noj实验11报告

0.0 题目//需求分析

File Name: T011.cpp

实验4.4：用弗洛伊德算法求赋权图中任意两点间的最短路径。

Time Limit: 3000ms , Memory Limit: 10000KB , Accepted: 0 , Total Submissions: 0

VIDEO1

Description

用弗洛伊德算法求任意两点间的最短路径，并输出指定的m对结点间的最短路径。

Input

先输入一个小于100的正整数n，然后输入图的邻接矩阵（10000表示无穷大，即两点之间没有边），之后再输入一个小于100的正整数m，最后的m行每行输入两个不同的0到n-1之间的整数表示两个点。

Output

用弗洛伊德算法求任意两点间的最短路径，并输出这些两个点之间的最短路径。

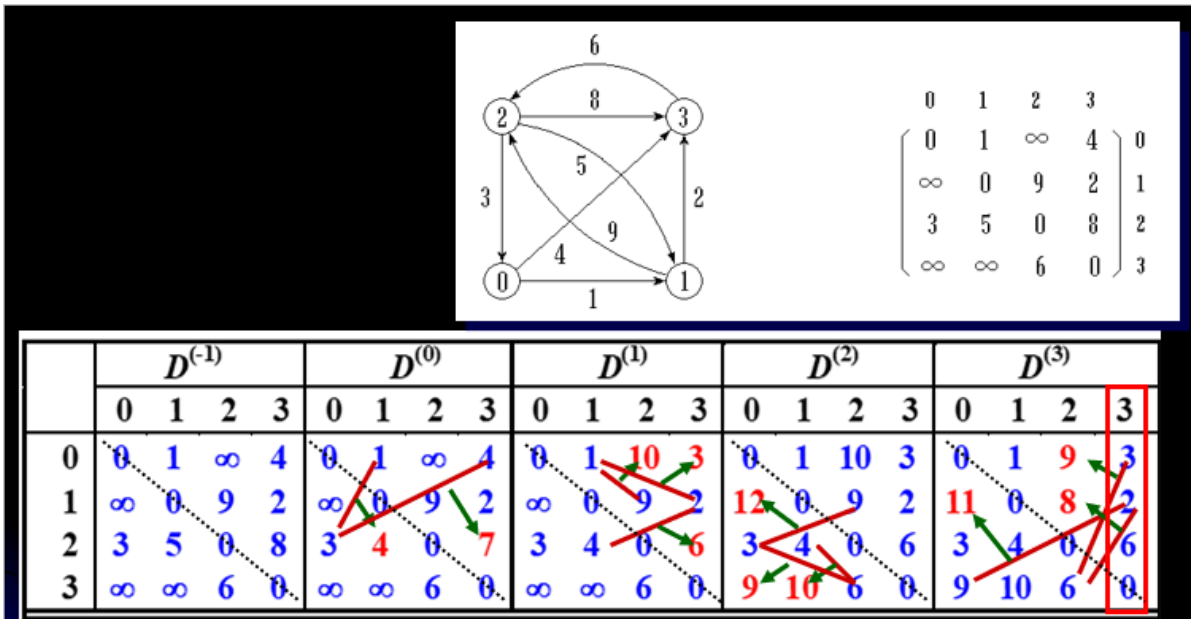
Sample Input

```
4
0 2 10 10000
2 0 7 3
10 7 0 6
```

需求分析：把第10题中的一个改为任一。算法相同。

1.0 实验思路

存储结构和以前的都一样。



就是一个起始点一列一列从上到下遍历，中点随起始点的遍历更新终点步数，终点就是起点和中点在矩阵中的交叉点，最后按需输出就可以。

2.0 代码

```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_VER_NUM 20


typedef struct {
    int vertex[MAX_VER_NUM];
    int weight[MAX_VER_NUM][MAX_VER_NUM];
    int num_vex;//弧的个数用不上
}AdjMatrix;

int TargetPath[MAX_VER_NUM];//记录所求路径

AdjMatrix* CreateMatrix(int n);   //构建邻接矩阵
void Floyd(AdjMatrix A, int D[MAX_VER_NUM][MAX_VER_NUM], int P[MAX_VER_NUM]
[MAX_VER_NUM]);
//弗洛伊德算法
void PrintMinPath(AdjMatrix A, int D[MAX_VER_NUM][MAX_VER_NUM]);
//输出最短路径

int main()
{
    int n;
    scanf("%d", &n);
    AdjMatrix* A;
    A = CreateMatrix(n);
    int D[MAX_VER_NUM][MAX_VER_NUM];//存储最短路径
    int P[MAX_VER_NUM][MAX_VER_NUM];//存储最短路线中终点的前驱结点
    Floyd(*A, D, P);
    PrintMinPath(*A, P);
    return 0;
}

AdjMatrix* CreateMatrix(int n)
{   //构建邻接矩阵
    AdjMatrix* A;
    A = (AdjMatrix*)malloc(sizeof(AdjMatrix));
    if (A == NULL) {
        return NULL;
    }
    A->num_vex = n;
    int i,j;
    for (i = 0; i < A->num_vex; i++) {
        for (j = 0; j < A->num_vex; j++) {
            scanf("%d", &A->weight[i][j]);
        }
    }
    return A;
}



void Floyd(AdjMatrix A, int D[MAX_VER_NUM][MAX_VER_NUM], int P[MAX_VER_NUM]
[MAX_VER_NUM])
{   //弗洛伊德算法
    int i, j,k;
```

```
    for (i = 0; i < A.num_vex; i++) {
        for (j = 0; j < A.num_vex; j++) {
            D[i][j] = A.weight[i][j];
            P[i][j] = j;
        }
    }
    for (i = 0; i < A.num_vex; i++) {//中间点
        for (j = 0; j < A.num_vex; j++) {//起点
            for (k = 0; k < A.num_vex; k++) {//终点
                if (D[j][k] > D[j][i] + D[i][k]) {
                    D[j][k] = D[j][i] + D[i][k];//更新最小路径
                    P[j][k] = P[j][i];//更新最小路径终点的前驱顶点
                }
            }
        }
    }
}

void PrintMinPath(AdjMatrix A, int P[MAX_VER_NUM][MAX_VER_NUM])
{   //输出最短路径
    int n;
    scanf("%d", &n);
    int j[MAX_VER_NUM][2];
    int i;
    for (i = 0; i < n; i++) {
        scanf("%d%d", &j[i][0], &j[i][1]);
    }
    for (i = 0; i < n; i++) {
        int tmp = j[i][0];
        while (tmp != j[i][1]) {
            printf("%d\n", tmp);
            tmp = P[tmp][j[i][1]];
        }
        printf("%d\n", tmp);
    }
}
```