Noip2016 解题报告

——By shenben

前言：

吐槽一下本次 noip。

NOIP2016 比 NOIP2015 的难度不知道涨了几个档次。出题人都丧心病狂了吗？
Day1 T2 主席树+差分
Day1 T3 期望 dp
Day2 T3 状压 dp
至少这三道题目，远超 noip 难度。
(当然并不排除有大牛直接 Ak)

正面评价：

本套试题，难度虽大。
部分分极高。
只要暴力打的好 400+，不是问题。
Zjk 暴力 396
我 264
……

暴力篇

Day1 t1

裸的模拟（不 A，我也没办法）

Day1 t2

裸暴力 25+，加上特判数据，例如一条链的情况，可以拿 65+（后者的前提是，你的代码能力足够好）

Day1 t3

dfs 套 dfs 暴力分居然 76 分（前提是你读懂期望值什么意思）

Day2 t1

杨辉三角或者分解质因数+前缀和（没有前缀和，最多 80 分，有了二维前缀和，就可以 AC）

Day2 t2

骗分可以骗到 45 分+，例如 q=0,用优先队列搞，具体看数据。

Day3 t3

枚举两个点，确定 a,b,在判断可以射掉几个点，dfs 搞一下。50 分+。(本题做时需要冷静)

如果你暴力打好了，并且不手残。（暴力分很良心）
粗略算一下：代码写的丑的话，380-400（SD 前 20）
代码写的漂亮点，400-430（SD 前 10）

正解篇

Day1 T1

就是顺时针转圈，还是逆时针转圈的问题。

Day1 T2

设起点为 u,v；
考虑到路径为树上的链（ lca (u,v)==u || lca (u,v)==v ） 或者经过 lca 的路径（lca(u,v)!= u ,v ），后一种可以统计成前两种情况：f=lca(u,v):
分成: (u,f)和(f,v);
这样之后，就只用处理
对于一个点 x;满足（dep[x]-dep[u]==tm[x] || dep[u]-dep[x]==tm[x]）&& x∈(u,v))的 u 的数量了；
考虑为链，所以如果满足属于，那么路径的终点（或者起点）一定在 x 的子树里
所以你在树上差分：
具体是这样的:
以 lca（u，v）==u 为例；
即 dep[x]==dep[u]+tm[x];
在 v 处打一个标记，使深度为 dep[u]的计数+1；(cnt[dep[u]]++)
在 u 处打一个标记，使深度为 dep[u]的计数-1（cnt[dep[u]]--）
答案统计时，
Ans[x]=cnt[dep[x]-tm[x]];
其实这样是错的，
因为该点的父亲的子树(还有更多的)答案放了进去没有清空(-1 标记在更上层)，统计时就会出错。
如何解决？
简单：进去该点时，先减去 cnt[dep[x]-tm[x]]即可，后面再+= cnt[dep[x]-tm[x]];
答案的增加只会在子树内进行，所以增加的点一定是有效的；
Lca(v,u)==v 时同理。

Day1 T3

思路很明了就是 floyd+dp，时间复杂度是 O(v^3+n^2)，空间复杂度 O(v^2+n^2)
可以滚动成 O(v^2+n)，考场上就没去管了
dp 思路:
dp[i][j][0]表示到第 i 次课时申请 j 次并且该次不申请的期望最小值
dp[i][j][1]表示到第 i 次课时申请 j 次并且该次申请的期望最小值
这个记录方式好处在于可以知道上次是否申请从而判断该次路径的起点
唯一的难点在于这个有点大的动态转移方程

```
dp[i][j][0]=min(dp[i-1][j][0]+(DB)f[c[i-1]][c[i]],
                dp[i-1][j][1]+K[i-1]*(DB)f[d[i-1]][c[i]]+(1-K[i-1])*(DB)f[c[i-1]][c[i]]);

dp[i][j][1]=min(dp[i-1][j-1][0]+K[i]*(DB)f[c[i-1]][d[i]]+(1-K[i])*(DB)f[c[i-1]][c[i]],
                dp[i-1][j-1][1]+K[i-1]*K[i]*(DB)f[d[i-1]][d[i]]+(1-K[i-1])*K[i]*(DB)f[
               c[i-1]][d[i]]+K[i-1]*(1-K[i])*(DB)f[d[i-1]][c[i]]+(1-K[i-1])*(1-K[i])*(D
               B)f[c[i-1]][c[i]]);
```

Day2 T1

杨辉三角或者分解质因数+前缀和
建议用：杨辉三角+前缀和
主要是一个组合数的递推公式 c(n,m)=c(n-1,m-1)+c(n-1,m)，然后用前缀和优化一下就可以 AC 了
预处理时间复杂度为 O（2000*2000），单次查询是 O（n）（貌似还有优的，O（1）），完全可以过

Day2 T2

读入一大堆蚯蚓，然后用堆把他们排序好，从小到大分别为 x1 x2 x3 x4....
第一只被切的蚯蚓显然是 x1 把他切成 y1 和 z1 此时再把 y1-q 和 z1-q 放到堆里面 我们假设 yi>=zi
第二只被切的蚯蚓显然是(x2+q)或 y1 切出来的是 y2=(x2+q)*u/v z2=... 或者是 y2=y1*u/v z2=...
然后再把 y2-2q z2-2q 放到堆里面去
我们比较一下 y1-q 和 y2-2q 假设切的是(x2+q)且 y1-q=y2-2q
就是 x1*u/v-q=(x2+q)*u/v-2q
化简一下 x1-q=(x2+q)-2q 显然只有 x2=x1 时成立
而 x2<=x1 所以 y2-2q<=y1-q
再看一下切的是 y1 的情况 y2=y1*u/v-2q 且 u/v<=1 这时候 y2 显然小于等于 y1-q
所以！ 切出来的 y1 y2 y3 y4 y5..... 是递减的！
同理 z1 z2 z3 z4 z5.....也是递减的
于是我们只要开三个队列 分别维护 xi yi zi 就可以了
最后再把三个队列归并一下 时间复杂度 O(n+m)

Day2 T3

状压 DP，对于每只猪 1 和 0 表示是否被打掉了
设 f[s]为当前状态的最小步数
我们知道，三个点可以确定一个抛物线
已知一个点是原点，那么再来两个点就可以确定一个抛物线，设点 i 和点 j 确定的抛物线表示为(i,j)
每次枚举一个状态 s，再枚举两只猪 i,j，当然 i,j 不在 s 里面

那么设所有经过(i,j)这条抛物线的点的集合为 s’
那么 f[s?s′]=f[s]+1,f[0]=1
那么现在还需要预处理的就是每个集合 s’
用 a[i,j]存放抛物线(i,j)所经过的点的集合
那么转移方程变成了
f[s|a[i,j](i,j 不∈s)]=min(f[s]+1),f[0]=1
时间复杂度 O(2^n+n^2)

代码详见：

https://www.cnblogs.com/shenben/p/6132085.html