《离散数学》课程实验报告 6 Warshall传递闭包算法举例

实验内容

用warshall算法实现传递闭包

实验原理和方法

> 传递闭包：存在一个有向图，能用布尔邻接矩阵表示（1、0）。存在一个矩阵，它能够给定图>的顶点之间是否存在任意长度的有向路径，这种矩阵称为有向图的传递闭包，是我们能够在常>数时间内判断第j个顶点是否可从第i个顶点到达。

通过Warshall算法可以计算一个布尔邻接矩阵的传递闭包,其具体过程如下，设在n个元素的有限集上关系R的关系矩阵为M：

1. 置新矩阵A=M;
2. 置k=1;
3. 对所有i如果A[i,k]=1，则对j=1..n执行：
   A[i,j]←A[i,j]∨A[k,j];
4. k增1;
5. 如果k≤n，则转到步骤（3），否则停止。
   所得的矩阵A即为关系R的传递闭包t(R)的关系矩阵。

下面将以示例的形式具体展示Warshell算法的流程：

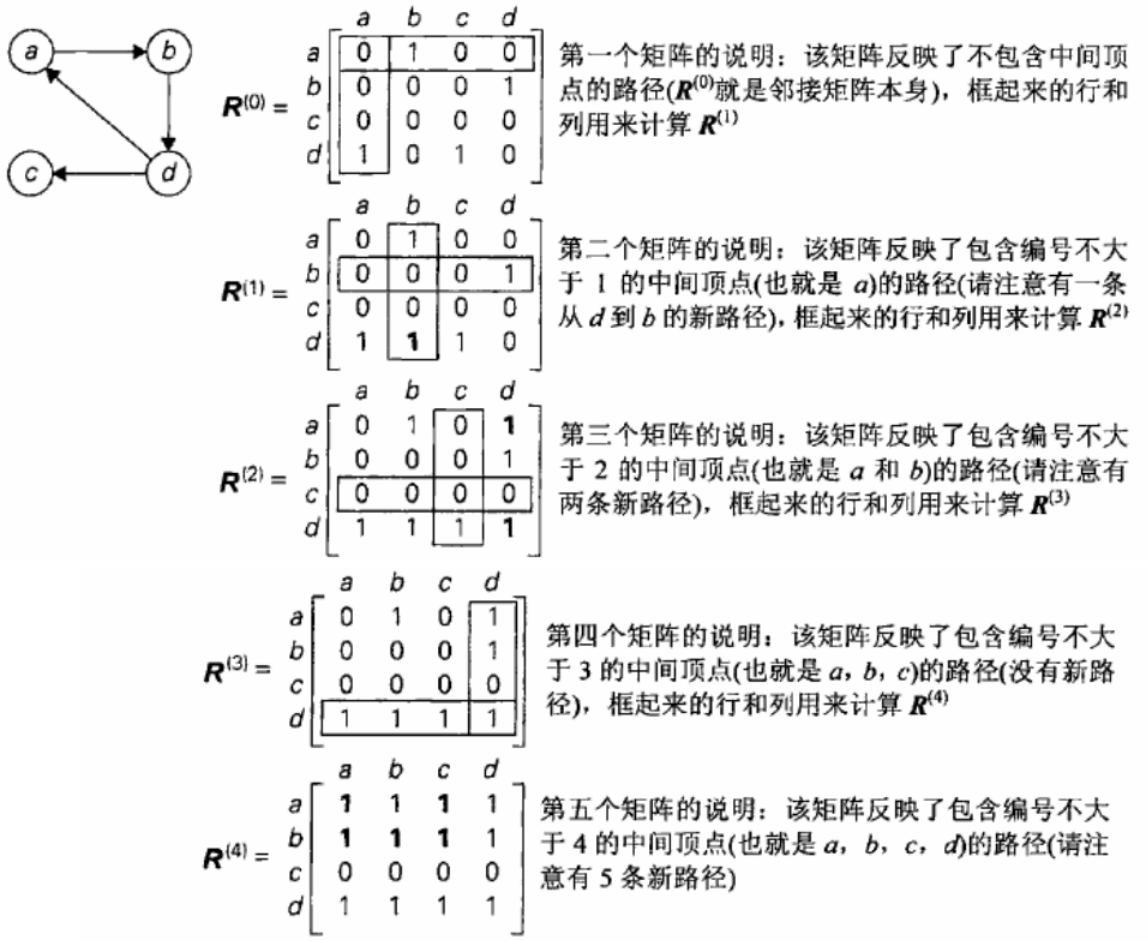

图 8.13　对图中的有向图应用 Warshall 算法，新的路径用粗体字表示

c++语言源代码

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


//对数据进行输入判断
int get_matrix(std::vector<std::vector<int>>& a,int n)
{
    for (int i = 0;i < n;i++)
    {
        std::vector<int> temp;
        a.push_back(temp);
        for (int j = 0;j < n;j++)
        {
            int t;
            std::cin>>t;
            a[i].push_back(t);
            //对输入数据进行判断，如果输入不合法就返回1
            if (a[i][j] != 0 && a[i][j] != 1)
                return 1;
         }
    }
    return 0;
```

```
}


//输出矩阵
int output_matrix(std::vector<std::vector<int>> a)
{
    int i = 0,j = 0;
    for (i = 0;i < a.size();i++) {
        for (j = 0;j < a[i].size();j++) {
            std::cout<<a[i][j]<<" ";
        }
        std::cout<<std::endl;
    }
}
int warshall(std::vector<std::vector<int>>& a)
{
//(1)i=1;
//(2)对所有j如果a[j，i]=1，则对k=0，1，…，n-1，a[j，k]=a[j，k]∨a[i，k]；
//(3)i加1；
//(4)如果i<n，则转到步骤2，否则停止
    int i = 0;
    int j = 0;
    int k = 0;
    for (i = 0;i < a.size();i++)
    {
        for (j = 0;j < a[i].size();j++)
        {
            if (a[j][i])
            {
                for (k = 0;k < a.size();k++)
                {
                    a[j][k] = a[j][k] | a[i][k];//逻辑加
                }
            }
        }
    }
}
int main()
{
    std::vector<std::vector<int>> a;
    std::cout<<"please input n:";
    int n;
    std::cin>>n;
    std::cout<<"please input a matrix with "<<n<<" * "<< n<<" :"<<std::endl;
    if (get_matrix(a,n))
     {
        printf("Get matrix error!Only 0 or 1 in matrix!\n");
        return 1; //错误返回主函数，返回值为1；
     }
    warshall(a);
    std::cout<<"The answer is:"<<std::endl;
    output_matrix(a);
    return 0;//正常返回主函数，返回值为0；
}
```

实验结果

```
please input n:4
please input a matrix with 4 * 4 :
0 1 0 0
1 1 0 1
1 0 0 0
1 1 1 1
The answer is:
1 1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
```