基于邻接矩阵初等变换算法的图同构识别

一、实验环境和地点：

1、硬件环境：个人机，CPU 主频：1.80GHz　　内存：4GB

2、软件环境：操作系统：Windows 7

　　编程语言：C++

3、地点：智能信息处理实验室

二、实验任务解决方案：

1、邻接矩阵初等变换算法流程图。

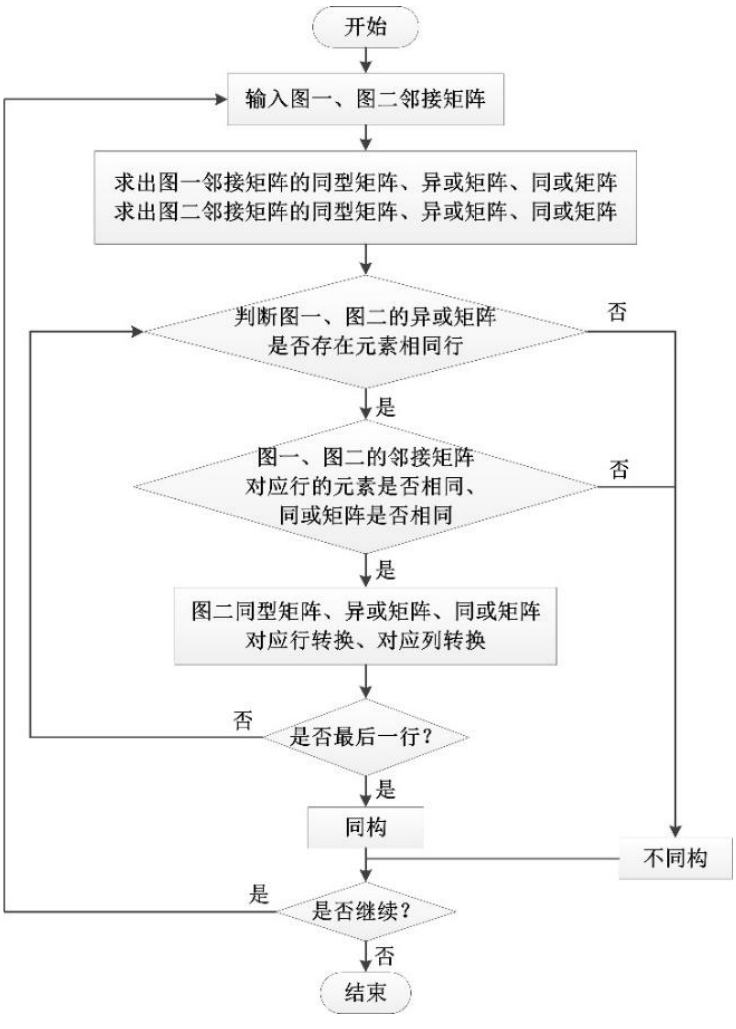

图 1 邻接矩阵算法流程

2、邻接矩阵初等变换算法实现关键代码。

（1）同型矩阵

```
void SimilarMatrix(int **p1,int **p2,int n)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(p1[i][j]>0)
            {
                p2[i][j]=1;
            }
            else
            {
                p2[i][j]=0;
            }
        }
    }
}
```

（2）异或矩阵

```
void XORMatrix(int **p1,int **p2,int **p3,int n)
{
    for( int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(i==j)
            {
                p3[i][j]=p1[i][i];
            }
            else
            {
                int sum1,sum12;
                    sum1=0;
                for(int k=0;k<n;k++)
                {
                    if(p2[i][k]==p2[j][k])
                    {
                        sum12=0;
                    }
                    else
                    {
                        sum12=1;
                    }
                    sum1=sum1+(p1[i][k]+p1[j][k])*sum12;
                }
```

```
                p3[i][j]=sum1;
            }
        }
    }
}
```

（3）同或矩阵

```
void AORMatrix(int **p1,int **p2,int **p4,int n)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            if(i==j)
            {
                p4[i][j]=p1[i][i];
            }
            else
            {
                int sum1,sum12;
                    sum1=0;
                for(int k=0;k<n;k++)
                {
                    if(p2[i][k]==p2[j][k])
                    {
                        sum12=1;
                    }
                    else
                    {
                        sum12=0;
                    }
                    sum1=sum1+(p1[i][k]+p1[j][k])*sum12;
                }
                p4[i][j]=sum1;
            }
        }
    }
}
```

（4）行行置换实现代码

```
void Transf(int **p1,int **p13,int **p14,int **p2,int **p23,int **p24,int n)
{
    int *p77=new int[n]; //临时一维数组，存放 p13
    int *p88=new int[n]; //临时一维数组，存放 p23
    int *p33=new int[n]; //临时一维数组，存放 p1
    int *p44=new int[n]; //临时一维数组，存放 p14
```

```
int *p55=new int[n]; //临时一维数组，存放 p2
int *p66=new int[n]; //临时一维数组，存放 p24
int *p99=new int[n]; //临时一维数组，用于行行替换
int t;
int tt;                    //跳转判断
int ttt=0;                 //跳转判断
//行行替换
for( int i=0;i<n;i++)
{
    //行赋值给另外一个数组 p13
    for(int i77=0;i77<n;i77++)
    {
        p77[i77]=p13[i][i77];
    }
    //行赋值给另外一个数组 p1
    for(int i33=0;i33<n;i33++)
    {
        p33[i33]=p1[i][i33];
    }
    //行赋值给另外一个数组
    for(int i44=0;i44<n;i44++)
    {
        p44[i44]=p14[i][i44];
    }
    //p77,p33,p44 冒泡排序
    BubbleSort(p77,n);
    BubbleSort(p33,n);
    BubbleSort(p44,n);
    //开始比较,p12 的每一行与 p23 的每一行进行比较
    for(int y=i;y<n;y++)
    {
        tt=0;
        //行赋值给另外一个数组
        for(int i88=0;i88<n;i88++)
        {
            p88[i88]=p23[y][i88];
        }
        //行赋值给另外一个数组
        for(int i55=0;i55<n;i55++)
        {
            p55[i55]=p2[y][i55];
        }
        //行赋值给另外一个数组
        for(int i66=0;i66<n;i66++)
        {
            p66[i66]=p24[y][i66];
        }
        //p88,p55,p66 冒泡排序
        BubbleSort(p88,n);
```

```
BubbleSort(p55,n);
BubbleSort(p66,n);
//开始比较
for(int a=0;a<n;a++)
{
    if(p77[a]==p88[a])
    {
        tt=a;
        if(a==n-1)    //如果各个都相等,找到匹配
        {
            //邻接矩阵对应位置比较
            for(int b=0;b<n;b++)
            {
                if(p33[b]==p55[b])
                {
                    continue;
                }
                else if(b<n-1)
                {
                    cout<<"不同构\n";
                    return;
                }
            }
            //同或矩阵
            for(int c=0;c<n;c++)
            {
                if(p44[c]==p66[c])
                {
                    continue;
                }
                else if(c<n-1)
                {
                    cout<<"不同构\n";
                    return;
                }
            }
            ttt++;          //成功匹配一行
            //行行转换 p2
            for(int u1=0;u1<n;u1++)
            {
                t=p2[i][u1];
                p2[i][u1]=p2[y][u1];
                p2[y][u1]=t;
            }
            for(int u11=0;u11<n;u11++)
            {
                t=p2[u11][i];
                p2[u11][i]=p2[u11][y];
                p2[u11][y]=t;
```

```
                    }
                    //行行转换 p23
                    for(int u2=0;u2<n;u2++)
                    {
                        t=p23[i][u2];
                        p23[i][u2]=p23[y][u2];
                        p23[y][u2]=t;
                    }
                    for(int u22=0;u22<n;u22++)
                    {
                        t=p23[u22][i];
                        p23[u22][i]=p23[u22][y];
                        p23[u22][y]=t;
                    }
                    //行行转换 p24
                    for(int u3=0;u3<n;u3++)
                    {
                        t=p24[i][u3];
                        p24[i][u3]=p24[y][u3];
                        p24[y][u3]=t;
                    }
                    for(int u33=0;u33<n;u33++)
                    {
                        t=p24[u33][i];
                        p24[u33][i]=p24[u33][y];
                        p24[u33][y]=t;
                    }
                break;
                }
                else
                {
                    continue;
                }
            }
            else if(y==n-1)         //一直循环到最后都未找到匹配
            {
                cout<<"-------------------------------------\n";
                cout<<"图同构判断结果：";
                cout<<"不同构\n";
                return;
            }
            else
            {
                break;
            }
        }
        //匹配无误，则进行行替换
        if(tt==n-1)
        {
```

```
                    if(ttt==n)
                    {
                        cout<<"-----------------------------------\n";
                        cout<<"> 图同构判断结果：";
                        cout<<"同构\n";
                        cout<<"-----------------------------------\n";
                        return;
                    }
                    else
                    {
                       break;//成功跳出循环判断下一行
                    }
                }
            }
        }
    }
```

（5）冒泡排序

```
void BubbleSort(int mp[],int n)
{
    int t;
    for(int i=0;i<n-1;i++)
    {
        for(int j=0;j<n-1-i;j++)
        {
            if(mp[j]>mp[j+1])
            {
                t=mp[j];
                mp[j]=mp[j+1];
                mp[j+1]=t;
            }
        }
    }
}
```

三、邻接矩阵算法的计算复杂度分析：

（1）最好情况：

每一行都互相对应，所以复杂度为：3n^2+3n^3+8n^2，时间复杂度为 O(n^3)。

（2）最差情况：

每一行都与最后一行对应，所以复杂度为：3n^2+3n^3+8n*n!，时间复杂度为 O(n*n!)。

（3）平均时间复杂度为 O(n*n!)

四、总结综合设计心得体会：

1、实例演示

例：图 $G=(V_G, E_G)$，图 $H=(V_H, E_H)$ 图形表示如下：

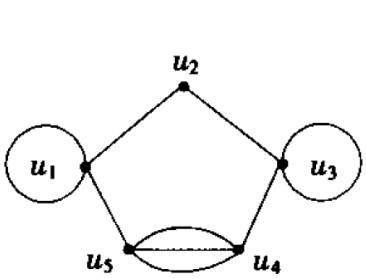

图 G

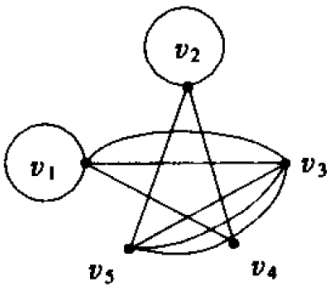

图 H

（1）根据图形，求得邻接矩阵如下：

$$A_G=\begin{array}{c} \begin{matrix} u_1 & u_2 & u_3 & u_4 & u_5 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 \\ 0 & 0 & 2 & 0 & 3 \\ 1 & 0 & 0 & 3 & 0 \end{bmatrix} \end{array} \qquad A_H=\begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 2 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 & 3 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 3 & 0 & 0 \end{bmatrix} \end{array}$$

（2）运行图同构判断程序，输入图顶点数以及图一、图二的邻接矩阵，得出图同构判断结果。程序运行界面如下：

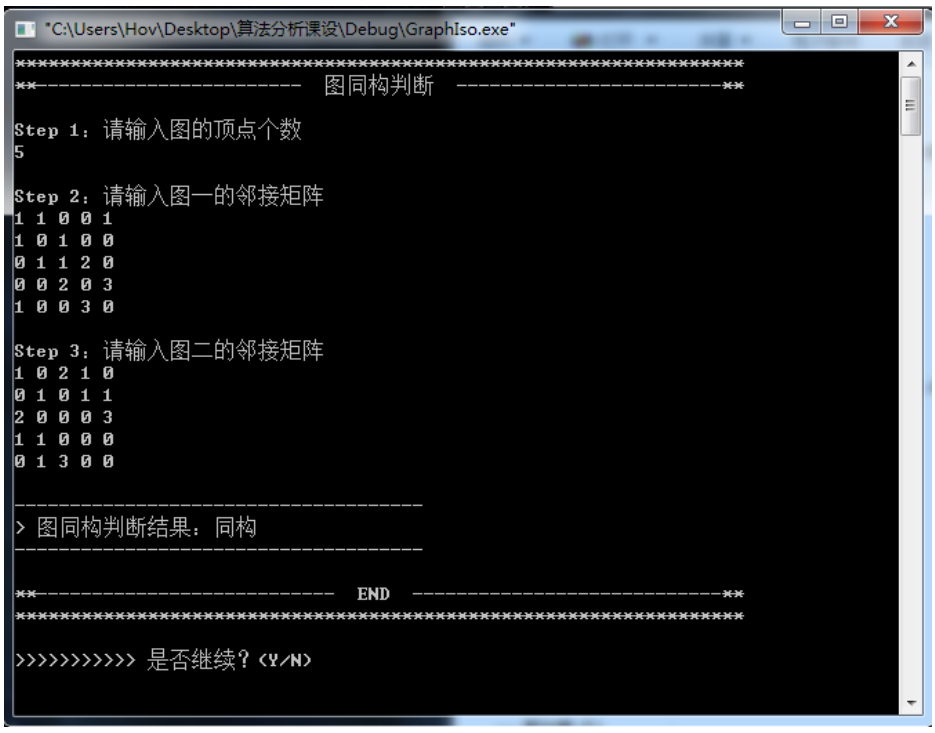

图 2 程序运行界面：以邻接矩阵判断两个图是否同构

由图同构判断结果可知，图 G 与图 H 同构。

附录：程序完整源代码

```
#include<iostream>
using namespace std;

/******************函数定义*********************/
//同型矩阵
void SimilarMatrix(int **p1,int **p2,int n)
{
    for(int i=0;i<n;i++)
    {

        for(int j=0;j<n;j++)
        {
            if(p1[i][j]>0)
            {
                p2[i][j]=1;
            }
            else
            {
                p2[i][j]=0;
            }
        }
    }
}

//异或矩阵
void XORMatrix(int **p1,int **p2,int **p3,int n)
{
        for( int i=0;i<n;i++)
    {

        for(int j=0;j<n;j++)
        {

            if(i==j)
            {
                p3[i][j]=p1[i][i];
            }
            else
            {
                int sum1,sum12;
                    sum1=0;
                for(int k=0;k<n;k++)
                {

                    if(p2[i][k]==p2[j][k])
                    {
                        sum12=0;
```

```
                    }
                    else
                    {
                        sum12=1;
                    }

                    sum1=sum1+(p1[i][k]+p1[j][k])*sum12;
                }

                p3[i][j]=sum1;
            }
        }
    }
}

//同或矩阵
void AORMatrix(int **p1,int **p2,int **p4,int n)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {

            if(i==j)
            {
                p4[i][j]=p1[i][i];
            }
            else
            {
                int sum1,sum12;
                    sum1=0;
                for(int k=0;k<n;k++)
                {
                    if(p2[i][k]==p2[j][k])
                    {
                        sum12=1;
                    }
                    else
                    {
                        sum12=0;
                    }
                    sum1=sum1+(p1[i][k]+p1[j][k])*sum12;
                }
                p4[i][j]=sum1;
            }
        }
    }
}
```

```
//输出函数
void Display(int **p,char *s,int n)
{
    cout<<s;
    cout<<"\n";
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            cout<<p[i][j];
            cout<<"\t";
        }
        cout<<"\n";
    }
}

//冒泡排序
void BubbleSort(int mp[],int n)
{
    int t;
    for(int i=0;i<n-1;i++)
    {
        for(int j=0;j<n-1-i;j++)
        {
            if(mp[j]>mp[j+1])
            {
                t=mp[j];
                mp[j]=mp[j+1];
                mp[j+1]=t;
            }
        }
    }
}


//异或矩阵行转换
void Transf(int **p1,int **p13,int **p14,int **p2,int **p23,int **p24,int n)
{
    int *p77=new int[n]; //临时一维数组，存放 p13
    int *p88=new int[n]; //临时一维数组，存放 p23
    int *p33=new int[n]; //临时一维数组，存放 p1
    int *p44=new int[n]; //临时一维数组，存放 p14
    int *p55=new int[n]; //临时一维数组，存放 p2
    int *p66=new int[n]; //临时一维数组，存放 p24
    int *p99=new int[n]; //临时一维数组，用于行行替换

    int t;
    int tt;                 //跳转判断
    int ttt=0;              //跳转判断
```

```
//行行替换
for( int i=0;i<n;i++)
{
    //行赋值给另外一个数组 p13
    for(int i77=0;i77<n;i77++)
    {
        p77[i77]=p13[i][i77];
    }
    //行赋值给另外一个数组 p1
    for(int i33=0;i33<n;i33++)
    {
        p33[i33]=p1[i][i33];
    }
    //行赋值给另外一个数组
    for(int i44=0;i44<n;i44++)
    {
        p44[i44]=p14[i][i44];
    }
    //p77,p33,p44 冒泡排序
    BubbleSort(p77,n);
    BubbleSort(p33,n);
    BubbleSort(p44,n);

    //开始比较,p12 的每一行与 p23 的每一行进行比较
    for(int y=i;y<n;y++)
    {
        tt=0;
        //行赋值给另外一个数组
        for(int i88=0;i88<n;i88++)
        {
            p88[i88]=p23[y][i88];
        }
        //行赋值给另外一个数组
        for(int i55=0;i55<n;i55++)
        {
            p55[i55]=p2[y][i55];
        }
        //行赋值给另外一个数组
        for(int i66=0;i66<n;i66++)
        {
            p66[i66]=p24[y][i66];
        }
        //p88,p55,p66 冒泡排序
        BubbleSort(p88,n);
        BubbleSort(p55,n);
        BubbleSort(p66,n);
        //开始比较
        for(int a=0;a<n;a++)
```

```
{
    if(p77[a]==p88[a])
    {
        tt=a;
        if(a==n-1)    //如果各个都相等,找到匹配
        {
            //邻接矩阵对应位置比较
            for(int b=0;b<n;b++)
            {
                if(p33[b]==p55[b])
                {
                    continue;
                }
                else if(b<n-1)
                {
                    cout<<"不同构\n";
                    return;
                }
            }
            //同或矩阵
            for(int c=0;c<n;c++)
            {
                if(p44[c]==p66[c])
                {
                    continue;
                }
                else if(c<n-1)
                {
                    cout<<"不同构\n";
                    return;
                }
            }
            ttt++;          //成功匹配一行
            //行行转换 p2
            for(int u1=0;u1<n;u1++)
            {
                t=p2[i][u1];
                p2[i][u1]=p2[y][u1];
                p2[y][u1]=t;
            }
            for(int u11=0;u11<n;u11++)
            {
                t=p2[u11][i];
                p2[u11][i]=p2[u11][y];
                p2[u11][y]=t;
            }
            //行行转换 p23
            for(int u2=0;u2<n;u2++)
            {
```

```
                    t=p23[i][u2];
                    p23[i][u2]=p23[y][u2];
                    p23[y][u2]=t;
                }
                for(int u22=0;u22<n;u22++)
                {
                    t=p23[u22][i];
                    p23[u22][i]=p23[u22][y];
                    p23[u22][y]=t;
                }
                //行行转换 p24
                for(int u3=0;u3<n;u3++)
                {
                    t=p24[i][u3];
                    p24[i][u3]=p24[y][u3];
                    p24[y][u3]=t;
                }
                for(int u33=0;u33<n;u33++)
                {
                    t=p24[u33][i];
                    p24[u33][i]=p24[u33][y];
                    p24[u33][y]=t;
                }
            break;
            }
            else
            {
                continue;
            }
        }
        else if(y==n-1)         //一直循环到最后都未找到匹配
        {
            cout<<"-------------------------------------\n";
            cout<<"图同构判断结果： ";
            cout<<"不同构\n";
            return;
        }
        else
        {
            break;
        }
    }

    //匹配无误，则进行行替换
    if(tt==n-1)
    {
        if(ttt==n)
        {
            cout<<"-------------------------------------\n";
```

```
                    cout<<"> 图同构判断结果：";
                    cout<<"同构\n";
                    cout<<"-----------------------------------\n";
                    return;
                }
                else
                {
                   break;//成功跳出循环判断下一行
                }
            }
        }
    }
}

//主程序
int main()
{
    int Vertex_Num;      //图的顶点数
    char *s;             //字符串提示
    char Run_Flag='y';
    cout<<"*************************************************************************\n";
    cout<<"**----------------------   图同构判断   ----------------------**\n";
    cout<<"\n";
    while(Run_Flag=='y')
    {
        cout<<"Step 1：请输入图的顶点个数\n";
        cin>>Vertex_Num;  //接收图的顶点个数
        if(cin.fail())
        {
            cout<<"**********输入错误，请重新输入**********\n";
            continue;
        }
        else
        {
            //创建图一邻接矩阵数组
            int **p1=new int*[Vertex_Num];
            for(int i1=0;i1<Vertex_Num;i1++)
            {
                p1[i1]=new int[Vertex_Num];
            }
            //创建图一同型矩阵
            int **p12=new int*[Vertex_Num];
            for(i1=0;i1<Vertex_Num;i1++)
            {
                p12[i1]=new int[Vertex_Num];
            }
            //创建图一行异或矩阵
            int **p13=new int*[Vertex_Num];
            for(i1=0;i1<Vertex_Num;i1++)
```

```
{
    p13[i1]=new int[Vertex_Num];
}
//创建图一行同或矩阵
int **p14=new int*[Vertex_Num];
for(i1=0;i1<Vertex_Num;i1++)
{
    p14[i1]=new int[Vertex_Num];
}

//创建图二邻接矩阵数组
int **p2=new int*[Vertex_Num];
for(int i2=0;i2<Vertex_Num;i2++)
{
    p2[i2]=new int[Vertex_Num];
}
//创建图二同型矩阵
int **p22=new int*[Vertex_Num];
for(i1=0;i1<Vertex_Num;i1++)
{
    p22[i1]=new int[Vertex_Num];
}
//创建图二行异或矩阵
int **p23=new int*[Vertex_Num];
for(i1=0;i1<Vertex_Num;i1++)
{
    p23[i1]=new int[Vertex_Num];
}
//创建图二行同或矩阵
int **p24=new int*[Vertex_Num];
for(i1=0;i1<Vertex_Num;i1++)
{
    p24[i1]=new int[Vertex_Num];
}

//接收第一个邻接矩阵的二维数组
cout<<"\nStep 2：请输入图一的邻接矩阵\n";
for(int i11=0;i11<Vertex_Num;i11++)
{
    for(int j11=0;j11<Vertex_Num;j11++)
    {
        cin>>p1[i11][j11];
    }

}
//接收第二个邻接矩阵的二维数组
cout<<"\nStep 3：请输入图二的邻接矩阵\n";
for(int i22=0;i22<Vertex_Num;i22++)
{
```

```
                for(int j22=0;j22<Vertex_Num;j22++)
                {
                    cin>>p2[i22][j22];
                }
            }

            //图一同型矩阵
            SimilarMatrix(p1,p12,Vertex_Num);
            //图一异或矩阵
            XORMatrix(p1,p12,p13,Vertex_Num);
            //图一同或矩阵
            AORMatrix(p1,p12,p14,Vertex_Num);
            //图二同型矩阵
            SimilarMatrix(p2,p22,Vertex_Num);
            //图二异或矩阵
            XORMatrix(p2,p22,p23,Vertex_Num);
            //图二同或矩阵
            AORMatrix(p2,p22,p24,Vertex_Num);

            cout<<"\n";
            Transf(p1,p13,p14,p2,p23,p24,Vertex_Num);   //核心代码，进行行转换
            cout<<"\n";
            cout<<"**--------------------------   END   --------------------------**\n";

    cout<<"*****************************************************************\n";
            cout<<"\n>>>>>>>>>>> 是否继续？(Y/N)";
            cin>>Run_Flag;
        }
    }
    return 0;
}
```