联赛层次水平测试

author: paper_

前言

本场比赛四道题均为原创题，出题人水平有限，所以不满意的话请骂轻一点 /kt/kt。

T1 火（fire）

签到题。

可以分别计算每一棵树对期望产生的贡献。

发现 k 秒后可以点燃的第 i 棵树的树所在区间是 $[\max(1,i-k),\min(i+k,n)]$，不妨记作 $[l_i,r_i]$。

容易得到第 i 棵树燃着的概率为 $1-\prod_{j=l_i}^{r}(1-a_j)$，其中 a_i 是第 i 棵树最初燃着的概率。

这样就可以很容易做到 $O(n^2)$。

接着发现所有区间的左端点或右端点都是单调不减的，所以双指针优化即可。

这里还要注意一个点，由于 0 没有逆元，所以直接单独开一个变量维护区间中 1 的数量，特判一下就行。

时间复杂度 $O(n\log n)$

T2 捐赠（donate）

暴力直接排序就行，不讲。$y=1$ 的情况可以发现 k 每次至多加一，可以平衡树加一个指针来维护。

来看正解吧：

首先透露一下，本题的灵感来源于范德蒙恒等式：

$$\binom{n\times 2}{n}=\sum_{i=0}^{n}\binom{n}{i}^2 \tag{1}$$

关于这个式子的组合意义相信你会这道题之后一定能搞懂，所以在这里不多赘述。

令 A 类物品的总个数为 cnt_A，B 类总个数为 cnt_B。则你可以发现**选择的 A 类物品个数加没选择的 B 类物品个数和固定为 cnt_B**。

因此，转变一下思路：将 B 类物品的贡献改为所有 B 类物品的贡献减去不选则的 B 类物品的贡献。用权值线段树记录所有 A 类物品的权值和 B 类物品权值的相反数，求最大的 cnt_B 个数的和，再加上所有 B 类物品的总贡献即可。

复杂度 $O(m\log v)$ 其中 v 为值域。

T3 染色（color）

或许是通过人数最少的一道题?

Sub 0

直接暴搜枚举出所有的情况，直接判断就行。

Sub 2

~~Sub1 出题人也没想到 n^3 的解法。~~

考虑树形 dp。

单独考虑一棵子树，染色完之后要么只剩下单一颜色的叶子节点还没有遇到染色了的祖先，要么子树内所有叶子节点都已经找到了对应的祖先。

$dp_{u,i}$ 表示 u 为根的子树中仅存在某些颜色为 i 的叶子节点还没有遇到染色祖先的情况数，$i=0$ 时表示全部都遇到了。

考虑转移。

$$\forall i\in[1,m]\ dp_{u,i}=\prod_{x\in son_u}(dp_{x,0}+dp_{x,i})-\prod_{x\in son_u}dp_{x,0} \tag{2}$$

意思就是叶子节点可以被染色完，也可以剩下 i 颜色。最后减去所有都被染色完了的情况。

$$dp_{u,0}=(m+1)\prod_{x\in son_u}dp_{x,0}+\sum_{i=1}^{m}dp_{u,i} \tag{3}$$

前一部分很好理解，就是所有儿子节点都没有剩下的未匹配的颜色，所以当前一位随便填。后面一部分代表再 $dp_{u,i}$ 的所有情况中把 u 从不染色变为染 i。

答案就是 $dp_{1,0}$。

Sub 3

其实跟 Sub 2 一样，直接特判一下重新写个 $dp_{i,0/1}$ 的树形 dp 就行。

Sub 4

发现只有子树内出现过 i 颜色的叶子 $dp_{u,i}$ 才不会等于 0，所以考虑线段树合并（不知道启发式合并行不行，不过可能有点卡常）。

这里会有一个问题：我们处理 $dp_{u,i}$ 时是先计算 $\prod_{x\in son_u}(dp_{x,0}+dp_{x,i})$ 再统一减去 $\prod_{x\in son_u}dp_{x,0}$，这会导致所有 $dp_{u,i}$ 在中间的计算过程中都会有值。

我们可以考虑对于每一棵子树记录一个 $sameval_u$，表示所有 u 子树内不存在的颜色当前的 $dp_{u,i}$ 的值，显然它们都是一样的。此时线段树上所有空节点的值就可以用 $sameval_u$ 代替。

当然，$dp_{u,0}$ 要单独开个数组维护。

T4 孤独旅者（loneliness）

比较容易发现追忆边就是不同强连通分量间的连边，而期许边就是同一个强连通分量内的边。

由于可以走任意多次，而且经过一条边时还不一定写游记，所以同一个强连通分量内的所有点都是等价的，所以直接缩点，此时原强连通分量内的边就变成了这个强连通分量所代表的点的自环。

容易发现 A 强连通分量内的期许边的贡献（即可描写地点个数）就是 A 强连通分量可达的点的个数，令其为 a_A。而 A 强连通分量到 B 强连通分量的追忆边的贡献其实就是 A 中向 B 有连边的点的数量，令其为 $w_{A,B}$。

- 先考虑如果经过追忆边时**一定**会写下一篇游记。

问题就转化为每个点数不为一强连通分量初始向自己连边权为 a_i 的自环，i 强连通分量向 j 连边权为 $w_{i,j}$。每次询问就是给定初始强连通分量和终点强连通分量，求走 k 步到达终点的所有路径的贡献和，每条路径的贡献为经过的所有边的贡献乘积。哈哈实际上这就是个自动机。

这很容易 $\mathrm{O}(n^2k)$ 暴力按拓扑序 dp 解决。当然，因为 k 过大考虑矩阵加速，$\mathrm{O}(n^3\log k)$ 解决。由于在 DAG 上 dp 矩阵是个下三角，所以带了个 $\frac{1}{6}$ 的常数，复杂度没有问题。

此时问题就出在了经过追忆边是也有可能不写游记。笔者一开始想过方案中经过的追忆边数量只有 $\mathrm{O}(n)$，能否考虑对于 $k\sim k+n$ 都算一遍，这不失为一个好的思路，但是笔者最终并没有从这方面想到正解，感兴趣可以尝试一下。

考虑一下问题产生的变化，其实就是经过追忆边时有可能只产生乘 1 的贡献并且不消耗 k 篇游记的次数。

改了之后你依然可以直接 $\mathrm{O}(n^2k)$ dp 得到 30 分，不过要得出正解，显然是逃不开矩阵的。那么变换后的矩阵怎么求呢？显然不可能直接本层转移，矩阵也不支持这个操作。欸！那我们能否在上一层就把本层转移的事办好？

可以！具体来说，若原本 i 到 j 有 w 倍的转移权值，问题更改后就直接改成 i 到 j 的所有可达点都加上 w 倍的转移权值。

这样就又有一个问题，初始情况怎么办呢？显然初始的时候要将 s 的所有可达点都当作起点。这其实也很好办，先求出转移矩阵 A^k，然后对于不同的 n 种 s 构造初始矩阵，然后直接全部乘上 A^k。由于初始矩阵只有 $\mathrm{O}(n)$ 个非零点，所以复杂度是 $\mathrm{O}(n^3)$，总复杂度为 $\mathrm{O}(n^3\log k)$，可以通过。