# 黛黛进贾府——day1 题解

## 前缀？

（a.cpp/c/pas）

Time Limit:1 Sec　Memory Limit:128 MB

### 题意简述：

求 N 位的由‘0’、‘1’、‘2’组成的所有前缀都满足 num2（‘2’的个数）≥num1≥num0 的字符串个数。

### 解题思路：

这题是很简单的 DP。设 f[i][j][k]表示 i+j+k 位的字符串的 num0=i，num1=j，num2=k，易得转移方程为 f[i][j][k]=f[i-1][j][k]+f[i][j-1][k]+f[i][j][k]。边界条件自己处理一下即可。这样是 80%的数据，100%的数据滚存一下即可。

## 前缀！

（b.cpp/c/pas）

Time Limit:3 Sec　Memory Limit:256 MB

### 题意简述：

动态维护多重前缀和。

## 解题思路：

这道题数据范围与 M 有关。

我们可以知道 $f_i(j)=f_i(j+1)-f_{i-1}(j+1)$。

于是我们再把 $f_i(j+1)$ 和 $f_{i-1}(j+1)$ 化开来，得：$f_i(j)=f_i(j+2)-2f_{i-1}(j+2)+f_{i-2}(j+2)$

继续化：$f_i(j)=f_i(j+3)-3f_{i-1}(j+3)+3f_{i-2}(j+3)-f_{i-3}(j+3)$。接下来，我们可以发现等式右边部分的系数为（1,-3,3,-1），一正一负，同时，不管符号的话，就是杨辉三角的某一行。这样，我们可以继续把 $f_i(j)$ 化开来，直到 $f_i(j)=\sum(a_x*f_{i-x}(n))$。

这样，我们只要维护 $f_i(n)(-n\leq i\leq m)$ 即可，那么问题来了，我们怎么动态维护这个呢？我们可以考虑每次 Add 操作对 $f_i(n)$ 贡献，发现规律与杨辉三角类似，变通一下即可。

这题的效率为(N+Q)*(N+M)。

当然还有 Q*Q 的更简洁、时空复杂度更优的方法，但傻逼的我没有想到。

# 前缀……

（c.cpp/c/pas）

Time Limit:1 Sec　Memory Limit:128 MB

## 题意简述：

动态维护多重前缀和。

## 解题思路：

以前做过一题求前缀和的前缀和，我从那题受到启发想到了这题，但应该有比我的方法更好的方法。

对于 15%的数据，一棵树状数组动态维护前缀和。

对于 40%的数据，两棵树状数组：一棵维护前缀和，一棵维护 Ai*(n-i+1)的和，答案就是 Tree2[i]-Tree1[i]*(n-i)。

对于 60%和 80%的数据，其实是差一点就想到标算了，方法和标算类似，这里不再赘述。

考虑满分算法，由 15%和 40%的方法可以猜想，维护前缀和用一棵树状数组，维护前缀和的前缀和用两棵树状数组，那维护 10 重前缀和，是不是用 10 棵树状数组呢？如果真的是用 10 棵树状数组，那维护什么呢？这时我们很容易想到 Ai*(n-i+1)中的(n-i+1)，这个数的意义是什么呢？它表示第 i 个数对第二重前缀和的最后一位的贡献（即加了几次）。

我们假设前四个数分别为 a、b、c、d，要维护的是三重前缀和。

| | | | | |
|---|---|---|---|---|
| a | b | c | d | 贡献 |
| a | a+b | a+b+c | a+b+c+d | 1 1 1 1 |
| a | 2a+b | 3a+2b+c | 4a+3b+2c+d | 4 3 2 1 |
| a | 3a+b | 6a+3b+c | 10a+6b+3c+d | 10 6 3 1 |

有没有看出什么规律来呢？如果用 num[j][i]表示第 j 层（即第 j 重前缀和）时，Ai 对第 j 层的最后一位的贡献的话，num[j][i]=num[j][i+1]+num[j-1][i]。我们就可以很容易地求出每个数的贡献，O(n*10)预处理出来。这样思路就明朗了：第 j 棵树状数组维护 Ai*num[j][i]的和。

知道了怎么维护，接下来就是想怎么求了。我们用 Ans[j][i]表示第 j 重前缀和的第 i 个数（即答案）。同样，我们来思考 Tree2[i]-Tree1[i]*(n-i)中(n-i)的意义，它表示前 i 个数多算的贡献，我们能否也像 num[j][i]一样打出一张表 f[j][i]呢？我们以上表为例，进一步去寻找规律。

我们把上表的字母去掉，只剩下系数。

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 贡献 |
| 1 | 1 1 | 1 1 1 | 1 1 1 1 | 1 1 1 1 |
| 1 | 2 1 | 3 2 1 | 4 3 2 1 | 4 3 2 1 |
| 1 | 3 1 | 6 3 1 | 10 6 3 1 | 10 6 3 1 |

发现 Ans[j][i]的系数是 Ans[j][i+1]的系数的后缀。

Ans[3][2](3 1 0 0)有一种求法：

（10 6 3 1）-（4 3 2 1）=（6 3 1 0）

（6 3 1 0）-（3 2 1 0）=（3 1 0 0）

考虑到对于 Ans[j][i]来说，若 k>i,则 Ak 对 Ans[j][i]的贡献必然为 0。这样的话，Ans[j][i]的值就只与 Tree[k][i](k≤j)有关了。

由于上表太小找不到规律，我再画大一点。求 Ans[5][2]（5 1 0 0 0），即（5 1）。

| | |
|---|---|
| 1 1 1 1 1<br>5 4 3 2 1<br>15 10 6 3 1<br>35 20 10 4 1<br>70 35 15 5 1 | (5 1) = (70 35) - (35 20) - (20 10) - (10 4)<br>(20 10) = (35 20) - (15 10)<br>(10 4) = (20 10) - (10 6)<br>(10 6) = (15 10) - (5 4)<br>即（可能有点长）： |

(5 1)=(70 35)-(35 20)-((35 20)-(15 10))-((35 20)-(15 10)-((15 10)-(5 4)))
=(70 35)-(35 20)-(35 20)+(15 10)-(35 20)+(15 10)+(15 10)-(5 4)
=(70 35)-3(35 20)+3(15 10)-(5 4)

如果你再多举几个例子就会发现：这和容斥原理有些类似，一加一减。

再举一个例子，还是 5x5 的方阵，求 Ans[5][1]（1 0 0 0 0），即（1）

(1)=(70)-4(35)+6(15)-4(5)+1(1)，其中

4=4；6=3+2+1；4=2+1 +1；1=1；

不知不觉已经说那多了，那我再举一个更大的例子

5=5；4+3+2+1=10；3+2+1 +2+1 +1=10；2+1 +1 +1=5；1=1

你是否有所感悟？似乎是一个递归函数 f(n,m)=f(n-1,m)+f(n,m-1)。

而边界条件是 f(1,m)=m，f(n,0)=0。

别问我是怎么想到的，我想了一晚上。其实这个函数，只是杨辉三角的变形 T_T。

我们可以打一张 f[j][i]的表。那么——

Ans[j][i]=Tree[j][i]-f[1][n-i]*Tree[j-1][i]+f[2][n-i-1]*Tree[j-2][i]……

接下来的一切都迎刃而解了。