CCPC 深圳 2023

Elegia, gyh20, Itst, liuzhangfeiabc, SpiritualKhorosho†
2023 年 11 月 12 日

清华大学算法协会

除命题人外, 感谢:

- 另有 E · Space, ix35, Mys.C.K., JohnVictor, xzn, 一念之间[†] 等人参与题目准备工作.
- 校内队伍 雾里看花, 有手有脚[†] 集体验题.

† 按照字典顺序排列.

Part I (讲者 *gyh20*)

A 一道好题

D 机器人游戏

E 二合一

K 四国军棋

Part II (讲者 *Elegia*)

M 三数之和

B 借口

G 相似基因序列问题

Part III (讲者 *liuzhangfeiabc*)

C 报数 III

H 快速散列变换

J 计数器清零问题

Part IV (讲者 *Mys.C.K.*)

F 见面礼

I 不定方程

L Mary 有颗有根树

Part I (讲者 *gyh20*)

A 一道好题 by *gyh20*

回顾题意: 给定一个长为 n 的序列 $0 \le b_i \le n$, 你可以要把一个全零序列变换成 b, 允许以下两种操作:

- 把所有值为 x 的数 $+1$.
- 把第 x 个数 $+1$.
- 对于 $n \le 1000$, 你要用 ≤ 20000 次操作完成.

可以发现，只使用 1 操作或只使用 2 操作都是不行的，我们一定需要一个 1 操作和 2 操作的结合。

可以发现，只使用 1 操作或只使用 2 操作都是不行的，我们一定需要一个 1 操作和 2 操作的结合。

我们发现，数变大之后不能变小。所以主要的问题是，1 操作会让一些我们不希望变大的数变大。

A 一道好题 by *gyh20*

所以我们大致需要，用 2 操作分裂出一个集合，用 1 操作将这个集合整体加。而为了降低分裂出的集合大小，同时保持加操作不太多，可以想到分治。

A 一道好题 by *gyh20*

所以我们大致需要，用 2 操作分裂出一个集合，用 1 操作将这个集合整体加。而为了降低分裂出的集合大小，同时保持加操作不太多，可以想到分治。

具体的，我们可以对值域进行分治。每次递归到一个值域区间 $[l,r]$，选取 $mid=\frac{l+r}{2}$，将所有需要 $\geq mid$ 的数先用 2 操作 $+1$，与其他数区分开，然后将这些数用 1 操作一起 $+1$ 直到均变为 mid。然后递归操作 $[l,mid-1]$ 和 $[mid,r]$。

A 一道好题 by *gyh20*

所以我们大致需要，用 2 操作分裂出一个集合，用 1 操作将这个集合整体加。而为了降低分裂出的集合大小，同时保持加操作不太多，可以想到分治。

具体的，我们可以对值域进行分治。每次递归到一个值域区间 $[l,r]$，选取 $mid=\frac{l+r}{2}$，将所有需要 $\geq mid$ 的数先用 2 操作 $+1$，与其他数区分开，然后将这些数用 1 操作一起 $+1$ 直到均变为 mid。然后递归操作 $[l,mid-1]$ 和 $[mid,r]$。

容易发现 1 操作和 2 操作的个数均不超过 $n\log n$，所以总操作次数 ≤ 20000。

A 一道好题 by *gyh20*

实际上从做题的角度来看，20000 次操作是 $O(n \log n)$ 级别，更容易联想到分治。

A 一道好题 by *gyh20*

实际上从做题的角度来看，20000 次操作是 $O(n\log n)$ 级别，更容易联想到分治。

如果更精细地分析可以发现操作是远远不到 20000 这个上界的，不过由于本题应该并不存在较优秀的高复杂度解（验题人的 $O(n\sqrt{n})$ 的代码只能做到 30000 次左右），所以并没有设置更紧的界。

D 机器人兄弟 by *gyh20*

回顾题意: 有一棵 n 个点的树，m 个叶子，编号为 $1 \sim m$。两人在树上博弈，均从根出发，轮流行动，每次走向一个当前所在节点的子节点，如果在叶子就不移动。最终如果两人所在叶子编号一个是另一个 $+1$（$\bmod m$ 意义下），则 $+1$ 的一方获胜。

$m < n \le 10^5, \sum n \le 5 \times 10^5$。

D 机器人兄弟 by *gyh20*

首先，后手是必不败的，因为无论怎样都能模仿先手的操作，最终两人到达相同的叶子节点。

D 机器人兄弟 by *gyh20*

首先，后手是必不败的，因为无论怎样都能模仿先手的操作，最终两人到达相同的叶子节点。

剩下的问题是判断后手是否必胜。

D 机器人兄弟 by *gyh20*

首先，后手是必不败的，因为无论怎样都能模仿先手的操作，最终两人到达相同的叶子节点。

剩下的问题是判断后手是否必胜。

初步观察：可以发现，大部分较随机的树，游戏都是平局。

D 机器人兄弟 by *gyh20*

考虑一个这样的方式，对于每个子树，维护 S_x 表示子树内的叶子集合，T_x 表示子树内每个叶子 $+1$ 后构成的集合。

D 机器人兄弟 by *gyh20*

考虑一个这样的方式，对于每个子树，维护 S_x 表示子树内的叶子集合，T_x 表示子树内每个叶子 $+1$ 后构成的集合。

若某时刻，先手在 x，后手在 y，T_x 不是 S_y 的子集，那么 x 直接朝向一个不在 S_y 中的点，后手就无法打败先手。所以后手需要任意时刻保持 T_x 是 S_y 的子集。

D 机器人兄弟 by *gyh20*

考虑一个这样的方式，对于每个子树，维护 S_x 表示子树内的叶子集合，T_x 表示子树内每个叶子 +1 后构成的集合。

若某时刻，先手在 x，后手在 y，T_x 不是 S_y 的子集，那么 x 直接朝向一个不在 S_y 中的点，后手就无法打败先手。所以后手需要任意时刻保持 T_x 是 S_y 的子集。

首先若一个叶子的父亲只有一个儿子，可以删掉这个父亲。这样我们可以避免一个人能动另一个人不能动的情况。然后注意到若后手必胜，同一层的点 S_x 的大小必须一样，且如果有 x 个点，则 S_x 形如 $\{a, a+x, a+2x\ldots\}$（否则存在一个点 x 使得不存在 S_y 是 T_x 的超集）。

D 机器人兄弟 by *gyh20*

所以当后手可以必胜时，任意时刻，$T[x] = S[y]$!

D 机器人兄弟 by *gyh20*

所以当后手可以必胜时，任意时刻，$T[x] = S[y]$！

因此一个简单的做法就是直接 $dfs(x, y)$ 表示判断先手在 x 后手在 y 是否一定必胜，因为必须要 $T_x = S_y$ 所以只有至多 n 种可能的状态，每次枚举 x y 所有子节点，找到 $T_x = S_y$ 的点向下检查即可。

D 机器人兄弟 by *gyh20*

所以当后手可以必胜时，任意时刻，$T[x] = S[y]$!

因此一个简单的做法就是直接 $dfs(x, y)$ 表示判断先手在 x 后手在 y 是否一定必胜，因为必须要 $T_x = S_y$ 所以只有至多 n 种可能的状态，每次枚举 x y 所有子节点，找到 $T_x = S_y$ 的点向下检查即可。

总复杂度 $O(n) \sim O(n \log n)$。

D 机器人兄弟 by *gyh20*

如果就这样结束实在有点不优美，因为实际上树的形态更加简单。

D 机器人兄弟 by *gyh20*

如果就这样结束实在有点不优美，因为实际上树的形态更加简单。

注意缩树之后叶子父亲的一层，这一层的 S 一定都是 $\{a, a+x, a+2x\ldots\}$ 的形式。我们发现，$a+1$ 的集合打败 a 的集合，也就是说，我们可以把叶子父亲的一层也当成叶子！

D 机器人兄弟 by *gyh20*

如果就这样结束实在有点不优美，因为实际上树的形态更加简单。

注意缩树之后叶子父亲的一层，这一层的 S 一定都是 $\{a, a+x, a+2x\dots\}$ 的形式。我们发现，$a+1$ 的集合打败 a 的集合，也就是说，我们可以把叶子父亲的一层也当成叶子！

所以理论上来说本题可以带修/计数等等，不过由于各种原因没有加强。

E 二合一 by *gyh20*

回顾题意：有一个长度为 n 的序列，找到一个区间和两种颜色使得两种颜色出现次数的二进制或最大。

$n \le 10^5, \sum n \le 5 \times 10^5$。

E 二合一 by *gyh20*

本题其实是一个结论题。

E 二合一 by *gyh20*

本题其实是一个结论题。

结论：令出现次数最多的两个颜色的出现次数分别为 x, y，答案为 $x \| y \| (highbit(x \& y) - 1)$。即在 $[0, x]$ 和 $[0, y]$ 中分别选一个数，能达到的 or 的最大值。这个值明显是一个上界，接下来我们可以证明这个值一定能被取到。

证明：仅考虑这两种颜色，且仅考虑选择一个前缀的情况。枚举这个前缀，存在一个时刻，使得在这一个时刻两种颜色的 or 在 $highbit(x\|y)$ 更高的位和 $x\|y$ 一致，在此之后更高位的 or 不会改变，但存在一个时刻使得其中一个数的低位取到 $highbit(x\|y)-1$。

K 四国军棋 by *gyh20*

回顾题意：给定两个军棋序列，判断最终结果是否为平局，支持动态修改。

$n \le 10^5, m \le 2 \times 10^5$。

可以发现若一个棋子后面有一个严格比自己大的棋子，那么这个棋子是没有用的，因为我们可以直接认为这个棋子无论怎样都战败，并不会影响最终的局势。

可以发现若一个棋子后面有一个严格比自己大的棋子，那么这个棋子是没有用的，因为我们可以直接认为这个棋子无论怎样都战败，并不会影响最终的局势。

所以，对于一个军棋序列，只有那些后缀最大值是有用的。

K 四国军棋 by *gyh20*

对于两个后缀最大值序列，比较其大小时，实际上我们是在比较字典序。判断平局我们可以直接维护字符串哈希即可。所以最终的做法是用线段树维护后缀最大值的哈希值，总复杂度 $O(m\log^2 n)$。

K 四国军棋 by *gyh20*

对于两个后缀最大值序列，比较其大小时，实际上我们是在比较字典序。判断平局我们可以直接维护字符串哈希即可。所以最终的做法是用线段树维护后缀最大值的哈希值，总复杂度 $O(m\log^2 n)$。

最终数据没有针对任何哈希算法，错的不太离谱的哈希应该都能过。

Part II (讲者 *Elegia*)

M 三数之和 by *Elegia*

- 回顾题意: 给定 N 个大整数, 求出它们中所有 $a+b+c=0 \pmod{M}$ 的解.
- $N \le 500,\ M = 10^K - 1,\ K \le 2 \times 10^4$.

M 三数之和 by *Elegia*

我们首先将每个数计算出取模到 $[0, M)$ 的结果, 由于给出的 M 比较特殊, 我们可以根据 $10^K = 1$ 把最高位不断降下来, 在 $O(\text{digits})$ 的时间内完成取模运算. 以及需要特判最后刚好是 $10^K - 1$ 的情况.

M 三数之和 by *Elegia*

我们首先将每个数计算出取模到 $[0, M)$ 的结果, 由于给出的 M 比较特殊, 我们可以根据 $10^K = 1$ 把最高位不断降下来, 在 $O(\text{digits})$ 的时间内完成取模运算. 以及需要特判最后刚好是 $10^K - 1$ 的情况.

现在 $a + b + c = 0 \pmod{M}$ 当且仅当它们加起来等于 0, M 或 $2M$, 我们随机选取模数进行 hash 判断就能确定是否满足.

我们首先将每个数计算出取模到 $[0, M)$ 的结果，由于给出的 M 比较特殊，我们可以根据 $10^K = 1$ 把最高位不断降下来，在 $O(\text{digits})$ 的时间内完成取模运算．以及需要特判最后刚好是 $10^K - 1$ 的情况．

现在 $a + b + c = 0 \pmod M$ 当且仅当它们加起来等于 0，M 或 $2M$，我们随机选取模数进行 hash 判断就能确定是否满足．

预处理 hash 的时间为 $O(N \cdot \text{digits})$，之后检验所有答案的时间为 $O(N^3)$，或者再用一个 `map` 解决最后一层循环，时间为 $\tilde{O}(N^2)$．

M 三数之和 by *Elegia*

我们首先将每个数计算出取模到 $[0,M)$ 的结果，由于给出的 M 比较特殊，我们可以根据 $10^K=1$ 把最高位不断降下来，在 $O(\text{digits})$ 的时间内完成取模运算．以及需要特判最后刚好是 10^K-1 的情况．

现在 $a+b+c=0 \pmod M$ 当且仅当它们加起来等于 0，M 或 $2M$，我们随机选取模数进行 hash 判断就能确定是否满足．

预处理 hash 的时间为 $O(N\cdot\text{digits})$，之后检验所有答案的时间为 $O(N^3)$，或者再用一个 `map` 解决最后一层循环，时间为 $\tilde{O}(N^2)$．

一次进位操作会让所有数码的总和 -9，而一开始有最多 $9\cdot\text{digits}$ 的总和，所以进位不超过 digits 次．

Hash 的正确率

我们随机选取模数进行 hash 判断就能确定是否满足.

选取三个 $[10^8, 10^9]$ 内的素数进行取模.

Hash 的正确率

我们随机选取模数进行 hash 判断就能确定是否满足.

选取三个 $[10^8, 10^9]$ 内的素数进行取模.

- 一个不超过 10^{10^5} 大小的数最多有 $10^5/8$ 个超过 10^8 的素因子.

Hash 的正确率

我们随机选取模数进行 hash 判断就能确定是否满足.

选取三个 $[10^8, 10^9]$ 内的素数进行取模.

- 一个不超过 10^{10^5} 大小的数最多有 $10^5/8$ 个超过 10^8 的素因子.
- 在 $[10^8, 10^9]$ 之间有 $4.5 \cdot 10^7$ 个素数.

Hash 的正确率

我们随机选取模数进行 hash 判断就能确定是否满足.

选取三个 $[10^8, 10^9]$ 内的素数进行取模.

- 一个不超过 10^{10^5} 大小的数最多有 $10^5/8$ 个超过 10^8 的素因子.
- 在 $[10^8, 10^9]$ 之间有 $4.5 \cdot 10^7$ 个素数.
- 简单计算可得, 一次测试的错误率为 $2 \cdot 10^{-11}$, 因此一组数据的错误率为 10^{-3} 量级.
- ~~经过考虑, 测试数据里没有针对特定模数构造数据.~~

B 借口 by *Elegia*

- 回顾题意: 给定 N, 求 $p = 1/2$ 的 N 个独立几何分布的数的 mex 的期望.
- $N \leq 10^5$.

B 借口 by *Elegia*

考虑对于一个正整数 k 计算 $\mathrm{mex} \ge k$ 的概率, 记为 p_k, 那么答案可以写作所有 p_k 求和.

B 借口 by *Elegia*

考虑对于一个正整数 k 计算 $\mathrm{mex} \ge k$ 的概率, 记为 p_k, 那么答案可以写作所有 p_k 求和.

对于计算 p_k: 也就是说, 对于 $0, 1, \ldots, k-1$ 这些数都应当出现至少一次.

考虑对于一个正整数 k 计算 $\operatorname{mex} \geq k$ 的概率, 记为 p_k, 那么答案可以写作所有 p_k 求和.

对于计算 p_k: 也就是说, 对于 $0, 1, \ldots, k-1$ 这些数都应当出现至少一次.

我们考虑用指数型生成函数来枚举每种数占据了多少个位置. 数字 i $(0 \leq i \leq k-1)$ 出现概率是 $2^{-(i+1)}$, 且至少出现一次, 更大的数都看做一个数, 出现概率是 2^{-k}, 于是我们有

考虑对于一个正整数 k 计算 $\mathrm{mex} \geq k$ 的概率, 记为 p_k, 那么答案可以写作所有 p_k 求和.

对于计算 p_k: 也就是说, 对于 $0, 1, \ldots, k-1$ 这些数都应当出现至少一次.

我们考虑用指数型生成函数来枚举每种数占据了多少个位置. 数字 i $(0 \leq i \leq k-1)$ 出现概率是 $2^{-(i+1)}$, 且至少出现一次, 更大的数都看做一个数, 出现概率是 2^{-k}, 于是我们有

$$p_k = n![x^n]\left(\mathrm{e}^{x/2} - 1\right)\left(\mathrm{e}^{x/2^2} - 1\right)\cdots\left(\mathrm{e}^{x/2^k} - 1\right)\mathrm{e}^{x/2^k}.$$

B 借口 by *Elegia*

$$p_k = n![x^n]\left(e^{x/2} - 1\right)\left(e^{x/2^2} - 1\right)\cdots\left(e^{x/2^k} - 1\right)e^{x/2^k}.$$

B 借口 by *Elegia*

$$p_k = n![x^n]\left(\mathrm{e}^{x/2}-1\right)\left(\mathrm{e}^{x/2^2}-1\right)\cdots\left(\mathrm{e}^{x/2^k}-1\right)\mathrm{e}^{x/2^k}.$$

记 $xf(x) = \mathrm{e}^x - 1$.

B 借口 by *Elegia*

$$p_k = n![x^n]\frac{x^k}{2^{k(k+1)/2}} f(x/2) \cdots f(x/2^k) \mathrm{e}^{x/2^k}.$$

记 $xf(x) = \mathrm{e}^x - 1$.

$$p_k = n![x^n]\frac{x^k}{2^{k(k+1)/2}} f(x/2)\cdots f(x/2^k)\mathrm{e}^{x/2^k}.$$

记 $xf(x) = \mathrm{e}^x - 1$.

如果我们可以计算无穷乘积 $g(x) = \prod_{k=1}^{\infty} f(x/2^k)$ 的话...

B 借口 by *Elegia*

$$p_k = n![x^n]\frac{x^k}{2^{k(k+1)/2}}g(x)g(x/2^k)^{-1}e^{x/2^k}.$$

记 $xf(x) = e^x - 1$.

如果我们可以计算无穷乘积 $g(x) = \prod_{k=1}^{\infty} f(x/2^k)$ 的话...

可以把连乘积简化成 $g(x)/g(x/2^k)$.

B 借口 by *Elegia*

$$p_k = n![x^n]\frac{x^k}{2^{k(k+1)/2}} g(x) g(x/2^k)^{-1} \mathrm{e}^{x/2^k}.$$

记 $xf(x) = \mathrm{e}^x - 1$.

如果我们可以计算无穷乘积 $g(x) = \prod_{k=1}^{\infty} f(x/2^k)$ 的话...

可以把连乘积简化成 $g(x)/g(x/2^k)$.

这是可行的, 我们已经整理成的 f 的常数项是 1, 取对数之后有

$$\log g(x) = \sum_{k=1}^{\infty} \log f(x/2^k),$$

右侧对应于把第 n 项系数变成 $\sum_{k=1}^{\infty} a_n/2^{nk} = a_n/(2^n-1)$.

$$p_k = n![x^n]\frac{x^k}{2^{k(k+1)/2}}g(x)g({}^{x}\!/\!{}_{2^k})^{-1}\mathrm{e}^{x/2^k}.$$

记 $xf(x) = \mathrm{e}^x - 1$.

如果我们可以计算无穷乘积 $g(x) = \prod_{k=1}^{\infty} f({}^{x}\!/\!{}_{2^k})$ 的话...

可以把连乘积简化成 $g(x)/g({}^{x}\!/\!{}_{2^k})$.

这是可行的, 我们已经整理成的 f 的常数项是 1, 取对数之后有

$$\log g(x) = \sum_{k=1}^{\infty} \log f({}^{x}\!/\!{}_{2^k}),$$

右侧对应于把第 n 项系数变成 $\sum_{k=1}^{\infty} a_n/2^{nk} = a_n/(2^n - 1)$.

另一种思路是利用 $g(x) = g({}^{x}\!/\!{}_{2})f({}^{x}\!/\!{}_{2})$ 这个式子递推计算 $g(x)$ 的系数, 可以半在线卷积 (a.k.a. cdq 分治 FFT). 可能是赛场上更易于实现的做法.

最后我们有

$$\mathrm{Ans} = n![x^n] \sum_{k=1}^{\infty} \frac{x^k}{2^{k(k+1)/2}} g(x)\, \mathrm{g}(x/2^k)^{-1} \mathrm{e}^{x/2^k}.$$

B 借口 by *Elegia*

最后我们有

$$\mathrm{Ans} = n![x^n] \sum_{k=1}^{\infty} \frac{x^k}{2^{k(k+1)/2}} g(x) h(x/2^k).$$

记 $h(x) = g(x)^{-1} \mathrm{e}^x$.

最后我们有

$$\text{Ans} = n![x^n]g(x)\sum_{k=1}^{\infty}\frac{x^k}{2^{k(k+1)/2}}h(x/2^k).$$

记 $h(x) = g(x)^{-1}\mathrm{e}^x$.

最后我们有

$$\text{Ans} = n![x^n]g(x)\sum_{k=1}^{\infty}\frac{x^k}{2^{k(k+1)/2}}h(x/2^k).$$

记 $h(x) = g(x)^{-1}\mathrm{e}^x$.

里面这个卷积展开, 写作

$$a_n = \sum_{k=1}^{\infty} 2^{-k(k+1)/2}\cdot 2^{-k(n-k)}h_{n-k}.$$

B 借口 by *Elegia*

最后我们有

$$\mathrm{Ans} = n![x^n]g(x)\sum_{k=1}^{\infty}\frac{x^k}{2^{k(k+1)/2}}h(x/2^k).$$

记 $h(x) = g(x)^{-1}\mathrm{e}^x$.

里面这个卷积展开, 写作

$$a_n = \sum_{k=1}^{\infty} 2^{-k(k+1)/2}\cdot 2^{-k(n-k)}h_{n-k}.$$

根据做 Chirp Z 变换的经验, 我们可以使用

$$k(n-k) = \binom{n}{2} - \binom{k}{2} - \binom{n-k}{2}$$

分离 k 和 $n-k$ 纠缠的部分. (你会发现分离完这个卷积可以 $O(n)$ 算).

瓶颈是解

$$g(x) = \prod_{k=1}^{\infty} f\left(x/2^k\right)$$

的时候需要做 ln–exp 或者半在线卷积, 复杂度 $O(n\log n)$, 甚至 $O(n\log^2 n)$, $O\left(\frac{n\log^2 n}{\log\log n}\right)$ 或者$\cancel{O\left(n\log n e^{\sqrt{2\log 2\cdot\log\log n}\sqrt{\log\log n}}\right)}$.

G 相似基因序列问题 by *SpiritualKhorosho*

定义两个长度都为 M 的字符串 w_i, w_j 的距离

$$d\left(w_i, w_j\right) = \sum_{m=1}^{M} \left[w_{i,m} \neq w_{j,m}\right].$$

给定模式串 $s_1, \ldots, s_N$. Q 组询问, 每次给出 t_j, 求满足 $d\left(s_i, t_j\right) \leq K$ 的模式串 s_i 的数量.

$1 \leq N, Q \leq 600, 1 \leq M \leq 60,000, 1 \leq K \leq 12.$

我会 $K = 0$

如果 $K = 0$, 问题等价于比较两个字符串是否相同.

可以对模式串预先计算 hash, 每次输入目标串时计算 hash 并比较有多少个模式串的 hash 值相同.

如果 $K = 1$

假设已知两个字符串 s_i, t_j 只在下标 m 处不同. 如果想要验证这个结论是否成立, 只需分别对前缀及后缀求 hash, 即判断 $h\left(s_{i,1}s_{i,2}\cdots s_{i,m-1}\right) = h\left(t_{j,1}t_{j,2}\cdots t_{j,m-1}\right)$ 和 $h\left(s_{i,m+1}s_{i,m+2}\cdots s_{i,M}\right) = h\left(t_{j,m+1}t_{j,m+2}\cdots t_{j,M}\right)$ 是否成立.

现在的问题是, 我们不知道 m 是多少, 甚至不确信两个字符串对应位置不同的下标是否唯一.

如果 $K = 1$

假设已知两个字符串 s_i, t_j 只在下标 m 处不同. 如果想要验证这个结论是否成立, 只需分别对前缀及后缀求 hash, 即判断
$h\left(s_{i,1} s_{i,2} \cdots s_{i,m-1}\right) = h\left(t_{j,1} t_{j,2} \cdots t_{j,m-1}\right)$ 和
$h\left(s_{i,m+1} s_{i,m+2} \cdots s_{i,M}\right) = h\left(t_{j,m+1} t_{j,m+2} \cdots t_{j,M}\right)$ 是否成立.

现在的问题是, 我们不知道 m 是多少, 甚至不确信两个字符串对应位置不同的下标是否唯一.

注意到前缀 hash 值是否相同具有单调性. 记满足 $s_{i,m} \neq t_{j,m}$ 的最小 m 为 m_0, 则:

- 对于 $m < m_0$, 一定有 $h\left(s_{i,1} s_{i,2} \cdots s_{i,m}\right) = h\left(t_{j,1} t_{j,2} \cdots t_{j,m}\right)$;
- 对于 $m \geq m_0$, (有很大的概率) $h\left(s_{i,1} s_{i,2} \cdots s_{i,m}\right) \neq h\left(t_{j,1} t_{j,2} \cdots t_{j,m}\right)$.

因此, 可以先二分查找 m_0, 再通过后缀 hash 是否相等来推测是否只有一个位置不同.

一生二, 二生三, 三生万物

如果会做 $K=1$ 的情况, 那么更大的 K 也可以同理求解:

- 二分求出前 K 个满足 $s_{i,m} \neq t_{j,m}$ 的下标 m (如果找不到 K 个, 那么必有 $d\left(s_i, t_j\right) < K$);
- 假设第 k 个找到的下标为 m_k, 判断 $h\left(s_{i,m_k+1} \cdots s_{i,M}\right)$ 和 $h\left(t_{j,m_k+1} \cdots t_{j,M}\right)$ 是否相同即可.

总复杂度 $O\big((N+Q)M + NQK\log M\big)$.

你说得对，但是我选择分块

可以考虑对字符串分块, 每一块通过 hash 判断是否可能包含差异, 如果 hash 不同才对块内逐字符比较.

本题只需判断 $d\left(s_i, t_j\right)$ 是否不超过 K, 故最多只有 K 块需要块内比较: 若有 $(K+1)$ 块 hash 不同, 则至少有 $(K+1)$ 个字符不同.

假设块大小取 B, 则最坏情况下 hash 的比较次数和逐字符的比较次数之和 $M/B + KB \geq \sqrt{MK}$, 当且仅当 $B = \sqrt{M/K}$ 时取等.

取 $B = O\left(\sqrt{M/K}\right)$, 总复杂度 $O\left((N+Q)M + NQ\sqrt{MK}\right)$.

同样比较暴力, 并且 (当 $M \gg K$ 时) 渐进复杂度没有二分 hash 优秀, 但是跑得比二分 hash 快.

一道题的命运啊, 当然要靠自我奋斗, 但也要考虑到历史的行程.

我实在也不是谦虚, 我一个别的比赛的题怎么到了深圳来了呢.

但是, (已隐藏) 讲, ‘‘大家已经研究决定了”.

所以这道题就到了深圳.

- 题目数量 $3 \to 4$, 喜提深圳出题数最多的出题人 (并列).
- 从别的尚未举行的比赛临时换了这道相对简单的题过来, 所以可能别的某场比赛会多一道防 AK 好题.
- 没有专门构造数据卡特定的 hash 写法, 实现正确的应该都通过了.
- ~~出题人希望, 这道较为简单的题目, 可以给拼搏于 CCPC 逐梦之路上的你, 提供一个有力的援助.~~

Part III (讲者 *liuzhangfeiabc*)

C 报数 III by *liuzhangfeiabc*

回顾题意: 给定 01 串 s, 问有多少个 01 串 t (不含前导 0) 满足 $(t)_2 \le (s)_2$, 且对于任意整数 $k \ge 2$ 均满足 $(t)_k$ 不是 7 的倍数, 其中 $(s)_k$ 表示将串 s 视为 k 进制数的值.

花絮

- 无奖竞猜: 报数 I 和 II 分别是什么题
- 其实这题本来是 $n \cdot 7^6$ 的送温暖题, 结果出题人在写 std 的时候顺手加强了……

容易注意到如下几件事情:

1. 只需要考虑 $1,2,3,4,5,6,7$ 进制 (1 进制是各个数位的和), 因为 k 进制和 $k+7$ 进制 $\bmod 7$ 的值是一样的;
2. 7 进制不是 7 的倍数, 其实就是最低位不是 0;
3. 由费马小定理, 所有 $\bmod 6$ 相同的位其实贡献都是一样的 (除了最低位).

- 然后能想到两个直接的数位 dp:
- $f_1(i,p_1,p_2,p_3,p_4,p_5,p_6,0/1)$: 从高到低考虑到第 i 位,
- 在 1,2,3,4,5,6 进制下的值分别是 p_1,p_2,p_3,p_4,p_5,p_6, 是否卡上界;
- $f_2(i,p_1,p_2,p_3,p_4,p_5,p_6,0/1)$: 从高到低考虑到第 i 位, 下标 $\bmod 6 = 1,2,3,4,5,0$ 的位上 1 的个数分别为 p_1,p_2,p_3,p_4,p_5,p_6，是否卡上界;
- 但是二者的状态数都有 7^6, 需要优化

- 接下来从 f_1 入手优化 (从 f_2 入手可能也有类似的做法, 略);
- 不妨先考虑一个没有上界的问题, 也就是 t 可以是任意 n 位 01 串, 也暂时不考虑最末位;
- 注意 t 进制和 $7-t$ 进制下的区别: 奇数位的贡献取相反数, 偶数位的贡献不变;
- 把奇偶数位拆开, 分别做一个只有 $1,2,3$ 进制的 DP, 状态数就只有 7^3;
- 然后把奇偶数位 $n/2$ 的结果拼起来 ($n/2$ 可能要分别上下取整).

- 然后把奇偶数位 $n/2$ 的结果拼起来:
- 对于 $k = 1,2,3$, 要同时考虑 k 进制和 $-k$ 进制的限制;
- 假设已知偶数位在 k 进制下的值为 p, 则奇数位在 k 进制下的取值 q 必须满足:$p+q \neq 0, p-q \neq 0 \pmod 7$;
- 就相当于至多两个 q 是不能选的;
- 7^3 枚举偶数位 DP 值的一项, 奇数位能与其匹配的项的三维坐标各分别有至多两个不能选;
- 可以容斥掉, 然后前缀和一下, 额外带一个 3^3 即可.

- 至于有了上界以及最低位的限制, 每次枚举卡上界多少位即可, 然后低位是若干个随便填的位, 高位和最低位的 1 定死, 在合并时要加一个 offset;
- $p+p_0+q+q_0 \neq 0, p+p_0-q-q_0 \neq 0 \pmod 7$, 其中 p_0 和 q_0 分别是奇数为和偶数位由那些被定死的位贡献的 offset
- 复杂度: 计算 DP 数组是 $n \cdot 7^3$, 每次合并是 21^3, 总复杂度 $n \cdot 21^3$, 或 $O(n \cdot (3p)^{(p-1)/2})$

H 快速散列变换 by *SpiritualKhorosho*

令 $\mathcal{V} = \left\{0, 1, \cdots, 2^{64} - 1\right\}$. 记单轮散列函数族 $\mathscr{F}$,

$$\begin{aligned} f \in \mathscr{F} \Leftrightarrow \exists A_j \in \mathcal{V}, B \in \mathcal{V}, m, s_j, \circ_j \in \{\vee, \wedge\}, \\ \text{s.t.} \quad f(X) = B \oplus \bigoplus_{j=1}^{m} \left(\left(X \lll s_j\right) \circ_j A_j\right). \end{aligned}$$

给定 N 个 $f_1, f_2, \cdots, f_N \in \mathscr{F}$, 要求完成 Q 组操作, 每组操作为求 $f_r\left(\cdots\left(f_{l-1}\left(f_l(x)\right)\right)\cdots\right)$ 或修改某个 f_i 的参数 (修改最多 C 组) .

$1 \le N, Q \le 20{,}000, C \le 400.$

分析

如果忽略 A_j 及 B, $f(X) = \bigoplus_{j=1}^{m} \left(X \lll s_j\right)$ 可以看成在 $\mathcal{V}$ 上的线性变换. 以 $\mathbb{Z}_2^4$ 为例:

- $T_0 = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$ 是恒等变换, $T_0 \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$;
- $T_1 = \begin{bmatrix} & & & 1 \\ 1 & & & \\ & 1 & & \\ & & 1 & \end{bmatrix}$ 是往高位循环移 1 位, $T_1 \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} d \\ a \\ b \\ c \end{bmatrix}$;
- 变换合成:$(T_0 + T_1)(a, b, c, d)^{\mathsf{T}} = (a + d, b + a, c + b, d + c)^{\mathsf{T}}$.

线性变换可以用矩阵高效维护, 考虑如何扩展矩阵形式, 从而快速维护本题变换.

位运算的性质

对于或运算和与运算,

- 短路:$1 \vee x = 1, 0 \wedge x = 0$;
- 简化:$0 \vee x = 1 \wedge x = x$.

如果 $\circ_j = \vee$ 且对应位上为 0, 或 $\circ_j = \wedge$ 且对应位上为 1, 则与原来处理方式相同.

如果 $\circ_j = \wedge$ 且对应位上为 0, 则直接把矩阵对应的位置设为 0.

- 例:$\overline{dcba} \wedge 1101 = \overline{dc0a} \Rightarrow \begin{bmatrix} 1 & & & \\ & 0 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} a \\ 0 \\ c \\ d \end{bmatrix}$.

如果 $\circ_j = \vee$ 且对应位上为 1, 则按位或结果必定为 1. 等价于矩阵对应位置设为 0, 同时结果对应位异或 1;后者可以用 B 对应位异或 1 实现.

从线性变换到仿射变换

本题中可以看成在 $\mathbb{Z}_2$ 下进行矩阵运算, $\oplus$ 和 $+$ 等价.

$$
\begin{aligned}
f(X) &= B \oplus \bigoplus_{j=1}^{m} \left(\left(X \lll s_j \right) \circ_j A_j \right) \\
&\Rightarrow b + \sum_{j=1}^{m} T_j X = b + \left(\sum_{j=1}^{m} T_j \right) X.
\end{aligned}
$$

可以用仿射变换维护本题的操作. 仿射变换可以按如下方式进行复合:

$$
A'(Ax+b)+b' = A'Ax + A'b + b' = (A'A)x + (A'b+b').
$$

故用线段树维护区间 A, b 信息即可.

位运算加速位运算

令 $w=64$, 直接存 $w\times w$ 矩阵 A 和长度为 w 的向量 b, 建树 $O\left(Nw^3\right)$, 查询 $O\left(Qw^2\log N\right)$, 修改 $O\left(Cw^3\log N\right)$, 不能通过本题.

注意到是在 $\mathbb{Z}_2$ 下进行运算, 压位后用异或等位运算计算即可.

- 例: $\begin{bmatrix} a_0 & a_1 & a_2 & a_3 \end{bmatrix}(1,0,1,1)^{\mathsf{T}}+b=a_0\oplus a_2\oplus a_3\oplus b.$

使用位运算加速, 总复杂度 $O\left(Nw^2+Qw\log N+Cw^2\log N\right)$.

查询时不是每次矩阵乘向量, 而是先求出 $[l,r]$ 变换的复合再代值的做法 (即询问多个 $O(w)$) 理论上被卡掉了, 除非常数足够小.

对于补全到 N 位的非负整数 $X=\overline{x_1x_2\cdots x_N}$, 对 X 进行一次变换 $f(X)=\overline{y_1y_2\cdots y_N}$ 的规则为:

- $y_1=(x_1+1) \bmod 10$;
- $\forall i\ge 2, y_i=\begin{cases}(x_i+1) \bmod 10, & x_i=x_1\land\forall j<i, x_j\le x_i,\\ x_i, & \text{otherwise.}\end{cases}$

求 $\sum_{X=L}^{R}\min\left\{t\,\middle|\,t\ge 0\land f^{(t)}(X)=\overline{0\cdots 0}\right\}$, 其中 $f^{(t)}$ 表示 f 迭代 t 次.

$1\le N\le 5000, 0\le L\le R<10^N$.

观察

考虑能否单独算每一位的贡献. 如果根据某种简单的规则可以确定当前位是否贡献次数, 就可以使用数位 DP 统计答案.

对于最高位, 一定要转 $(10 - x_1) \bmod 10$ 次才能清零; 对于其它位, 有可能贡献 10 次旋转, 也可能不贡献额外次数.

- 对于 0070, 7 贡献 10 次旋转;
- 对于 0370, 7 不贡献额外次数 (随着 3 一起清零) ;
- 对于 0970, 7 贡献 10 次旋转 $(0970 \to 9970 \to 0070)$.

猜测: 需要根据上一个数码判断是否贡献?

反例

小反例
对于 0070, 7 贡献 10 次旋转.

- 对于 3070, 7 会随着 3 一起清零, 因此不额外贡献次数.

猜测: 需要根据上一个非 0 位判断?

反例

小反例
对于 0070, 7 贡献 10 次旋转.

- 对于 3070, 7 会随着 3 一起清零, 因此不额外贡献次数.

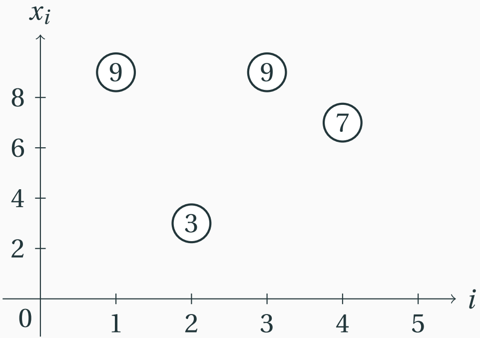
猜测: 需要根据上一个非 0 位判断?

大反例
对于 0970, 7 贡献 10 次旋转 $(0970 \to 9970 \to 0070)$.

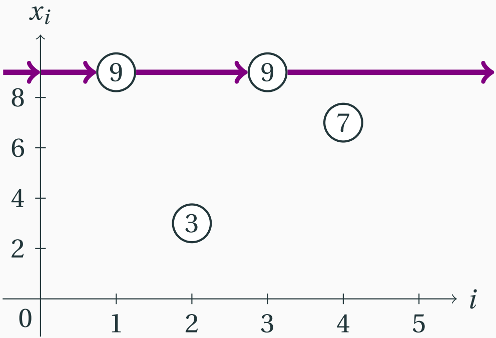
- 对于 9397, 7 会随着 3 一起清零, 因此不额外贡献次数.

如果不幸踩了以上反例的坑, 大概会在样例上卡一段时间.

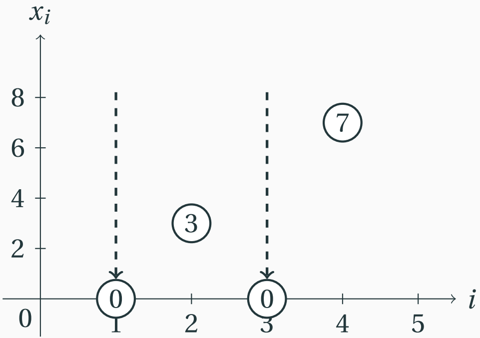
分析



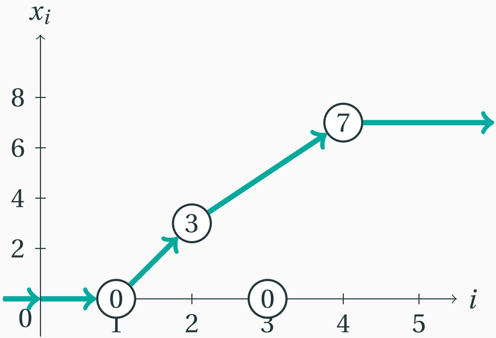
分析



分析

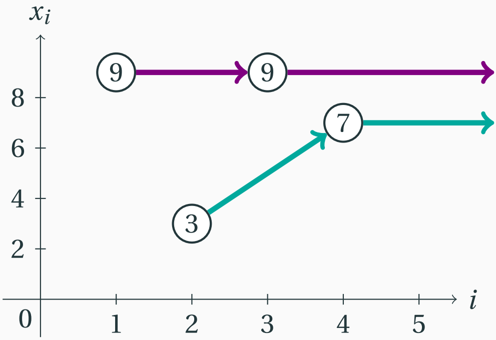


分析



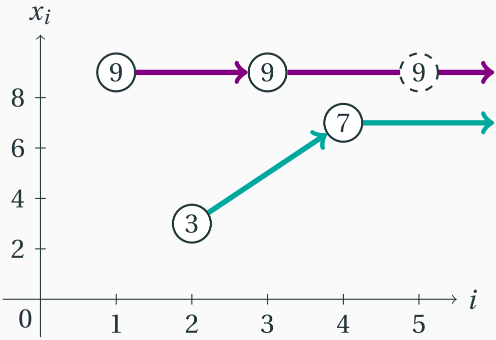
分析

- 可以看出，需要维护前缀拆分成若干个不降子序列的信息。拆分出的每条不降子序列同时完成清零。
- 加入新的一位时，0 不影响当前的子序列；其它数位 d 都贪心地找不大于 d 的最大结尾，或开启一条新的子序列。



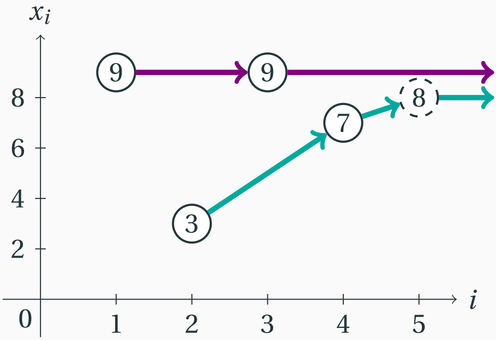
分析

- 可以看出，需要维护前缀拆分成若干个不降子序列的信息.
  拆分出的每条不降子序列同时完成清零.
- 加入新的一位时，0 不影响当前的子序列；其它数位 d 都贪心地找不大于 d 的最大结尾，或开启一条新的子序列.



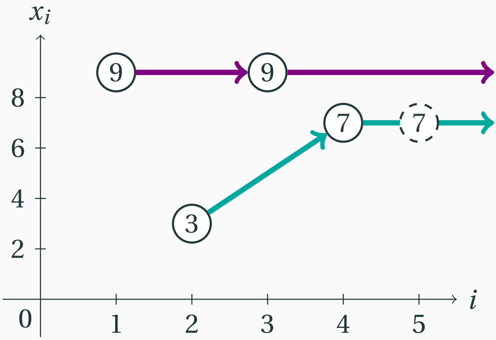
分析

- 可以看出，需要维护前缀拆分成若干个不降子序列的信息.
  拆分出的每条不降子序列同时完成清零.
- 加入新的一位时，0 不影响当前的子序列；其它数位 d 都贪心地找不大于 d 的最大结尾，或开启一条新的子序列.



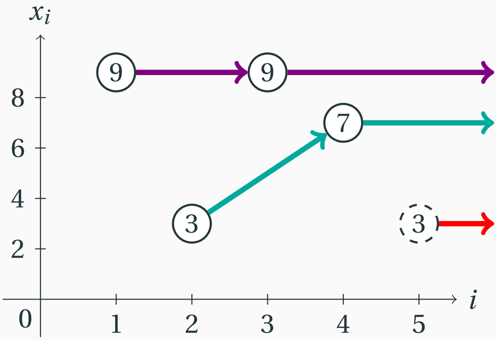
分析

- 可以看出，需要维护前缀拆分成若干个不降子序列的信息.
  拆分出的每条不降子序列同时完成清零.
- 加入新的一位时，0 不影响当前的子序列；其它数位 d 都贪心地找不大于 d 的最大结尾，或开启一条新的子序列.



分析

- 可以看出，需要维护前缀拆分成若干个不降子序列的信息．拆分出的每条不降子序列同时完成清零．
- 加入新的一位时，0 不影响当前的子序列；其它数位 d 都贪心地找不大于 d 的最大结尾，或开启一条新的子序列．



解法

显然任意时刻每个数字最多只会出现在一条子序列的结尾, 因为有相同数字加入时一定会连在同一子序列后面. 因此, 可以通过状压是否存在以 d 为结尾的子序列表示当前所有子序列状态.

记 $f(i,S,u;X)$ 表示对 $[\overline{0\cdots0},X]$ 进行统计, 已确定前 i 位时子序列状态为 S, 确定的前缀是否顶上界 ($u=1$ 顶上界) 时可能的前缀所需次数之和. 为方便统计答案, 另记辅助 $Count(i,S,u;X)$ 表示该状态对应的前缀数量. 枚举第 $(i+1)$ 位为 d,

- 如果 d 对应一条新的子序列, 则 $f\left(i+1,S\cup\{d\},u';X\right)$ 需要加上 $f(i,S,u;X)+10Count(i,S,u;X)$;
- 否则, $f\left(i+1,S',u';X\right)$ 只加 $f(i,S,u;X)$, 其中 S' 为加入 d 后的新状态.

答案即 $\sum_S\sum_{u=0}^{1}\left(f(N,S,u;R)-f(N,S,u;L-1)\right)$. 记 $\Sigma=\{0,1,\cdots,9\}$ 表示字符集, 则总复杂度为 $O\left(N|\Sigma|2^{|\Sigma|}\right)$. 也可以直接根据 $Count$ 统计答案, 具体做法此处不详述.

坑

如果想到做法但没有通过本题, 可能的原因包括但不限于:

- 需要求 $(L-1)$, 但是没考虑 $L=\overline{\cdots 00}$ 甚至 $\overline{0\cdots 0}$ 的情况;
- 处理顶上界的时候没有考虑中间有 0, 或者有前导 0 的情况 (如果细节没处理清楚, 也有可能在上界有 1 的时候, 或者没顶上界但前缀全 0 时出错) ;
  - 根据参与验题的校内队伍的反馈, 补充了样例 3.

坑

如果想到做法但没有通过本题, 可能的原因包括但不限于:

- 需要求 $(L-1)$, 但是没考虑 $L=\overline{\cdots 00}$ 甚至 $\overline{0\cdots 0}$ 的情况；
- 处理顶上界的时候没有考虑中间有 0, 或者有前导 0 的情况 (如果细节没处理清楚, 也有可能在上界有 1 的时候, 或者没顶上界但前缀全 0 时出错)；
    - 根据参与验题的校内队伍的反馈, 补充了样例 3.
- 分别统计 $[\overline{0\cdots 0}, L-1]$ 和 $[\overline{0\cdots 0}, R]$ 的答案, 直接对减得到负数；

坑

如果想到做法但没有通过本题, 可能的原因包括但不限于:

- 需要求 $(L-1)$, 但是没考虑 $L=\overline{\cdots 00}$ 甚至 $\overline{0\cdots 0}$ 的情况;
- 处理顶上界的时候没有考虑中间有 0, 或者有前导 0 的情况 (如果细节没处理清楚, 也有可能在上界有 1 的时候, 或者没顶上界但前缀全 0 时出错) ;
  - 根据参与验题的校内队伍的反馈, 补充了样例 3.
- 分别统计 $[\overline{0\cdots 0}, L-1]$ 和 $[\overline{0\cdots 0}, R]$ 的答案, 直接对减得到负数;
- 用 $f(\cdot,\cdot,\cdot;\cdot)=0$ 判断当前状态是否有效, 没有考虑 $Count$ 是否为 0 (刚好随到了一个点) .

彩蛋 (部分)

样例 2: 100084 518118
邮政编码.

样例 3: 068493 205479
好像在哪里见过......?

样例 4: 040139021316 234700825190
md5("counter") = 0c40fb1390213162b347eb0fd0825190.

Part IV (讲者 *Mys.C.K.*)

F 见面礼 by *ltst*

给定一个 n 个点 n 条边的无向图，图没有重边自环。

你需要求有多少种选择图上的一个点 p 和一条边 (x, y) 的方案，使得删去 (x, y) 后图变成一棵树，且这棵树以 p 为根时每个节点的儿子个数均不超过 3。

保证至少存在一种这样的方案。

$2 \le n \le 10^5$

题解

由于保证存在一个方案，所以给出的图一定是一棵基环树。找到基环树的环后，删去的边一定在环上。

题解

由于保证存在一个方案，所以给出的图一定是一棵基环树。找到基环树的环后，删去的边一定在环上。

枚举被删去的边（也就是环上的边），我们需要快速地确定选根的方案数。

题解

由于保证存在一个方案，所以给出的图一定是一棵基环树。找到基环树的环后，删去的边一定在环上。

枚举被删去的边（也就是环上的边），我们需要快速地确定选根的方案数。

考虑一个点作为根的条件。注意到每个点的儿子个数不超过 3 的充要条件是：根的度数不超过 3，其余节点的度数不超过 4。

题解

由于保证存在一个方案，所以给出的图一定是一棵基环树。找到基环树的环后，删去的边一定在环上。

枚举被删去的边（也就是环上的边），我们需要快速地确定选根的方案数。

考虑一个点作为根的条件。注意到每个点的儿子个数不超过 3 的充要条件是：根的度数不超过 3，其余节点的度数不超过 4。

于是维护每种度数的出现次数（注意到保证有解时，所有节点的度数均不会超过 5），删边时修改相邻的两个节点的度数。当不存在度数为 5 的节点时，选根的方案数即为度数不超过 3 的节点个数，否则不存在选根的方案。

回顾题意: 求不定方程 $a^k - b^k = n$ 的正整数解的数量.

20 组询问, $1 \le n \le 10^{18}$, $3 \le k \le 64$.

注意到, $a^k - b^k = (a-b)(a^{k-1} + a^{k-2}b + \cdots + b^{k-1}) \geq (a-b)^k$, 我们有 $a - b \leq n^{1/k}$, 可以枚举 $a - b$ 的值, 然后解剩下的多项式方程.

注意到, $a^k - b^k = (a-b)(a^{k-1} + a^{k-2}b + \cdots + b^{k-1}) \geq (a-b)^k$, 我们有 $a - b \leq n^{1/k}$, 可以枚举 $a - b$ 的值, 然后解剩下的多项式方程. 对于 $k = 3$, 可以使用二次方程求根公式较快计算, 对于 $k \geq 4$, 二分解方程的时间是足够的.

L Mary 有颗有根树 by *SpiritualKhorosho*

给定一棵只有根结点的有根树. 每次操作时随机选取一个叶结点, 并为该结点添加恰好 M 个子结点. 求 K 次操作后整棵树的结点深度总和的期望, 答案对 $1,000,000,009$ 取模. 定义根结点的深度为 0.

$1 \leq M \leq 100, 1 \leq K \leq 10^7$.

Mary 是谁?
Wikipedia 上表示一棵树的子结点个数不超过某个给定常数的条目名称是 m-ary tree.

分析

期望具有线性性. 记第 i 次操作后树为 T_i, 第 i 次添加的 M 个子结点的深度均为 D_i, 则

$$\begin{aligned}E\left(\sum_{v\in T_K} depth(v)\right) &= E\left(\sum_{v\in T_0} depth(v)+\sum_{i=1}^{K}\sum_{v\in T_i\setminus T_{i-1}} depth(v)\right)\\ &= 0+M\sum_{i=1}^{K} E\left(D_i\right).\end{aligned}$$

$E\left(D_i\right)$ 取决于被选出的结点的深度. 对于有根树 T, 记 $\mathscr{L}(T)$ 为 T 的所有叶结点组成的集合, $\mathscr{I}(T)=T\setminus\mathscr{L}(T)$ 为所有内部结点组成的集合, 则

$$E\left(D_i|T_{i-1}\right)=\frac{\sum_{v\in\mathscr{L}(T_{i-1})} depth(v)}{\left|\mathscr{L}\left(T_{i-1}\right)\right|}+1=\frac{\sum_{v\in\mathscr{L}(T_{i-1})} depth(v)}{(M-1)(i-1)+1}+1.$$

转换

根据重期望公式:

$$E(D_i) = E(E(D_i | T_{i-1})) = \frac{E\left(\sum_{v \in \mathscr{L}(T_{i-1})} depth(v)\right)}{(M-1)(i-1)+1} + 1,$$

问题转换为求每次操作后叶结点深度和的期望.

转换

根据重期望公式：

$$E(D_i) = E(E(D_i | T_{i-1})) = \frac{E\left(\sum_{v \in \mathscr{L}(T_{i-1})} depth(v)\right)}{(M-1)(i-1)+1} + 1,$$

问题转换为求每次操作后叶结点深度和的期望.假设第 i 次从 $\mathscr{L}(T_{i-1})$ 中选出一个深度为 d 的结点 v, 则

$$\begin{aligned} \sum_{v \in \mathscr{L}(T_i)} depth(v) &= \sum_{v \in \mathscr{L}(T_{i-1})} depth(v) - d + M(d+1) \\ &= \sum_{v \in \mathscr{L}(T_{i-1})} depth(v) + (M-1)d + M. \end{aligned}$$

根据重期望公式:

$$E(D_i) = E(E(D_i | T_{i-1})) = \frac{E\left(\sum_{v \in \mathscr{L}(T_{i-1})} depth(v)\right)}{(M-1)(i-1)+1} + 1,$$

问题转换为求每次操作后叶结点深度和的期望.假设第 i 次从 $\mathscr{L}(T_{i-1})$ 中选出一个深度为 d 的结点 v, 则

$$\begin{aligned}\sum_{v \in \mathscr{L}(T_i)} depth(v) &= \sum_{v \in \mathscr{L}(T_{i-1})} depth(v) - d + M(d+1) \\ &= \sum_{v \in \mathscr{L}(T_{i-1})} depth(v) + (M-1)d + M.\end{aligned}$$

可以对两边分别关于 T_{i-1} 求期望, 得到与叶结点深度和的期望有关的表达式.

推式子

两边分别关于 T_{i-1} 求期望得：

$$\begin{aligned}
&E\left(\sum_{v\in\mathscr{L}(T_i)} depth(v)\right)\\
=&E\left(\sum_{v\in\mathscr{L}(T_{i-1})} depth(v)\right)+(M-1)E(d)+M\\
=&\left(1+\frac{M-1}{(M-1)(i-1)+1}\right)E\left(\sum_{v\in\mathscr{L}(T_{i-1})} depth(v)\right)+M\\
=&\frac{(M-1)i+1}{(M-1)(i-1)+1}E\left(\sum_{v\in\mathscr{L}(T_{i-1})} depth(v)\right)+M.
\end{aligned}$$

递推求解

记 $A_i = E\left(\sum_{v\in\mathcal{L}(T_i)} depth(v)\right)$, 则有递推式

$$A_i = \frac{(M-1)i+1}{(M-1)(i-1)+1} A_{i-1} + M$$
$$\Rightarrow Ans = M\sum_{i=1}^{K} E(D_i) = M\sum_{i=0}^{K-1}\left(\frac{A_i}{(M-1)i+1}+1\right),$$

可以递推求解. 另外, 如果记 $B_i = E\left(\sum_{v\in\mathcal{I}(T_i)} depth(v)\right)$, 根据 $\sum_{v\in\mathcal{I}(T_i)} depth(v) = \sum_{v\in\mathcal{I}(T_{i-1})} depth(v) + d$, 同理可得 $B_i = B_{i-1} + A_{i-1}/((M-1)(i-1)+1)$, 答案即 $A_K + \sum_{k=1}^{K} B_k$. 两种做法的复杂度都是 $O\left(K+\log P\right)$ (P 表示模数) .

- 用类似阶乘逆元的方式预处理, 就只需一次 $O(\log P)$ 逆元.
- 没有特意卡 $O\left(K\log P\right)$ ~~除非常数偏大, 比如算了 $2K$ 次逆元~~

简化

根据递推式中系数的性质可以简化表达式，但是...

$$A_i = \frac{(M-1)i+1}{(M-1)(i-1)+1}\left(\frac{(M-1)(i-1)+1}{(M-1)(i-2)+1}A_{i-2}+M\right)+M$$

$$= \cdots = M\sum_{j=1}^{i}\frac{(M-1)i+1}{(M-1)j+1}$$

$$\Rightarrow Ans = M\sum_{i=0}^{K-1}\left(\frac{M\sum_{j=1}^{i}\frac{(M-1)i+1}{(M-1)j+1}}{(M-1)i+1}+1\right) = M\sum_{j=1}^{K}\frac{MK+1-j}{(M-1)j+1}$$

- 求等差数列的倒数和存在 $O(\sqrt{K}\log K)$ 的算法，不过需要分块 FFT，超出了本题定位.

思考
复杂度和 M 没有关系，为什么 M 只出到 100?

感谢倾听!