20240801 NOIP模拟赛题解

——by $lcrh$

整数（integer）

签，这个题目过于经典了，以至于我觉得应该没有什么讲的必要。

本题中整数4位，小数9位，一共13位精度，使用double 不会直接产生精度误差，当然，也可以当个字符串输入，这样就一定不会有精度误差了。

做法还是很经典的，将所有实数乘上 10^9 转成整数之后，”两个实数乘积为整数“就被转化为了”两个整数的乘积为 10^{18}的倍数“。这个问题就非常简单了。

我们知道 $10^k = 2^k \times 5^k$。因此实际上我们只需要对于所有整数提取其对于质数 $2, 5$ 的质因子次数即可。因此我们可以将 a_i 改写为：$a_i' 2^{p_i} 5^{q_i}$，其中 $\gcd(a_i', 10) = 1$。那么我们要求的答案如下：

$$\begin{aligned} Ans &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} [\min(p_i + p_j, q_i + q_j) \geq 18] \\ &= \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} [\min(p_i + p_j, q_i + q_j) \geq 18] - \sum_{i=1}^{n} [\min(2p_i, 2q_i) \geq 18] \right) \end{aligned}$$

简单重构一下式子，我们就得到了上边这个两项都比较简洁的式子（如果直接拿第一行的式子做当然也是没有问题的，无非是一个三位偏序而已）。

上面的式子中，$\sum_{i=1}^{n} [\min(2p_i, 2q_i) \geq 18]$ 可以非常简单的 $O(n)$ 求出。对于其余部分，做法很多：

- 做法一：我们考虑枚举一个元素 (p_i, q_i)，这样问题就变成了一个简单的二维偏序。我们将所有二元组按照 p_i 排个序解决一维，另一维使用数据结构解决。复杂度 $O(n \log n)$
- 做法二：观察到上面的数据结构没有意义，值域只有 $O(\log n)$ 级别，因此枚举元素其实也是没有意义的，我们直接使用二维前缀和，记 $S(x, y) = \sum_{i=1}^{n} [p_i \geq x][q_i \geq y]$，求出 $S(x, y)$ 之后答案即为 $\sum_{i=1}^{n} S(18 - p_i, 18 - q_i)$。复杂度为 $O(\log^2 n + n)$。

无论使用哪种做法，由于需要提取因子，总复杂度为 $O(n \log n)$。

Grievous Lady（gl）

> 出题人出在洛谷的题目 Luogu P8946

本套题最难的一道。想要在本题拿分，首先肯定要注意的是，由于我们运算的顺序是从左到右，因此每一次我们都会把之前的运算结果作为上指标，新的数字 a_i 作为下指标。这两个数字中，下指标 a_i 的值域位于 $[1,m]$，因此过大的上指标只会生成 0，就上指标而言大于 m 的数字之间没有本质区别，因此不妨将大于 m 的数字统一记为 $m+1$。

我们假设 $f_{i,j}$ 表示填了前 i 位数字，运算结果为 j 的方案数。根据上面的推论，我们直接朴素转移，可以得到 24 分。出题人自称实现精细可以得到 40 分。

如何精细实现呢？其实我们可以发现，除了一些特殊的转移，其他转移的扩张速度是极快的，很容易到达 $m+1$，不妨考虑一下各种转移。

考虑所有转移，对于 A:

1. 所有数字都可以往后接一个更小的数字转移到 0。。
2. 1 可以接下任意数字转移到对应的数字。
3. 0 接下任意数字转移到 1。
4. 其余的转移数量极少。

对于字符 C:

5. 所有数字都可以往后接一个更小的数字转移到 0。
6. 所有数字都可以往后接上自己转移到 1。
7. 所有数字都可以完后接自己 $+1$ 转移到自己 $+1$。
8. 1 可以接任意数转移到该数。
9. 其余转移不多。

这里面的转移有一些会有重复部分（例如 1 C 1 就被 $2,4$ 两条转移到了），需要剪掉。重复部分太少不妨直接归在最后一类的特殊转移内。我们依次考虑每一种转移(将数组 f 转移到数组 g):

1. 求和 $\sum f_i(i-1)$，最后对 g 进行一次单点修改。
2. 全局加上 f_1。
3. 求 f_0，修改 g_1。
4. f 单点查询，g 单点修改。
5. 同 1。
6. f 求全局和，g 单点修改
7. f 整体向右平移一位
8. 全局加 f_1。
9. 同 3。

功能很多很杂，单次操作复杂度至多不能超过 $O(\sqrt{m})$ 级别，个别操作复杂度必须为 $O(1)$。具体来说：

- $O(1)$ 单点修改。
- $O(1)$ 单点求值。
- $O(\sqrt{m})$ 以下全局加法。

- $O(\sqrt{m})$ 以下全局求和。
- $O(\sqrt{m})$ 以下求 $\sum f_i(i-1)$。
- $O(\sqrt{m})$ 以下向右平移一位。

注意这个转移后一列基于前一列但是不完全继承前一列，所以我们还需要支持：

- $O(\sqrt{m})$ 以下全部清零。

大家可以思考以下使用怎样的方法维护。

我们使用一个 $O(1)$ 的数据结构维护，我们保存 DP 数组 f 和一个时间标记数组 t，另外记 $s_1=\sum_{i=0}^{m} f_i, s_2=\sum_{i=0}^{m} f_i(i-1)$，以及一个清零值 v_0，一个清零时间 t_0，整体加的值 v_A，考虑所有操作：

- 单点修改：这里主要是单点加操作，下放标记（如果 $t_i<t_0$，将时间修改到 t_0，f_i 修改到 v_0）之后直接修改数组并更新 s_1, s_2 即可，时间复杂度为 $O(1)$。
- 单点求值：下放标记之后返回 f_i+v_A 即可， $O(1)$。
- 全局加：修改 v_A, s_1, s_2 即可，$O(1)$。
- 全局求和：返回 s_1 即可，$O(1)$。
- 全局求 $\sum f_i(i-1)$：返回 s_2+f_0 即可（注意 s_2 中 f_0 的系数为 -1）,$O(1)$。
- 向右平移一位：修改 f, t 起始的指针位置（可以开一个两倍长的数组，初始把指针指在中间，这样每次直接自减 1 即可），并修改 s_1, s_2 即可，$O(1)$。
- 清零：将 t_0 设为当前时间，v_0 设为 $-v_A$，s_1 和 s_2 均设为 0 即可。$O(1)$。

注意，最后一个运算符需要特殊处理，使用组合数上指标前缀和即可，经典问题，可以 $O(1)$ 求出来。

本题复杂度可以分析到 $O\left(n\sum_{i=2}^{\sqrt{m}}(i!m)^{\frac{1}{i}}\right)$,不过实际上 $m=10^5$ 的时候，两种转移的数量分别只有 $393/1195$ 个，通过还是很简单的。

conflict(cl)

思路很明显的题目。

显然枚举被割开的边是一个很合适的思路，并且每个连通块也恰只有一条合法的割边，因此这样的统计是不重且不漏的。

因此我们需要计算以该边连接的两个点为根，不包含另一个点的大小为 x 的连通块的数量。

我们设 $f_{u,s}$ 表示选出一个以 u 为根的，大小为 s 的连通块的方案数。 $g_{u,s}$ 为选出一个包含 u 的父亲，不包含 u 的大小为 s 的连通块的方案数。（也就是普通的树上背包的换根）。

直接做，求 f 的复杂度是 $O(n^2)$ 的，没有问题，但是求 g 复杂度是 $O(n^3)$ 的。

注意到两个 dp 数组的复杂度差异在于 f 的第二维受子树大小约束。而 g 不受。

然而由于答案是 $\sum f_{u,s} \times g_{u,s}$，只有 $f_{u,s}$ 有值的位置 $g_{u,s}$ 才是有用的。因此我们向子节点换根的时候，可以强行将 g_u 数组截断到 u 的子树大小进行暴力转移，复杂度可以沿用树上背包的复杂度分析，$O(n^2)$。

种树(plant)

一点小小的分讨。

考虑两棵有根树同构的一个必要条件——它们的深度相同。于是，设 T 的直径长度（边数）为 D，其中一条直径为 $a_0 - a_1 - \dots a_{D-1} - a_D$。则以 a_i 为根时，树的深度为 $\max\{i, D-i\}$，于是一共分为了 $\lfloor \frac{D}{2} \rfloor + 1$ 个等价类。

而当我们在不断"加叶子"的过程中，树的直径不会减少，于是 $\lfloor \frac{D}{2} \rfloor + 1$ 就是颜色数量的一个下界。下面以构造的方法说明，这个下届也是可以达到的。和大多数题一样，我们取这棵树的中心（直径中点）为根。当然，首先还是需要根据 D 的奇偶性讨论它是一条边还是一个点。

1. 树的中心时一条边(u, v)。

   此时，将这条边断开后，形成了两棵子树，分别以 u, v 为根。

   记 b_d 为两棵树中，所有深度为 d 的点中，子节点数的最大值。

   然后，我们把每个点都补子节点，知道将每个深度为 d 的点的子节点数都补齐到 b_d。

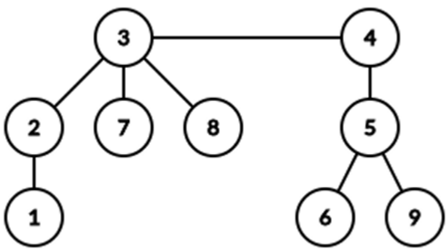
   可以证明，最终得到的树具有高度的对称性，对于每个深度 d，深度为 d 的所有点都处于同一个等价类——它们都可以被染成相同的颜色。

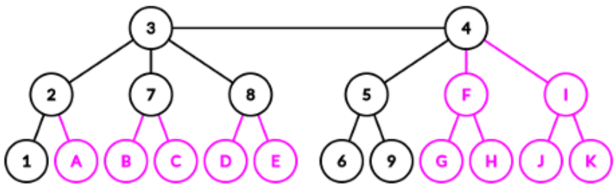
   于是最终不同等价类的数目就等于不同深度的个数，而深度的最大值为 $\lfloor \frac{D}{2} \rfloor$，因此 $\lfloor \frac{D}{2} \rfloor + 1$ 种颜色就足够了。

2. 树的中心时一个点 v。

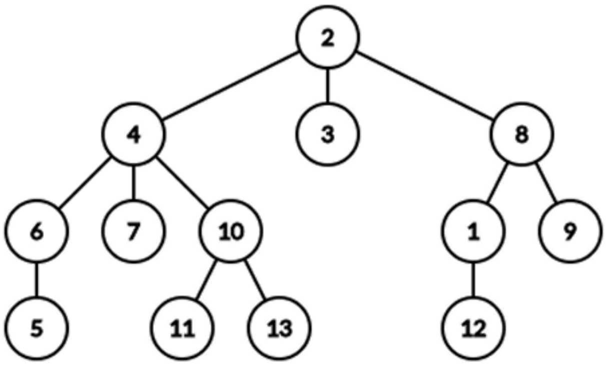
   模仿上面的方法，上面像"边分治"，这里就沿用"点分治"了。

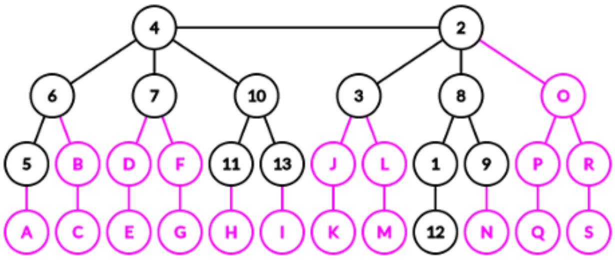
   我们将这个点去掉后，得到若干棵子树，用上面类似的方法，将对应深度的点的子节点数补齐，最终得到的树的颜色数量仍然为 $\lfloor \frac{D}{2} \rfloor + 1$。



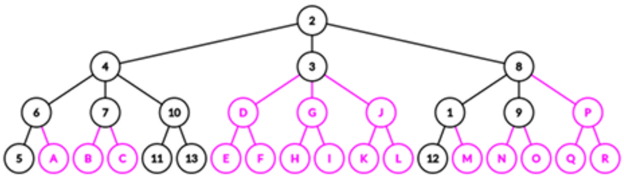


综上，直径为 D 的树的颜色数量最小值为 $\lfloor \frac{D}{2} \rfloor + 1$。接下来考虑，在满足颜色数量为 $\lfloor \frac{D}{2} \rfloor + 1$ 的情况下，叶节点数量的最小值。首先，对于中心时一条边的情况下，直径时不能增大的，否则直径又会变为偶数，从而使得颜色数量变多。因此，我们只能在不改变直径的情况下添加叶节点——如上图所示。由上面的必要条件可知，欲使颜色数量为$\lfloor \frac{D}{2} \rfloor + 1 = \frac{D+1}{2}$，深度相同的点必须等价，也就是说它们的颜色必须相同。由等价的必要条件知，这些点的度数也要相等。由于度数只能增不能减，因此上面所说的构造是必要的，同时也是最优的。在这种情况下，最终的叶子总数就等于每层的度数之积的 2 倍，即 $2b_0b_1b_2 \dots b_{\frac{D-3}{2}}$。对于中心是一个点 v 的情况，直径是可以增大的，且直径增大有可能使得答案变得更优，如：





对于上面这个例子，直接加叶子的结果是下图，共有 18 个叶子，而上图的直径虽然比原图大 1，但是只有 12 个叶子。因此，我们要分两种情况讨论，一种是不增大直径的方法，另一种是增大直径的方法。



对于不增大直径的方法，处理方法和「中心是一条边」的情况基本一样。对于增大直径的方法，容易证明新树的中心边一定以 v 为其中一个顶点。因此我们只需枚举与 v 关联的边，哪条作为中心，然后分别求解，最后再取最小值。每次计算显然是 $O(N)$ 的，如果要枚举边，外边再套一层 $O(N)$，总时间复杂度 $O(N^2)$。

最后简单解释一下为什么答案(以及所有的候选答案)都不会超过 $long\ long$。考虑 b_d，它代表了某个点的子节点数量，因此不同的 b_d 所代表的点是互不相交的。因此有 $b_0+b_1+b_2+\cdots \leq N \leq 100$，且 b_i 是正整数。由小学的正整数拆分的结论知，$b_0b_1b_2\cdots \leq 3^{32}\cdot 4 < 2^{53}$，因此这种算法运算所得到的所有数都不会超过 $long\ long$，这也是出题人为什么放 100 而不是 1000 的原因之一(避免写高精度)。