实验 8 经典算法应用（动态规划）

计科 2004 202008010404 赖业智

一、问题分析

1. 问题与功能描述

1）需要处理的是两个 int 型数字，它们组成了一个方格矩阵的行和列。

2）实现的功能有，输入两个正整数 m，n，找出从图的左下角到图右上角路径数，即从坐标（1，1）到（m，n），蚂蚁只能向上或者向右移动。

3）输出不同移动路线的数目。

2. 样例分析

1）输入样例

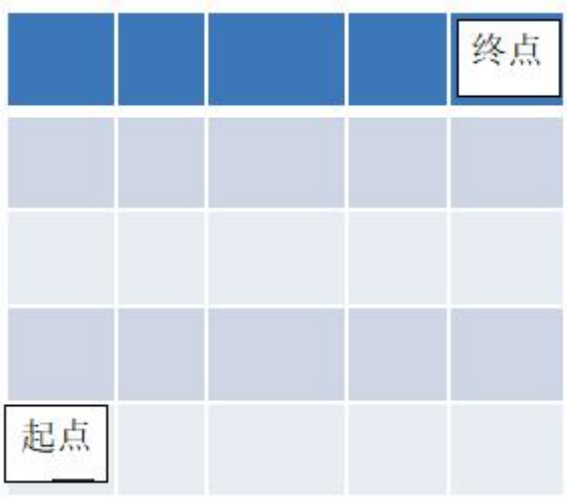
10 10

2）输出样例

48620

3）样例说明

我们可以利用组合数学的知识分析一下这个题目



如现在行为 5，列为 5，那么我们从起点到终点，无论怎么走，都必须向右走 4 步，向上走 4 步，从这 8 步中找更小步的走，因此可以是 C(4+4,4)=70。同理，样例是 C(9+9,9)=48620。

二、数据结构分析

1. 数据对象：本题处理 m，n 两个 int 整数。

2. 数据关系：m 为行，n 为列，数据规模已经确定，因此我们可以开一个二维数组表示这个 m 行 n 列的方格矩阵。

3. 物理实现

int a[20][20]={0};

三、算法分析

1.算法思想：我们可以直接是组合数学的知识，利用公式去推导出结果，但是我们可以采用动态规划的思想，获得更简便的结果。先将数组 a[1][1]起初始化为 1，我们要找出状态转移方程，从 a[1][1]到 a[2][2]的路径数可以分解为是 a[1][1]到 a[1][2]的路径数和 a[1][1]到 a[2][1]的路径数的和，因此通过累加可以得到每一个坐标的路径数，状态转移方程为

a[i][j] = a[i - 1][j] + a[i][j - 1];

3. 算法实现：

```
static int a[20][20] = { 0 };
int main() {
	int m, n;
	cin >> m >> n;
	for (int i = 1; i <= m; i++) {
		for (int j = 1; j <= n; j++) {
			if (i == 1 && j == 1)
				a[i][j] = 1;
			else
				a[i][j] = a[i - 1][j] + a[i][j - 1];
		}
	}
```

```
    cout << a[m][n];}
```

3.性能分析：

【时间复杂度】两个 for 循环，时间复杂度为 O(m*n)。

【空间复杂度】开辟了一个二维数组，由于数据规模已经固定，空间复杂度为 O(20*20)。