题解

Harry27182

2024 年 11 月 19 日

正解

- 设 $f_{u,s}$ 表示以 u 为根，出现过 s 内颜色的链数量，转移显然。

正解

- 设 $f_{u,s}$ 表示以 u 为根，出现过 s 内颜色的链数量，转移显然。
- 计算答案的时候考虑让每条路径在 LCA 处合并，枚举两个方向的 s 判断并集是否包含三种颜色即可。复杂度 $O(n)$。

设计 dp

- 设 f_i 表示将 $1 \sim i$ 划分成若干个能拿一等奖的段的方案数，$w(i,j)$ 表示 $[i,j]$ 是否满足条件。

设计 dp

- 设 f_i 表示将 $1 \sim i$ 划分成若干个能拿一等奖的段的方案数，$w(i,j)$ 表示 $[i,j]$ 是否满足条件。
- 显然有转移 $f_i = \sum_{j=1}^{i-1} f_j w(j+1,i)$，复杂度 $O(n^2)$，期望得分 40 分。

优化

- 考虑优化，首先发现 i 只能从与其奇偶性相同的 j 转移过来。

优化

- 考虑优化，首先发现 i 只能从与其奇偶性相同的 j 转移过来。
- 在这个基础上，不难发现，j 能够转移到的 i 是一段区间，能够转移到 i 的 j 也是一段区间。

优化

- 考虑优化，首先发现 i 只能从与其奇偶性相同的 j 转移过来。
- 在这个基础上，不难发现，j 能够转移到的 i 是一段区间，能够转移到 i 的 j 也是一段区间。
- 用树状数组维护，对于每个 j，在处理完之后加入树状数组，在 i 已经超出它能转移的区间之后删除树状数组，就可以处理前一个限制。

优化

- 考虑优化，首先发现 i 只能从与其奇偶性相同的 j 转移过来。
- 在这个基础上，不难发现，j 能够转移到的 i 是一段区间，能够转移到 i 的 j 也是一段区间。
- 用树状数组维护，对于每个 j，在处理完之后加入树状数组，在 i 已经超出它能转移的区间之后删除树状数组，就可以处理前一个限制。
- 对于后一个限制，树状数组区间查询即可。复杂度 $O(n \log n)$，期望得分 $75 \sim 85$ 分。

正解

- 考虑从 $i-2$ 变成 i 的时候继承部分信息。实时维护对于后一个限制合法的 j 的范围 $[l,r]$，处理它的变化。

正解

- 考虑从 $i-2$ 变成 i 的时候继承部分信息。实时维护对于后一个限制合法的 j 的范围 $[l,r]$，处理它的变化。
- 如果 $s_{i-1}\neq 0, s_i\neq 0$，不难发现范围变为 $[l-2,r+2]$。

正解

- 考虑从 $i-2$ 变成 i 的时候继承部分信息。实时维护对于后一个限制合法的 j 的范围 $[l,r]$，处理它的变化。
- 如果 $s_{i-1}\neq 0, s_i\neq 0$，不难发现范围变为 $[l-2,r+2]$。
- 否则如果 $s_i\neq 0$，那么变为只能从 $i-2$ 转移过来。否则不存在合法转移位置。这些操作都可以在 $O(1)$ 做到。

正解

- 考虑从 $i-2$ 变成 i 的时候继承部分信息。实时维护对于后一个限制合法的 j 的范围 $[l,r]$，处理它的变化。
- 如果 $s_{i-1}\neq 0, s_i\neq 0$，不难发现范围变为 $[l-2,r+2]$。
- 否则如果 $s_i\neq 0$，那么变为只能从 $i-2$ 转移过来。否则不存在合法转移位置。这些操作都可以在 $O(1)$ 做到。
- 由于对于每个 j，能转移到的 i 是一段区间，那么只需要在 $[l,r]$ 变化的时候判断一下发生变化的位置是否能转移到 i 并同时维护转移量即可。还需要在 i 超出能转移的范围时从转移量中删除。复杂度 $O(n)$，期望得分 100 分。

部分分

- 首先观察到，query 操作的复杂度不会超过 $O(\log n)$。

部分分

- 首先观察到，query 操作的复杂度不会超过 $O(\log n)$。
- 对于 $n \leq 2 \times 10^5$ 的情况，容易套路地想到将 x 向 $x + lowbit(x)$ 连一条边，这样可以构成一个森林的结构。

部分分

- 首先观察到，query 操作的复杂度不会超过 $O(\log n)$。
- 对于 $n \leq 2 \times 10^5$ 的情况，容易套路地想到将 x 向 $x + lowbit(x)$ 连一条边，这样可以构成一个森林的结构。
- 修改操作就是对一条从根到 x 的链异或上一个值，查询操作是 $O(\log n)$ 次单点查询。使用树剖线段树即可做到 $O((n+q)\log^2 n)$，期望得分 55 分。

K 为奇数

- K 为奇数的情况下，可以发现，在修改操作的过程中，$lowbitv(x)$ 是固定的。

K 为奇数

- K 为奇数的情况下，可以发现，在修改操作的过程中，$lowbitv(x)$ 是固定的。
- 也就是说，对于每个数 x，可以求出其在 $lowbitv(x)$ 以上的位发生变化时，这一位变成了什么，这个值显然是唯一的。

K 为奇数

- K 为奇数的情况下，可以发现，在修改操作的过程中，$lowbitv(x)$ 是固定的。
- 也就是说，对于每个数 x，可以求出其在 $lowbitv(x)$ 以上的位发生变化时，这一位变成了什么，这个值显然是唯一的。
- 而对于上一个值的逆运算，即一次操作下 $lowbitv(x)$ 发生变化且到达 x 的数也是唯一的，如果多次到达 x 是先不断最低位乘 2，然后进行上述操作。

K 为奇数

- K 为奇数的情况下，可以发现，在修改操作的过程中，$lowbitv(x)$ 是固定的。
- 也就是说，对于每个数 x，可以求出其在 $lowbitv(x)$ 以上的位发生变化时，这一位变成了什么，这个值显然是唯一的。
- 而对于上一个值的逆运算，即一次操作下 $lowbitv(x)$ 发生变化且到达 x 的数也是唯一的，如果多次到达 x 是先不断最低位乘 2，然后进行上述操作。
- 所以可以对每一位下的每一个值求出唯一的前驱，使得这个前驱是奇数并且经过若干次变化后可以在一次 $lowbitv(x)$ 以上的位变化后到达 x。

K 为奇数

- 发现这种前驱结构构成若干条链，所以修改就是对一条链的后缀进行异或操作，利用倍增前驱数组得到链头后也就可以定位到时哪一条链，在对应的链上进行后缀异或即可。

K 为奇数

- 发现这种前驱结构构成若干条链，所以修改就是对一条链的后缀进行异或操作，利用倍增前驱数组得到链头后也就可以定位到时哪一条链，在对应的链上进行后缀异或即可。
- 对于查询操作，可以变为 $O(\log n)$ 次线段树上单点查询。用同样的方式定位每个点的位置查询即可。这里注意转为单点修改前缀查询实现更简单一些。复杂度 $O(q \log^2 n)$，结合上述做法期望得分 75 分。

正解

- 扩展 K 为奇数的结论。如果 x 在 $lowbitv(x)$ 这一位的值的因子 2 个数不少于 K 的因子 2 个数，那么变化过程中 $lowbitv(x)$ 不变。

正解

- 扩展 K 为奇数的结论。如果 x 在 $lowbitv(x)$ 这一位的值的因子 2 个数不少于 K 的因子 2 个数，那么变化过程中 $lowbitv(x)$ 不变。
- 所以在 x 满足上述条件的时候，可以沿用奇数的处理方式。

正解

- 扩展 K 为奇数的结论。如果 x 在 $lowbitv(x)$ 这一位的值的因子 2 个数不少于 K 的因子 2 个数，那么变化过程中 $lowbitv(x)$ 不变。
- 所以在 x 满足上述条件的时候，可以沿用奇数的处理方式。
- 将 x 暴力跳至满足上述条件即可。可以证明，暴力跳的次数不超过 $O(\log n)$。

正解

- 扩展 K 为奇数的结论。如果 x 在 $lowbitv(x)$ 这一位的值的因子 2 个数不少于 K 的因子 2 个数，那么变化过程中 $lowbitv(x)$ 不变。
- 所以在 x 满足上述条件的时候，可以沿用奇数的处理方式。
- 将 x 暴力跳至满足上述条件即可。可以证明，暴力跳的次数不超过 $O(\log n)$。
- 用 map 维护暴力跳的位置，复杂度 $O(q \log^2 n)$，期望得分 100 分。

暴力

- 有一些基于数据随机的搜索剪枝，期望得分 $0 \sim 32$ 分。

观察

- 覆盖操作是很困难的，因为会改变大量信息，不可避免地会出现指数级的复杂度。

观察

- 覆盖操作是很困难的，因为会改变大量信息，不可避免地会出现指数级的复杂度。
- 考虑时光倒流，容易发现每个时刻被覆盖的是一段区间，所以每次只需要考虑在区间的左边和右边填数即可。

观察

- 覆盖操作是很困难的，因为会改变大量信息，不可避免地会出现指数级的复杂度。
- 考虑时光倒流，容易发现每个时刻被覆盖的是一段区间，所以每次只需要考虑在区间的左边和右边填数即可。
- 但是仍然需要记录最长上升子序列的全部信息，复杂度仍然很差。

观察

- 覆盖操作是很困难的，因为会改变大量信息，不可避免地会出现指数级的复杂度。
- 考虑时光倒流，容易发现每个时刻被覆盖的是一段区间，所以每次只需要考虑在区间的左边和右边填数即可。
- 但是仍然需要记录最长上升子序列的全部信息，复杂度仍然很差。
- 我们将最后呈现在表面上的数叫做好的，显然 a_n 是好的。

观察

- 覆盖操作是很困难的，因为会改变大量信息，不可避免地会出现指数级的复杂度。
- 考虑时光倒流，容易发现每个时刻被覆盖的是一段区间，所以每次只需要考虑在区间的左边和右边填数即可。
- 但是仍然需要记录最长上升子序列的全部信息，复杂度仍然很差。
- 我们将最后呈现在表面上的数叫做好的，显然 a_n 是好的。
- 注意到，存在一种操作方式，使得除了 a_n，其余好的数都在 LIS 中。

观察

- 覆盖操作是很困难的，因为会改变大量信息，不可避免地会出现指数级的复杂度。
- 考虑时光倒流，容易发现每个时刻被覆盖的是一段区间，所以每次只需要考虑在区间的左边和右边填数即可。
- 但是仍然需要记录最长上升子序列的全部信息，复杂度仍然很差。
- 我们将最后呈现在表面上的数叫做好的，显然 a_n 是好的。
- 注意到，存在一种操作方式，使得除了 a_n，其余好的数都在 LIS 中。
- 只需要分类讨论 a_n 是否在 LIS 中即可。

设计 DP

- 有了上述观察，不难想到，设 $f_{i,j,0/1}$ 表示当前考虑了 $i \sim n$，填了数的区间长度为 j，a_i 在左边，右边的最小值或 a_i 在右边，左边的最大值。

设计 DP

- 有了上述观察，不难想到，设 $f_{i,j,0/1}$ 表示当前考虑了 $i \sim n$，填了数的区间长度为 j，a_i 在左边，右边的最小值或 a_i 在右边，左边的最大值。
- 转移枚举下一个填入最长上升子序列的数，再考虑放在左边或右边，复杂度 $O(n^3)$，期望得分 $44 \sim 48$ 分。

优化

- 根据某个经典结论，随机排列的最长上升子序列是 $O(\sqrt{n})$ 级别的。

优化

- 根据某个经典结论，随机排列的最长上升子序列是 $O(\sqrt{n})$ 级别的。
- 所以 f 数组的 j 这一维的范围只需要开到 $O(\sqrt{n})$，复杂度 $O(n^2\sqrt{n})$，期望得分 $48 \sim 64$ 分。

正解

- 状态已经难以优化，考虑优化转移。

正解

- 状态已经难以优化，考虑优化转移。
- 将转移写成填表的形式，形如
  $f_{k+1,i,0} = \min((\min_{j>i,a_j>a_i} f_{k,j,0}), (\min_{j\geq i+k,f_{k,j,1}\geq a_i} a_j))$，
  $f_{k+1,i,1} = \max((\max_{j>i,a_j<a_i} f_{k,j,1}, (\max_{j\geq i+k,f_{k,j,0}\leq a_i} a_j))$。

正解

- 状态已经难以优化，考虑优化转移。
- 将转移写成填表的形式，形如
  $f_{k+1,i,0} = \min((\min_{j>i,a_j>a_i} f_{k,j,0}), (\min_{j\geq i+k,f_{k,j,1}\geq a_i} a_j))$，
  $f_{k+1,i,1} = \max((\max_{j>i,a_j<a_i} f_{k,j,1}, (\max_{j\geq i+k,f_{k,j,0}\leq a_i} a_j))$。
- 上述两个转移都是二维偏序的形式，可以使用树状数组优化转移，复杂度 $O(n\sqrt{n}\log n)$，期望得分 100 分。

致谢

谢谢大家！