

中山大学数据科学与计算机学院本科生实验报告

(2016 学年春季学期)

课程名称: Algorithm design

任课教师: 张子臻

年级	1501	专业(方向)	移动信息工程
学号	15352015	姓名	曹广杰
电话	13727022190	Email	1553118845@qq.com
开始日期	2017/3/25	完成日期	2017/3/25

1. 实验题目

SubDiagonal paths

2. 实验目的

题意: 给出 n , 在 $n \times n$ 的格子中, 一开始你位于 $(0,0)$, 然后你每次可以从 (x,y) 走到 $(x+1,y)$, 或者 $(x,y+1)$, 全程需满足 $x \geq y$, 问这样走到 (n,n) 处有几种走法。

3. 程序设计

已知数据结构;

已知数据关联;

求算状态方程;

调用函数计算;

具体解释如下:

1. 题中欲求从 $(0,0)$ 到 (n,n) 在特定的运动方式下的所有可行数, 数据结构已经给出了——矩阵;
2. 影响可行数的因素即为位置, 也已经非常明显, 矩阵可以很好地将可行数与位置信息联系起来;
3. 由于矩阵的出现以及状态的划分, 使用动态规划是很有利的。所以, 我们使用动态规划, 根据行走路径得知每一个状态的可行数都是来自于其下及其左侧的状态, 此处需要对情况进行限制;
4. 于是获得状态转移方程:

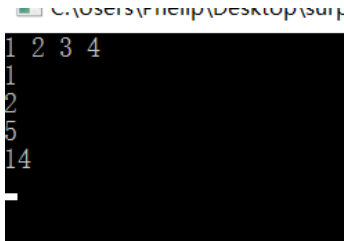
```
long long ans=0;
if(i>j){
    ans=dp(i-1,j)+dp(i,j-1);
}else{
    ans=dp(i,j-1);
}
```

使用 if 语句对条件进行了限制;

5. 这样每一个状态都非常清晰了，只需要对每一次输入的情况进行函数的调用，使用状态转移方程计算即可：

```
while(cin>>n&&){
    memset(p, -1, sizeof(p));
    cout << dp(n, n)<<endl;
}
```

4.程序运行与测试



5.实验总结与心得

除了以上的使用递归的方法，递推也是完全一样的，这一点笔者在 1345 中提出过。依然是状态转移方程，不过不同的情况有不同的状态，如下：

```
if(i==0){
    dp[i][j]=1;
}else if(j>i){
    dp[i][j]=dp[i-1][j]+dp[i][j-1];
}else if(i==j){
    dp[i][j]=dp[i-1][j];
}
```



可以发现状态转移方程都是一样的，只不过形式不同罢了。

但是，

也有例外，这里是笔者的一个比较困惑的点，在网上搜到了一个算法但是不是很理解，采用的状态转移方程是对称状态乘积，如下：

```
for (int i = 2; i <= 30; i++) {
    dp[i] = 0;
    for (int j = 0; j < i; j++) {
        dp[i] += dp[j] * dp[i-1-j];
    }
}
```

此题亲做，有图为证：

 http://soj.sysu.edu.cn/problem_list.php		
译_有道		
	10359	Valuable Jewellery
	4313	Boys and Girls
	5232	Flood
	13062	SubDiagonal Paths
	17956	Maximum Multiple
	2712	继承与多态
	6767	Making Decisions

附录、提交文件清单

15352015-caogj-13062-v0;

13062 (1) .cpp;

13062 (2) .cpp;

13062 (3) .cpp;