf("%d", &edges1);

flag= vertex;

generation(A, &vertex, edges1, 0, check);

vertex=0;

for(int i=0;i<max;i++)

{

check[i]=0;

}

printf("\n\tEnter number of vertex of Graph B: ");

scanf("%d", &vertex);

printf("\n\tEnter number of nodes of Graph B: ");

scanf("%d", &edges2);

generation(B, &vertex, edges2, 1, check);

printf("\n\tInputted graph A:\n");

neighbourchecking(A,B);

display(A, &flag, edges1);

printf("\n\tInputted graph B:\n");

display(B, &vertex, edges2);

if(flag == vertex)

{

degreeA= checkdegree(A, &vertex, degA);

degreeB= checkdegree(B, &vertex, degB);

flag=0;

if(degreeA == degreeB)

{

sort(degA, &vertex);

sort(degB, &vertex);

flag= isomorphic(degA, degB, &vertex);

if(flag == 1)

printf("\n\tBoth graphs are Not Isomorphic!\n");

else if(flag == 0)

printf("\n\tBoth graphs are Isomorphic!\n");

}

else

printf("\n\tBoth graphs are Not Isomorphic!\n");

}

else

printf("\n\tBoth graphs are Not Isomorphic!\n");

}

void generation(int A[max][max], int \*vertex, int edges1, int a1, int check[])

{

int a,b,c=0,d=0;

for(int i=0;i<edges1;i++)

{

printf("\n\tenter start and end point for edge %d : \n",i);

scanf("%d%d",&a, &b);

A[a][c++]=b;

A[b][d++]=a;

if(a1==0)

{

if(check[a]==0)

{

A1[count1++]=a;

check[a]=1;

}

if(check[b]==0)

{

A1[count1++]=b;

check[b]=1;

}

}

else

{

if(check[a]==0)

{

B1[count2++]=a;

check[a]=1;

}

if(check[b]==0)

{

B1[count2++]=b;

check[b]=1;

}

}

}

}

void display(int A[max][max], int \*vertex, int edges)

{

for(int i=0;i<\*vertex;i++)

{

int c=0;

for(int j=0;j<\*vertex;j++)

{

if(j==0)

{

printf("\n\tdegree of %d : ",i);

}

if(A[i][j]>=1)

{

c++;

}

}

printf("%d\n",c);

}

int i=0, j;

for(;i<(\*vertex);i++)

{

j=0;

for(;j<(\*vertex);j++)

{

printf(" %d", A[i][j]);

}

printf("\n");

}

return;

}

int checkdegree(int A[max][max], int \*vertex, int degA[max])

{

int i=0, j, degreeA, k=0;

for(;i<(\*vertex);i++, k++)

{

j=0, degreeA=0;

for(;j<(\*vertex);j++)

{

if(A[i][j] == 1)

degreeA++;

}

degA[k]= degreeA;

}

degreeA=0;

for(k=0;k<(\*vertex);k++)

{

degreeA= (degreeA + degA[k]);

}

return(degreeA);

}

void sort(int degA[max], int \*vertex)

{

int i=0, j, temp=0;

for(;i<(\*vertex);i++)

{

j=0;

for(;j<(\*vertex)-i-1;j++)

{

if(degA[j]>degA[j+1])

{

temp= degA[j];

degA[j]= degA[j+1];

degA[j+1]= temp;

}

}

}

return;

}

int neighbourchecking(int A[max][max], int B[max][max])

{

for(int i=0;i<count1;i++)

{

printf("\n\tNeighbor of %d : ", A1[i]);

for(int j=0;j<5;j++)

{

printf("%d ", A[i][j]);

}

printf("\n");

}

printf("\n\n");

for(int i=0;i<count2;i++)

{

printf("\n\tNeighbor of %d : ", B1[i]);

for(int j=0;j<5;j++)

{

printf("%d ", B[i][j]);

}

printf("\n");

}

printf("\n\n");

}

int isomorphic(int degA[max], int degB[max], int \*vertex)

{

int flag=0, i=0, j=0;

for(;i<(\*vertex);i++, j++)

{

if(degA[i] != degB[j])

{

flag= 1;

break;

}

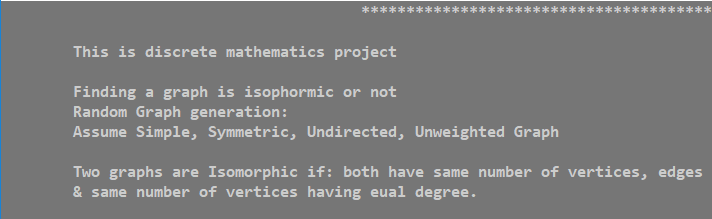
}

return(flag);

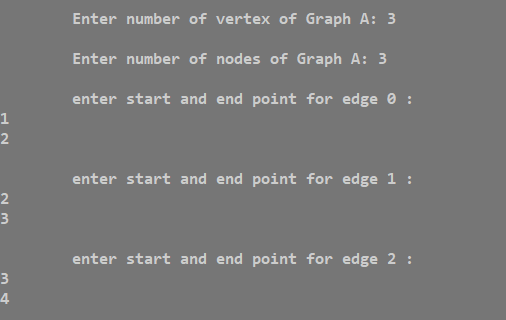
}

Output

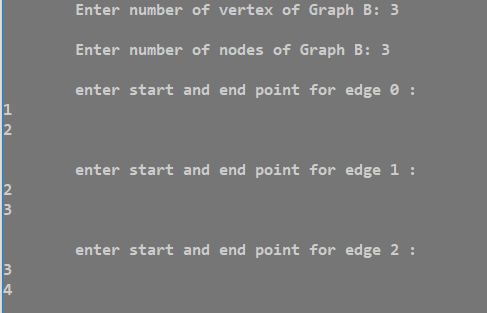
Introduction part



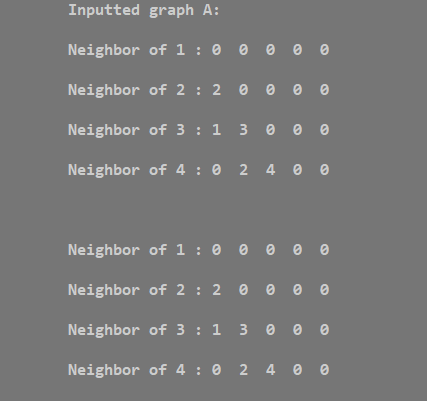
1st graph input



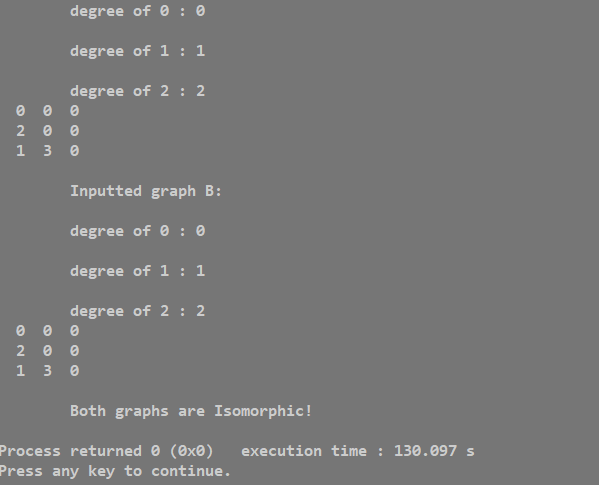
2nd graph input



Matrix of the graph



Final result



Another two graph

