线段树与平衡树

fsfdgdg

2024.3.29

线段树(Segment Tree)

线段树是一种高效处理区间问题的数据结构，能够做到 $O(n)$ 建树，$O(logn)$ 修改，$O(logn)$ 查询，其解决的问题要具有可合并性（即能快速地将两个相邻的区间的信息合并起来，符合的有 $max, min, and, or, xor, sum$ 等，不符合的有 mex 等）。

线段树(Segment Tree)

线段树是一种高效处理区间问题的数据结构，能够做到 $O(n)$ 建树，$O(logn)$ 修改，$O(logn)$ 查询，其解决的问题要具有可合并性（即能快速地将两个相邻的区间的信息合并起来，符合的有 $max, min, and, or, xor, sum$ 等，不符合的有 mex 等）。
线段树的思想基于将一个完整的区间拆成两个互不相交的区间分别计算答案再合并起来，通常将区间拆成两个长度一样的区间，不难证明树高是 $O(logn)$ 的（每次递归长度至少减半，至多递归 $O(logn)$ 次），节点数是 $O(n)$ 的（线段树本质是一棵有 n 个叶子的二叉树）。

线段树(Segment Tree)

线段树是一种高效处理区间问题的数据结构，能够做到 $O(n)$ 建树，$O(logn)$ 修改，$O(logn)$ 查询，其解决的问题要具有可合并性（即能快速地将两个相邻的区间的信息合并起来，符合的有 $max, min, and, or, xor, sum$ 等，不符合的有 mex 等）。
线段树的思想基于将一个完整的区间拆成两个互不相交的区间分别计算答案再合并起来，通常将区间拆成两个长度一样的区间，不难证明树高是 $O(logn)$ 的（每次递归长度至少减半，至多递归 $O(logn)$ 次），节点数是 $O(n)$ 的（线段树本质是一棵有 n 个叶子的二叉树）。
一般线段树节点的编号方式可以采取传统二叉树的编号方式，即 p 的左右儿子分别为 $2p, 2p+1$ 。

线段树(Segment Tree)

单点修改/查询只需直接修改/查询叶子的权值即可，显然时间复杂度为 $O(\log n)$ 。

线段树(Segment Tree)

单点修改/查询只需直接修改/查询叶子的权值即可，显然时间复杂度为 $O(\log n)$ 。
区间查询则需要将查询区间拆分成若干个线段树上的区间，再依次将答案合并起来。
考虑什么位置的区间会被询问到。设询问区间为 $[l, r]$ ，不难发现询问区间会被拆分为根节点到 $l-1$ 位置的叶子的路径的所有右儿子和到 $r+1$ 位置的叶子的路径的所有左儿子，因此时间复杂度显然为 $O(\log n)$ 。

线段树(Segment Tree)

单点修改/查询只需直接修改/查询叶子的权值即可，显然时间复杂度为 $O(\log n)$ 。
区间查询则需要将查询区间拆分成若干个线段树上的区间，再依次将答案合并起来。
考虑什么位置的区间会被询问到。设询问区间为 $[l, r]$ ，不难发现询问区间会被拆分为根节点到 $l-1$ 位置的叶子的路径的所有右儿子和到 $r+1$ 位置的叶子的路径的所有左儿子，因此时间复杂度显然为 $O(\log n)$ 。
区间修改同样需要将修改区间拆分为若干个线段树上的区间，每次修改后需要在修改区间打上懒标记，在向下递归时下放标记，时间复杂度依然为 $O(\log n)$ 。
有多个懒标记时需要注意懒标记修改和下放的顺序。

线段树(Segment Tree)

动态开点线段树：当需要存储 n 棵线段树且修改数只有 $O(n)$ 时，暴力建出 n 棵完整的线段树的空间复杂度为 $O(n^2)$ ，不可接受。考虑线段树上有非常多的空节点，因此只对修改遍历到的非空节点分配空间，此时需要放弃传统二叉树的编号方式，而用两个变量记录左右儿子的编号。$O(n)$ 次修改只会遍历到 $O(n \log n)$ 个节点，因此时空复杂度为 $O(n \log n)$ 。

线段树(Segment Tree)

动态开点线段树：当需要存储 n 棵线段树且修改数只有 $O(n)$ 时，暴力建出 n 棵完整的线段树的空间复杂度为 $O(n^2)$ ，不可接受。考虑线段树上有非常多的空节点，因此只对修改遍历到的非空节点分配空间，此时需要放弃传统二叉树的编号方式，而用两个变量记录左右儿子的编号。$O(n)$ 次修改只会遍历到 $O(n \log n)$ 个节点，因此时空复杂度为 $O(n \log n)$ 。

可持久化线段树：当需要存储每个版本的线段树且单点修改数只有 $O(n)$ 时，每次只新建单点修改遍历到的节点，其他没有变化的节点从上一个版本继承，不难发现每次只会新建 $O(\log n)$ 个节点，因此时空复杂度为 $O(n \log n)$ 。

线段树(Segment Tree)

P3373

给定长度为 n 的序列和模数 mod ，初始时序列所有元素为 0 。
接下来有 m 次操作，每次操作有三种：区间加 k ，区间乘 k ，询问区间和模 mod 。
$n, m \leq 10^6$

线段树(Segment Tree)

P3373

给定长度为 n 的序列和模数 mod ，初始时序列所有元素为 0 。
接下来有 m 次操作，每次操作有三种：区间加 k ，区间乘 k ，询问区间和模 mod 。
$n, m \leq 10^6$

Solution

线段树板题。
维护加法标记 add ，乘法标记 tim ，区间和 sum 。区间加 k 则 $add+ = k, sum+ = len * k$ ，区间乘 k 则 $add* = k, sum* = k, tim* = k$ ，下放标记先下放 tim 再下放 add 。
时间复杂度 $O(m \log n)$ 。

线段树(Segment Tree)

P3586

给定长度为 n 的序列，初始时序列所有元素为 0 。
接下来有 m 次操作，每次操作有两种：单点修改，询问能否执行 x 次"从序列中选出 y 个正数，并将它们都减去 1 " 。
$n, m \leq 10^6$

线段树(Segment Tree)

P3586

给定长度为 n 的序列，初始时序列所有元素为 0 。
接下来有 m 次操作，每次操作有两种：单点修改，询问能否执行 x 次"从序列中选出 y 个正数，并将它们都减去 1 " 。
$n, m \leq 10^6$

Solution

询问的本质为 $[\Sigma_{val_i \leq x} val_i + \Sigma_{val_i > x} x \geq x \times y]$ 。
因此离散化后用值域线段树维护区间和和区间元素个数即可，时间复杂度 $O(n \log n)$ 。

线段树(Segment Tree)

P1502

（二维滑动窗口）二维平面上有 n 个点，坐标为 (x_i, y_i)，权值为 val_i。
给定 W, H，选择一个 $W \times H$ 的矩形，使得包含在矩形内的点权值和最大（包括边界），输出最大权值和。
$n \leq 10^6$

线段树(Segment Tree)

Solution

考虑一维滑动窗口的实现过程：坐标排序后双指针尺取长度尽可能长的一段区间，至多有 $O(n)$ 个区间，对所有可能区间取 max 。
显然有一个 $O(n^2)$ 的暴力：横纵坐标分别排序，横坐标双指针尺取后再纵坐标双指针尺取，至多有 $O(n^2)$ 个矩形，对所有可能矩形取 max 。
考虑优化。发现每次纵坐标尺取到的区间完全相同，因此在横坐标双指针尺取时维护每个纵坐标尺取区间的权值和，插入/删除点时增加/减少对应区间权值，每次取全局最大值即可。再发现每个点所处的尺取区间也是一个区间，因此插入/删除点时相当于区间加法。
问题转化为 $O(n)$ 次区间加法，$O(n)$ 次询问全局最大值，线段树维护即可。时间复杂度 $O(n \log n)$ 。

线段树(Segment Tree)

P4198

二维平面上有 n 个点，初始时第 i 个点坐标为 $(i,0)$ 。
接下来有 m 次操作，每次操作有两种：修改某个点的纵坐标，查询 $\Sigma_{i=1}^{n}[\forall j \neq i,(i,y_i)$ 与 $(0,0)$ 的连线不与 (j,y_j) 与 $(j,0)$ 的连线相交]。
$n,m \leq 10^5$

线段树(Segment Tree)

Solution

询问的本质为 $\Sigma_{i=1}^{n}[\forall j < i, \frac{y_i}{i} > \frac{y_j}{j}]$，即序列 $\frac{y_i}{i}$ 的单调上升栈大小。
考虑用线段树维护区间单调上升栈大小 ans 和区间最大值 max。查询即为 $ans_{1,n}$。修改即为单点修改。
合并区间时，设当前区间为 $[l, r]$，有 $ans_{l,r} = ans_{l,mid} + [mid + 1, r]$ 单调栈中大于 $max_{l,mid}$ 的数的个数。
后者可以在 $[mid + 1, r]$ 中二分得到。具体地，设当前二分区间为 $[L, R]$，$[L, R]$ 单调栈中大于 $max_{l,mid}$ 的数的个数为 $query(L, R)$。如果 $max_{L,Mid} > max_{l,mid}$，则有 $query(L, R) = ans_{L,R} - ans_{L,Mid} + query(L, Mid)$（不是 $ans_{Mid+1,r} + query(L, Mid)$，因为 $ans_{Mid+1,r}$ 中可能含有小于等于 $max_{l,mid}$ 的数，但是 $ans_{L,R} - ans_{L,Mid}$ 不会）；否则有 $query(L, R) = query(Mid + 1, R)$。
每次合并需要 $O(\log n)$ 的二分，单次修改需要 $O(\log n)$ 次合并，因此总时间复杂度为 $O(n \log^2 n)$。

线段树(Segment Tree)

P2824

给定长度为 n 的排列。
接下来有 m 次操作，每次操作有两种：区间升序排列，区间降序排列。
最后单点查询。
$n, m \leq 10^5$

线段树(Segment Tree)

P2824

给定长度为 n 的排列。
接下来有 m 次操作，每次操作有两种：区间升序排列，区间降序排列。
最后单点查询。
$n, m \leq 10^5$

Solution

首先考虑一种特殊情况：01 序列如何排序。显然升序排序即将所有 1 移到最右侧，所有 0 移到最左侧，降序同理。即 $O(n)$ 次区间求和，$O(n)$ 次区间覆盖，显然线段树维护，时间复杂度 $O(m \log n)$ 。
现在考虑原序列，二分最后的答案 mid ，设当前二分区间为 $[l, r]$ 。改写所有数为 $[num \geq mid]$ ，依次 m 次操作后，如果最后是 1 则向 $[mid, r]$ 递归，否则向 $[l, mid - 1]$ 递归。
总共二分 $O(\log n)$ 次，每次复杂度 $O(m \log n)$ ，总时间复杂度为 $O(m \log^2 n)$ 。

线段树(Segment Tree)

P5324

设当前数列长度为 k ，定义一次数列操作为删去数列中所有等于 k 的数。定义一个数列为合法的，当且仅当该数列能经过若干次数列操作变成空数列。
现在给定长度为 n 的数列。
接下来有 m 次操作，每次操作有两种：单点修改，整体 $+1/-1$ 。每次操作后询问当前数列至少需要修改几个数使得数列合法。
$n, m \leq 10^6$

线段树(Segment Tree)

Solution

首先考虑如何计算答案。设 $i(i \in [1,n])$ 的个数为 cnt_i ，每次将 $[i-cnt_i+1,i]$ 权值加 1 ，最后 $[1,n]$ 内 0 的个数即为答案。

正确性证明：如果最后 $[1,n]$ 内不含有 0 则显然序列合法，且易证这是充要条件。所有位置的权值和为 n ，因此 $[1,n]$ 内有 k 个 0 则说明有 k 个位置重叠或者超出范围，显然至少且可以用 k 次操作将序列修改为合法，因此答案为 k 。

考虑如何处理修改。单点修改显然更新对应数的个数并修改对应位置的权值即可。整体 $+1/-1$ 显然不能暴力修改，需要维护一个懒标记 eps 表示整个数列的偏移量，查询即为查询 $[1-st,n-st]$ 的答案。注意 st 变化时可能有数列中的数大于 n ，此时它们的贡献无效（它们不可能被删除，只能被修改），因此将对应区间权值减 1 ，当它们小于等于 n 时再还原。

问题转化为 $O(n)$ 次单点加法，$O(n)$ 次区间加法，$O(n)$ 次询问区间 0 的个数。注意到此题中权值始终非负，因此转化为询问区间最小值是否是 0 和区间最小值个数即可。时间复杂度 $O(n\log n)$ 。

线段树(Segment Tree)

P5490

（扫描线）求 n 个四边平行于坐标轴的矩形的面积并。

$n \leq 10^6$

线段树(Segment Tree)

P5490

（扫描线）求 n 个四边平行于坐标轴的矩形的面积并。
$n \leq 10^6$

Solution

首先离散化坐标。
用一个扫描线从左到右扫过所有矩形，维护每个纵坐标段的覆盖次数。
设矩形的左上角和右下角坐标分别为 $(x_1, y_1), (x_2, y_2)(x_1 \leq x_2, y_1 \leq y_2)$，则在 x_1 处 $[y_1, y_2]$ 覆盖次数加 1，$x_2 + 1$ 处 $[y_1, y_2]$ 覆盖次数减 1，每次横坐标移动时答案加上横坐标变化量乘以覆盖次数非 0 纵坐标段长度。
问题转化为 $O(n)$ 次区间加法，$O(n)$ 次询问序列非 0 段长度，线段树维护即可。观察到覆盖次数始终非负，因此维护区间最小值和区间最小值出现次数，查询整体最小值是否为 0 与整体最小值出现次数即可。
时间复杂度 $O(n \log n)$ 。

线段树(Segment Tree)

P4556

（线段树合并）给定有 n 个节点的树，每个节点有一个初始为空的可重集合。
接下来有 m 次操作，每次操作将路径上所有点的可重集合内插入一个数。所有操作结束后，输出每个点的可重集合内出现次数最多的数。
$n, m \leq 10^6$

Solution

路径加，单点查转化为单点加，子树查。因为所有询问在修改之后，因此先处理所有修改，最后再向上合并。
修改即为线段树上单点加，因为有 n 棵线段树，因此用动态开点线段树即可。

线段树(Segment Tree)

Solution

现在的问题是如何高效合并两棵线段树，我们采用如下策略：同步遍历两棵线段树，如果一个节点为空则返回另一个节点；否则如果两个节点非空且为叶子，直接合并两个节点的信息，并返回合并后的节点；否则递归两边合并左右子树，再返回合并后的节点。

时间复杂度证明：每次暴力递归合并都会将两个点合并为一个点，总点数减少 1 。因为有 m 次修改，因此动态开点线段树的总点数是 $O(m \log V)$ 的，因此合并的时间复杂度为 $O(m \log V)$ 。

回答询问是简单的，只需要维护区间最大出现次数和出现次数最大的数即可。

总时间复杂度 $O(m \log V)$ 。

因为节点有可能是从儿子处继承，因此合并时直接合并两个节点可能会导致儿子的线段树信息出错。本题合并后不会再询问儿子，因此没有影响。如果需要维护儿子的线段树信息，只需要在合并两个节点时新建一个节点存储合并后的信息并返回新建节点，不难证明时间复杂度依然为 $O(n \log n)$ 。

线段树(Segment Tree)

P6242

（吉司机线段树）给定长度为 n 的数列，初始时所有元素为 0 。
接下来有 m 次操作，每次操作有五种：区间加，区间取 min ，区间求和，区间求 max ，区间求历史 max 。
$n, m \leq 10^5$

线段树(Segment Tree)

Solution

区间加，区间求和，区间求 max 是简单的，不再赘述。

考虑如何区间取 min ，我们采用如下策略：设区间最大值为 fir ，严格次大值为 sec ，当前区间对 k 取 min 。如果 $fir > k > sec$ 则将所有最大值修改为 k ；否则如果 $fir > sec \geq k$ ，则继续向两边递归；否则有 $k \geq fir > sec$ ，当前区间不需要操作，直接返回。

时间复杂度证明：设 $\phi(u) = \Sigma_{u \in subtree_v} fir_u \neq fir_v$ ，显然初始时 $\Sigma\phi(u) = O(n \log n)$ （总共有 $O(n)$ 个节点，每个节点有 $O(\log n)$ 个祖先）。

设当前节点为 u ，当向两边递归时，说明 $\exists v \in subtree_u, fir_v \neq fir_u, fir_v > k$ ，且 $\forall w \in u \to v \&\& w \neq v, fir_w \neq fir_v$ ，显然修改后 $\forall w \in u \to v, fir_w = k$ ，因此 $\phi(v)$ 减少了 $dep_v - dep_u$ ，且额外的递归次数恰好为 $dep_v - dep_u$ ，因此均摊每次额外的递归都会使得 $\Sigma\phi(u)$ 减少 1 ，所以总额外递归次数是 $O(n \log n)$ 的。

当存在区间加操作时，因为每次区间加至多影响 $O(\log n)$ 个节点，所以 $\Sigma\phi(u)$ 至多增加 $O(\log^2 n)$ ，因此总时间复杂度为 $O(n \log^2 n)$ 。

线段树(Segment Tree)

Solution

考虑如何区间求历史 max ，难点其实在于如何下放标记时快速更新儿子的历史 max ，容易发现只有历史懒标记 max 可能可以更新历史 max ，因此还需要维护历史懒标记 max 。
注意各类标记的下放顺序。
时间复杂度 $O(n \log^2 n)$ 。

线段树(Segment Tree)

P4097

（李超线段树）给定初始为空的二维坐标系。
接下来有 n 次操作，每次操作有两种：插入一条线段，查询与 $x = k$ 相交的线段中交点纵坐标最大的线段编号。
$n \leq 10^5$

线段树(Segment Tree)

P4097

（李超线段树）给定初始为空的二维坐标系。
接下来有 n 次操作，每次操作有两种：插入一条线段，查询与 $x = k$ 相交的线段中交点纵坐标最大的线段编号。
$n \leq 10^5$

Solution

线段树维护在区间中点取值最大的线段，由线段的单调性，其他线段至多只在左端点或者只在右端点取值比当前线段大，因此向左或者向右递归修改即可，单次修改时间复杂度显然 $O(\log n)$ 。如果只插入到一个区间，需要将插入区间拆分为 $O(\log n)$ 个线段树上的区间，再分别递归，单次修改时间复杂度 $O(\log^2 n)$ 。
询问时向对应位置的叶子递归，从根节点到叶子路径上所有线段的取值最大值即为答案，单次询问时间复杂度 $O(\log n)$ 。
总时间复杂度 $O(n \log^2 n)$ 。

线段树(Segment Tree)

CF997E

（历史版本和线段树）给定长度为 n 的排列。
接下来有 m 次询问，每次询问
$\Sigma_{[l,r]\subseteq[L,R]}[max_{l,r} - min_{l,r} + 1 = r - l + 1]$ 。
$n, m \leq 10^6$

线段树(Segment Tree)

Solution

区间子区间问题依然考虑扫描线。扫描右端点，维护每个左端点的答案，询问 $[L,R]$ 即为在 R 处查询 $[L,R]$ 的区间和。

$[max_{l,r}-min_{l,r}+1=r-l+1]=[max_{l,r}-min_{l,r}+l-r=0]$ ，因此对于所有左端点维护 $max_{l,r}-min_{l,r}+l-r$ ，初始时每个左端点的权值为 l 。每次右端点右移相当于整体减 1 ；再维护两个单调上升栈和单调下降栈，每次单调栈变化时对应于一个区间的 $max_{l,r},min_{l,r}$ 变化，即一次区间加法，由单调栈的复杂度分析总共有 $O(n)$ 次区间加法。每次修改完后对所有权值为 0 的位置答案加 1 。

观察到权值始终非负，因此对最小值为 0 的位置更新答案即可。为了快速更新区间答案和，需要维护区间最小值个数。为了下传标记时快速更新儿子的区间答案和，需要维护答案更新懒标记。

时间复杂度 $O(n\log n)$ 。

线段树(Segment Tree)

P3834

给定长度为 n 的序列。
接下来有 m 次询问，每次询问区间第 k 大的数。
$n, m \leq 10^6$

线段树(Segment Tree)

P3834

给定长度为 n 的序列。
接下来有 m 次询问，每次询问区间第 k 大的数。
$n, m \leq 10^6$

Solution

主席树板题。
线段树维护值域区间出现次数和，依次插入 n 个数建立主席树。询问 $[l, r]$ 即在第 r 个版本和第 $l-1$ 个版本线段树上二分即可。
时间复杂度 $O(n \log n)$ 。

线段树(Segment Tree)

P3834

给定长度为 n 的序列 val 。
接下来有 m 次询问，每次询问 $max_{i=l}^{r}(val_i + y) \oplus x$ 。
$n \leq 10^6, m \leq 10^5$

线段树(Segment Tree)

P3834

给定长度为 n 的序列 val 。
接下来有 m 次询问，每次询问 $max_{i=l}^{r}(val_i + y) \oplus x$ 。
$n \leq 10^6, m \leq 10^5$

Solution

首先考虑如何计算答案，加法和异或混合，难以直接处理。考虑从高到低贪心确定每一位，设当前考虑到第 k 位，$val_i + y$ 的更高位确定为 A ，则第 k 位能填 1 当且仅当 $val_i \in [A + 2^k - y, A + 2^{k+1} - y - 1]$ 。
转化为 $O(m \log V)$ 次区间求出现次数，主席树处理即可。
时间复杂度 $O(n \log n + m \log V \log n)$ 。

线段树(Segment Tree)

P4587

给定长度为 n 的序列。
接下来有 m 次询问，每次询问序列下标区间的所有数组成的可重集合的所有子集和的 mex 。
$n, m \leq 10^6$ 。

线段树(Segment Tree)

Solution

首先考虑如何计算答案。将区间内所有数从小到大排序，依次扫过每个数，设当前数为 i ，前面所有数之和为 sum ，如果 $sum \geq i-1$ 则 $sum+=i$ ，否则答案为 $sum+1$ 。正确性显然。
考虑用数据结构优化计算过程。主席树维护区间出现次数和与区间和，查询时同时在第 r 个版本和第 $l-1$ 个版本线段树上查询。首先检查是否有 1 ，然后向上回溯，设当前区间为 $[1,R]$ ，如果 $sum_{1,mid} \geq min_{mid+1,R}-1$ ，则继续向上回溯（如果 $[mid+1,R]$ 内没有数则显然可以回溯；否则因为 $sum_{1,mid} \geq min_{mid+1,R}-1 \geq mid, min_{mid+1,R} \geq mid+1 \rightarrow sum_{1,mid}+min_{mid+1,R} \geq R$，因此可以回溯）；否则答案为 $sum_{1,mid}+1$ 。$min_{mid+1,R}$ 可以在主席树上二分得到。时间复杂度 $O(n\log^2 n)$ 。

线段树(Segment Tree)

P3302

给定一片有 n 个点 m 条边的森林，点有点权。
接下来有 q 次操作，每次操作有两种：连接两个尚未连通的点，询问两个已经连通的点间路径的第 k 大点权。
$n, q \leq 10^5$,**强制在线**

线段树(Segment Tree)

P3302

给定一片有 n 个点 m 条边的森林，点有点权。
接下来有 q 次操作，每次操作有两种：连接两个尚未连通的点，询问两个已经连通的点间路径的第 k 大点权。
$n, q \leq 10^5$,**强制在线**

Solution

首先考虑没有连边操作如何处理。同样建立主席树，每个点在其父亲的版本上修改，类似于序列查询即可。
再考虑没有强制在线如何处理。显然可以建出最后的森林，再建主席树即可。
最后考虑有强制在线如何处理。显然可以从某个端点出发，以其为根暴力重建其子树。不难想到启发式合并，每次暴力重建子树大小较小的一端，根据启发式合并的复杂度分析，每个点只会被重建 $O(\log n)$ 次，因此时间复杂度为 $O(n \log^2 n)$ 。因为树的形态不固定，因此采用倍增法求 lca ，更新倍增表的时间复杂度依然为 $O(n \log^2 n)$ 。

平衡树(Self-balancing Binary Search Tree)

二叉搜索树，一种支持动态插入，删除，修改，查询元素的数据结构，其思想为将元素按一定的顺序联结成二叉树形结构，即左子树的数都比根节点小，右子树的数都比根节点大，方便快速查找与修改。
普通二叉搜索树的复杂度是 $O(dep)$ 的，在随机数据下时间复杂度是 $O(\log n)$ 的，但在构造数据里时间复杂度会退化为 $O(n)$ 。平衡树，即自平衡二叉搜索树，能够在严格 $O(\log n)$ 的时间内完成插入，删除，修改与查询操作，其思想为适当地对不平衡的树进行重构，使其树高始终保持在 $O(\log n)$ 的范围内。
平衡树有非常多的种类，如替罪羊树，AVL 树，红黑树，Splay 树，Treap，Fhq-treap。算法竞赛中，最常使用的两种是 Splay 树和 Fhq-treap。
如果只是为了查询一个元素的前驱和后继这样的简单任务，而没有更加复杂的操作，那么可以用 STL 中的 set 与 multiset 维护一棵平衡树（前者不可重复，后者可重复）。

平衡树(Self-balancing Binary Search Tree)

Splay树的思想为向上旋转不平衡的位置，使其变得平衡。复杂度保证来源于其独特的单旋与双旋机制：当当前节点与父节点处于同侧时，先旋转父亲再旋转当前节点；否则旋转两次当前节点，证明用到了势能分析（证明篇幅太长，就不一一赘述），复杂度是均摊 $O(\log n)$ 的，因此不能可持久化，且常数较大，但其灵活多变的旋转使得其在作为 LCT 的辅助树时，复杂度依然只有 $O(\log n)$ （其他平衡树的复杂度是 $O(\log^2 n)$ 的）。

Treap的思想基于随机数据下，普通二叉搜索树的复杂度就是 $O(\log n)$ 的。其给所有数随机一个附加权值，使得其不仅本身权值满足二叉搜索树的性质，附加权值还要满足堆的性质，不满足则一直向上旋转直到满足为止。Fhq-treap 又称非旋 treap，其维护二叉平衡树性质的方式不是旋转而是分裂与合并：先在插入位置分裂成两棵子树，再将插入元素与两棵子树一一合并，合并时由附加权值的大小来决定哪个数作为根节点。两者的复杂度都是严格 $O(\log n)$ 的，Fhq-treap 由于其不要求旋转，结构相对稳定，易于实现可持久化，且码量极短，常数优秀，使用更加频繁。

平衡树(Self-balancing Binary Search Tree)

P3369

总共有 n 次操作，维护一种数据结构，支持插入/删除元素，查询排名为 k 的数，查询 k 的排名，求 k 的前驱与后继。

$n \leq 10^6$

平衡树(Self-balancing Binary Search Tree)

P3369

总共有 n 次操作，维护一种数据结构，支持插入/删除元素，查询排名为 k 的数，查询 k 的排名，求 k 的前驱与后继。
$n \leq 10^6$

Solution

平衡树板题。
任选一种平衡树维护每个节点的权值和子树大小，插入删除元素是简单的，查询时直接二分查询。
时间复杂度 $O(n)$ 。

平衡树(Self-balancing Binary Search Tree)

P3285

给定长度为 n 的序列，初始时为长度为 n 的递增排列。
接下来有 m 次操作，每次操作有五种：修改序列中的元素，将元素移至序列开头，将元素移至序列末尾，查询第 k 个元素，查询元素 k 的位置。
$n \leq 10^9, m \leq 10^6$,**强制在线**

平衡树(Self-balancing Binary Search Tree)

P3285

给定长度为 n 的序列，初始时为长度为 n 的递增排列。
接下来有 m 次操作，每次操作有五种：修改序列中的元素，将元素移至序列开头，将元素移至序列末尾，查询第 k 个元素，查询元素 k 的位置。
$n \leq 10^9, m \leq 10^6$,**强制在线**

Solution

不难发现序列由三部分构成：移到序列第一位的元素，初始位置的元素，移到序列最后一位的元素。第一个和第三个的元素用平衡树维护，位置用 map 维护；第二个平衡树维没有被移动过的元素，初始时为 $[1, n]$ 的区间，每次取出元素就将对应区间分裂成两个，区间个数至多只有 m 个，时间复杂度 $O(m \log m)$ 。
查询第 k 个位置时二分查询，依次查询第一、二、三个部分；查询元素 k 的位置时首先查询是否在第一、三部分的 map 里，有则直接输出，没有则在第二部分的平衡树上二分查询，时间复杂度 $O(m \log m)$ 。

平衡树(Self-balancing Binary Search Tree)

P3391

给定长度为 n 的序列，初始时为长度为 n 的递增排列。
接下来有 m 次操作，每次操作翻转一个区间，输出所有操作后的序列。

$n, m \leq 10^6$

平衡树(Self-balancing Binary Search Tree)

P3391

给定长度为 n 的序列，初始时为长度为 n 的递增排列。
接下来有 m 次操作，每次操作翻转一个区间，输出所有操作后的序列。

$n, m \leq 10^6$

Solution

文艺平衡树板题。用平衡树维护序列的位置。
Splay 做法：翻转时将 $l-1$ 旋转到根，将 $r+1$ 旋转到根的右儿子（需要提前插入 0 与 $n+1$ ），此时 $[l, r]$ 的所有元素一定在 $r+1$ 的左子树内，将 $r+1$ 的左儿子节点打上翻转懒标记，注意每次 Splay 时要先从根到旋转位置下放翻转懒标记，再旋转。
Fhq-treap 做法：每次将 $[l, r]$ 区间的子树分裂出来，在根节点打上翻转懒标记，每次在分裂和合并之前下放懒标记。
两者的时间复杂度都是 $O(n \log n)$ 。

平衡树(Self-balancing Binary Search Tree)

P7880

给定有 n 个节点的以 1 为根的树，定义 dep_i 表示根节点到 i 节点的路径权值和。
接下来有 m 次询问，每次询问对于所有 $l \leq i, j \leq r$ ，$dep_{lca(i,j)}$ 的种类数。
$n \leq 10^5, q \leq 10^6$

平衡树(Self-balancing Binary Search Tree)

Solution

首先离散化所有 dep 。
区间子区间问题显然考虑扫描线。扫描右端点 r ，对于每种 dep ，维护最大的左端点 l 满足 $dep_{lca(l,r)} = dep$ ，并在 l 处权值加 1 ，询问 $[l, r]$ 即为在 r 处查询 $[l, r]$ 的区间和。因为可能成为答案的点对有 $O(n^2)$ 个，所以此时时间复杂度为 $O(n^2 \log n + q \log n)$ 。
处理询问的复杂度难以优化，考虑优化点对个数。不难想到枚举 lca ，每个点与其他子树内的点都可以贡献答案 dep_{lca} 。因为只有数列上最接近的点对有可能有作用，因此查询每个点在其他子树的前驱后继即可，显然可以用 set 维护。
此时随机数据下的时间复杂度为 $O(n \log^2 n + q \log n)$（因为树高是 $O(\log n)$ 的，因此每个点会被插入 set 和查询前驱后继 $O(\log n)$ 次），但是链上的时间复杂度依然为 $O(n^2 \log n + q \log n)$ 。不难想到启发式合并，每次只插入和查询轻儿子子树内所有点，显然此时依然正确，且每个点只会被插入 set 和查询前驱后继严格 $O(\log n)$ 次，时间复杂度优化为 $O(n \log^2 n + q \log n)$ 。