数据结构选讲

Harry27182

2025 年 1 月 27 日

luogu P9334

- 给定一个长度为 n 的序列 d，需要支持 q 次操作。每次操作形如 x, y, l, r，修改 $d_x = y$ 并查询 $[l, r]$ 划分为若干区间满足相邻两段之间的区间和大小关系交替。求最大划分段数。
- $n \leq 2.5 \times 10^5, q \leq 5 \times 10^4$。

luogu P9334

- 首先有一些观察：显然小段的长度是 1 时最优。所以对于一个大段 $[l, r]$，需要满足的条件为 $\sum_{i=l}^{r} a_i > \max(a_{l-1}, a_{r+1})$。

- 首先有一些观察：显然小段的长度是 1 时最优。所以对于一个大段 $[l, r]$，需要满足的条件为 $\sum_{i=l}^{r} a_i > \max(a_{l-1}, a_{r+1})$。
- 但这个条件并不是划分合法的充分条件，不过由于最优性，可以发现，满足这个条件但不合法的一定存在划分数不少于当前方案的合法方案。所以就可以直接把它当做充分条件进行处理，答案不变。

luogu P9334

- 首先有一些观察：显然小段的长度是 1 时最优。所以对于一个大段 $[l, r]$，需要满足的条件为 $\sum_{i=l}^{r} a_i > \max(a_{l-1}, a_{r+1})$。
- 但这个条件并不是划分合法的充分条件，不过由于最优性，可以发现，满足这个条件但不合法的一定存在划分数不少于当前方案的合法方案。所以就可以直接把它当做充分条件进行处理，答案不变。
- 对于每个 r 求出最大的 $f(r) = l$ 满足 $[l, r]$ 是合法的大段，$nxt(i)$ 表示 i 作为上一个大段右端点时下一个大段右端点的最小值。如果没有修改，建出 nxt 树的结构并倍增即可求解，需要分类讨论一下头尾的情况。

- 注意到 $nxt(i) - i \leq 2\log V$。所以每次修改只会 $O(\log V)$ 个位置的 $nxt(i)$。

- 注意到 $nxt(i) - i \leq 2\log V$。所以每次修改只会 $O(\log V)$ 个位置的 $nxt(i)$。
- 考虑查询，可以直接用一个线段树来维护起点在这个区间前 $\log V$ 个位置，跳出这个区间之后跳出的长度和跳的次数分别是多少。

luogu P9334

- 注意到 $nxt(i) - i \leq 2\log V$。所以每次修改只会 $O(\log V)$ 个位置的 $nxt(i)$。
- 考虑查询，可以直接用一个线段树来维护起点在这个区间前 $\log V$ 个位置，跳出这个区间之后跳出的长度和跳的次数分别是多少。
- 这个信息是区间可合并的，单次复杂度 $O(\log V)$。总复杂度 $O(n\log V + q\log^2 V)$。

- 给定一个长度为 n 的序列，划分成若干段长度不超过 k 的区间，每段区间的贡献为 $mex \times sum$，求最大贡献值。
- $n \leq 2 \times 10^5$。

- 如果贡献只有 mex，就可以维护 (l, r, w) 表示当前右端点在 i，左端点位于 $[l, r]$ 时 mex 为 w。

- 如果贡献只有 mex，就可以维护 (l, r, w) 表示当前右端点在 i，左端点位于 $[l, r]$ 时 mex 为 w。
- 修改时找到 $w = a_i$ 的 (l, r, w)，用线段树维护每个值最晚出现的位置，那么可以用线段树二分进行 (l, r, w) 的分裂。维护区间的同时容易维护 dp 转移最大值。总的分裂次数是 $O(n)$ 的，复杂度 $O(n \log n)$。

- 如果贡献只有 mex，就可以维护 (l, r, w) 表示当前右端点在 i，左端点位于 $[l, r]$ 时 mex 为 w。
- 修改时找到 $w = a_i$ 的 (l, r, w)，用线段树维护每个值最晚出现的位置，那么可以用线段树二分进行 (l, r, w) 的分裂。维护区间的同时容易维护 dp 转移最大值。总的分裂次数是 $O(n)$ 的，复杂度 $O(n \log n)$。
- 写出转移式 $dp_i = dp_{j-1} - s_{j-1} mex + s_i mex$，对于前两项只和 j, mex 有关，对于每个 mex 相同的连续段维护前两项的最大值。

- 将 $-s_{j-1}$ 看做斜率，dp_{j-1} 看做截距，使用线段树套李超树维护一段区间内的直线即可，这部分复杂度为 $O(n\log^2 n)$。

- 将 $-s_{j-1}$ 看做斜率，dp_{j-1} 看做截距，使用线段树套李超树维护一段区间内的直线即可，这部分复杂度为 $O(n\log^2 n)$。
- 能转移的是后缀的一些区间和一个不完整区间。完整的区间可以在外层再使用一个线段树套李超树，只不过这次是关于 $k = mex, b = \max\{dp_{j-1} - s_{j-1}mex\}$ 的直线。复杂度 $O(n\log^2 n)$。

- 将 $-s_{j-1}$ 看做斜率，dp_{j-1} 看做截距，使用线段树套李超树维护一段区间内的直线即可，这部分复杂度为 $O(n\log^2 n)$。
- 能转移的是后缀的一些区间和一个不完整区间。完整的区间可以在外层再使用一个线段树套李超树，只不过这次是关于 $k = mex, b = \max\{dp_{j-1} - s_{j-1}mex\}$ 的直线。复杂度 $O(n\log^2 n)$。
- 不完整的那一段 mex 是定值，在第一个树套树上进行区间查询即可。复杂度同样是 $O(n\log^2 n)$。

- 给定一个长度为 n 的序列 a，和一个初始为空集的 S_0。
- 定义 $\mathrm{xor}(S)$ 表示 S 集合所有元素的异或和，$g(x,S)=\max_{T\subset S} x\oplus \mathrm{xor}(T)$，$f(l,r)=g(\oplus_{i=l}^{r} a_i, S_0)$。
- 操作 1：$a_x \oplus= v$；操作 2：将 x 加入集合 S_0 中；操作 3：查询 $\sum_{i=l}^{r} f(l,i)$。
- $n\le 5\times 10^5, q\le 10^5$。

- 如果不存在 1 操作，且询问 $l = 1$，那么只需要维护线性基对应 n 个位置对应的 g 值。

- 如果不存在 1 操作，且询问 $l = 1$，那么只需要维护线性基对应 n 个位置对应的 g 值。
- 由于线性基只会变化 $O(\log V)$ 次，所以暴力重构就可以做到 $O(n \log n \log V)$。

- 如果不存在 1 操作，且询问 $l = 1$，那么只需要维护线性基对应 n 个位置对应的 g 值。
- 由于线性基只会变化 $O(\log V)$ 次，所以暴力重构就可以做到 $O(n \log n \log V)$。
- 注意到，每次线性基发生变化时，所有 g 需要变化的位置对应都是异或上同一个数 x，所以可以做到 $O(n \log V)$。

- 如果不存在 1 操作，且询问 $l = 1$，那么只需要维护线性基对应 n 个位置对应的 g 值。
- 由于线性基只会变化 $O(\log V)$ 次，所以暴力重构就可以做到 $O(n \log n \log V)$。
- 注意到，每次线性基发生变化时，所有 g 需要变化的位置对应都是异或上同一个数 x，所以可以做到 $O(n \log V)$。
- 对于有修改操作的情况，记录 $h(x, S) = \min_{T \subset S} x \oplus \mathrm{xor}(T)$。那么有 $g(x \oplus y, S) = g(x, S) \oplus h(y, S)$。

- 这样修改操作对于 g 数组的影响就变成了区间异或一个数，查询操作 $l \neq 1$ 也可以用区间异或解决对 g 数组的影响。

- 这样修改操作对于 g 数组的影响就变成了区间异或一个数，查询操作 $l \neq 1$ 也可以用区间异或解决对 g 数组的影响。
- 但是此时线段树需要记录每一位上是 1 的有多少个，所以 pushup 的复杂度为 $O(\log V)$，总复杂度 $O((n+q)\log n \log V)$，瓶颈在重构线段树。

- 这样修改操作对于 g 数组的影响就变成了区间异或一个数，查询操作 $l \neq 1$ 也可以用区间异或解决对 g 数组的影响。
- 但是此时线段树需要记录每一位上是 1 的有多少个，所以 pushup 的复杂度为 $O(\log V)$，总复杂度 $O((n+q)\log n \log V)$，瓶颈在重构线段树。
- 压缩信息，我们每个 int 变量只用了 $O(\log len)$ 位，有些浪费。我们将记录的信息看做一个 01 矩阵，将矩阵转置一下，就可以只用 $O(\log len)$ 个变量记录信息。pushup 变为手动模拟加法器。建树的复杂度优化为 $T(n) = 2T(\frac{n}{2}) + O(\log n)$，也就是 $O(n)$。复杂度 $O(n \log V + q \log n \log V)$。

- q 个事件以下两种：新建一个区间 $[l, r]$，或者给定一个 x，将所有覆盖 x 的区间以一半一半的概率变成 $[l, x]$ 或者 $[x, r]$。
- 求最后所有区间的期望长度之和。$q \leq 50000$。

- 考虑一个暴力做法，对于每个区间 $[l,r]$ 分别去考虑最后会变成哪个区间。如果变成 $[x,y]$，那么 $[l,x]$ 中的分裂会选择右边，$[y,r]$ 的分裂会选择左边，分别用单调栈维护有几个有效分裂即可。复杂度 $O(n^2)$。

- 考虑一个暴力做法，对于每个区间 $[l,r]$ 分别去考虑最后会变成哪个区间。如果变成 $[x,y]$，那么 $[l,x]$ 中的分裂会选择右边，$[y,r]$ 的分裂会选择左边，分别用单调栈维护有几个有效分裂即可。复杂度 $O(n^2)$。
- 如果所有新建操作都在分裂操作之前，每个区间任意时刻的状态只有以下几种情况：没被分裂过；是某个分裂点左侧或右侧的一段，是两个相邻分裂点之间的一段。

- 考虑一个暴力做法，对于每个区间 $[l, r]$ 分别去考虑最后会变成哪个区间。如果变成 $[x, y]$，那么 $[l, x]$ 中的分裂会选择右边，$[y, r]$ 的分裂会选择左边，分别用单调栈维护有几个有效分裂即可。复杂度 $O(n^2)$。
- 如果所有新建操作都在分裂操作之前，每个区间任意时刻的状态只有以下几种情况：没被分裂过；是某个分裂点左侧或右侧的一段，是两个相邻分裂点之间的一段。
- 用两棵线段树分别维护每个点 i 到其左边第一个或右边第一个分裂点的区间期望个数。对于每个区间的初始形态虽然无法维护，但可以放在预处理里面。

- 考虑一个暴力做法，对于每个区间 $[l,r]$ 分别去考虑最后会变成哪个区间。如果变成 $[x,y]$，那么 $[l,x]$ 中的分裂会选择右边，$[y,r]$ 的分裂会选择左边，分别用单调栈维护有几个有效分裂即可。复杂度 $O(n^2)$。
- 如果所有新建操作都在分裂操作之前，每个区间任意时刻的状态只有以下几种情况：没被分裂过；是某个分裂点左侧或右侧的一段，是两个相邻分裂点之间的一段。
- 用两棵线段树分别维护每个点 i 到其左边第一个或右边第一个分裂点的区间期望个数。对于每个区间的初始形态虽然无法维护，但可以放在预处理里面。
- 加入分裂点的操作是可以用线段树区间乘，区间查询，单点修改来维护。

- 对操作序列分块，每 B 个一块。对于第 i 块的新建操作，需要考虑第 $i+1 \sim \frac{m}{B}$ 块的分裂操作。

- 对操作序列分块，每 B 个一块。对于第 i 块的新建操作，需要考虑第 $i+1 \sim \frac{m}{B}$ 块的分裂操作。
- 所以我们对每个块内的加入操作用暴力的做法处理出它在块尾的时候有可能被分裂成哪些区间，概率是多少。然后进行上述线段树的操作。

- 对操作序列分块，每 B 个一块。对于第 i 块的新建操作，需要考虑第 $i+1 \sim \frac{m}{B}$ 块的分裂操作。
- 所以我们对每个块内的加入操作用暴力的做法处理出它在块尾的时候有可能被分裂成哪些区间，概率是多少。然后进行上述线段树的操作。
- 预处理的复杂度为 $O(\frac{q^2}{B})$，每次加入都需要 $O(\log n)$ 的复杂度，这部分复杂度为 $O(\frac{q^2 \log n}{B})$。对于线段树部分的复杂度为 $O(qB \log n)$。取 $B = \sqrt{q}$，复杂度 $O(q\sqrt{q} \log n)$。

qoj5098

- 给定一个长度为 n 的序列 a 和颜色序列 c，需要进行 q 次操作：询问区间 $[l, r]$ 中没有重复颜色的最大子段和；修改某一个点的颜色。
- $n, q \leq 2 \times 10^5$。

- 注意到，一个位置 y 对应的左端点 $x = \max_{i=l}^{y} pre_i$。

- 注意到，一个位置 y 对应的左端点 $x = \max_{i=l}^{y} pre_i$。
- 考虑单侧递归线段树，维护一段区间如果只有区间内部的限制，右半边能贡献的最优答案，记为 val_k。

- 注意到，一个位置 y 对应的左端点 $x = \max_{i=l}^{y} pre_i$。
- 考虑单侧递归线段树，维护一段区间如果只有区间内部的限制，右半边能贡献的最优答案，记为 val_k。
- 考虑单侧递归，如果左边限制的最大值 $\geq$ 当前 lim，那么当前 lim 对右半边没用，调用 val_k 并递归左边。否则左边答案不受其内部限制，其答案为 $sum_{mid} - sum_{lim-1}$，递归右边即可。

- 注意到，一个位置 y 对应的左端点 $x = \max_{i=l}^{y} pre_i$。
- 考虑单侧递归线段树，维护一段区间如果只有区间内部的限制，右半边能贡献的最优答案，记为 val_k。
- 考虑单侧递归，如果左边限制的最大值 $\geq$ 当前 lim，那么当前 lim 对右半边没用，调用 val_k 并递归左边。否则左边答案不受其内部限制，其答案为 $sum_{mid} - sum_{lim-1}$，递归右边即可。
- 每个颜色开一个 set 记录每个点的 pre，复杂度 $O(n \log^2 n)$。

- 定义一个集合的 *xormex* 为选择一个数 x 之后给集合内所有数异或上 x 之后集合的 *mex*。给定长度为 2^n 的序列 a，多次询问区间子区间的 *xormex* 和。
- $n \le 18, q \le 10^6, 0 \le a_i < 2^n$。

- 对于 a 是排列的问题，尝试离线，考虑扫右端点，维护所有左端点的答案。
- 考虑计算答案是选择 x 为每个叶子节点的情况，这个值等于 x 所有祖先的兄弟中，所有满子树的 size 和。
- 对于右端点扫描线维护所有左端点的答案，每次考虑 Trie 上被新元素填满的子树。这些子树会更新其兄弟子树内的所有叶子的答案。
- 暴力枚举兄弟子树内所有叶子，求出其对答案的 n 段贡献。用线段树维护区间取 max 和历史和，总的线段树操作次数为 $O(2^n n^2)$。

- 优化上述做法。对子树内 2^d 种可能的答案分别求出最大左端点，再和子树外的 n 个决策进行归并，而且这个 n 是每次操作共享的。操作次数就变成了 $O(2^n n)$。
- 对于 a 不是排列的情况，一个子树可能被更新多次。给原树的每个节点再开一个 Trie 树，进行插入删除节点并维护 xormex，单次复杂度 $O(n)$。
- 每次右端点移动时有 $O(n)$ 个节点上的 Trie 树会发生变化。对于这 $O(n)$ 个点内部子树已经填满的进行处理。同样是将兄弟子树内的点与子树外的 n 个决策进行归并。
- 这里每次操作都会导致在 Trie 树上删除节点，所以找到的最优区间数是 $O(2^n n)$ 的。总复杂度 $O(2^n n^2 + nm)$。

致谢

谢谢大家！