牛客暑期ACM多校训练营

第六场 - sd0061

牛客网
NOWCODER

A. Singing Contest

由于每个选手的策略都是尽可能赢，所以他该认输的时候只能认输。

能赢的时候只要选权值大于对方最大值的最小值，大的留在后面不会更差。

直接模拟即可。

B. Endless Pallet

设 x_i 为第 i 个节点第一次染成黑色的时间，所求即 $E(\max\{x_i\})$

由 $E(X) = \sum_{k=0}^{\infty} P(X \geq k)$

$$P(\max\{x_i\} \geq k) = 1 - P(\max\{x_i\} < k)$$

$$P(\max\{x_i\} < k) = \Pi_i P(x_i < k)$$

将 Π 展开后，所求式子就是 $2^n - 1$ 个等比数列求和，每一项的公比为每次操作某一个节点集合不会被染成黑色的概率，这样我们就正着推出了容斥做法。

B. Endless Pallet

每个等比数列只在乎它的公比（不被染色的概率）以及容斥因子（±1）。

每个等比数列对应着树上结点的一个非空子集 S，不被染色的概率即不经过集合内部节点的链数除总链数，容斥因子即 (-1)^|S|

所以可以直接 dp，设 f[i][j][k][2] 表示 i 这棵子树，现在与根节点相连的不在子集 S 内部的节点构成的连通块大小为 j，k 表示子树内部已经有 k 条不经过子集 S 内部结点的链，以及子树内 S 大小的奇偶性。

复杂度看上去很像 O(n^7) 但实际上是 O(n^5) 除很大的常数。

C. Generation I

简单排列组合题

考虑枚举最后有 k 种颜色，那么有 $A_k^m = \dfrac{m!}{(m-k)!}$ 种排列它们的方法。

由于每种操作对一个后缀有影响，区分方案只要考虑第一个被影响的位置即可。

n 个位置放 k 种球，每个位置可以放多个，由隔板法方案数为 $\dbinom{n-1}{k-1}$。

所以答案为 $\sum_{k=1}^{\min\{n-1,m\}} \dfrac{m!}{(m-k)!}\dbinom{n-1}{k-1}$

D. Bulbasaur

签到题

给每个 body 配个权值最大的人即可。

E. Charmander

因为 $|S_i| > 1$，考虑单个字母 t 秒生成的串，长度一定不小于 2^t，至多 10 秒之后，单个字母生成的串长度就不小于目标串 T 了。

不妨先处理答案不大于 10 秒的情况，对 T kmp 之后可以递推出 f[i][j][k] 表示 i 这个字符经过 j 秒变成的串，从 T 的 kmp 构成的 DFA 状态 k 进去后，出来的是哪个状态。这样就可以知道 S 串经过 k 秒后的串从 DFA 的初始状态进入，出来的是不是终态了。

那么考虑最终 T 第一次出现的位置开始回溯。最早要么是某个单个字符，要么是两个相邻的字符。且是在它 10 秒之内生成的。这里的某个单个字符和某两个相邻字符不一定是初始串中，可能是初始串经过若干秒的变化后得到的。所以还需要一个 BFS 来求出每个字符以及每个相邻的字符对第一次出现的时间，然后再用之前求出的 f[i][j][k] 判断即可。

F. Squirtle

f[i][0/1] 表示以 i 为根的子树能得到 0/1 的最大数量，答案即 f[1][1]。

考虑转移，容易证明当前节点的 f[][0/1] 由儿子的 f[][0/1] 转移来不会更差。

这里我们仅对异或非运算符进行证明，异或运算符的证明与异或非运算符类似，其余的运算符结论较为显然。对于异或非运算符，只有 00 和 11 时值为1，否则为 0。

不妨固定左子树 0 / 1 的数量：
- 若左子树 0 多于 1，显然右子树 0 取最大数量时，f[x][1] 最大。
- 若左子树 1 多于 0，显然右子树 1 取最大数量时，f[x][1] 最大。
- 若左子树 0 等于 1 ，那么不论右子树如何选取，f[x][1] 的值都一样。
- 对于右子树同理。也就是说，上述四种情况中，至少有一种取得最大值。

F. Squirtle

题目的数据范围下需要用到大整数，但实际上平方的多项式乘法就可以通过。

由于 f[x][0/1] 的位数与 x 子树大小是一个级别。

所以由经典的树 dp 复杂度分析可知这样做的复杂度为 O(n^2)。

实现时用 Java 最方便，C++ 可能需要压位才能通过。

FFT 反而是 O(n^2logn)。

G. Pikachu

如果给定了在图 G 上的源点 s 和汇点 t，则割为：

$$\sum_{x\in S}\sum_{y\in T} dist(x,y), s\in S, t\in T$$

dist(x , y) 为在树 T 上 x 到 y 的距离。

因为要求最小割，相当于要求一种节点的染色方案，使得 s 点是白的，t 点是黑的，并且同色节点间的距离之和最大。

显然除了 s, t 两点，其他节点都同色时距离和最大，所以 s 到 t 的最小割即：

$$\min\{\sum_{i=1}^{n} dist(s,i), \sum_{i=1}^{n} dist(t,i)\}$$

所以只需要算出树上每个节点到其他节点的距离之和即可。

H. Eevee

若 $c^z=1$，则有 $a^x=b^y=1$，答案显然为 $(2m)^2=4m^2$，之后不考虑这种情况。

若 $a^x=1$ 或 $b^y=1$，那么答案即为满足 $a^x=c^z$ 的 $<a,x>$组数乘以 $4m$。组数的计算可以枚举 a 或者枚举 x。

剩下的情况我们就有 $a,b > 1$, $x,y > 0$，且 a,b 的质因子都是 c 的质因子。

于是我们对于 c 的所有质因子的指数可以列出一个关于 x, y 的方程，从而得到一个关于 x, y 的一元二次方程组。

H. Eevee

由于 c 不大于 10^5, 最坏情况下 c = 2 x 3 x 5 x 7 x 11 x 13，这时 a, b 最多只会有 1848 种可能的取值。

设 $a = \Pi p_i^{d_i}, b = \Pi p_i^{e_i}, c = \Pi p_i^{f_i}$, 对于每个 i 有 $d_i x + e_i y = f_i$

不妨先枚举 a ，那么就知道了所有的 d_i，再枚举一个 e_1。

这时有两种情况：

1.所有 (d_i, e_i) 之间线性相关，方程有无数整数解。
这时因为已经知道了(d_1, e_1)，就能知道所有 (d_i , e_i)，可以通过 f_i 的值判定是否存在这种情况。如果存在，就可以用扩展欧几里得求出 x,y 的一组解，从而求出所有满足 1 <= x, y <= m 的解。

H. Eevee

2. 方程有唯一解，那么可以枚举第一个和 (d_1, e_1) 不线性相关的 (d_i, e_i)，不妨设是 j，然后再枚举 e_j 的取值，这样就可以解出 x, y ，从而再解出所有的 (d_i, e_j) 并判断这种情况是否满足条件。

由于 (d_i , e_i) 的取值最大为 $\lfloor \log_{p_i} m \rfloor$，最多有 6 个方程，

单组数据复杂度最坏为 $O(5 * 1848 \log^2 m)$ ，实际不可能跑满。

I. Team Rocket

喜闻乐见的简单基础数据结构题。

考虑把每个区间当做平面上的一个点 (l, r)，每次操作相当于是删去所有的满足 (l <= x <= r) 的点，相当于是把横坐标 <= x 的所有纵坐标 >= y 的点删去。

离散化横坐标后建线段树，每个节点用一个 vector 保存所有左端点在区间内，按右端点从小到大排序的点。如果用归并的方法初始化，时间复杂度和保存的线段总数均为 O(nlogn).

每次询问时在线段树上走到 [1,x] 要问的节点，将右端点不小于 x 的线段都从该节点的 vector 中删去。同时删的时候查看该线段是否为第一次从线段树中删除，是则更新答案。

因为线段树中的每个节点最多被删除一次，最终的时间复杂度就是 O(nlogn).

由于每次操作影响的是一个前缀，也可以用常数更小的树状数组实现，但初始化时注意不能多 log。

实现较好的 k-d tree 应该也可以通过。

J. Heritage of skywalkert

由于数据~~看上去像~~是随机生成的，只需要选出前 100 大的数平方暴力即可。

随机两个正整数互质的概率为 $\frac{6}{\pi^2}$。

Thanks

牛客网