Multiple School Training by WJMZBMR 题解

HFLS WJMZBMR

Contents

题解可能比较简略，有问题可以参考代码。

1 A:Life Game

定位：难题

考虑死人为染黑色，幸存为染白色。

考虑最小割建模。对每个位置建一个点u.

点跟源S联通表示该点是黑色，不然是白色。

那么首先连边$S->u:b_u, u->T:w_u$.表示染黑色就要割掉白色的收入，反之亦然。

那么考虑一个矩阵，首先我们为该矩阵建一个点r，代价为c。假设需要该矩阵内全为白色，不防令矩阵内一点为x，那么如果x是黑色，跟S联通，那么代价c就要被割掉。

我们可以从x向r连一条容量为无穷大的边，同时从r向T连一条容量为c的边。

需要全为黑色也类似建图。

但是这样图的边的数量会很多。

我们注意到这里每次要从一个点连边无穷大的边到一个矩阵的所有点。我们可以使用二维RMQ来优化。

二维线段树优化也可以过。

这样复杂度就可以接受了。具体可以参考代码。

2 B:Reincarnation

定位：中等题

Sol 1

我们先用hash拿出所有字串，考虑一个子串t,假如它分别在位置$[l_1, r_1], [l_2, r_2], [l_3, r_3], ..., [l_k, r_k]$出现了.

那么我们来它考虑在哪些询问中出现了，我们可以枚举某询问第一个包含的是出现的哪次，比如是$[l_i, r_i]$。

那么该询问就要满足$[L, R] : L > l_{i-1p}, L \leq l_i, R \geq r_i$.我们可以根据这个进行预处理，那么问题就可以解决了。

复杂度$O(n^2) + O(1)$。

Sol 2

当然一个更加粗暴的做法就是枚举子串的开头l，然后一个个把后面的字符加入一个后缀自动机。

复杂度$O(n^2) + O(1)$

Extra

很残念的是因为我Q太小了。。。导致很多人以为$O(nQ)$就能过。。。

其实是这样的，一开始我的范围是$n = 5000, Q = 25000$，因为我觉得$O(n^2)$嘛，5000正好。

但是发现这样时限比较吓人，于是后来就同比放缩了一下变成$n = 2000, Q = 10000$了。忘记了考虑可能产生误导的问题。

不过如果你仔细算一下，可以发现$O(nQ \log n)$的做法是过不了的，$O(nQ)$的话常数又太大了。

3 C:Crime

定位：中等题

我们来考虑一下，首先可以注意到6和12是等价的，因为每个数有意义的只是它的素因子有哪些。

同时比如$n = 28$，我们可以发现对于一个> 14的大素数p，它跟任何数互质，也就是说它跟1是等价的。

那么我们在状态中对每个数的等价类，只记录它的个数，并且也记录最后一个数是哪个等价类。

可以发现状态只有几千万级别，可以接受了。

当然就算慢一点的其他做法，只要你内存开的下，就可以打表。

4 D:Endless Spin

定位：难题

我们注意到令L_i为进行多于i步的概率。

那么答案等于$\sum L_i$。

我们来考虑怎么算L_i，进行多于i步也就是说，前i步没有覆盖完所有的位置。

那么我们可以使用容斥原理来计算，我们枚举哪些位置的集合不被覆盖，那么整个就被分成了很多段。

那么每个选择区间就要在不被覆盖的连续段内。设每次的概率为p。那么就要给L_i根据集合的奇偶性加或减p^i。

注意到对每个i，奇偶性都是一样的。那么就是要给答案加上$\sum p^i = \frac{1}{1-p}$。

那么我们注意到这里可以dp，我们令状态为$dp_{i,j,k}$表示考虑了前i个，不覆盖的集合奇偶性为j，概率和为k的。

那么就可以了。注意到这里精度要求有点高，C++就手写个高精度吧，Java就用BigDecimal好了。

话说为什么精度要求这么高呢，其实一开始精度要求只有6位，但是如果那样的话，在本地随机非常多次然后取平均值就能得出结果了，为了防止这种做法，精度要求就变高了。

（可能比较像TC某题，但TC那题是这题改了下来解决打表方面的问题出的。这题是先出的。）

5 E:JZPTREE

定位：难题

注意到我们可以将n^k表示为$n^k = \sum_{i=0}^{k} A_i \binom{n}{i}$。

那么我们可以转化一下问题，对每个点求出$\sum_v \binom{dist_{u,v}}{i}$。

而这个是很容易在$O(nk)$的时间内求出的。

6 F:Jinkeloid

定位：难题

首先我们不妨预处理cnt_{set}表示只有set内部的数字出现了奇数次的前缀的数量是多少。

那么对于每一个询问$S = c_1, c_2, ..., c_k$，考虑它的每一个子集$s = a_1, ..., a_t$。如果有C个前缀在S中只包含s中的询问奇数次，那么答案就要加上$C * (C - 1)/2$。

考虑到这一点，我们先通过一次dp，算出sum_{set}表示set内部的数字出现了奇数次的前缀的数量是多少。sum跟cnt的不同是前者是只有。

这个通过一个简单的dp就能实现：

```
for (int i = 0; i < MAX_LETTER; ++i) {
    for (int j = 0; j < (1 << MAX_LETTER); ++j)
        if (j >> i & 1) {
            cnt[j - (1 << i)] += cnt[j];
        }
}
```

那么接下来我们对于每个S的子集s，通过查sum，可以知道s出现了奇数次的前缀的数量，由于我们需要知道的是只有s出现了奇数次的数量。考虑到按大小从大到小计算，对于每个s，现在不合法的实际上就是s的所有超集的结果和（s出现了但并不只是s出现了也就是说还出现了其它的）。减去即可。

7 G:The Unsolvable Problem

定位：签到题

先特判$n=2$的情况。然后如果$n=2k+1$，那么答案显然是k和$k+1$。

如果$n=2k$，那么进一步讨论，如果k是奇数，那么答案是$k-1$和$k+1$。

否则答案是$k-2$和$k+2$。

8 H:Pieces

定位：简单题

很容易的题目，我们用dp_{set}表示还剩下set这个集合的子串，还需要的最小步数，然后进行$O(3^n)$的dp就可以了。

9 I:Burning

定位：中等题

比较标准的题目，n给的很小，所以很多做法都可以解决这个问题。

比如我们可以求出所有的交点，然后依次处理两个相邻交点之间的情况，它的内部只有梯形和三角形，比较好处理。这样是$O(n^3 \log n)$的。可以参考kac.cpp。

或者也可以直接切凸多边形，那么是$O(n^4)$的。可以参考标程。

范围很小，精度也很宽松，simpson估计也可以过。

10 J:No Pain No Game

定位：中等题

这题挺容易的，我们考虑从左往右扫描，对每个数i，我们记录$prev_i$表示i的倍数中，在当前位置之前，最靠右的在哪个位置。那么考虑当前数a_i的所有约数x，对于所有$r \geq i$,$l \leq prev_x$的询问，x都是可能的答案了。

那么不妨使用一个树状数组来维护，然后每次回答r在当前i的所有询问，就能解决这个问题了。

11 K:Sad Love Story

定位：中等题

由于数据是随机的，所以做法就比较多了，这题蛮有趣的。

其实一开始的标程是在线并且可以解决任意数据的，但是数据比较难构造，而且难度也太大了，所以就改成随机数据了。

Sol 1

容易发现答案其实下降的很快，所以下面的方法效果很好。

用一个set保存所有点按照x排序的顺序，然后根据插入点从x的近到远更新结果，一旦不可能更新答案了就跳掉。

因为答案下降的很快，并且点是随机分布，所以更新的点其实没几个。

虽然不怎么好证明，但复杂度应该小于$O(n \log n)$。

Sol 2

我们也可以套用KD树的经典做法，但这里速度其实挺慢的，如果要通过可能需要点常数优化。

复杂度不好说。

Sol 3

这个做法就比较有趣了，首先我们先生成所有点，然后运行一次求平面最近点对的算法。

不防设他们是p_i和p_j，$i < j$，注意到点都是随机的，所以i和j也当然是随机的了。

那么我们可以发现p_j之后的答案都已经确定了，那么我们现在就要求出$j-1$之前的答案。

注意到此时变成了子问题，由于i, j是随机的，每次平均可以由$O(n)$的问题规模变成$O(\frac{2}{3}n)$的问题规模。

那么复杂度就是$O(n \log n)$也就是求平面最近点对的复杂度。