湖南大学

HUNAN UNIVERSITY

数 据 结 构 与 算 法 分 析

实 验 报 告

学生姓名/学号　梅炳寅　202108010206

专业班级　计科 2102

指导老师　夏艳

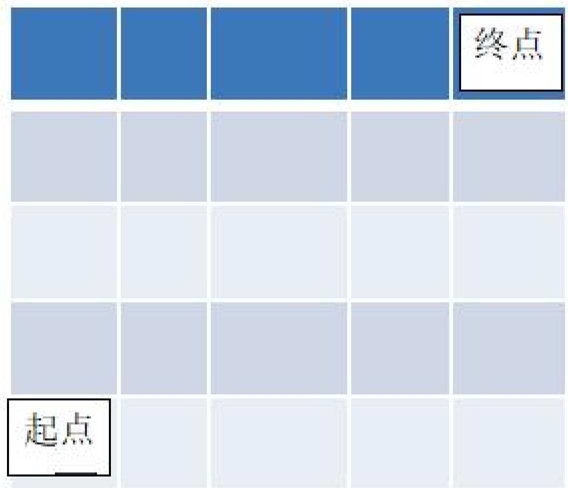
2022 年 5 月 31 日

目录

1.问题分析

桌子上有一个 m 行 n 列的方格矩阵，将每个方格用坐标表示，行坐标从下到上依次递增，列坐标从左至右依次递增，左下角方格的坐标为(1,1)，则右上角方格的坐标为(m,n)且（0<m+n<=20）。

小明是个调皮的孩子，一天他捉来一只蚂蚁，不小心把蚂蚁的右脚弄伤了，于是蚂蚁只能向上或向右移动。小明把这只蚂蚁放在左下角的方格中，蚂蚁从左下角的方格中移动到右上角的方格中，每步移动一个方格。蚂蚁始终在方格矩阵内移动，请计算出不同的移动路线的数目。



1.1 处理的对象（数据）

第一行有两个数 n、m 表示 m 行 n 列。

1.2 实现的功能

输出不同的移动路线的数目。

1.3 处理后的结果如何显示

一个整数，表示路径种数。

1.4 请用题目中样例，详细给出样例求解过程。

【输入样例】

7 8

【输出样例】

1716

【分析】

方法一、数学分析法

不难发现，答案可以通过排列组合计算得出（具体证明后面会给出），结果为 C（7+8-2，7-1）即 C（13，6），只需编程计算组合数即可。最终答案为 1716。
方法二、动态规划法

不难发现，m[1][1]=1,第一行也都是 1。到第二行第二列开始出现变化，m[2][2]=m[1][2]+m[2][1]=2，此后按照 m[i][j]=m[i-1][j]+m[i][j-1]类推。最终答案为 1716。

2.算法设计与复杂度分析

2.1 数学方法

2.1.1 算法思想

仔细观察，不难发现，这道题本质上是数学上一个比较简单的模型。我们发现：不论怎么走，总步数都是有限的，并且向右的步数与向上的步数分别都是一定的。即向上的一定为 m-1，向右的一定为 n-1。

所以我呢提的本质就变成了从 m+n-2 次中选取 m-1 次，使他们成为向上的。答案就是 C（m+n-2，m-1）。只需计算组合数即可。

2.1.2 关键功能

关键的功能是两个，一个是计算组合数，而计算组合数则需计算三个阶乘。所以需要编程实现计算阶乘的过程以及通过阶乘计算组合数的过程。

在下面的代码中 long long int jiecheng(int n)函数展示了计算阶乘的方法，long long int combination(int n,int m)展示了计算组合数的方法。

2.1.3 代码实现

```
#include<bits/stdc++.h>
using namespace std;

long long int jiecheng(int n)
{
    long long int temp=1;
    for (int i=1;i<=n;i++)
    {
        temp*=i;
    }
    return temp;
}

long long int combination(int n,int m)
{
    return (jiecheng(n)/(jiecheng(m)*jiecheng(n-m)));
}
```

```
int main()
{
    int n,m;
    cin>>n>>m;
    cout<<combination(n+m-2,m-1);
  return 0;
}
```

2.1.4 复杂度分析

时间复杂度：主要使用一层 for 循环计算阶乘，复杂度为 O（n）。

空间复杂度：几乎可以忽略不计。

2.2 动态规划法

2.2.1 算法思想

实际上本题也可以看作是一个简单的动态规划。对于图中不在边界上（即第一行或第一列的）节点来说，它们的值显然有 m[i][j]=m[i-1][j]+m[i][j-1]，即到达该节点的方法等于到达该节点前的一个节点的方法，再加上到达该节点方法。

用这种方法就可以将问题的规模缩小。最终将被压缩到起点到达起点的方法，而从起点到起点显然为 1。

2.2.2 关键功能

构建状态转移方程，

m[i][j]=1;（i 为 1 或 j 为 1）

m[i][j]=m[i-1][j]+m[i][j-1];（i，j 不为 1）

构建双重循环进行动态规划过程。

2.2.3 代码实现

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n,m;
    cin>>n>>m;
    long long int map[m+1][n+1];
    for (int i=1;i<=m;i++)
        for (int j=1;j<=n;j++)
        {
            if (i==1 || j==1) map[i][j]=1;
            else map[i][j]=map[i-1][j]+map[i][j-1];
        }
    cout<<map[m][n];
  return 0;
}
```

2.2.4 复杂度分析

时间复杂度：动态规划，双重循环，为 O（n^2）。

空间复杂度：n*m 的数组，为 O（n^2）。