CAT2024 题解

哈尔滨工程大学

2024 年 7 月 16 日

A. 2026 美加墨世界杯

by A_Big_Jiong

题意概述

在世界杯小组赛中，一个小组一共有 4 支球队。排名前 2 的队伍可以晋级。
给定韩国、中国、泰国、新加坡这 4 支队伍打完倒数第 3 场小组赛之后的的积分表，
根据输入的 a, b, c, d 计算，如果在最后两场比赛中，韩国 $a : b$ 中国，
并且泰国 $c : d$ 新加坡，那么国足能否顺利出线。

本场比赛的签到题。
注意到，不管这两场比赛的结果如何，韩国必定是第一，而新加坡必定是第四，
因此我们实际只需要去比较中国与泰国这两支队伍谁排在更前面即可。
计算一下这两支队伍的积分、净胜球、总进球，并挨个进行比较。
需要特别注意一下上述三项全都相同的情况，由于中国队相互比赛的积分高于泰国队，
因此在这种情况下，根据规则，最终出线的是中国队。

B. Stop! High School Maths Please No More

by Frost_Ice (special thanks to fxss && XFish)

题意概述

给定一个长度不超过 10^5 的数列，问至少需要修改其中几个数，
才能将其修改成**首项和公比都为正整数的等比数列**。

本场比赛的另一个签到题。

首先考虑公比 q 为 1 的情况。在这种情况下，全部元素都应该相等。
使用 `std::map` 给全部元素计一下数，拿个数最多的更新答案即可。

之后是 $q \geq 2$ 的情况。
不难注意到，即使是在最极限的情况下，也只有前 60 项是有意义的。
（$1, 2, 4, 8, 16, \cdots, 2^{59}$，再往后面就会超过 10^{18}）
因此我们只需要去关注前 60 个数即可。

B. Stop! High School Maths Please No More

by Frost_Ice (special thanks to fxss && XFish)

注意到答案再坏也至多是 $n-1$ 。因此，如果前 60 项当中有若干个数成功留下，只有留下的数字数量大于等于 2 的情况才是有意义的。
因此，考虑枚举前 60 个数当中，哪两个数会最终留在新数列里，check 一下并更新答案即可。时间复杂度 $O(n \cdot \log(n) + 60^3)$ 。

本题有几个坑点。
首先是，所有项都必须是整数。因此 $0.5, 1, 2, 4, \cdots$ 这种就是不合法的。
其次是需要特别注意一下溢出以及精度的问题。
（如果实在搞不定的话不妨用用 `__int128`）
当然，本题样例基本把这些都给涵盖了。
没过的话不妨检查一下是不是有什么细节写挂了。

C. Diamond

by aldric_li && Frost_Ice (special thanks to ImmortaLimit && StarSilk && Su_Zipei)

题意概述

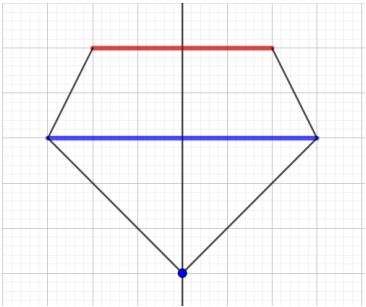
给定二维平面上的不超过 300 个整数点，坐标范围 $\pm 10^8$ 。
从其中选出 5 个点，有多少中选法，使得其能够组成钻石。

首先，我们先考虑一下是否会有枚举重复的问题。
假设有 5 个点，能在不同的朝向上构成钻石，
我们可以证明，它们必定会构成正五边形。
（由对称关系，推一下角度以及边长之间的相等关系即可）
但很显然，平面上由 5 个整数点是够不成正五边形的。
因此，我们无需担心枚举重复的问题。

C. Diamond

by aldric_li && Frost_Ice (special thanks to ImmortaLimit && StarSilk && Su_Zipei)

之后我们来观察一下钻石这个图形的特点。

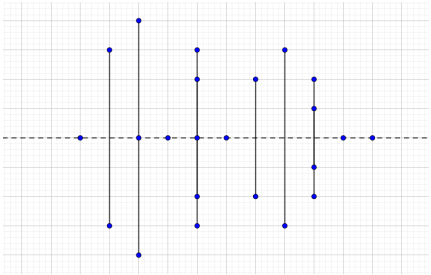


不难发现，钻石其实本质是由共用中垂线的两条线段，
以及该中垂线上的一个点，
所组成。并且排列顺序是：短线段—长线段—点。
显然，这样的中垂线数量至多是 n^2 的。
因此，我们可以使用 `std::map` ，
在 $O(n^2 \cdot \log(n))$ 的复杂度内，
预处理出每条中垂线各自垂直平分了哪些线段。
同样，预处理出每条中垂线上有哪些点，
也只需要 $O(n^3 \cdot \log(n))$ 的复杂度。
之后我们考虑如何统计答案。

C. Diamond

by aldric_li && Frost_Ice (special thanks to ImmortaLimit && StarSilk && Su_Zipei)

对于每根中垂线，上面都分布有若干线段以及点。

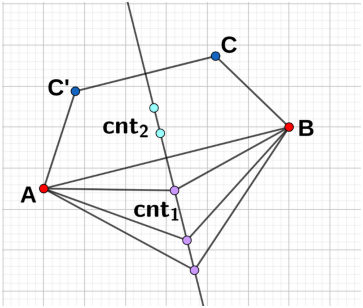


我们需要知道，从当中选出「短线段—长线段—点」这样的组合，有多少种选法。
这是一个经典的线性计数问题。使用离散化 + 树状数组即可统计答案。
总时间复杂度 $O(n^3 \cdot \log(n))$ 。

C. Diamond

by aldric_li && Frost_Ice (special thanks to ImmortaLimit && StarSilk && Su_Zipei)

这里再补充另一种复杂度同样为 $O(n^3 \cdot \log(n))$ 的解法。这是验题队伍想到的。



我们可以先 $O(n^2)$ 的枚举长边的两个端点 AB，之后，我们先 $O(n)$ 的处理出，在直线 AB 的两侧，分别有多少个点在 AB 中垂线上，记为 cnt_1 与 cnt_2。之后，我们去枚举 AB 以外的点 C，如果点 C 关于直线 AB 的对称点 C' 也在点集中，并且 $|CC'|$ 小于 $|AB|$，那么它可以对答案造成「另一侧的 cnt 数量」的贡献。(比如图中 C 在 AB 左上角，那么它的贡献是 cnt_1) 显然，这一步是 $O(n \cdot \log(n))$ 的。
因此，总时间复杂度也为 $O(n^3 \cdot \log(n))$。

D. A xor B problem

by Rubyonly && Frost_Ice (special thanks to ImmortaLimit && StarSilk && Su_Zipei)

题意概述

我们有一片长度为 2^{32} 的内存空间，以及 26 个寄存器。
寄存器 a 和 b 的初始值分别为 A 和 B 。其中 $0 \leq A, B \leq 65535$ 。
我们可以进行 3 种操作，分别是：

- 选取三个寄存器 x, y, z ，将 x 的值写到 $y + z$ 的地址处。
- 选取三个寄存器 x, y, z ，将 $y + z$ 的地址处的值赋给 x 。
- 给一个寄存器 x 赋一个指定的值。

所有涉及到的值都是无符号 32 位整型。
如果计算地址的过程中超出了范围，则会自然溢出。
我们需要用这三种操作，计算出 $A \oplus B$ 。总操作次数不能超过 10^6 。

D. A xor B problem

by Rubyonly && Frost_Ice (special thanks to ImmortaLimit && StarSilk && Su_Zipei)

注意到，我们可以写寄存器，并且可以根据寄存器去写内存，
也就意味着，整片内存其实都是可以通过操作去赋值的。

我们可以考虑通过赋值再访问的方式，进行一些基本计算。
比方说，我们给一片区域进行赋值，第 i 个数赋值为 `i>>1`。
然后我们用寄存器 x 对这篇区域的第 x 个数进行访问，就能得到 `x>>1` 的值。

之后我们考虑如何去进行构造。

D. A xor B problem

by Rubyonly && Frost_Ice (special thanks to ImmortaLimit && StarSilk && Su_Zipei)

这里具体讲 3 种构造方式。

(一) 出题人想到的构造

我们从高位往低位考虑。

- 先把 $S[i]$ 赋值为 `i>>15` $(0 \le i \le 65535)$
- 取出 $S[a]$ 和 $S[b]$，赋值给 d 和 e，从而得到 A 和 B 的最高位。
- 把 $S[0]$ 到 $S[2]$ 赋值为 $\{0, 32768, 0\}$，取出 $S[x+y]$，从而得到最高位的异或结果。
- 把 $S[i]$ 赋值为 `i-((i>>15)<<15)` $(0 \le i \le 65535)$
- 把 a 变成 $S[a]$，把 b 变成 $S[b]$，从而去掉 A 和 B 的最高位。

  重复上述过程，不断去掉最高位，并计算 (当前) 最高位的异或结果。

  最后，把这些异或结果相加，即可得到答案。复杂度 $O(n)$ 。

D. A xor B problem

by Rubyonly && Frost_Ice (special thanks to ImmortaLimit && StarSilk && Su_Zipei)

（二）验题队伍想到的构造

我们分成两个部分来考虑，分别算前 8 个二进制位与后 8 个二进制位的异或结果，位移并相加即可得到答案。

- 把 $S[i]$ 赋值为 `i>>8` $(0 \le i \le 65535)$
- 取出 $S[a]$ 和 $S[b]$，赋值给 u 和 v，从而得到 A 和 B 的前 8 位。
- 把 $S[i]$ 赋值为 `i&0xFF` $(0 \le i \le 65535)$
- 取出 $S[a]$ 和 $S[b]$，赋值给 x 和 y，从而得到 A 和 B 的后 8 位。
- 把 $S[i]$ 赋值为 `i<<8` $(0 \le i \le 255)$
- 通过取出 $S[u]$ 与 $S[v]$，把 u 和 v 左位移 8 位。
- 把 $S[i]$ 赋值为 `((i<<8)>>8)xor(i>>8)` $(0 \le i \le 65535)$
- 取出 $S[u+x]$ 与 $S[v+y]$，即可分别得到前 8 位与后 8 位的异或结果。
- 最后，利用类似方法，将前者左位移 8 位与后者相加，即可得到答案。

D. A xor B problem

by Rubyonly && Frost_Ice (special thanks to ImmortaLimit && StarSilk && Su_Zipei)

（三）另一只验题队伍想到的构造

我们先准备三片区域，第一片用于进行右位移。第二片区域用于进行加法 (并只保留最低的 16 个二进制位)。第三片用于左位移 (并只保留最低的 16 个二进制位)。

对于从低到高的第 i 个二进制位 $(0 \leq i \leq 15)$:

- 把 a 和 b 复制一份，赋值给 x 和 y 。
- 依次通过 i 次右位移，i 次左位移，$15 - i$ 次左位移，使得 d 和 e 只剩一个有效二进制位，并且该二进制位位于最高位。
- 计算 d 和 e 的加法。因为我们强行让超出 16 个二进制位的部分溢出，因此这里其实相当于做了异或。
- 之后进行 $15 - i$ 次右位移，回到该有的位置上，并累加到答案上。

E. Rolling for the Destination

by Frost_Ice

题意概述

一个人初始站在数轴 0 点，每次随机往前走 $1/2/3/4$ 步。
问这个人能走到 n 这个点的概率是多少，对 998244353 取模。
其中 n 是 10 的 5E7 次方。

我们令走到 i 这个点的概率是 P_i。显然，$P_0 = 1$，并且对于小于 0 的 i，有 $P_i = 0$。
如果这个人成功到达了点 i，显然这其中有四种可能的情况。

- 先到达点 $i-1$，然后再往前走 1 步。概率 $P_{i-1}/4$。
- 先到达点 $i-2$，然后再往前走 2 步。概率 $P_{i-2}/4$。
- 先到达点 $i-3$，然后再往前走 3 步。概率 $P_{i-3}/4$。
- 先到达点 $i-4$，然后再往前走 4 步。概率 $P_{i-4}/4$。

由此，我们得到了递推公式 $P_i = (P_{i-1} + P_{i-2} + P_{i-3} + P_{i-4})/4$。

E. Rolling for the Destination

by Frost_Ice

写成矩阵的话，就是下面这种形式：

$$\begin{bmatrix} P_{i-3} \\ P_{i-2} \\ P_{i-1} \\ P_i \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix} \cdot \begin{bmatrix} P_{i-4} \\ P_{i-3} \\ P_{i-2} \\ p_{i-1} \end{bmatrix}$$

这样，我们即可得到 P_n 的计算公式：

$$\begin{bmatrix} P_{n-3} \\ P_{n-2} \\ P_{n-1} \\ P_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix}^n \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

E. Rolling for the Destination

by Frost_Ice

使用矩阵快速幂可以在 $4^3 \cdot \log(n)$ 的复杂度内完成上述计算。

不过在本题的数据范围下，使用矩阵快速幂大概率会超时。
因此我们需要去找一些其他的性质。

不难打表发现，在模数 $M = 998244353$ 的情况下，
这个矩阵的循环节是 $M - 1$。
因此，只需要输入对 $M - 1$ 取模，再使用矩阵快速幂计算即可。

时间复杂度 $O(T \cdot (\log_{10}(n) + 4^3 \cdot \log(M)))$

F. 静流的路径

by skytyf

题意转化

对于一条路径，由于 $a \to b$，那么一定有 $b = p \times a$，其中 p 为质数，因此必有 $f(b) > f(a)$。
也就是说，一条路径至多包含 1 个罚抄数。
因此，原问题等价于求罚抄数的个数，即求 $b_1 + \cdots + b_k = T$，且 $b_i \le m$ 的序列 $\{b_1, \cdots, b_k\}$ 的个数。

考虑生成函数，原问题的生成函数显然为：

$$F(x) = (1 + x + \cdots + x^m)^k$$

注意到罚抄数的个数即为 $F(x)$ 在 x^T 的系数，即 $ans = [x^T]F(x)$。
直接使用多项式快速幂求解的时间复杂度是 $O(mk \log(mk) \log k)$，无法通过。
考虑对生成函数的形式进行化简。

F. 静流的路径

by skytyf

使用等比数列求和公式，有：$F(x)=\dfrac{(x^{m+1}-1)^k}{(x-1)^k}$

利用二项式定义展开：

$$F(x)=\frac{x^{k(m+1)}-\binom{k}{1}x^{(k-1)(m+1)}+\cdots+\binom{k}{k}(-1)^k}{x^k-\binom{k}{1}x^{k-1}+\binom{k}{2}x^{k-2}+\cdots+(-1)^k}$$

考虑将上式分子分离，即可得出 $F(x)$ 在 x^T 处的系数，答案形式为：

$$[x^T]F(x)=\sum_{i=0}^{k}(-1)^i\binom{k}{i}\binom{T-i(m+1)+k-1}{k-1}$$

直接计算即可，时间复杂度 $O(T)$。

F. 静流的路径

by skytyf

使用生成函数求解的做法计算量较大。本题也可以从组合意义入手。

注意到 $b_i \le m$ 中的小于等于的条件不容易转化，考虑容斥至少有几个 b_i 超过 m。

如果一个数 s 超过 m，那么等价于 $s = m + 1 + \delta$，其中 $\delta \ge 0$ 代表超过的部分。

设当前容斥的数中有至少 $|S|$ 个超过 m 的个数。那么，将 T 减去 $|S| \times (m+1)$ 之后，原问题就转化为了无限制条件的简单组合问题，直接插板法即可。

计算发现发现，答案同样为：

$$\sum_{i=0}^{k} (-1)^i \binom{k}{i} \binom{T - i(m+1) + k - 1}{k - 1}$$

直接计算即可，时间复杂度 $O(T)$。

G. Candidate Master of Both (VI)

by Frost_Ice

题意概述

给定一个长度为 n 的序列，要求对于每一个 $1 \le k \le n$，
计算所有满足 $\gcd(l, r) = k$ 的区间的异或和的和。
其中 n 为 10^5 ，$0 \le a_i \le 10^5$ 。

首先，我们先考虑在去掉 $\gcd(l, r) = k$ 的限制的情况下，
如何去计算 $\sum_{1 \le l \le r \le n} a_l \oplus a_{l+1} \oplus \cdots \oplus a_r$ 的值。

不难发现，每个二进制位对答案的贡献是可以拆开来考虑的。
我们用 $d[i][j]$ 表示 a_i 的从低往高的第 j 个二进制位。

我们记 $sum[i][j] = \bigoplus_{x=1}^{j} d[i][j]$ 表示 $d[i][j]$ 的异或前缀和。特别的，$sum[i][0] = 0$ 。

G. Candidate Master of Both (VI)

by Frost_Ice

我们依次考虑每个二进制位。对于第 i 位，
如果一个区间 $[l, r]$ 满足：$sum[i][l-1] \oplus sum[i][r] = 1$ ，
那么，根据异或的性质，该区间的异或和在第 i 位上是 1 ，
因此，它将对答案产生 2^{i-1} 的贡献。
我们可以从左往右扫一遍，j 从 0 枚举到 n ，
遇到 $sum[i][j]$ 为 1 的，看一下它前面有多少个 0；
遇到 $sum[i][j]$ 为 0 的，看一下它前面有多少个 1。
依次对于每个二进制位这样计一下数，即可得到答案。
至此，我们完成了 $\sum_{1 \le l \le r \le n} a_l \oplus a_{l+1} \oplus \cdots \oplus a_r$ 的计算。

由于在本题的数据范围内，至多需要考虑 20 个二进制位，
因此这样计算的时间复杂度是 $O(20 \cdot n)$ 的。

G. Candidate Master of Both (VI)

by Frost_Ice

让我们回到原题。

我们令 $f(l,r)$ 为：

$$f(l,r)=\begin{cases}\bigoplus_{i=l}^{r} a_i & \text{, if } l\le r\\ 0 & \text{, if } l>r\end{cases}$$

也就是说，当 $l\le r$ 时，$f(l,r)$ 表示区间 $[l,r]$ 的异或和，否则 $f(l,r)$ 为 0 。

我们用 $g(x)$ 表示 $k=x$ 情况下的答案，则有：

$$g(x)=\sum_{i=1}^{n}\sum_{j=1}^{n}[\gcd(i,j)=k]\cdot f(i,j)$$

G. Candidate Master of Both (VI)

by Frost_Ice

显然，只有那些为 x 的倍数的 i 和 j 才可能会对答案产生贡献，因此：

$$g(x) = \sum_{i=1}^{\lfloor \frac{n}{x} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{x} \rfloor} \epsilon(\gcd(i,j)) \cdot f(i \cdot x, j \cdot x)$$

根据莫比乌斯反演经典结论，上式可以化成：

$$= \sum_{i=1}^{\lfloor \frac{n}{x} \rfloor} \sum_{j=1}^{\lfloor \frac{n}{x} \rfloor} \sum_{d|i,d|j} \mu(d) \cdot f(i \cdot x, j \cdot x)$$

G. Candidate Master of Both (VI)

by Frost_Ice

考虑交换求和顺序，先枚举 d，于是有：

$$g(x)=\sum_{d=1}^{\lfloor \frac{n}{x} \rfloor}\mu(d)\sum_{d|i}^{\lfloor \frac{n}{x} \rfloor}\sum_{d|j}^{\lfloor \frac{n}{x} \rfloor}f(i\cdot x,j\cdot x)$$

把 i 和 j 同除 d，得到：

$$g(x)=\sum_{d=1}^{\lfloor \frac{n}{x} \rfloor}\mu(d)\sum_{i=1}^{\lfloor \frac{n}{x\cdot d} \rfloor}\sum_{j=1}^{\lfloor \frac{n}{x\cdot d} \rfloor}f(i\cdot x\cdot d,j\cdot x\cdot d)$$

G. Candidate Master of Both (VI)

by Frost_Ice

我们令 $h(x)$ 为:

$$h(x) = \sum_{i=1}^{\lfloor \frac{n}{x} \rfloor} \sum_{i=1}^{\lfloor \frac{n}{x} \rfloor} f(i \cdot x, j \cdot x)$$

这样原式就可以写成:

$$g(x) = \sum_{d=1}^{\lfloor \frac{n}{x} \rfloor} \mu(d) \cdot h(d \cdot x)$$

G. Candidate Master of Both (VI)

by Frost_Ice

之后我们考虑 $h(x)$ 如何去计算。
不难发现，$h(x)$ 表示的其实就是：
由 x 的倍数为端点构成的区间的异或和的和。
仿照我们之前计算 $h(1)$ 的过程。预处理完 $sum[i][j]$ 之后，我们依旧是从左往右扫。
对于 $sum[i][p \cdot x]$ 为 1 的，我们需要数一下

$$sum[i][1 \cdot x - 1]\ , sum[i][2 \cdot x - 1]\ , sum[i][3 \cdot x - 1]\ , \cdots\ , sum[i][p \cdot x - 1]$$

这些 x 的若干倍减 1 当中，一共有多少个 0 。
对于 $sum[i][p \cdot x]$ 为 0 的情况也是同理。
这样一来，我们可以用 $O(20 \cdot n/x)$ 的复杂度完成 $h(x)$ 的计算。
因此，我们预处理 $h(1)$ 到 $h(n)$，只需要 $O(20 \cdot n \cdot \log(n))$ 的复杂度。

G. Candidate Master of Both (VI)

by Frost_Ice

至于莫比乌斯函数 $\mu(x)$ 的话，拿一个线性筛即可线性的预处理。
预处理完 $\mu(x)$ 以及 $h(x)$ 之后，根据我们之前的公式：

$$g(x) = \sum_{d=1}^{\lfloor \frac{n}{x} \rfloor} \mu(d) \cdot h(d \cdot x)$$

我们即可在 $O(n/x)$ 的时间复杂度内计算出 $g(x)$ 。
这样一来，预处理完之后的计算是 $O(n \cdot \log n)$ 的，
因此总时间复杂度 $O(20 \cdot n \cdot \log n)$ 。

H. Duel on the Chessboard

by Febleaf && Frost_Ice

题意概述

一片 $n \times m$ 的棋盘，上面有两枚棋子 A 和 B，初始相邻。
部分其他位置存在有障碍物，障碍物上不能摆放棋子。
两个人轮流移动棋子，每次移动，可以选择一个棋子，将其绕着另一个旋转 $90°$。
不能走重复状态。轮到谁没法操作，则输掉游戏。

不难发现，在某个人移动完棋子后，如果两枚棋子之前是横着的，则必然会变成竖着。
同理，之前是竖着的话，移动完之后必然会变成横着。
也就是说，我们可以把所有状态分成两类，一类是两枚棋子横着，另一类是竖着。
同类的状态之间不存在转移，而不同类的状态之间存在。
也就是说，所有的状态以及它们之间的转移构成一个二分图。
所以这是一个二分图上不能走重复状态的博弈问题。

H. Duel on the Chessboard

by Febleaf && Frost_Ice

引理 1

对于二分图上不能走重复状态的博弈问题，先手必胜的充分必要条件为：起始状态必定位于二分图的最大匹配上。

- 假设起始状态必定位于最大匹配上。我们任选一组最大匹配。可以证明，两人走出的路径必然是类似 $A_1 - B_1 - A_2 - B_2 - \cdots - A_i - B_i$ 这样的，其中 A_1 与 B_1 匹配，A_2 与 B_2 匹配，以此类推。先手采取的策略是：从当前状态，移动到与当前状态相匹配的状态。这样一来，后手行动时，要么没有状态可走，要么移动到一个也在最大匹配中的新的状态上去。为什么呢？因为，如果后手的某步操作，移动到了一个不在最大匹配上的 C 状态上去，此时就存在了一条 $A_1 - B_1 - A_2 - B_2 - \cdots A_i - B_i - C$ 的交错路。我们把已有的匹配替换一下，匹配数没有减少，而起始状态却不在这组匹配中了。这就与我们的假设相矛盾。这样一来，先手总是有操作可以走，最终输掉的只会是后手。

H. Duel on the Chessboard

by Febleaf && Frost_Ice

- 另一种情况也是类似的。假设起始状态不一定位于最大匹配上。不妨记起始状态为 A_0 。我们任选一组不包含起始状态的最大匹配。两人走出的路径也必然是类似 $A_0 - B_1 - A_1 - B_2 - A_2 - \cdots B_i - A_i$ 这样的，其中 A_1 与 B_1 匹配，A_2 与 B_2 匹配，以此类推。后手操作时，只需要从当前状态走到与之相匹配的状态就行了。而轮到先手的时候，他必然会走到一个新的也处在最大匹配上的状态。（否则，假设某步操作，先手走到了一个不在最大匹配上的 C 状态，此时起始就存在了一条 $A_0 - B_1 - A_1 - B_2 - A_2 - \cdots B_i - A_i - C$ 的增广路，因为 A_0 和 C 都不在匹配中，这就表明这组匹配不是最大匹配，因此矛盾）这样一来，后手总是有操作可以走，最终输掉的只会是先手。

H. Duel on the Chessboard

by Febleaf && Frost_Ice

让我们回到原题。不难发现，把所有状态建成图，
其中至多只会有 $4 \cdot n \cdot m$ 个点，$8 \cdot n \cdot m$ 条边。
因此，使用 dinic 算法即可在 $O(n \cdot m \cdot \sqrt{n \cdot m})$ 的时间复杂度内求解。
（原图跑一遍，去掉起始点再跑一遍，比较最大匹配是否减小即可）

I. 黄金树

by Rubyonly && Frost_Ice

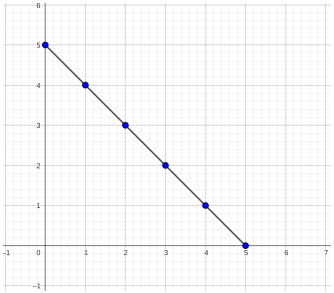
题意概述

给定一棵 10^6 个节点的有根树。每个节点初始时刻有个点权。点权不超过 10^9 。
之后，每个时刻，只要某个节点满足：
它在当前时刻的点权，小于其父亲节点在当前时刻的点权，或者这个节点为根节点，
那么下一时刻，该节点的点权就会减 1 。最多减少到 0 。
求每个节点在所有时刻的点权之和。

I. 黄金树

by Rubyonly && Frost_Ice

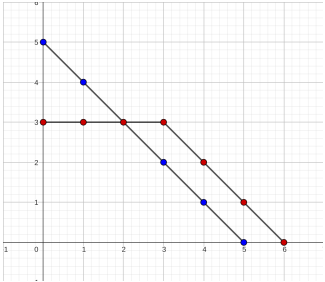
首先我们不难注意到，根节点的答案是十分好算的。
因为根节点每一时刻必定会减 1 ，直到减少到 0 。
画出函数图像的话，就会是类似这样。（横轴代表时间，纵轴代表点权）

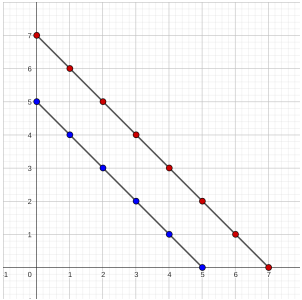


I. 黄金树

by Rubyonly && Frost_Ice

之后我们考虑一下根节点的儿子。此处会有两种情况。
第一种是，该节点点权小于等于根节点。第二种是，点权大于根节点。
分别如左图和右图所示。（其中红色点代表该节点）

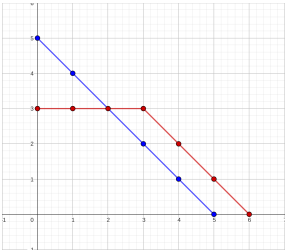


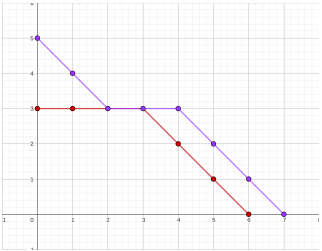


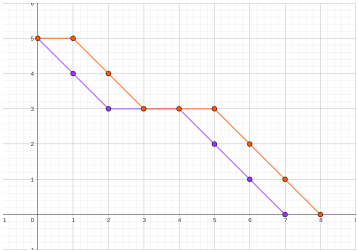
I. 黄金树

by Rubyonly && Frost_Ice

我们再手推一下从根节点往下分别是 $[5,3,5,5]$ 的这样一组样例。







I. 黄金树

by Rubyonly && Frost_Ice

我们不难观察到以下几条性质：

- 每个节点的函数图像都必定是一个分段函数。其中有些段水平，有些段以 -1 的斜率下降。
- 函数图像必定是水平和下降交替。并且分界点都是整数点。
- 如果一个节点的初始点权小于等于其父亲节点，那么该节点的函数图形会先保持水平。否则，则是先开始下降。
- 从左往右看，在某个节点的函数图像第一次与其父亲节点的函数图形发生“接触”后，之后都将重复其父亲节点的图像，但是时间轴上会向右偏移 1 格。

I. 黄金树

by Rubyonly && Frost_Ice

也就是说，对于某个节点而言，它的图像必定先是一段水平或下降，之后全都是在重复其父亲节点的图像，只不过水平上会右移一格。（当然，如果初始点权大到一定程度，也可能在下降的全过程中都没遇到其父亲节点的函数图像）

因此，如果这棵树是一条链的话，我们即可在线性的时间复杂度内，通过用栈维护函数图像的方式求解。

不过，因为这是在树上，因此直接用栈去维护是无法保证复杂度的。因此我们考虑用二分的方式，代替暴力弹栈。

因此，我们可以在 `dfs` 的过程中，用数组维护当前节点的函数图像。到一个新的节点，二分出它与父亲节点的图像的交点，对数组进行修改并计算答案。回溯时再撤回修改。

注意到每次弹栈之后，我们至多会为函数图像增加两段新的线段，

因此时间复杂度 $O(n \cdot \log n)$ 。

J. 简单的指数运算

by A_Big_Jiong (&& Rubyonly && Frost_Ice && Nanako)

题意

给定 N 和 P 计算以下式子模 P 的值。其中，$N \leq 10^6$ ，P 是质数。

$$\prod_{i=1}^{N}\prod_{j=1}^{N}(i \cdot j)^{d(i \cdot j)}$$

不难发现，式子中最难以处理的是 $d(i \cdot j)$ ，因为 $i \cdot j$ 最大会到 10^{12} ，
并且会有大约 10^{11} 这个数量级种不同的取值，
所以即使是一些很优秀的筛法，也不太能预处理出 $d(i \cdot j)$ 的值。
因此考虑转换 $d(i \cdot j)$。

J. 简单的指数运算

by A_Big_Jiong (&& Rubyonly && Frost_Ice && Nanako)

引理

$$d(i\cdot j)=\sum_{x|i}\sum_{y|j}[\gcd(x,y)=1]$$

记一个数对 (x,y) 贡献的的是 $x\times\frac{j}{y}$ 这个因数。

先证若 $\gcd(x,y)=g\neq 1$，则必然算重：$x\times\frac{j}{y}=gx'\times\frac{j}{gy'}=x'\times\frac{j}{y'}$。

即 $\gcd(x,y)=g\neq 1$ 与 $\gcd(x',y')=1$ 产生了相同的贡献。

再证对于不同的互质数对，它们贡献的数不同。

假设对于互质数对 $(x_1,y_1),(x_2,y_2),x_1>x_2$，假设他们贡献了相同的因数，有 $\frac{x_1}{y_1}=\frac{x_2}{y_2}$。

因为上述数对均为互质数对，$x_1\neq x_2$，而分数的最简形式唯一，必不存在 $\frac{x_1}{y_1}=\frac{x_2}{y_2}$，假设不成立。因此引理成立。

J. 简单的指数运算

by A_Big_Jiong (&& Rubyonly && Frost_Ice && Nanako)

同时，根据莫比乌斯反演经典结论：

$$[\gcd(u,v)=1]=\sum_{p|u,p|v}\mu(p)$$

因此：

$$\begin{aligned} d(i\cdot j)&=\sum_{u|i}\sum_{u|j}[\gcd(u,v)=1]\\ &=\sum_{u|i}\sum_{v|j}\sum_{p|u,p|v}\mu(p) \end{aligned}$$

（为了避免与 $d(x)$ 这个函数相混淆，我们这里使用的是字母 p）

J. 简单的指数运算

by A_Big_Jiong (&& Rubyonly && Frost_Ice && Nanako)

还是按照莫比乌斯反演题目的经典做法，我们考虑先去枚举上式中的 p：

$$d(i\cdot j)=\sum_{p|\gcd(i,j)}\mu(p)\sum_{p|u,u|i}\sum_{p|v,v|j}1$$

注意到只有为 p 的倍数的 u 和 v 才能对这个式子造成贡献，因此：

$$\begin{aligned}d(i\cdot j)&=\sum_{p|\gcd(i,j)}\mu(p)\sum_{u|\frac{i}{p}}\sum_{v|\frac{i}{p}}1\\&=\sum_{p|\gcd(i,j)}\mu(p)\cdot d(\frac{i}{p})\cdot d(\frac{j}{p})\end{aligned}$$

至此，我们完成了 $d(i\cdot j)$ 的转化。

J. 简单的指数运算

by A_Big_Jiong (&& Rubyonly && Frost_Ice && Nanako)

将上述等式带入原式，则有：

$$\prod_{i=1}^{N}\prod_{j=1}^{N}(i\cdot j)^{\sum_{p|\gcd(i,j)}\mu(p)\cdot d(\frac{i}{p})\cdot d(\frac{j}{p})}$$

我们再次考虑交换求和顺序，先去枚举上式中的 p。原式可以转化为：

$$\prod_{p=1}^{N}\{\prod_{i=1}^{\lfloor\frac{N}{p}\rfloor}\prod_{j=1}^{\lfloor\frac{N}{p}\rfloor}(i\cdot j\cdot p^2)^{d(i)\cdot d(j)}\}^{\mu(p)}$$

J. 简单的指数运算

by A_Big_Jiong (&& Rubyonly && Frost_Ice && Nanako)

考虑枚举 p，则上式中大括号内的部分，由 $\prod_{i=1}^{\lfloor \frac{N}{p} \rfloor} \prod_{j=1}^{\lfloor \frac{N}{p} \rfloor} p^{d(i)\cdot d(j)}$ 和 $\prod_{i=1}^{\lfloor \frac{N}{p} \rfloor} \prod_{j=1}^{\lfloor \frac{N}{p} \rfloor} i^{d(i)\cdot d(j)}$（$i$ 和 j 完全对称）组成，尝试分别化简以上部分。为方便计算，我们记 $M = \lfloor \frac{N}{p} \rfloor$。

对于 $\prod_{i=1}^{M} \prod_{j=1}^{M} p^{d(i)\cdot d(j)}$，在本式中 p 为常量，可以化简为：

$$p^{\sum_{i=1}^{M} \sum_{j=1}^{M} d(i)\cdot d(j)}$$

$$p^{\sum_{i=1}^{M} d(i)\cdot \sum_{j=1}^{M} d(j)}$$

$$p^{\{\sum_{i=1}^{M} d(i)\}^2}$$

显然，我们只需要预处理出 $\sum_{i=1}^{M} d(i)$ 的值，即可在 $O(n \log n)$ 的时间复杂度内完成本部分的计算。

J. 简单的指数运算

by A_Big_Jiong (&& Rubyonly && Frost_Ice && Nanako)

对于 $\prod_{i=1}^{M}\prod_{j=1}^{M} i^{d(i)\cdot d(j)}$，观察到 j 只出现在指数位上，可以化简为：

$$\prod_{i=1}^{M} i^{d(i)\cdot\{\sum_{j=1}^{M} d(j)\}}$$

$$(\prod_{i=1}^{M} i^{d(i)})^{\sum_{j=1}^{M} d(j)}$$

我们可以先预处理出 $i^{d(i)}$ 的前缀积和 $\sum_{i=1}^{M} d(i)$ 的值，然后对于不同的 M 计算出不同的值，时间复杂度同样为 $O(n\log n)$。

最后，我们将上面预处理的值带入原式，即可求出答案，时间复杂度为 $O(n\log n)$。