

作业

- 问题描述：给定一个 m 行 n 列的矩阵，从左上角开始每次只能向右或者向下移动，最后到达右下角的位置，路径上的所有数字累加起来作为这条路径的路径和。编写一个实验程序使用动态规划方法求所有路径和中的最小路径和，并给出具体路径。
- 输入样例：

1	3	5	9
8	1	3	4
5	0	6	1
8	8	4	0

本题的动态规划方程为：

$$dp[i][j] = \min(dp[i-1][j], dp[i][j-1]) + a[i][j]$$

其中 $a[i][j]$ 表示原矩阵该点处的权值， $dp[i][j]$ 为到达 $a[i][j]$ 对应位置的最小值。

源代码如下所示（具体思路和分析均已包含在注释中）：

```
#include <iostream>
#include <math.h>

using namespace std;
// 存储矩阵行列数
int m,n;
// 存储路径权值数组
int a[1000][1000];
// dp 数组用于进行动态规划
int dp[1000][1000];
// 用于递归判断
```

```

bool flag = false;

// 用于输出最短路径
void minIndex(int row,int col)
{
    // 抵达终点时递归结束，且修改 flag，告诉上层的递归函数也可以结束，直至返回
    // 到第一层结束所有递归
    if( flag == true ){
        return ;
    }
    if( row == m-1 && col == n-1){
        cout << "End" << endl;
        flag = true;
        return ;
    }
    // 若抵达最下面一行，则只能往右走
    if(row == m-1){
        // 输出下一步
        cout << a[row][col+1] << " -> ";
        // 递归寻找下一步的最小值
        minIndex(row,col+1);
        if( flag == true ){
            return ;
        }
    }
    // 若抵达右边一列，则只能往下走
    if(col == n-1){
        // 输出下一步
        cout << a[row+1][col] << " -> ";
        // 递归寻找下一步的最小值
        minIndex(row+1,col);
        if( flag == true ){
            return ;
        }
    }
    // 找出 dp 数组中具有更小权值的下一步（往右 or 往下）
    // 这里是假设往下的权值更小
    if(dp[row+1][col] >= dp[row][col+1]){
        // 输出下一步
        cout << a[row][col+1] << " -> ";
        // 递归寻找下一步的最小值
        minIndex(row,col+1);
        if( flag == true ){
            return ;
        }
    }
}

```

```

    }
}
// 若往右的权值更小
else{
    // 输出下一步
    cout << a[row+1][col] << " -> ";
    // 递归寻找下一步的最小值
    minIndex(row+1,col);
    if( flag == true ){
        return ;
    }
}
}

int main()
{
    // 输入矩阵行数和列数
    cin>>m>>n;
    // 输入矩阵每一点的路径权值
    for(int i = 0;i <= m-1;i++){
        for(int j = 0;j <= n-1;j++){
            cin>>a[i][j];
        }
    }
    /**
     * dp[m][n]为到达 a[m][n]对应位置的最小值
     * 第一行只能从左往右:
     * dp[0][0] = a[0][0], dp[0][j] = a[0][j] + dp[0][j-1];
     * 而第一列元素只能从上往下:
     * dp[i][0] = dp[i-1][0] + a[i][0]
     * 第二行第二列元素的可能从当前节点的左节点(向右走)和上节点(向下走)过来
     * 那么该节点的最小值应为当前节点的值加上 min(上节点,左节点)
     * 即 dp[i][j] = min(dp[i-1][j],dp[i][j-1]) + a[i][j];
     * 所以最右下方的节点 dp[m-1][n-1]的值就为最小的路径和
     */

    // 动态规划
    dp[0][0] = a[0][0];
    // 第一列
    for(int i = 1;i <= m-1;i++){
        dp[i][0] = dp[i-1][0] + a[i][0];
    }
    // 第一行
    for(int j=1;j <= n-1;j++){

```

```

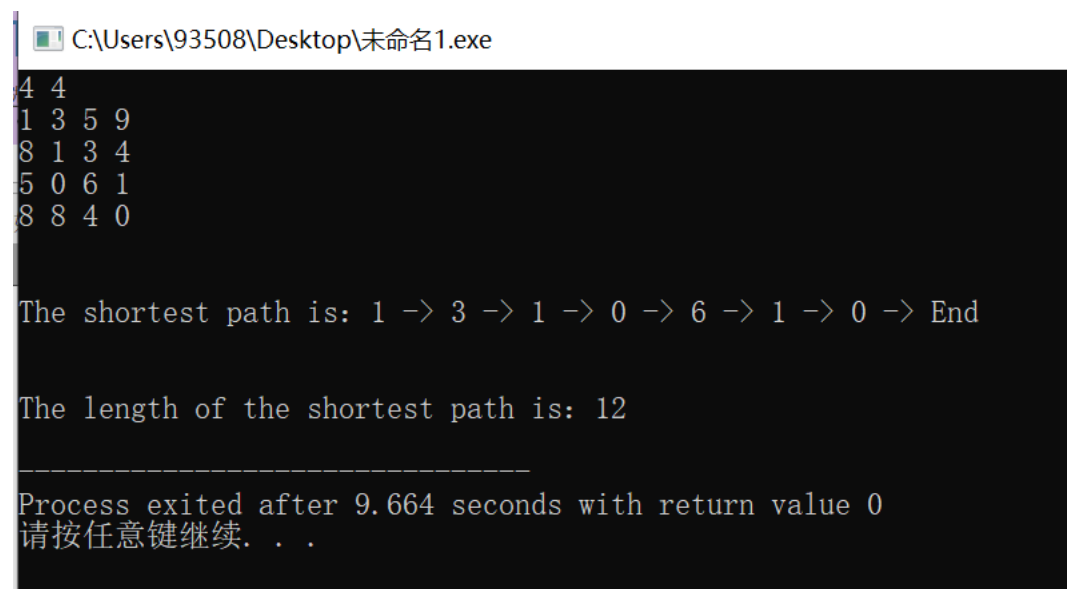
        dp[0][j] = dp[0][j-1] + a[0][j];
    }
    for(int i = 1; i <= m-1; i++){
        for(int j = 1; j <= n-1; j++){
            dp[i][j] = min(dp[i-1][j], dp[i][j-1]) + a[i][j];
        }
    }
    cout << endl << endl;

    //先输出起点
    cout << "The shortest path is: " << a[0][0] << " -> ";
    //找最短路径
    minIndex(0,0);
    cout << endl << endl;
    //输出结果
    cout << "The length of the shortest path is: " << dp[m-1][n-1] <<
endl;
    return 0;
}

```

输出截图如下所示:

样例 1:



```

C:\Users\93508\Desktop\未命名1.exe
4 4
1 3 5 9
8 1 3 4
5 0 6 1
8 8 4 0

The shortest path is: 1 -> 3 -> 1 -> 0 -> 6 -> 1 -> 0 -> End

The length of the shortest path is: 12

-----
Process exited after 9.664 seconds with return value 0
请按任意键继续. . .

```

样例 2:

```
C:\Users\93508\Desktop\未命名1.exe
5 4
1 2 3 4
6 8 9 7
4 1 2 3
4 2 8 7
9 8 4 3

The shortest path is: 1 -> 2 -> 3 -> 4 -> 7 -> 3 -> 7 -> 3 -> End

The length of the shortest path is: 30

-----
Process exited after 13.18 seconds with return value 0
请按任意键继续. . .
```

样例 3:

```
C:\Users\93508\Desktop\未命名1.exe
4 5
1 2 3 4 5
5 4 3 2 1
1 3 5 4 2
2 4 5 3 1

The shortest path is: 1 -> 2 -> 3 -> 3 -> 2 -> 1 -> 2 -> 1 -> End

The length of the shortest path is: 15

-----
Process exited after 17.85 seconds with return value 0
请按任意键继续. . .
```