## A – lmh不会写高精度

这道题要拿60分就是大概A+B Problem的难度。

没想到一些选手，包括比较厉害的选手，居然栽在了这道题上。

什么没打换行啊，没清空数组啊，居然还有编译不过的。

也是让我有点惊讶。

祝贺龙实初一庄邦宇和金实初三李根AC本题。

### Case 1-6

1≤a,b≤1000。

直接按照题意模拟即可。

复杂度：O(t)。

### Case 7-10

1≤a≤1000，1≤b≤10100，1≤t≤100。

算法1：

注意到a不超过1000，所以当b>2000000时，结果一定是0.000。

读入整数a和字符串b，如果b的位数过大则直接输出0.000，否则将b转换为整数计算。

转换的时候可以用sscanf函数。

复杂度：O(t)。

算法2：

高精度除法。

复杂度：O(tlg2b)。

## B - lmh得到了一个网站

这一道题是排序的模板题。

本来这道题不怎么难的，结果出来了我也很意外，还很担心自己数据出错了。

然后就把各位的程序都看了一遍，发现很多人都错在同一个细节，让我哭笑不得。

王佳栋写了个hash，不过我看不懂（逃），也不清楚是什么意思。

张周骁写了和别人差不多的算法，不过不知道为什么有10分。

总之，B题比ACD都惨（悲）。

### Case 1-6

1≤n≤500。

直接模拟。

注意只要比较该选手有没有在之前的选手中出现过即可，减掉的时候也只减一次，不用两两比较，否则会重复。

大部分的选手都出现了这个问题。

for(i=1; i<=n; i++){

flag=true;

for(j=1; j<i; j++)

if(a[i]==a[j])

flag=false;

if(flag) count++;

}

复杂度：O(n2)。

### Case 7-10

1≤n≤50000。

算法1：

注意到整个过程其实就是一个去重的过程。

对所有的选手排序一遍，然后遍历一遍数组去重即可。

在这里可以用algorithm库中的std::sort和std::unique。

复杂度：O(nlogn)。

算法2：

当然，像王佳栋dalao那样写个hash，或者写个trie也没问题。

复杂度：O(n)。

## **C – 被厌恶者的哲学**

C题这次的确出得有点过难了，在这里给各位选手道歉，毕竟历年的NOIP普及T3基本都是些模板题和一些简单的动规嘛。

不过我部分分还是有给的，给了30分的贪心分和30分的暴搜分，还有5分的水分，结果却没什么人拿。

### Case 1

n=3，m=1。

很明显，直接输出a+b+c即可。

复杂度：O(n)。

### Case 1, 2, 7, 8, 13, 14

1≤n, m≤50，保证同一个活动的遗忘值单调不上升。

因为保证同个活动持续时间越长收益越低，因此可以采用贪心的策略。

选择第一天获得的遗忘值最大的活动P，持续进行这个活动，直到有其他的活动第一天获得的遗忘值超过这个活动的效益。

就是说，不停地进行活动P，直到烦了，进行另一个活动一天，然后再回来继续进行活动P。

就像lmh刷题刷烦了就去玩一会儿slay.one，玩完了再过来刷题一样。

复杂度：O(n)。

### Case 1-6

1≤n≤20，1≤m≤2。

暴搜，枚举每天进行的活动。

复杂度：O(mn)。

### Case 7-12

1≤n≤50，1≤m≤2。

动规。

设f[i][j][k]表示在第i天，连续进行j天活动1，连续进行k天活动2，所能得到的最大遗忘值。

推式子即可。

复杂度：O(n2m)。

### Case 13-20

1≤n,m≤50。

容易注意到上面那个动规的状态有很多是多余的，因为我们不可能同时持续进行多项活动多天。

设f[i][j][k]表示在第i天，连续进行j天活动k，所能得到的最大遗忘值。

答案就是max{f[n][x][y]}，其中1≤x≤n且1≤y≤m。

状态转移方程：

f[i][j][k]=f[i-1][j-1][k]+a[k][j] (当j>1时)

f[i][1][k]=max{f[i-1][x][y]}+a[k][1] (1≤x<i 且1≤y≤m且y≠k)

当j>1时，共有约O(n2m)个状态，而这个方程是O(1)的，因而总时间复杂度为O(n2m)。

当j=1时，共有O(nm)个状态，而这个方程是O(nm)的，因而总时间复杂度为O(n2m2)。

综上，复杂度：O(n2m2)。

### Extra

1≤n≤50,m≤5000。

这一部分数据点我本来想出的，不过觉得这道题本身就很难，因此就不出了，各位学有余力的可以试试。

设

maxf[i][j]=max{f[i][x][j]} (1≤x≤i)

可以用O(nm)✕O(n)=O(n2m)的时间求出所有的maxf[i][j]。

而第二个状态转移方程可以写成

f[i][1][k]=max{f[i-1][y]}+a[k][1] (1≤y≤m且y≠k)

这样就可以用O(nm)✕O(m)=O(nm2)的时间求出所有的f[i][1][k]。

这样朴素的实现，复杂度为O(n2m+n2m+nm2)=O(n2m+nm2)。

注意到第二个状态转移方程可以用线段树等数据结构优化。

线段树优化后的时间复杂度为O(nm)✕O(logm)=O(nmlogm)。

综上，复杂度为O(n2m+n2m+nmlogm)=O(n2m+nmlogm)。

## D – lmh必须继续努力

OLE 对象