题解

Harry27182

2024 年 11 月 19 日

正解

- 考虑构造若干个完全图，并以一条链的形式连接这些完全图。

正解

- 考虑构造若干个完全图，并以一条链的形式连接这些完全图。
- 一个大小为 n 的完全图能形成 $\frac{n(n-1)(n-2)}{6}$ 个三元环，由于当 $n=3$ 时三元环数量为 1，所以显然可以通过上述方式构造三元环数量为 n 的图。

正解

- 考虑构造若干个完全图，并以一条链的形式连接这些完全图。
- 一个大小为 n 的完全图能形成 $\frac{n(n-1)(n-2)}{6}$ 个三元环，由于当 $n=3$ 时三元环数量为 1，所以显然可以通过上述方式构造三元环数量为 n 的图。
- 使用尽可能大的完全图来拼接，容易发现需要的点数级别为 $O(n^{\frac{1}{3}})$，打表发现 500 个点是够用的。

设计 dp

- 考虑容斥原理，钦定若干个位置满足限制。我们发现在知道相邻两个钦定位置和对应的钦定值的时候，中间的方案是可以计算的。

设计 dp

- 考虑容斥原理，钦定若干个位置满足限制。我们发现在知道相邻两个钦定位置和对应的钦定值的时候，中间的方案是可以计算的。
- 设 $dp_{i,j}$ 表示考虑了前 i 个位置且钦定第 i 个位置满足限制，$\max\{a_1, a_2..., a_i\} = j$ 的容斥系数之和。

设计 dp

- 考虑容斥原理，钦定若干个位置满足限制。我们发现在知道相邻两个钦定位置和对应的钦定值的时候，中间的方案是可以计算的。
- 设 $dp_{i,j}$ 表示考虑了前 i 个位置且钦定第 i 个位置满足限制，$\max\{a_1, a_2..., a_i\} = j$ 的容斥系数之和。
- 考虑转移，计算从 $dp_{a,b}$ 转移到 $dp_{c,d}$ 的方案数。$[a+1, c]$ 的值显然在 $[b, d]$ 之间，且必须有 b 和 d，方案数可以简单计算。

设计 dp

- 考虑容斥原理，钦定若干个位置满足限制。我们发现在知道相邻两个钦定位置和对应的钦定值的时候，中间的方案是可以计算的。
- 设 $dp_{i,j}$ 表示考虑了前 i 个位置且钦定第 i 个位置满足限制，$\max\{a_1, a_2 ..., a_i\} = j$ 的容斥系数之和。
- 考虑转移，计算从 $dp_{a,b}$ 转移到 $dp_{c,d}$ 的方案数。$[a+1, c]$ 的值显然在 $[b, d]$ 之间，且必须有 b 和 d，方案数可以简单计算。
- 复杂度 $O(n^2m^2)$，期望得分 60 分。

优化

- 把转移系数写出来，是 $(d-b+1)^{c-a}-2(d-b)^{c-a}+(d-b-1)^{c-a}$，对于每一项分别计算。

优化

- 把转移系数写出来，是 $(d-b+1)^{c-a}-2(d-b)^{c-a}+(d-b-1)^{c-a}$，对于每一项分别计算。
- 每一项可以写成 x^{c-a} 的形式，也就可以写成 $\frac{x^c}{x^a}$ 的形式。

优化

- 把转移系数写出来，是 $(d-b+1)^{c-a}-2(d-b)^{c-a}+(d-b-1)^{c-a}$，对于每一项分别计算。
- 每一项可以写成 x^{c-a} 的形式，也就可以写成 $\frac{x^c}{x^a}$ 的形式。
- 所以可以把系数拆成关于 a 和 c 独立的两部分，可以配合前缀和算法做到 $O(nm^2)$，期望得分 80 分。

正解

- 容易发现，答案是关于 m 的 $n+1$ 次多项式，可以通过拉格朗日插值处理 m 很大的情况。

正解

- 容易发现，答案是关于 m 的 $n+1$ 次多项式，可以通过拉格朗日插值处理 m 很大的情况。
- 由于 $n \leq 400$，不会拉格朗日插值也可以使用高斯消元通过待定系数法求出多项式系数。

正解

- 容易发现，答案是关于 m 的 $n+1$ 次多项式，可以通过拉格朗日插值处理 m 很大的情况。
- 由于 $n \leq 400$，不会拉格朗日插值也可以使用高斯消元通过待定系数法求出多项式系数。
- 复杂度 $O(n^3)$，期望得分 100 分。

观察

- 首先观察到，一个 a_i 经过 x 次操作之后得到的数为 $(a_i - x)2^x$。证明考虑归纳。

观察

- 首先观察到，一个 a_i 经过 x 次操作之后得到的数为 $(a_i - x)2^x$。证明考虑归纳。
- 如果 $a_i - x$ 为偶数，那么最终序列要得到这个数，初值并不是 a_i，可以用比 a_i 更小的数代替。

观察

- 首先观察到，一个 a_i 经过 x 次操作之后得到的数为 $(a_i - x)2^x$。证明考虑归纳。
- 如果 $a_i - x$ 为偶数，那么最终序列要得到这个数，初值并不是 a_i，可以用比 a_i 更小的数代替。
- 所以对于每个数 x，以 x 为初值的位置个数至少有 $\lfloor \frac{x}{2} \rfloor$ 个，可以在 $O(\sqrt{n})$ 时间内解决第一问。

第二问

- 考虑对于确定的初值 a_i 和操作次数 c_i 求出对应的排名。将最终数的形态记作 $x2^y$，其中 x 为奇数。

第二问

- 考虑对于确定的初值 a_i 和操作次数 c_i 求出对应的排名。将最终数的形态记作 $x2^y$，其中 x 为奇数。
- 显然只需要关心 $y' \in [y - \log n, y + \log n]$ 的 (x', y') 和 (x, y) 的大小关系，对于每个 y' 可以 $O(1)$ 计算，也就是对于一个确定的 a_i 和 c_i 可以 $O(\log n)$ 计算它的排名。

第二问

- 可以通过二分找到最小的 y 满足 $x=1$ 时 2^y 的排名在 b_i 之后的 y，那么同理排名为 b_i 的 y' 一定满足 $y' \in [y-\log n, y]$。

第二问

- 可以通过二分找到最小的 y 满足 $x=1$ 时 2^y 的排名在 b_i 之后的 y，那么同理排名为 b_i 的 y' 一定满足 $y' \in [y-\log n, y]$。
- 对于每一个 y' 二分求出第一个 x 使得 $x2^{y'}$ 的排名在 b_i 之后就可以得到答案，复杂度 $O(q\log^3 n)$，期望得分 80 分。

优化

- 瓶颈在于 $O(\log n)$ 次二分。考虑优化。

优化

- 瓶颈在于 $O(\log n)$ 次二分。考虑优化。
- 在第 p 行的 x 确定之后，可以通过 $O(1)$ 次 check 来得到第 $p+1$ 行对应的 x。

优化

- 瓶颈在于 $O(\log n)$ 次二分。考虑优化。
- 在第 p 行的 x 确定之后，可以通过 $O(1)$ 次 check 来得到第 $p+1$ 行对应的 x。
- 具体来说，第 $p+1$ 行的 x 一定在第 p 行的 $\frac{x}{2}$ 附近 $O(1)$ 个位置。时间复杂度优化到 $O(q\log^2 n)$，可以通过。

操作二

- 对于只有操作二和操作四的情况，显然操作二的操作次数不会超过 $d(n)$，也就是不会超过 $O(\sqrt{n})$。

操作二

- 对于只有操作二和操作四的情况，显然操作二的操作次数不会超过 $d(n)$，也就是不会超过 $O(\sqrt{n})$。
- 所以问题变为了 $O(q\sqrt{n})$ 次单点修改和 $O(q)$ 次链查询。首先可以套路地通过 dfs 序转化为区间修改和单点查询，使用分块进行平衡即可做到 $O(q\sqrt{n})$。

操作三

- 设阈值 B，对于 $X > B$ 的修改，单点修改的位置数不会超过 $O(\frac{n}{B})$，所以可以用类似操作二的做法来完成。

操作三

- 设阈值 B，对于 $X > B$ 的修改，单点修改的位置数不会超过 $O(\frac{n}{B})$，所以可以用类似操作二的做法来完成。
- 对于每个 $x \leq B$，可以通过前缀和差分通过 $O(n)$ 时间的预处理，$O(1)$ 查询任意一条链上是 x 的倍数的点的数量。

操作三

- 设阈值 B，对于 $X > B$ 的修改，单点修改的位置数不会超过 $O(\frac{n}{B})$，所以可以用类似操作二的做法来完成。
- 对于每个 $x \le B$，可以通过前缀和差分通过 $O(n)$ 时间的预处理，$O(1)$ 查询任意一条链上是 x 的倍数的点的数量。
- 所以可以开桶记录每个 x 上修改之和，查询的时候枚举 x 查询链上是 x 的倍数的点数量。视为 n, q 同阶，取 $B = \sqrt{n}$ 可得到最优复杂度 $O(n\sqrt{n})$。

操作一

- 考虑定期重构，每 B 个重构一次。每次重构可以通过 dfs 预处理这次重构之前的操作对每个点的贡献，前缀和预处理后可以 $O(1)$ 查询。复杂度 $O(\frac{nq}{B})$。

操作一

- 考虑定期重构，每 B 个重构一次。每次重构可以通过 dfs 预处理这次重构之前的操作对每个点的贡献，前缀和预处理后可以 $O(1)$ 查询。复杂度 $O(\frac{nq}{B})$。
- 对于本块内的贡献，考虑每个修改对每个询问的贡献，可以将询问链差分成四段根到某个点 x 的链。

操作一

- 考虑定期重构，每 B 个重构一次。每次重构可以通过 dfs 预处理这次重构之前的操作对每个点的贡献，前缀和预处理后可以 $O(1)$ 查询。复杂度 $O(\frac{nq}{B})$。
- 对于本块内的贡献，考虑每个修改对每个询问的贡献，可以将询问链差分成四段根到某个点 x 的链。
- 在重构时对上一块块内贡献统一处理，进行一遍 dfs，dfs 到每个点时用数据结构维护从根到当前点每个位置是否出现。对于每个查询操作去枚举每个修改操作，在数据结构上查询当前这条链上多少点位于 $[l, r]$ 之间。

操作一

- 考虑定期重构，每 B 个重构一次。每次重构可以通过 dfs 预处理这次重构之前的操作对每个点的贡献，前缀和预处理后可以 $O(1)$ 查询。复杂度 $O(\frac{nq}{B})$。
- 对于本块内的贡献，考虑每个修改对每个询问的贡献，可以将询问链差分成四段根到某个点 x 的链。
- 在重构时对上一块块内贡献统一处理，进行一遍 dfs，dfs 到每个点时用数据结构维护从根到当前点每个位置是否出现。对于每个查询操作去枚举每个修改操作，在数据结构上查询当前这条链上多少点位于 $[l, r]$ 之间。
- 数据结构使用分块可以做到 $O(qB + \frac{nq}{B})$，视为 n, q 同阶，取 $B = \sqrt{n}$ 可得到最优复杂度 $O(n\sqrt{n})$

正解

- 加法操作满足交换律和结合律，也就是三种操作对答案的贡献可以独立计算之后相加。

正解

- 加法操作满足交换律和结合律，也就是三种操作对答案的贡献可以独立计算之后相加。
- 复杂度 $O(n\sqrt{n})$，期望得分 100 分。

致谢

谢谢大家！