《离散数学》课程实验报告

3 求关系的自反、对称和传递闭包

实验原理和方法

对以矩阵表示的关系，其自反闭包只要将矩阵的主对角线全部置为1，对称闭包则由关系矩阵加上其转置矩阵得到（逻辑加），传递闭包根据定义为R的零次方至n次方的并集，通过定义获得传递闭包的答案。

C++语言源代码

```
#include<iostream>
#include<cstdlib>
#include<vector>

//output函数用于输出当前矩阵
void output(std::vector<std::vector<int>> arr);
/*
求自反闭包
在离散数学中，对于既不是自反也不是反自反的关系，适当的添加一些序偶使之变成自反关系，同时要求添加的序偶尽可能的少
*/
void reflexive(std::vector<std::vector<int>> arr);
/*
求对称闭包
对称闭包是X上包含R的最小的对称关系。即R'=R∪R^-1
*/
void symmetric(std::vector<std::vector<int>> arr);
/*
求传递闭包
传递闭包、即在数学中，在集合X上的二元关系R的传递闭包是包含R的X上的最小的传递关系。
*/
void transitive(std::vector<std::vector<int>> arr);
//测试函数
void select();
//退出函数
void exit();

std::vector<std::vector<int>> s;
int col,row;

int main()
{
    select();
    return 0;
```

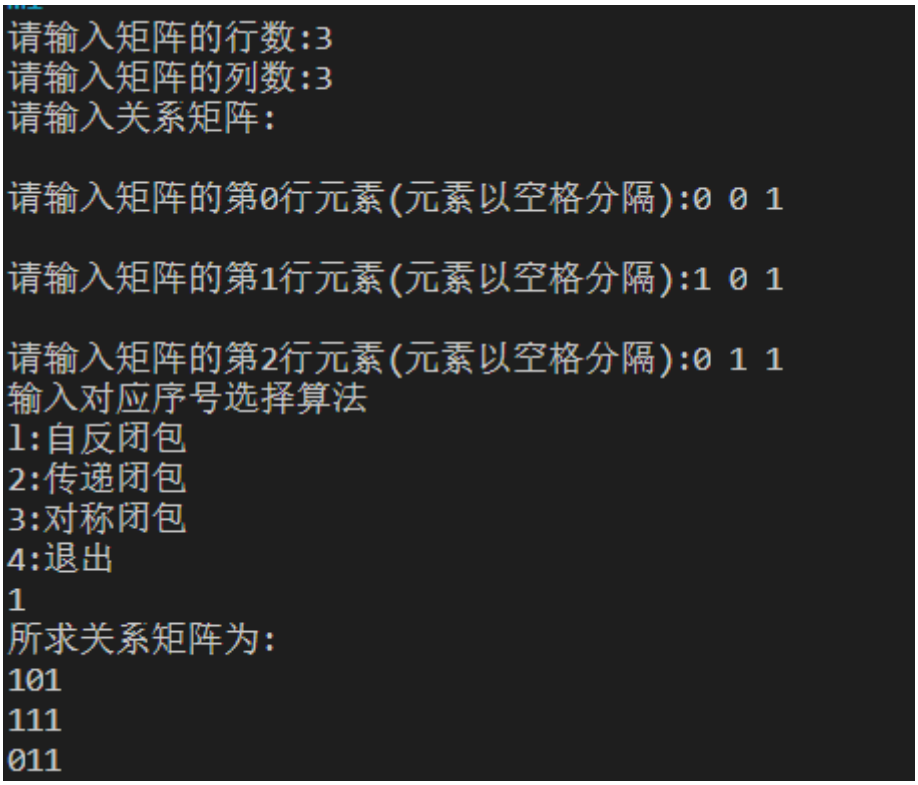
```
}

void select()
{
    int sign;

    //输入矩阵
    std::cout<<"请输入矩阵的行数:";
    std::cin>>row;
    std::cout<<"请输入矩阵的列数:";
    std::cin>>col;
    std::cout<<"请输入关系矩阵:"<<std::endl;
    for(int i = 0;i < row;i++)
    {
        std::cout<<std::endl;
        std::cout<<"请输入矩阵的第"<<i<<"行元素(元素以空格分隔):";
        std::vector<int> t;
        s.push_back(t);
        for(int j = 0;j < col;j++){
            int temp;
            s[i].push_back(temp);
            std::cin>>s[i][j];
        }
    }

    //选择算法
    std::cout<<"输入对应序号选择算法"<<std::endl<<"1:自反闭包"<<std::endl<<"2:传递闭包"
<<std::endl<<"3:对称闭包"<<std::endl<<"4:退出"<<std::endl;
    std::cin>>sign;
    switch(sign)
    {
        case 1:reflexive(s); break;
        case 2:transitive(s);break;
        case 3:symmetric(s);break;
        case 4:exit();break;
    }
}

void output(std::vector<std::vector<int>> arr)
{
    std::cout<<"所求关系矩阵为:"<<std::endl;
    for(int i = 0;i < row;i++)
    {
        for(int j = 0;j < col;j++)
            std::cout<<arr[i][j];
        std::cout<<std::endl;
    }
}

void reflexive(std::vector<std::vector<int>> arr)
{
    for(int i = 0;i < row;i++)
        arr[i][i] = 1;
    output(arr);
    select();
}

void symmetric(std::vector<std::vector<int>> arr)
```

```
{
    std::vector<std::vector<int>> temp=arr;
    for(int i = 0;i < row;i++)
        for(int j = 0;j < col;j++)
        {
            arr[i][j] = arr[i][j] + temp[i][j];
            if(arr[i][j] > 1)
                arr[i][j] = 1;
        }
    output(arr);
    select();
}

void transitive(std::vector<std::vector<int>> arr)
{
    std::vector<std::vector<int>> m=arr,t=arr;
    int k,h;
    std::vector<std::vector<int>> a;
    //预处理
    for(int i = 0;i < row;i++){
        std::vector<int> temp;
        a.push_back(temp);
        for(int j = 0;j < col;j++)
        {
            a[i].push_back(0);
        }
    }
    //求传递闭包
    for(int h = 0;h < row;h++)
    {
        //求n次方传递闭包
        for(int i = 0;i < row;i++)
            for(int j = 0;j < col;j++)
                if (m[i][j] == 1)
                {
                    for(int k = 0;k < row;k++)
                        if(arr[j][k] == 1)
                            a[i][k] = 1;
                }
        //求逻辑加获得的最终结果
        for(int i = 0;i < row;i++)
            for(int j = 0;j < col;j++)
            {
                m[i][j] = a[i][j];
                t[i][j] += a[i][j];
                a[i][j] = 0;
                if(t[i][j] > 1)
                    t[i][j] = 1;
            }
    }

    output(t);
    select();
}

void exit()
{
    exit(1);
```

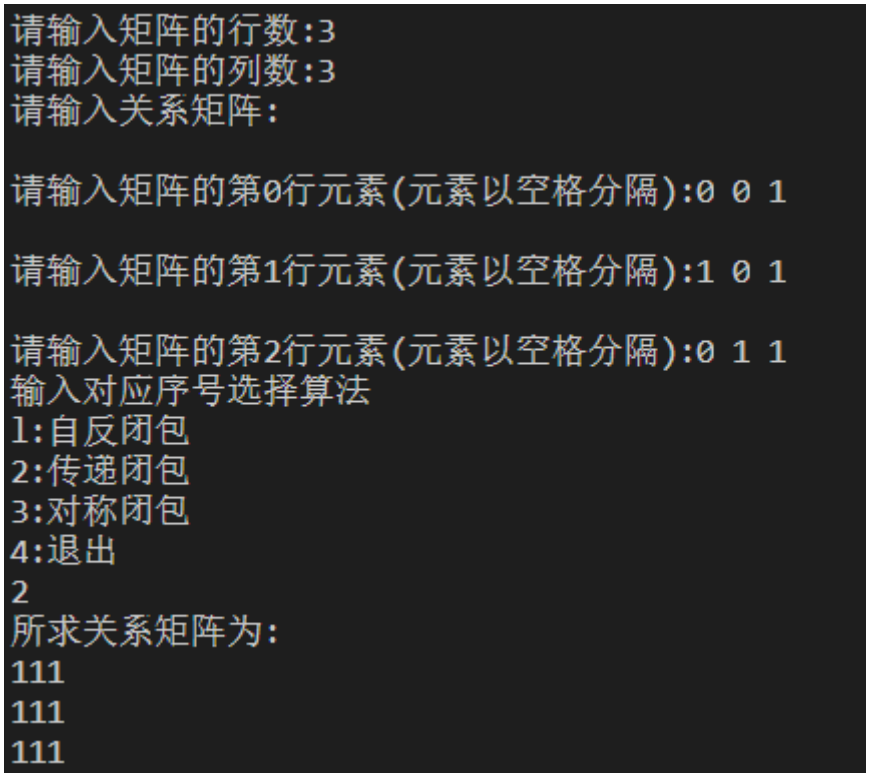
```
}
```

实验结果

1. 求关系的自反闭包



2. 求关系的传递闭包



3. 求关系的对称闭包

```
请输入矩阵的行数:3
请输入矩阵的列数:3
请输入关系矩阵:

请输入矩阵的第0行元素(元素以空格分隔):0 0 1

请输入矩阵的第1行元素(元素以空格分隔):1 0 1

请输入矩阵的第2行元素(元素以空格分隔):0 1 1
输入对应序号选择算法
1:自反闭包
2:传递闭包
3:对称闭包
4:退出
3
所求关系矩阵为:
001
101
011
```