线段树及类似数据结构及相关应用

汪直方

宁波市镇海蛟川书院

2023 年 8 月 10 日

线段树（Segment Tree）是一种 leafy tree，即其用叶子结点去对应所有需要表示的、不能再细分的信息，每一个结点表示其子树内叶子构成的信息。我们将叶子结点的信息按照 dfs 序编号，那么每一个结点就对应着一个区间的信息，特别地，根节点对应全集对应的区间，叶子结点对应单点对应的区间。

线段树（Segment Tree）是一种 leafy tree，即其用叶子结点去对应所有需要表示的、不能再细分的信息，每一个结点表示其子树内叶子构成的信息。我们将叶子结点的信息按照 dfs 序编号，那么每一个结点就对应着一个区间的信息，特别地，根节点对应全集对应的区间，叶子结点对应单点对应的区间。

为了保证常见的性质，我们一般至少要求线段树是二叉且广义重量平衡的。严格的重量平衡（weight balanced）是指对于任意结点 u 及其的任意儿子结点 v，满足 v 的子树大小与 u 的子树大小之比在常数范围内，即 $\dfrac{\mathrm{size}(v)}{\mathrm{size}(u)} = \Theta(1)$。不过重量平衡带来的性质往往只需要满足一个更为宽松的条件：存在常数 $c_1 > 1$ 和 c_2，满足对于任意结点 u，存在一个祖先结点 a 满足 $\mathrm{dist}(a, u) \leqslant c_2$ 且 a 为根节点或 $\dfrac{\mathrm{size}(a)}{\mathrm{size}(u)} \geqslant c_1$。

为了方便实现，我们对于表示区间 $[l,r]$ 的结点（其中 $l<r$ 即非叶子结点），令 $m=\left\lceil\frac{l+r}{2}\right\rceil$，则其左儿子结点表示区间 $[l,m]$，其右儿子结点表示区间 $(m,r]$。这也决定了一般意义上的线段树的结构，无法对动态编号的问题产生比较良好的解法，需要离散化或动态开点等技巧规避。

为了方便实现，我们对于表示区间 $[l,r]$ 的结点（其中 $l<r$ 即非叶子结点），令 $m=\left\lceil\frac{l+r}{2}\right\rceil$，则其左儿子结点表示区间 $[l,m]$，其右儿子结点表示区间 $(m,r]$。这也决定了一般意义上的线段树的结构，无法对动态编号的问题产生比较良好的解法，需要离散化或动态开点等技巧规避。

在以下部分中在不引起歧义的前提下我们默认 n 为线段树的叶子个数。

基于以上定义（weight balanced binary leafy tree），我们可以证明以下性质：

1. 线段树的任意子树均为线段树。

基于以上定义（weight balanced binary leafy tree），我们可以证明以下性质：

1. 线段树的任意子树均为线段树。
2. 任意区间在线段树上可以表示为一个结点对应区间的后缀和另一结点对应区间前缀的不交并。

基于以上定义（weight balanced binary leafy tree），我们可以证明以下性质：

1. 线段树的任意子树均为线段树。
2. 任意区间在线段树上可以表示为一个结点对应区间的后缀和另一结点对应区间前缀的不交并。
3. 一个前缀或者后缀在线段树上可以表示成若干个结点对应区间的不交并，满足这些结点两两深度不同且这些结点到根的链并的节点数不超过两倍的这些结点深度最大值。

基于以上定义（weight balanced binary leafy tree），我们可以证明以下性质：

1. 线段树的任意子树均为线段树。
2. 任意区间在线段树上可以表示为一个结点对应区间的后缀和另一结点对应区间前缀的不交并。
3. 一个前缀或者后缀在线段树上可以表示成若干个结点对应区间的不交并，满足这些结点两两深度不同且这些结点到根的链并的节点数不超过两倍的这些结点深度最大值。
4. 线段树的结点个数为 $2n-1$，树高为 $\Theta(\log n)$。

第一个性质确保了我们可以递归地设计算法。

第一个性质确保了我们可以递归地设计算法。

后四个性质可以推导出一个重要的性质：任意区间可以在线段树上用不超过两倍的树高（即 $\Theta(\log n)$）的结点的对应区间的不交并表示，且这些结点到根的链并的结点数不超过四倍的树高。

基于上述性质，我们可以支持以下操作：

基于上述性质，我们可以支持以下操作：

1. 构造线段树：递归构造根结点的左子树和右子树，构造根结点的信息。时间复杂度为构造所有结点的时间之和，当每个结点可以 $\Theta(1)$ 构造时时间复杂度即 $\Theta(n)$。

基于上述性质，我们可以支持以下操作：

1. 构造线段树：递归构造根结点的左子树和右子树，构造根结点的信息。时间复杂度为构造所有结点的时间之和，当每个结点可以 $\Theta(1)$ 构造时时间复杂度即 $\Theta(n)$。
2. 单点修改：递归修改单点所在的子树，修改根节点的信息。时间复杂度为不超过单点深度的所有深度的修改结点信息的时间之和，当每个结点可以 $\Theta(1)$ 修改信息时时间复杂度即 $\Theta(\log n)$。

为了方便区间查询，我们可以对一个结点直接维护查询其对应区间的所有答案使得我们可以在 $\Theta(1)$ 的时间复杂度查询一个结点的信息。

为了方便区间查询，我们可以对一个结点直接维护查询其对应区间的所有答案使得我们可以在 $\Theta(1)$ 的时间复杂度查询一个结点的信息。

3. 区间查询：我们可以先用 $\Theta(n)$ 的时间定位出若干个结点满足其对应区间的不交并为给定区间（判断是否为根结点对应区间，如果否则递归左子树和右子树），再依次查询这些结点，时间复杂度为两者之和，当我们可以 $\Theta(1)$ 查询一个结点时时间复杂度即 $\Theta(\log n)$。

为了方便区间修改，我们可以对一个结点维护一个懒标记，表示其后代结点（一般写法不包含自身）的结点实际信息为子线段树维护的信息经过懒标记代表的操作后的信息。

为了方便区间修改，我们可以对一个结点维护一个懒标记，表示其后代结点（一般写法不包含自身）的结点实际信息为子线段树维护的信息经过懒标记代表的操作后的信息。

一种维护方式是在之后进行子树内操作时将标记下传，即对左右子树分别进行该操作（修改根结点信息并在根结点打上懒标记）后将自身懒标记清空。

为了方便区间修改，我们可以对一个结点维护一个懒标记，表示其后代结点（一般写法不包含自身）的结点实际信息为子线段树维护的信息经过懒标记代表的操作后的信息。

一种维护方式是在之后进行子树内操作时将标记下传，即对左右子树分别进行该操作（修改根结点信息并在根结点打上懒标记）后将自身懒标记清空。

另一种维护方式是当懒标记的操作关于其他操作有着良好交换律时，我们不必下传懒标记，在区间查询定位结点时处理到根链的懒标记信息，这种维护方式称为标记永久化。

为了方便区间修改，我们可以对一个结点维护一个懒标记，表示其后代结点（一般写法不包含自身）的结点实际信息为子线段树维护的信息经过懒标记代表的操作后的信息。

一种维护方式是在之后进行子树内操作时将标记下传，即对左右子树分别进行该操作（修改根结点信息并在根结点打上懒标记）后将自身懒标记清空。

另一种维护方式是当懒标记的操作关于其他操作有着良好交换律时，我们不必下传懒标记，在区间查询定位结点时处理到根链的懒标记信息，这种维护方式称为标记永久化。

我们可以通过懒标记实现在 $\Theta(1)$ 的时间复杂度修改一个结点的信息。

4 区间修改：我们可以先用 $\Theta(n)$ 的时间定位出若干个结点满足其对应区间的不交并为给定区间，再一次修改这些结点的信息，时间复杂度为两者之和，当我们可以 $\Theta(1)$ 修改一个结点的信息时时间复杂度即 $\Theta(\log n)$。

4 区间修改：我们可以先用 $\Theta(n)$ 的时间定位出若干个结点满足其对应区间的不交并为给定区间，再一次修改这些结点的信息，时间复杂度为两者之和，当我们可以 $\Theta(1)$ 修改一个结点的信息时时间复杂度即 $\Theta(\log n)$。

这里的区间修改指的是对一个区间的信息分别进行一种修改，而区间翻转等涉及动态编号的区间操作就没有很好的实现方法，只能翻转一个线段树结点对应的区间，为了方便实现可以将信息数量补全成 2 的幂后将线段树建为满二叉树。

此外，设计线段树维护的信息（含懒标记）时，我们不仅要保证每个操作的复杂度必须在可接受范围内，还要保证每个信息在每个操作下都能在可接受的时间复杂度下维护。

此外，设计线段树维护的信息（含懒标记）时，我们不仅要保证每个操作的复杂度必须在可接受范围内，还要保证每个信息在每个操作下都能在可接受的时间复杂度下维护。

Problem

两个问题的例子：

1. 区间加、区间乘、区间赋值、区间求和、区间求最大值。

此外，设计线段树维护的信息（含懒标记）时，我们不仅要保证每个操作的复杂度必须在可接受范围内，还要保证每个信息在每个操作下都能在可接受的时间复杂度下维护。

Problem

两个问题的例子：

1. 区间加、区间乘、区间赋值、区间求和、区间求最大值。
2. 区间加、区间取最大值、区间求最大值。

此外，设计线段树维护的信息（含懒标记）时，我们不仅要保证每个操作的复杂度必须在可接受范围内，还要保证每个信息在每个操作下都能在可接受的时间复杂度下维护。

Problem

两个问题的例子：

1. 区间加、区间乘、区间赋值、区间求和、区间求最大值。
2. 区间加、区间取最大值、区间求最大值。

而区间加、区间取最大值、区间求和就没有那么便于设计信息，我们会在后面作为一个进阶技巧的经典问题进行介绍。

Problem (区间最大子段和)

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$，有以下两个操作：

1. 单点修改：给定 x, y，将 a_x 修改为 y。
2. 求区间最大子段和：给定 x, y，求 $\max\limits_{x \leqslant l \leqslant r \leqslant y} \sum\limits_{i=l}^{r} a_i$。

查询次数和 n 同阶，要求时间复杂度 $O(n \log n)$。

Problem (区间最大子段和)

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$，有以下两个操作：

1. 单点修改：给定 x, y，将 a_x 修改为 y。
2. 求区间最大子段和：给定 x, y，求 $\max\limits_{x \leqslant l \leqslant r \leqslant y} \sum\limits_{i=l}^{r} a_i$。

查询次数和 n 同阶，要求时间复杂度 $O(n \log n)$。

Solution

我们可以对线段树的每一个结点维护区间和、区间最大子段和、区间最大前缀和、区间最大后缀和。

Problem (矩形覆盖问题)

给定 n 个矩形，坐标为 $[1, m] \cap \mathbb{Z}$ 中的数，求这些矩形并的面积。n, m 同阶，要求时间复杂度 $O(n \log n)$。

Problem (矩形覆盖问题)

给定 n 个矩形，坐标为 $[1, m] \cap \mathbb{Z}$ 中的数，求这些矩形并的面积。n, m 同阶，要求时间复杂度 $O(n \log n)$。

Solution

我们可以使用扫描线的技巧：我们依次维护并查询横坐标为 $[x, x+1]$ 的部分的面积。具体而言，我们从小到大枚举 x，维护横坐标在 $[x, x+1]$，纵坐标在 $[i, i+1]$ 的矩形是否被给定的矩形覆盖，查询即全局计数。

Solution

对于一个横坐标为 $[a,b]$，纵坐标为 $[c,d]$ 的矩形，我们在 a 的查询前加入一个纵坐标为 $[c,d]$ 的矩形，在 b 的查询前删去一个纵坐标为 $[c,d]$ 的矩形。这样我们就将问题转化成了一个坐标为 $[1,m-1]$ 的 n 次区间加 1，n 次区间减 1，m 次全局计数非 0 数的个数。

Solution

对于一个横坐标为 $[a,b]$，纵坐标为 $[c,d]$ 的矩形，我们在 a 的查询前加入一个纵坐标为 $[c,d]$ 的矩形，在 b 的查询前删去一个纵坐标为 $[c,d]$ 的矩形。这样我们就将问题转化成了一个坐标为 $[1,m-1]$ 的 n 次区间加 1，n 次区间减 1，m 次全局计数非 0 数的个数。

我们可以对线段树的每一个结点维护区间最小值、区间最小值个数和区间加的懒标记。显然我们可以将修改一般化为区间加，将查询的非 0 数的个数转化为若最小值为 0 则为数的个数减去最小值个数，否则即为数的个数。注意这种维护 0 的个数的方式基于查询时每个数非负的性质。

Solution

对于一个横坐标为 $[a,b]$，纵坐标为 $[c,d]$ 的矩形，我们在 a 的查询前加入一个纵坐标为 $[c,d]$ 的矩形，在 b 的查询前删去一个纵坐标为 $[c,d]$ 的矩形。这样我们就将问题转化成了一个坐标为 $[1,m-1]$ 的 n 次区间加 1，n 次区间减 1，m 次全局计数非 0 数的个数。

我们可以对线段树的每一个结点维护区间最小值、区间最小值个数和区间加的懒标记。显然我们可以将修改一般化为区间加，将查询的非 0 数的个数转化为若最小值为 0 则为数的个数减去最小值个数，否则即为数的个数。注意这种维护 0 的个数的方式基于查询时每个数非负的性质。

注意在代码中区分 n 和 m，如果你很习惯用 n 表示线段树的叶子结点个数，那么你可以在代码中继续沿用这一习惯，将题目中的 n 用其它标识符表示。

Problem (离线静态二维数点)

给定 n 个点，坐标为 $[1,m]\cap\mathbb{Z}$ 中的数，再给定 q 个询问点 (a,b)，询问多少个给定点 (x,y) 满足 $x\leqslant a, y\leqslant b$。

n,m,q 同阶，要求时间复杂度 $O(n\log n)$。

Problem (离线静态二维数点)

给定 n 个点，坐标为 $[1,m]\cap\mathbb{Z}$ 中的数，再给定 q 个询问点 (a,b)，询问多少个给定点 (x,y) 满足 $x\leqslant a, y\leqslant b$。

n,m,q 同阶，要求时间复杂度 $O(n\log n)$。

Solution

同样使用扫描线。我们将所有点按横坐标排序，依次加入给定点并处理询问，线段树维护当前纵坐标为下标的点的数量，查询即查询线段树上的前缀和。

Problem (离线静态二维数点)

给定 n 个点，坐标为 $[1,m]\cap\mathbb{Z}$ 中的数，再给定 q 个询问点 (a,b)，询问多少个给定点 (x,y) 满足 $x\leqslant a, y\leqslant b$。

n,m,q 同阶，要求时间复杂度 $O(n\log n)$。

Solution

同样使用扫描线。我们将所有点按横坐标排序，依次加入给定点并处理询问，线段树维护当前纵坐标为下标的点的数量，查询即查询线段树上的前缀和。

事实上，这个问题在离散化后有更加优秀的算法，但这里不加介绍。

Problem (【SHOI2008】堵塞的交通)

有一张 $2 \times n$ 的网格，实时更新相邻两点间连边的存在性，在线询问两点间连通性。

$1 \leqslant n \leqslant 10^5$。

Problem (【SHOI2008】堵塞的交通)

有一张 $2 \times n$ 的网格，实时更新相邻两点间连边的存在性，在线询问两点间连通性。

$1 \leqslant n \leqslant 10^5$。

Solution

线段树维护对应区间角上 6 对点间的连通性。

询问时注意可以通过两边绕回来。

Problem (【SCOI2010】序列操作)

对一个长度为 n 的 01 数组有如下几种操作：

1. 区间赋 0，区间赋 1。
2. 区间反转（0 变 1，1 变 0）。
3. 求区间 1 的个数。
4. 求区间最长的连续 1 的长度。

$1 \leqslant n, m \leqslant 10^5$。

Problem (【SCOI2010】序列操作)

对一个长度为 n 的 01 数组有如下几种操作：

1. 区间赋 0，区间赋 1。
2. 区间反转（0 变 1，1 变 0）。
3. 求区间 1 的个数。
4. 求区间最长的连续 1 的长度。

$1 \leqslant n, m \leqslant 10^5$。

Solution

维护区间 0,1 的个数，区间 0,1 的最长连续前、后缀和最长连续段。

Problem (【NOI2016】区间)

给定数轴上 n 个区间，你需要从中挑出 m 个，使得它们至少包含一个共同点，并且这 m 个区间的最大长度与最小长度尽可能接近。

$1 \leqslant m \leqslant n \leqslant 5 \times 10^5$，$0 \leqslant l_i \leqslant r_i \leqslant 10^9$。

Problem (【NOI2016】区间)

给定数轴上 n 个区间，你需要从中挑出 m 个，使得它们至少包含一个共同点，并且这 m 个区间的最大长度与最小长度尽可能接近。

$1 \leqslant m \leqslant n \leqslant 5 \times 10^5$，$0 \leqslant l_i \leqslant r_i \leqslant 10^9$。

Solution

将所有区间按照长度排序后双指针，用线段树维护当前每个点被覆盖了几次，相当于区间加减维护最大值。

Problem

维护一个集合，q 次操作：插入一个数、删除一个数、查询全局第 k 大。要求时间复杂度 $O(q \log q)$。

Problem

维护一个集合，q 次操作：插入一个数、删除一个数、查询全局第 k 大。要求时间复杂度 $O(q\log q)$。

对于维护一个集合的问题，我们可以发现数的顺序几乎没有任何作用，我们往往用线段树维护对每个值的一些信息，例如令 f_i 为 i 的出现次数然后维护 f。我们一般将这个技巧称为权值线段树。

Solution

我们可以通过建立权值线段树将问题转化为了单点修改，查询前缀和 $\geqslant k$ 的第一个下标。我们直接进行线段树上二分。

二分会进行 $\log n + \Theta(1)$ 次检验，如果我们在检验的时候进行了一次线段树操作，在直接套用的情况下我们的总时间复杂度会有 $\Theta(\log^2 n)$，但是我们观察二分的过程可以发现比较冗余，首先我们可以以线段树中包含查询区间的结点的左右子树分界点作为二分的判断点，然后我们一般来说就可以复用之前的结果做到 $\Theta(1)$ 的判断。

Problem

现在有一个大小为 $v = 10^9$ 的数组（初始值都是 0），你需要支持区间加，求区间和，强制在线。

Problem

现在有一个大小为 $v = 10^9$ 的数组（初始值都是 0），你需要支持区间加，求区间和，强制在线。

注意到虽然线段树很大，但是绝大多数结点都和初始状态一样为空信息。这个时候我们可以动态开点。

Problem

现在有一个大小为 $v = 10^9$ 的数组（初始值都是 0），你需要支持区间加，求区间和，强制在线。

注意到虽然线段树很大，但是绝大多数结点都和初始状态一样为空信息。这个时候我们可以动态开点。

具体地，对一个线段树节点，当它没有被使用到时（在这个问题中，相当于它不在修改定位的结点的到根链上且标记永久化或懒标记尚未下传到当前结点），我们将这个结点当成空节点。

只有当一个节点在修改中第一次被访问到时，我们才给它一个地址，并同时保存指向它两个儿子的链接。

只有当一个节点在修改中第一次被访问到时，我们才给它一个地址，并同时保存指向它两个儿子的链接。

这样每次修改只会最多增加 $\Theta(\log v)$ 个结点。

只有当一个节点在修改中第一次被访问到时，我们才给它一个地址，并同时保存指向它两个儿子的链接。

这样每次修改只会最多增加 $\Theta(\log v)$ 个结点。

询问时正常进行，遇到空节点就当成一个正常的，区间和为 0 的结点对待即可。

只有当一个节点在修改中第一次被访问到时，我们才给它一个地址，并同时保存指向它两个儿子的链接。

这样每次修改只会最多增加 $\Theta(\log v)$ 个结点。

询问时正常进行，遇到空节点就当成一个正常的，区间和为 0 的结点对待即可。

如果遇到可以离线的类似问题还是建议离散化后转化成一般的线段树问题以减小常数。

Problem

单点加，矩阵求和或求最大值。

Problem

单点加，矩阵求和或求最大值。

Solution

我们使用两层线段树嵌套。外层线段树维护横坐标为下标的矩形，内层线段树维护横坐标为外层对应结点对应区间纵坐标为下标的矩形。

Problem

单点加，矩阵求和或求最大值。

Solution

我们使用两层线段树嵌套。外层线段树维护横坐标为下标的矩形，内层线段树维护横坐标为外层对应结点对应区间纵坐标为下标的矩形。

内层线段树需要动态开点，时空复杂度为 $\Theta(n \log^2 n)$。

Problem

单点加，矩阵求和或求最大值。

Solution

我们使用两层线段树嵌套。外层线段树维护横坐标为下标的矩形，内层线段树维护横坐标为外层对应结点对应区间纵坐标为下标的矩形。

内层线段树需要动态开点，时空复杂度为 $\Theta(n \log^2 n)$。

更高维的情况类似。

Problem

一个序列 a 的逆序对为满足 $(i < j) \wedge (a_i > a_j)$ 的 (i, j)。
每次修改会给定 x, y，交换 a_x 和 a_y，每次修改后求 a 的逆序对数。

Problem

一个序列 a 的逆序对为满足 $(i < j) \wedge (a_i > a_j)$ 的 (i, j)。
每次修改会给定 x, y，交换 a_x 和 a_y，每次修改后求 a 的逆序对数。

Solution

我们考虑一次操作后逆序对的变化，可以发现只可能有 $x < k < y$ 且 a_k 在 a_x 与 a_y 之间的 (k, x) 和 (k, y) 以及 (x, y)。

Problem

一个序列 a 的逆序对为满足 $(i < j) \wedge (a_i > a_j)$ 的 (i, j)。
每次修改会给定 x, y，交换 a_x 和 a_y，每次修改后求 a 的逆序对数。

Solution

我们考虑一次操作后逆序对的变化，可以发现只可能有 $x < k < y$ 且 a_k 在 a_x 与 a_y 之间的 (k, x) 和 (k, y) 以及 (x, y)。
我们可以转化为带修改的二维数点问题，可以用二维线段树解决。

Solution

动态二维数点的另一种处理方式是二进制分组。

Solution

动态二维数点的另一种处理方式是二进制分组。

维护若干组大小为 2^i 的静态二维数点，插入相当于添加一个大小为 2^0 的，删除相当于点权变成负的。每当有大小相同的组时合并成更大的一组。

Solution

动态二维数点的另一种处理方式是二进制分组。

维护若干组大小为 2^i 的静态二维数点，插入相当于添加一个大小为 2^0 的，删除相当于点权变成负的。每当有大小相同的组时合并成更大的一组。

查询时在每个组中都查一遍。

Solution

动态二维数点的另一种处理方式是二进制分组。

维护若干组大小为 2^i 的静态二维数点，插入相当于添加一个大小为 2^0 的，删除相当于点权变成负的。每当有大小相同的组时合并成更大的一组。

查询时在每个组中都查一遍。

时间复杂度为均摊 $\Theta(q\log^2 q)$，空间复杂度为 $\Theta(q\log q)$。

通常的数据结构问题我们只需要数据结构的当前结构及信息，但有时我们需要之前的结构及信息，我们将这类需求称为可持久化。

通常的数据结构问题我们只需要数据结构的当前结构及信息，但有时我们需要之前的结构及信息，我们将这类需求称为可持久化。

常见的可持久化需求可以分为三种：

1. 支持撤回上一次操作。

通常的数据结构问题我们只需要数据结构的当前结构及信息，但有时我们需要之前的结构及信息，我们将这类需求称为可持久化。

常见的可持久化需求可以分为三种：

1. 支持撤回上一次操作。
2. 支持查询在一个历史版本上进行查询。

通常的数据结构问题我们只需要数据结构的当前结构及信息，但有时我们需要之前的结构及信息，我们将这类需求称为可持久化。

常见的可持久化需求可以分为三种：

1. 支持撤回上一次操作。
2. 支持查询在一个历史版本上进行查询。
3. 支持在一个历史版本上进行修改。

通常的数据结构问题我们只需要数据结构的当前结构及信息，但有时我们需要之前的结构及信息，我们将这类需求称为可持久化。

常见的可持久化需求可以分为三种：

1. 支持撤回上一次操作。
2. 支持查询在一个历史版本上进行查询。
3. 支持在一个历史版本上进行修改。

其中第三种需求包含了前两种需求，在可以离线的情况下后两种需求可以转化为第一种需求。

Path Copy

Path Copy 是指对需要修改的链先拷贝一份后再进行修改，不影响原先有的结点的值。一般可持久化树形数据结构都会使用这种技巧。

Fat Node

Fat Node 是指对原先的每个变量存储一个二元组的序列，表示这个变量在某个时间是某个值。可持久化数组可以采用这种技巧，一些可撤销数据结构可以转化为可撤销数组用类似的方式完成。这种技巧在内存上有时也有一定的优势，但在历史版本上修改有时会遇到一定的瓶颈。

Problem

在第 k 次版本上，单点加得到一个新的版本。
在第 k 次版本上，求区间和。

Problem

在第 k 次版本上，单点加得到一个新的版本。
在第 k 次版本上，求区间和。

我们像动态开点一样，对线段树的每个结点存储其左儿子结点和右儿子结点的编号，我们在修改一个结点的时候，新建一个结点拷贝其信息再在新的结点基础上进行修改。

实现上，我们可以函数式地设计程序，修改函数返回其新的结点编号，先新建一个结点拷贝当前根节点的状态，如果需要递归修改则将左右儿子改为对应的返回值，最后修改当前结点的信息并返回。

实现上，我们可以函数式地设计程序，修改函数返回其新的结点编号，先新建一个结点拷贝当前根节点的状态，如果需要递归修改则将左右儿子改为对应的返回值，最后修改当前结点的信息并返回。

可持久化数组一般用可持久化线段树实现。

实现上，我们可以函数式地设计程序，修改函数返回其新的结点编号，先新建一个结点拷贝当前根节点的状态，如果需要递归修改则将左右儿子改为对应的返回值，最后修改当前结点的信息并返回。

可持久化数组一般用可持久化线段树实现。

由于这种可持久化线段树被黄嘉泰在考场上独立实现并随之在 OI 中推广，因此也有“主席树”的俗称。

Problem (静态区间第 k 大)

给定一个长度为 n 的数组，q 次询问某个区间的第 k 大的数。要求时间复杂度 $O((n+q)\log n)$。

Problem (静态区间第 k 大)

给定一个长度为 n 的数组，q 次询问某个区间的第 k 大的数。要求时间复杂度 $O((n+q)\log n)$。

Solution

我们可以先通过离散化使得数组的元素属于 $[1,n]\cap\mathbb{Z}$，我们考虑用可持久化线段树处理出每个前缀的权值线段树（在前一个版本的基础上进行一次单点修改即可），然后在询问的区间的权值线段树中每个结点的信息可以表示为预处理出的两个前缀的对应位置的信息的差，我们可以在两棵树上一起进行线段树二分。

Problem (静态链上第 k 大)

给定一棵 n 个结点的树，每个点有点权，q 次询问某条链的第 k 大的点权。要求时间复杂度 $O((n+q)\log n)$。

Problem (静态链上第 k 大)

给定一棵 n 个结点的树，每个点有点权，q 次询问某条链的第 k 大的点权。要求时间复杂度 $O((n+q)\log n)$。

Solution

我们可以离散化后用可持久化线段树处理出每条到根链的权值线段树，然后询问的链 $\mathrm{path}(u,v)$ 的权值线段树可以用 u，v，$\mathrm{lca}(u,v)$ 和 $\mathrm{par}(\mathrm{lca}(u,v))$ 四个点的到根链的权值线段树的和差表示。类似上一题进行线段树上二分即可。

Problem (在线静态二维数点)

给定 n 个点，坐标为 $[1,m]\cap\mathbb{Z}$ 中的数。q 次询问，每次给定一个询问点 (a,b)，询问多少个给定点 (x,y) 满足 $x\leqslant a, y\leqslant b$。强制在线。

n,m,q 同阶，要求时间复杂度 $O(n\log n)$。

Problem (在线静态二维数点)

给定 n 个点，坐标为 $[1,m]\cap\mathbb{Z}$ 中的数。q 次询问，每次给定一个询问点 (a,b)，询问多少个给定点 (x,y) 满足 $x\leqslant a,y\leqslant b$。强制在线。

n,m,q 同阶，要求时间复杂度 $O(n\log n)$。

Solution

类似离线做法，我们同样使用扫描线，用可持久化线段树替代原先的线段树进行预处理，每次在历史版本上查询即可。

Problem (区间颜色数 /【SDOI2009】HH 的项链 /【HEOI2012】采花)

给定一个数组，数字代表颜色，多次询问区间 $[l, r]$ 内有几个不同的颜色。

Problem (区间颜色数 / 【SDOI2009】HH 的项链 / 【HEOI2012】采花)

给定一个数组，数字代表颜色，多次询问区间 $[l, r]$ 内有几个不同的颜色。

Solution

考虑区间 $[l, r]$ 中每种颜色第一次出现的位置，如果能够仅让这些位置产生贡献，就能不重不漏地统计出答案。

Problem (区间颜色数 / 【SDOI2009】HH 的项链 / 【HEOI2012】采花)

给定一个数组，数字代表颜色，多次询问区间 $[l, r]$ 内有几个不同的颜色。

Solution

考虑区间 $[l, r]$ 中每种颜色第一次出现的位置，如果能够仅让这些位置产生贡献，就能不重不漏地统计出答案。

构造辅助数组 pre_x，表示点 x 前第一个和 x 颜色相同的位置，那么一个点 $x, l \leqslant x \leqslant r$，它的颜色在 $[l, r]$ 中第一次出现等价于 $\mathrm{pre}_x < l$。

Problem (区间颜色数 / 【SDOI2009】HH 的项链 / 【HEOI2012】采花)

给定一个数组，数字代表颜色，多次询问区间 $[l, r]$ 内有几个不同的颜色。

Solution

考虑区间 $[l, r]$ 中每种颜色第一次出现的位置，如果能够仅让这些位置产生贡献，就能不重不漏地统计出答案。

构造辅助数组 pre_x，表示点 x 前第一个和 x 颜色相同的位置，那么一个点 $x, l \leqslant x \leqslant r$，它的颜色在 $[l, r]$ 中第一次出现等价于 $\mathrm{pre}_x < l$。

这样就转化为了静态二维数点的问题。

Problem (带修区间第 k 大)

给定一个长度为 n 的数组（值为 $[1,n]\cap\mathbb{Z}$ 中的数），q 次操作每次单点修改一个数或询问某个区间的第 k 大的数。n 与 q 同阶，要求时间复杂度 $O(n\log^2 n)$。

Problem (带修区间第 k 大)

给定一个长度为 n 的数组（值为 $[1,n]\cap\mathbb{Z}$ 中的数），q 次操作每次单点修改一个数或询问某个区间的第 k 大的数。n 与 q 同阶，要求时间复杂度 $O(n\log^2 n)$。

Solution

观察到静态区间第 k 大相当于一个前缀和的形式。我们考虑用树状数组替代前缀和。

Problem (带修区间第 k 大)

给定一个长度为 n 的数组（值为 $[1,n]\cap\mathbb{Z}$ 中的数），q 次操作每次单点修改一个数或询问某个区间的第 k 大的数。n 与 q 同阶，要求时间复杂度 $O(n\log^2 n)$。

Solution

观察到静态区间第 k 大相当于一个前缀和的形式。我们考虑用树状数组替代前缀和。

我们对树状数组的每个位置维护对应区间的权值线段树，时间复杂度 $\Theta(n\log n+q\log^2 n)$。实际上只是个树套树，并不需要可持久化。

Problem (bzoj2653 middle)

定义一个区间 $[l,r]$ 的价值为其第 $\lfloor (r-l+1)/2 \rfloor$ 大的数（中位数）的大小。

多次在线询问对于左端点在 $[a,b]$ 内，右端点在 $[c,d]$ 内的所有区间，最大价值是多少，保证 $a<b<c<d$。

Solution

首先二分答案 x，然后我们将所有 $\geqslant x$ 的数定为 1，$<x$ 的数定为 -1，并假设我们拥有一颗关于这个 $1,-1$ 数组的线段树，我们不妨称之为关于 x 的线段树。

Solution

首先二分答案 x，然后我们将所有 $\geqslant x$ 的数定为 1，$<x$ 的数定为 -1，并假设我们拥有一颗关于这个 $1,-1$ 数组的线段树，我们不妨称之为关于 x 的线段树。

接下来就需要知道是否存在符合题意的区间满足区间和 >0。

Solution

首先二分答案 x，然后我们将所有 $\geqslant x$ 的数定为 1，$<x$ 的数定为 -1，并假设我们拥有一颗关于这个 $1,-1$ 数组的线段树，我们不妨称之为关于 x 的线段树。

接下来就需要知道是否存在符合题意的区间满足区间和 >0。

我们在关于 x 的可持久化线段树上维护区间最大前、后缀和即可求出。

Solution

首先二分答案 x，然后我们将所有 $\geqslant x$ 的数定为 1，$<x$ 的数定为 -1，并假设我们拥有一颗关于这个 $1,-1$ 数组的线段树，我们不妨称之为关于 x 的线段树。

接下来就需要知道是否存在符合题意的区间满足区间和 >0。

我们在关于 x 的可持久化线段树上维护区间最大前、后缀和即可求出。

在关于 x 的线段树的基础上构造关于 $x+1$ 的线段树时，只需要找到所有值为 x 的点，将它们从 1 修改为 -1 即可。

Problem (【UNR #1】火车管理)

有 n 个栈，你需要维护共 m 次以下操作：

1. 区间压入一个数。
2. 单点弹出一个数。
3. 询问区间栈顶的数的和。

$1 \leqslant n, m \leqslant 5 \times 10^5$。强制在线。

Problem (【UNR #1】火车管理)

有 n 个栈，你需要维护共 m 次以下操作：

1. 区间压入一个数。
2. 单点弹出一个数。
3. 询问区间栈顶的数的和。

$1 \leqslant n, m \leqslant 5 \times 10^5$。强制在线。

Solution

用可持久化线段树维护当前每个栈的栈顶及区间和。

Solution

区间压入一个数直接对可持久化线段树做区间赋值即可。

Solution

区间压入一个数直接对可持久化线段树做区间赋值即可。

单点弹出时，我们可以通过查询当前栈顶的数 x 在压入之前的那个版本的栈顶的数 y，做一次单点赋值即可。

Solution

区间压入一个数直接对可持久化线段树做区间赋值即可。

单点弹出时，我们可以通过查询当前栈顶的数 x 在压入之前的那个版本的栈顶的数 y，做一次单点赋值即可。

时空复杂度 $\Theta(m \log n)$。

给定两棵动态开点线段树（其根分别为 x 和 y），我们要把两者的对应位置合并，形成一棵新的线段树。我们可以递归实现：

1. 如果其中一棵树是空树，即根节点是空结点，那么合并的结果就是另一棵树。

给定两棵动态开点线段树（其根分别为 x 和 y），我们要把两者的对应位置合并，形成一棵新的线段树。我们可以递归实现：

1. 如果其中一棵树是空树，即根节点是空结点，那么合并的结果就是另一棵树。
2. 否则我们把两棵树的左子树和右子树分别合并，再更新根结点的信息。

给定两棵动态开点线段树（其根分别为 x 和 y），我们要把两者的对应位置合并，形成一棵新的线段树。我们可以递归实现：

1. 如果其中一棵树是空树，即根节点是空结点，那么合并的结果就是另一棵树。
2. 否则我们把两棵树的左子树和右子树分别合并，再更新根结点的信息。

我们发现我们每次合并两棵树，合并结束后都有一些结点不再会被使用，这些结点数等于合并过程中经过的两者非空的节点数，那么我们合并两棵树的时间复杂度就是关于合并结束后不会再被使用的结点数成线性。

给定两棵动态开点线段树（其根分别为 x 和 y），我们要把两者的对应位置合并，形成一棵新的线段树。我们可以递归实现：

1. 如果其中一棵树是空树，即根节点是空结点，那么合并的结果就是另一棵树。
2. 否则我们把两棵树的左子树和右子树分别合并，再更新根结点的信息。

我们发现我们每次合并两棵树，合并结束后都有一些结点不再会被使用，这些结点数等于合并过程中经过的两者非空的节点数，那么我们合并两棵树的时间复杂度就是关于合并结束后不会再被使用的结点数成线性。

因此我们可以证明线段树合并的时间复杂度不超过总的空间复杂度。

Problem (子树内众数)

给定一棵树，每个节点上有一个数字，求每个子树内出现次数最多的数的出现次数。要求时间复杂度 $\Theta(n \log n)$。

Problem (子树内众数)

给定一棵树，每个节点上有一个数字，求每个子树内出现次数最多的数的出现次数。要求时间复杂度 $\Theta(n \log n)$。

Solution

对每个结点用一棵权值线段树维护子树内每个数的出现次数，我们可以直接类似树形动态规划求子树大小一样用线段树合并即可得到。每个结点一开始只会有 $\Theta(\log n)$ 个结点，因此总时空复杂度为 $\Theta(n \log n)$。

Problem (【SSBR #2】hypnotic)

给定一个 n 个结点的二叉树，每个结点有两个或者零个儿子，叶子有权值 v_i，其他结点的权值为 $|v_{\text{lson}(i)} - v_{\text{rson}(i)}| + w_i$。

现在有 q 次操作，每次修改一个 v 或 w，然后问根的权值。

$1 \leqslant n, q \leqslant 200000, 0 \leqslant v_i, w_i \leqslant 20$。

Problem (【SSBR #2】hypnotic)

给定一个 n 个结点的二叉树，每个结点有两个或者零个儿子，叶子有权值 v_i，其他结点的权值为 $|v_{\text{lson}(i)} - v_{\text{rson}(i)}| + w_i$。

现在有 q 次操作，每次修改一个 v 或 w，然后问根的权值。

$1 \leqslant n, q \leqslant 200000, 0 \leqslant v_i, w_i \leqslant 20$。

Solution

我们考虑离线，对于每个结点，我们用动态开点线段树维护每个时间的权值，非叶子结点的线段树为儿子结点线段树合并后若干次区间加即可。

Problem (【HNOI2012】永无乡)

给出 n 个点的图，每个点有一个权值，你需要支持 q 次下面两个操作：

1. 在两个点之间连一条边。
2. 询问某个点所在连通块的第 k 大权值。(k 由每次询问给定。)

$1 \leqslant n \leqslant 10^5$，$1 \leqslant q \leqslant 3 \times 10^5$。

Problem (【HNOI2012】永无乡)

给出 n 个点的图，每个点有一个权值，你需要支持 q 次下面两个操作：

1. 在两个点之间连一条边。
2. 询问某个点所在连通块的第 k 大权值。(k 由每次询问给定。)

$1 \leqslant n \leqslant 10^5$，$1 \leqslant q \leqslant 3 \times 10^5$。

Solution

并查集维护连通块情况，直接对每个连通块维护一个权值线段树，合并就是线段树合并，询问时在线段树上二分。

与线段树合并对应，我们可以将一棵线段树根据位置是否不超过一个给定值分裂成两棵线段树。

与线段树合并对应，我们可以将一棵线段树根据位置是否不超过一个给定值分裂成两棵线段树。

显然我们只会可能把给定位置到根链的结点分裂成两个结点，时空复杂度 $\Theta(\log n)$。

Problem (【模板】普通平衡树)

维护若干个集合 A，支持以下操作：

1. 在 A_i 中插入 x。
2. 删除 A_i 中的 x。
3. 求 A_i 中比 x 小的元素个数。
4. 求 A_i 中第 y 小（比它小的有 $y-1$ 个）的元素。
5. 将 A_i 中比 x 小的所有元素插入 A_j，并在 A_i 中删除它们。

$1 \leqslant i, y, q \leqslant 10^5, 1 \leqslant x \leqslant 10^9$。强制在线。

Solution

我们显然可以通过权值线段树合并与分裂维护集合。

Solution

我们显然可以通过权值线段树合并与分裂维护集合。

首先看分裂：首先新建分裂出去的当前节点，如果左儿子要全被分走就设置好分裂走的左儿子，然后递归右边，否则直接递归左边。显然会增加 $\Theta(\log v)$ 的节点。

Solution

我们显然可以通过权值线段树合并与分裂维护集合。

首先看分裂：首先新建分裂出去的当前节点，如果左儿子要全被分走就设置好分裂走的左儿子，然后递归右边，否则直接递归左边。显然会增加 $\Theta(\log v)$ 的节点。

然后看合并，如果至少存在一个空节点直接返回，否则递归合并左右儿子。由于每次会删除一个节点，所以复杂度不超过其他操作中新建结点的复杂度。

Solution

我们显然可以通过权值线段树合并与分裂维护集合。

首先看分裂：首先新建分裂出去的当前节点，如果左儿子要全被分走就设置好分裂走的左儿子，然后递归右边，否则直接递归左边。显然会增加 $\Theta(\log v)$ 的节点。

然后看合并，如果至少存在一个空节点直接返回，否则递归合并左右儿子。由于每次会删除一个节点，所以复杂度不超过其他操作中新建结点的复杂度。

由于合并操作会删除大量的节点，可以写一个内存回收来保证总的节点个数不会太多。

Problem (Nastya and CBS)

给定多种括号的一个括号序列 S，你需要支持 q 次以下操作：

1. 修改一个位置的括号种类与方向。
2. 询问一个区间括号是否完美匹配。

$1 \leqslant |S|, q \leqslant 10^5$。

Problem (Nastya and CBS)

给定多种括号的一个括号序列 S，你需要支持 q 次以下操作：

1. 修改一个位置的括号种类与方向。
2. 询问一个区间括号是否完美匹配。

$1 \leqslant |S|, q \leqslant 10^5$。

Solution

可以发现，一个括号序列可能成为完美匹配括号序列的子序列当且仅当它不断缩去所有相邻左右括号的过程中所有缩去的括号对全部匹配。显然缩完后序列由若干个右括号后接若干个左括号组成。

Problem (Nastya and CBS)

给定多种括号的一个括号序列 S，你需要支持 q 次以下操作：

1. 修改一个位置的括号种类与方向。
2. 询问一个区间括号是否完美匹配。

$1 \leqslant |S|, q \leqslant 10^5$。

Solution

可以发现，一个括号序列可能成为完美匹配括号序列的子序列当且仅当它不断缩去所有相邻左右括号的过程中所有缩去的括号对全部匹配。显然缩完后序列由若干个右括号后接若干个左括号组成。

我们可以用哈希快速的判断一组左括号和一组右括号拼起来是否是完美匹配括号序列。

Solution

我们考虑用动态开点线段树维护每个结点对应区间缩去后的左括号序列和右括号序列，合并时我们将左儿子的左括号序列和右儿子的右括号序列进行匹配，即将较长者的线段树分裂出长度与较短者相同的前缀或后缀，直接比较哈希值即可判定这个结点对应序列的相邻左右括号对是否匹配。将较长者剩余的部分与另一个儿子的相同方向括号一起建立一个父亲结点合并成一棵新的线段树即可在 $O(\log l)$ 的时间复杂度内完成对一个结点信息的维护。

Solution

我们考虑用动态开点线段树维护每个结点对应区间缩去后的左括号序列和右括号序列，合并时我们将左儿子的左括号序列和右儿子的右括号序列进行匹配，即将较长者的线段树分裂出长度与较短者相同的前缀或后缀，直接比较哈希值即可判定这个结点对应序列的相邻左右括号对是否匹配。将较长者剩余的部分与另一个儿子的相同方向括号一起建立一个父亲结点合并成一棵新的线段树即可在 $O(\log l)$ 的时间复杂度内完成对一个结点信息的维护。

我们直接暴力维护线段树信息即可做到 $O(n \log n + q \log^2 n)$ 的时间复杂度。

Problem (Gorgeous Sequence)

给定一个长度为 n 的数组 a，q 次操作：

1. 给定 l, r, x，对所有的 $a_i (l \leqslant i \leqslant r)$，把 a_i 变成 $\min(a_i, x)$。
2. 给定 l, r，询问区间最大值。
3. 给定 l, r，询问区间和。

$1 \leqslant n, q \leqslant 10^6$。

Problem (Gorgeous Sequence)

给定一个长度为 n 的数组 a，q 次操作：

1. 给定 l, r, x，对所有的 $a_i (l \leqslant i \leqslant r)$，把 a_i 变成 $\min(a_i, x)$。
2. 给定 l, r，询问区间最大值。
3. 给定 l, r，询问区间和。

$1 \leqslant n, q \leqslant 10^6$。

Solution

我们对每个结点 u 维护区间最大值 $\mathrm{mx}(u)$ 及其个数 $\mathrm{mxcnt}(u)$，区间严格次大值 $\mathrm{se}(u)$ 和区间和 $\mathrm{sum}(u)$。

Solution

我们考虑把结点 u 子树中的所有结点的值对 x 取最小值的结果：

Solution

我们考虑把结点 u 子树中的所有结点的值对 x 取最小值的结果：

1. 如果 $\mathrm{mx}(u) \leqslant x$，那么显然不会产生任何影响，直接返回即可。

Solution

我们考虑把结点 u 子树中的所有结点的值对 x 取最小值的结果：

1. 如果 $\mathrm{mx}(u) \leqslant x$，那么显然不会产生任何影响，直接返回即可。
2. 如果 $\mathrm{se}(u) < x < \mathrm{mx}(u)$，那么我们只需要将 $\mathrm{mx}(u)$ 改成 x，区间和对应减小，并在之后递归下去时下传即可。

Solution

我们考虑把结点 u 子树中的所有结点的值对 x 取最小值的结果：

1. 如果 $\mathrm{mx}(u) \leqslant x$，那么显然不会产生任何影响，直接返回即可。
2. 如果 $\mathrm{se}(u) < x < \mathrm{mx}(u)$，那么我们只需要将 $\mathrm{mx}(u)$ 改成 x，区间和对应减小，并在之后递归下去时下传即可。
3. 如果 $x \leqslant \mathrm{se}(u)$，那么我们递归对左儿子和右儿子分别执行操作即可。

Solution

我们考虑把结点 u 子树中的所有结点的值对 x 取最小值的结果：

1. 如果 $\mathrm{mx}(u) \leqslant x$，那么显然不会产生任何影响，直接返回即可。
2. 如果 $\mathrm{se}(u) < x < \mathrm{mx}(u)$，那么我们只需要将 $\mathrm{mx}(u)$ 改成 x，区间和对应减小，并在之后递归下去时下传即可。
3. 如果 $x \leqslant \mathrm{se}(u)$，那么我们递归对左儿子和右儿子分别执行操作即可。

我们注意到，每次第三种情况递归下去时，该结点子树内不同数的个数至少会减少 1，由于初始所有结点的子树内不同数的个数最多为 $\Theta(n \log n)$，因此这部分的总时间复杂度为 $O(n \log n)$。

Solution

我们考虑把结点 u 子树中的所有结点的值对 x 取最小值的结果：

1. 如果 $\mathrm{mx}(u) \leqslant x$，那么显然不会产生任何影响，直接返回即可。
2. 如果 $\mathrm{se}(u) < x < \mathrm{mx}(u)$，那么我们只需要将 $\mathrm{mx}(u)$ 改成 x，区间和对应减小，并在之后递归下去时下传即可。
3. 如果 $x \leqslant \mathrm{se}(u)$，那么我们递归对左儿子和右儿子分别执行操作即可。

我们注意到，每次第三种情况递归下去时，该结点子树内不同数的个数至少会减少 1，由于初始所有结点的子树内不同数的个数最多为 $\Theta(n \log n)$，因此这部分的总时间复杂度为 $O(n \log n)$。

其余部分的时间复杂度与一般的线段树相同，因此总时间复杂度为 $\Theta((n+q) \log n)$。

Problem (Picks loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间加、区间求和。

$1 \leqslant n, q \leqslant 3 \times 10^5$。

Problem (Picks loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间加、区间求和。

$1 \leqslant n, q \leqslant 3 \times 10^5$。

Solution

我们在上一题的基础上进一步维护区间加的标记，区间取最小值沿用前一题的做法。

Solution

我们定义结点对应区间的最大值与父亲结点对应区间的最大值不相同的结点为关键点。初始时关键点数量最多为 $\Theta(n \log n)$，每次修改做多增加 $\Theta(\log n)$ 个关键点。

Solution

我们定义结点对应区间的最大值与父亲结点对应区间的最大值不相同的结点为关键点。初始时关键点数量最多为 $\Theta(n \log n)$，每次修改做多增加 $\Theta(\log n)$ 个关键点。

我们可以发现区间取最小值定位后递归下去的区间子树内一定存在一个在操作结束后变成非关键点的关键点，由于关键点最多减少 $\Theta((n+q)\log n)$ 个，因此这部分时间复杂度 $O((n+q)\log^2 n)$。

Solution

我们定义结点对应区间的最大值与父亲结点对应区间的最大值不相同的结点为关键点。初始时关键点数量最多为 $\Theta(n \log n)$，每次修改做多增加 $\Theta(\log n)$ 个关键点。

我们可以发现区间取最小值定位后递归下去的区间子树内一定存在一个在操作结束后变成非关键点的关键点，由于关键点最多减少 $\Theta((n+q)\log n)$ 个，因此这部分时间复杂度 $O((n+q)\log^2 n)$。

其他部分时间复杂度仍与一般的线段树相同，总时间复杂度 $O((n+q)\log^2 n)$。

Solution

我们定义结点对应区间的最大值与父亲结点对应区间的最大值不相同的结点为关键点。初始时关键点数量最多为 $\Theta(n\log n)$，每次修改做多增加 $\Theta(\log n)$ 个关键点。

我们可以发现区间取最小值定位后递归下去的区间子树内一定存在一个在操作结束后变成非关键点的关键点，由于关键点最多减少 $\Theta((n+q)\log n)$ 个，因此这部分时间复杂度 $O((n+q)\log^2 n)$。

其他部分时间复杂度仍与一般的线段树相同，总时间复杂度 $O((n+q)\log^2 n)$。

值得注意的是，在随机数据下表现为 $\Theta((n+q)\log n)$，而且我尚未听说过能卡到 $\Theta((n+q)\log^2 n)$ 时间复杂度的数据。

Problem (AcrossTheSky loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间取最大值、求区间和。

Problem (AcrossTheSky loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间取最大值、求区间和。

Solution

我们只需要同时维护区间最小值及其个数、区间最大值及其个数、区间严格次小值、区间严格次大值、区间和即可。注意取最大值操作可能对区间最小值等产生影响。

Problem (AcrossTheSky loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间取最大值、求区间和。

Solution

我们只需要同时维护区间最小值及其个数、区间最大值及其个数、区间严格次小值、区间严格次大值、区间和即可。注意取最大值操作可能对区间最小值等产生影响。

套用之前的时间复杂度分析可以得到总时间复杂度 $\Theta((n+q)\log n)$。

Problem (Mzl loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间取最大值、区间加、求区间内每个数被修改的次数之和。

$1 \leqslant n, q \leqslant 3 \times 10^5$。

Problem (Mzl loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间取最大值、区间加、求区间内每个数被修改的次数之和。

$1 \leqslant n, q \leqslant 3 \times 10^5$。

Solution

显然我们在区间取最小值的过程中，会一直递归直到对区间内所有最大值加一个相同的非 0 数，区间取最小值同理，区间加非 0 数会对区间内每个数增加一次被修改次数。

Problem (Mzl loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间取最大值、区间加、求区间内每个数被修改的次数之和。

$1 \leqslant n, q \leqslant 3 \times 10^5$。

Solution

显然我们在区间取最小值的过程中，会一直递归直到对区间内所有最大值加一个相同的非 0 数，区间取最小值同理，区间加非 0 数会对区间内每个数增加一次被修改次数。

我们在维护修改所需的内容的基础上维护最大值的修改次数之和与最小值的修改次数之和以及所有数的修改次数之和即可。总时间复杂度 $O((n+q)\log^2 n)$。

Problem (ChiTuShaoNian loves segment tree)

给定长度为 n 的两个数组 a 和 b，q 次操作：对其中一个数组区间取最小值或区间加，求区间内两个数组对应数之和的最大值。

$1 \leqslant n, q \leqslant 3 \times 10^5$。

Problem (ChiTuShaoNian loves segment tree)

给定长度为 n 的两个数组 a 和 b，q 次操作：对其中一个数组区间取最小值或区间加，求区间内两个数组对应数之和的最大值。

$1 \leqslant n, q \leqslant 3 \times 10^5$。

Solution

我们在每个结点对在两个数组内是否为最大值的四种情况分别维护对应数之和的最大值即可。时间复杂度 $O((n+q)\log^2 n)$。

Problem (Dzy loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间加、求区间最大公约数。

$1 \leqslant n, q \leqslant 10^5$。

Problem (Dzy loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间加、求区间最大公约数。

$1 \leqslant n, q \leqslant 10^5$。

Solution

我们先考虑没有区间取最小值的问题。

Problem (Dzy loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间加、求区间最大公约数。

$1 \leqslant n, q \leqslant 10^5$。

Solution

我们先考虑没有区间取最小值的问题。

因为 $\gcd(a_1, a_2, \ldots, a_n) = \gcd(a_1, a_2 - a_1, \ldots, a_n - a_{n-1})$，我们可以维护差分数组的区间最大公约数和前缀和，时间复杂度 $\Theta(n \log v + q(\log n + \log v))$。

Problem (Dzy loves segment tree)

给定一个长度为 n 的数组 a，q 次操作：区间取最小值、区间加、求区间最大公约数。

$1 \leqslant n, q \leqslant 10^5$。

Solution

我们先考虑没有区间取最小值的问题。

因为 $\gcd(a_1, a_2, \ldots, a_n) = \gcd(a_1, a_2 - a_1, \ldots, a_n - a_{n-1})$，我们可以维护差分数组的区间最大公约数和前缀和，时间复杂度 $\Theta(n \log v + q(\log n + \log v))$。

我们对每个线段树维护对应区间的最大值及其个数、次大值、除最大值以外的数与次大值差的最大公约数。时间复杂度 $O((n+q) \log n(\log n + \log v))$。

Problem (赛格蒙特彼茨)

给定一个长度为 n 的数组 a，q 次操作：区间取最大值、区间加、求区间历史最小值之和。

$1 \leqslant n, q \leqslant 10^5$。

Problem (赛格蒙特彼茨)

给定一个长度为 n 的数组 a，q 次操作：区间取最大值、区间加、求区间历史最小值之和。

$1 \leqslant n, q \leqslant 10^5$。

Solution

我们令 b_i 为 a_i 的历史最小值，$c_i = a_i - b_i$。我们只需要分别求出 a_i 与 c_i 的区间和即可。

Problem (赛格蒙特彼茨)

给定一个长度为 n 的数组 a，q 次操作：区间取最大值、区间加、求区间历史最小值之和。

$1 \leqslant n, q \leqslant 10^5$。

Solution

我们令 b_i 为 a_i 的历史最小值，$c_i = a_i - b_i$。我们只需要分别求出 a_i 与 c_i 的区间和即可。

我们观察每次操作对 c_i 的影响：第一种，给定结点对应区间 $[l, r]$ 与正数 x，如果 $a_i (l \leqslant i \leqslant r)$ 是区间内的最小值，那么将 c_i 变成 $c_i + x$；第二种，给定结点对应区间 $[l, r]$ 与数 x，将 $c_i (l \leqslant i \leqslant r)$ 变成 $\max(c_i + x, 0)$。

Solution

我们可以将问题转化为不超过 $\Theta((n+q)\log n)$ 次上述操作，不超过 q 次询问 c_i 的区间和。

Solution

我们可以将问题转化为不超过 $\Theta((n+q)\log n)$ 次上述操作，不超过 q 次询问 c_i 的区间和。

我们在线段树中对每个结点维护区间内 a_i 是区间内最小值的数的 c_i 最小值及其个数与次小值和其它数的最小值及其个数与次小值，像 a_i 一样维护即可。

Solution

我们可以将问题转化为不超过 $\Theta((n+q)\log n)$ 次上述操作，不超过 q 次询问 c_i 的区间和。

我们在线段树中对每个结点维护区间内 a_i 是区间内最小值的数的 c_i 最小值及其个数与次小值和其它数的最小值及其个数与次小值，像 a_i 一样维护即可。

总时间复杂度 $O((n+q)\log^3 n)$，但据说没有构造出 $\omega((n+q)\log^2 n)$ 的数据。

Problem (上帝造题的七分钟 2 / 花神游历各国)

给定一个长度为 n 的数组 a，q 次操作：区间开平方根下取整，求区间和。

$1 \leqslant n, q \leqslant 10^5, 1 \leqslant v \leqslant 10^{12}$，其中 $v = \max_{i=1}^{n} a_i$。

Problem (上帝造题的七分钟 2 / 花神游历各国)

给定一个长度为 n 的数组 a，q 次操作：区间开平方根下取整，求区间和。

$1 \leqslant n, q \leqslant 10^5, 1 \leqslant v \leqslant 10^{12}$，其中 $v = \max_{i=1}^{n} a_i$。

Solution

注意到每个数至多变化 $\Theta(\log \log v)$ 次。

Problem (上帝造题的七分钟 2 / 花神游历各国)

给定一个长度为 n 的数组 a，q 次操作：区间开平方根下取整，求区间和。

$1 \leqslant n, q \leqslant 10^5, 1 \leqslant v \leqslant 10^{12}$，其中 $v = \max_{i=1}^{n} a_i$。

Solution

注意到每个数至多变化 $\Theta(\log \log v)$ 次。

我们对每个结点维护区间和。如果区间和为区间长度即区间内每个 a_i 均为 1 那么直接返回，否则递归直至叶子进行单点修改。总时间复杂度 $\Theta((n \min(q, \log \log v) + q) \log n)$。

Problem (基础数据结构练习题)

给定一个长度为 n 的数组 a，q 次操作：区间开平方根下取整、区间加、求区间和。

$1 \leqslant n, q \leqslant 10^5, 1 \leqslant v \leqslant 10^5$。

Problem (基础数据结构练习题)

给定一个长度为 n 的数组 a，q 次操作：区间开平方根下取整、区间加、求区间和。

$1 \leqslant n, q \leqslant 10^5, 1 \leqslant v \leqslant 10^5$。

Solution

注意到 $\sqrt{a} - \sqrt{b} < \sqrt{(\sqrt{a} - \sqrt{b})(\sqrt{a} + \sqrt{b})} = \sqrt{a - b}$,
$\lfloor \sqrt{a} \rfloor - \lfloor \sqrt{b} \rfloor < \sqrt{a} - \sqrt{b} + 1$。

Problem (基础数据结构练习题)

给定一个长度为 n 的数组 a，q 次操作：区间开平方根下取整、区间加、求区间和。

$1 \leqslant n, q \leqslant 10^5, 1 \leqslant v \leqslant 10^5$。

Solution

注意到 $\sqrt{a} - \sqrt{b} < \sqrt{(\sqrt{a} - \sqrt{b})(\sqrt{a} + \sqrt{b})} = \sqrt{a - b}$，$\lfloor \sqrt{a} \rfloor - \lfloor \sqrt{b} \rfloor < \sqrt{a} - \sqrt{b} + 1$。

我们对每个结点维护最小值、最大值、区间和与区间加懒标记。如果开根能使极差减小则递归，否则进行区间加操作，总时间复杂度 $O((n + q \log n) \log \log v)$。

Problem (【UNR #5】诡异操作)

给定一个长度为 n 的序列 a，以及 q 个操作：

1. 给定 l, r, v，将 $a_i (l \leqslant i \leqslant r)$ 变为 $\lfloor a_i / v \rfloor$。
2. 给定 l, r, v，将 $a_i (l \leqslant i \leqslant r)$ 变为 $a_i \operatorname{and} v$。
3. 给定 l, r，求 $\sum_{i=l}^{r} a_i$。

$1 \leqslant n \leqslant 3 \times 10^5, 1 \leqslant q \leqslant 2 \times 10^5, 0 \leqslant a_i, v < 2^{128}$。计算时间复杂度时可将字长视为至少 128。

Problem (【UNR #5】诡异操作)

给定一个长度为 n 的序列 a，以及 q 个操作：

1. 给定 l, r, v，将 $a_i (l \leqslant i \leqslant r)$ 变为 $\lfloor a_i / v \rfloor$。
2. 给定 l, r, v，将 $a_i (l \leqslant i \leqslant r)$ 变为 $a_i \operatorname{and} v$。
3. 给定 l, r，求 $\sum_{i=l}^{r} a_i$。

$1 \leqslant n \leqslant 3 \times 10^5, 1 \leqslant q \leqslant 2 \times 10^5, 0 \leqslant a_i, v < 2^{128}$。计算时间复杂度时可将字长视为至少 128。

Solution

显然我们可以忽略除以 1 的操作。由于一个结点最多进行 $\log \max a_i$ 次除法，我们考虑对线段树的每个结点维护对应区间中数每一位中的 1 的个数，并维护区间与的懒标记。

Solution

可以发现这样需要维护 $\log v$ 个数而这些数在绝大多数结点的位数即结点对应区间长度（后面记作 l）的对数都很小，我们考虑用二进制分组的思想，转而维护 $\lfloor \log l \rfloor + 1$ 个数，第 $i(0 \leqslant i \leqslant \lfloor \log l \rfloor)$ 个数表示每一位的 1 的出现个数在 2^i 位上是否为 1。因此我们可以在 $\Theta(\log l)$ 的时间复杂度内修改一个结点的信息。

Solution

可以发现这样需要维护 $\log v$ 个数而这些数在绝大多数结点的位数即结点对应区间长度（后面记作 l）的对数都很小，我们考虑用二进制分组的思想，转而维护 $\lfloor \log l \rfloor + 1$ 个数，第 $i(0 \leqslant i \leqslant \lfloor \log l \rfloor)$ 个数表示每一位的 1 的出现个数在 2^i 位上是否为 1。因此我们可以在 $\Theta(\log l)$ 的时间复杂度内修改一个结点的信息。

因此单次区间除法定位部分和区间与的时间复杂度为 $\Theta(\log^2 n)$。注意到区间除法中定位完成后修改一个结点时每个结点最多会被递归 $\log \max a_i$ 次，因此这部分所有操作总时间复杂度为 $\Theta(\log \max a_i \sum_u \log l_u) = \Theta(n \log \max a_i)$。整道题总时间复杂度 $\Theta(n \log \max a_i + q \log^2 n)$。

这里讨论比较广义的李超树，一般指的是修改时先定位到区间，然后通过一些性质使得每次只需要暴力递归一侧的子树从而达到整体 $O(\log^2 n)$ 的时间复杂度。

Problem (线段树维护半平面交)

每次插入一个在区间上的一次函数，询问单点最大值。

Problem (线段树维护半平面交)

每次插入一个在区间上的一次函数，询问单点最大值。

Solution

相当于给出若干线段并求某横坐标最高点。

Problem (线段树维护半平面交)

每次插入一个在区间上的一次函数，询问单点最大值。

Solution

相当于给出若干线段并求某横坐标最高点。

线段树每个节点维护一条线段，我们可以通过恰当的操作满足覆盖每个点的 $\Theta(\log n)$ 个线段一定有一个是答案：

Problem (线段树维护半平面交)

每次插入一个在区间上的一次函数，询问单点最大值。

Solution

相当于给出若干线段并求某横坐标最高点。

线段树每个节点维护一条线段，我们可以通过恰当的操作满足覆盖每个点的 $\Theta(\log n)$ 个线段一定有一个是答案：

插入一个线段时，首先把它按照线段树分割成 $\Theta(\log n)$ 段，然后根据交点讨论往哪一侧递归下传。

Problem (线段树维护半平面交)

每次插入一个在区间上的一次函数，询问单点最大值。

Solution

相当于给出若干线段并求某横坐标最高点。

线段树每个节点维护一条线段，我们可以通过恰当的操作满足覆盖每个点的 $\Theta(\log n)$ 个线段一定有一个是答案：

插入一个线段时，首先把它按照线段树分割成 $\Theta(\log n)$ 段，然后根据交点讨论往哪一侧递归下传。

单次修改时间复杂度 $O(\log^2 n)$。

Problem (楼房重建)

长度为 n 的数组，初始为全部 0，m 次操作，每次修改一个结点的值，求修改后全局前缀最大值有多少种不同的取值。

$1 \leqslant n, m \leqslant 10^5$。

Problem (楼房重建)

长度为 n 的数组，初始为全部 0，m 次操作，每次修改一个结点的值，求修改后全局前缀最大值有多少种不同的取值。

$1 \leqslant n, m \leqslant 10^5$。

Solution

对线段树的每个非叶结点，维护如果前缀最大值是左子树中的最大值，那么右子树对应区间中全局前缀最大值变化了几次。

Problem (楼房重建)

长度为 n 的数组，初始为全部 0，m 次操作，每次修改一个结点的值，求修改后全局前缀最大值有多少种不同的取值。

$1 \leqslant n, m \leqslant 10^5$。

Solution

对线段树的每个非叶结点，维护如果前缀最大值是左子树中的最大值，那么右子树对应区间中全局前缀最大值变化了几次。

无论是修改还是询问，如果当前前缀最大值比左子树的最大值小，那么完成左子树的递归后前缀最大值就变成了左子树的最大值，递归右子树会造成的影响已经被维护在结点上了，只需单侧递归左子树；否则无需递归左子树，只需单侧递归右子树。

Problem (楼房重建)

长度为 n 的数组，初始为全部 0，m 次操作，每次修改一个结点的值，求修改后全局前缀最大值有多少种不同的取值。

$1 \leqslant n, m \leqslant 10^5$。

Solution

对线段树的每个非叶结点，维护如果前缀最大值是左子树中的最大值，那么右子树对应区间中全局前缀最大值变化了几次。

无论是修改还是询问，如果当前前缀最大值比左子树的最大值小，那么完成左子树的递归后前缀最大值就变成了左子树的最大值，递归右子树会造成的影响已经被维护在结点上了，只需单侧递归左子树；否则无需递归左子树，只需单侧递归右子树。

单次操作时间复杂度 $O(\log^2 n)$。

Problem (前缀 min 求和)

给定一个长度为 n 的数组，q 次操作：单点修改，求一个数列的 $\sum_{i=l}^{r} \min\{a_l, a_{l+1}, \ldots, a_i\}$。

Problem (前缀 min 求和)

给定一个长度为 n 的数组，q 次操作：单点修改，求一个数列的 $\sum_{i=l}^{r} \min\{a_l, a_{l+1}, \ldots, a_i\}$。

Solution

我们同样对每个非叶结点维护如果之前的最小值是左子树中的最小值，右子树对上面式子的贡献。

Problem (前缀 min 求和)

给定一个长度为 n 的数组，q 次操作：单点修改，求一个数列的 $\sum_{i=l}^{r} \min\{a_l, a_{l+1}, \ldots, a_i\}$。

Solution

我们同样对每个非叶结点维护如果之前的最小值是左子树中的最小值，右子树对上面式子的贡献。

修改和询问时，如果当前最小值 x 比左子树的最小值大，那么单侧递归左子树并加上右子树的贡献；否则直接计算左子树内的贡献，递归计算右子树。

我们观察到，单点修改，求前缀和时线段树每个结点的右儿子信息都不会被用到，因此我们可以直接用一个长度为 n 的数组存储，第 i 个位置的值表示以 i 为右端点的深度最小的结点的信息。我们观察到当 n 为 2 的幂的时候可以用二进制运算优化常数，同时查询时只会用到不超过查询位置的位置的值，因此我们可以转而维护扩充成 2 的幂后的线段树对应数组的前 n 个位置的值，这就是树状数组了。

Listing 1: c++

```
void update(int p,const int v,const int n){
    for(tr[p]+=v; (p+=p&(-p))<=n; tr[p]+=v);
}
int query(int p,int res=0){
    for(res+=tr[p]; (p&=p-1); res+=tr[p]);
}
```

将 p+=(p&(-p)) 和 p&=p-1 互换即可将求前缀和转换为求后缀和。

Problem

几个问题的例子：

1. 单点加，求前缀和。

Problem

几个问题的例子：

1. 单点加，求前缀和。
2. 单点加非负值，求前缀最大值。

Problem

几个问题的例子：

1. 单点加，求前缀和。
2. 单点加非负值，求前缀最大值。
3. 前缀加，求单点值。

Problem

几个问题的例子：

1. 单点加，求前缀和。
2. 单点加非负值，求前缀最大值。
3. 前缀加，求单点值。
4. 前缀加，求前缀值。

Problem

几个问题的例子：

1. 单点加，求前缀和。
2. 单点加非负值，求前缀最大值。
3. 前缀加，求单点值。
4. 前缀加，求前缀值。
5. 插入，删除，查询第 k 小。

当我们有 q_1 次单点加，q_2 次求前缀和，其中 q_1 与 q_2 不同阶时，树状数组也有一种平衡手段：我们将树状数组由二叉改至 k 叉。

当我们有 q_1 次单点加，q_2 次求前缀和，其中 q_1 与 q_2 不同阶时，树状数组也有一种平衡手段：我们将树状数组由二叉改至 k 叉。

若 $q_1 < q_2$，我们令一个结点的值为同一层前缀所有子树的叶子结点值之和。修改操作时间复杂度 $\Theta(k \log_k n)$，查询操作时间复杂度 $\Theta(\log_k n)$，平衡后 $k = \Theta(q_2/q_1)$，总时间复杂度 $\Theta(q_2 \log n / \log(1 + q_2/q_1))$。

当我们有 q_1 次单点加，q_2 次求前缀和，其中 q_1 与 q_2 不同阶时，树状数组也有一种平衡手段：我们将树状数组由二叉改至 k 叉。

若 $q_1 < q_2$，我们令一个结点的值为同一层前缀所有子树的叶子结点值之和。修改操作时间复杂度 $\Theta(k \log_k n)$，查询操作时间复杂度 $\Theta(\log_k n)$，平衡后 $k = \Theta(q_2/q_1)$，总时间复杂度 $\Theta(q_2 \log n / \log(1 + q_2/q_1))$。

若 $q_1 > q_2$，我们令一个结点的值为自身子树的叶子结点值之和。修改操作时间复杂度 $\Theta(\log_k n)$，查询操作时间复杂度 $\Theta(k \log_k n)$，平衡后 $k = \Theta(q_1/q_2)$，总时间复杂度 $\Theta(q_1 \log n / \log(1 + q_1/q_2))$。

钱哥给的代码：https://loj.ac/s/824942，并自称一般不必要，可能要 q_1, q_2 差 100 倍才能拉出一点差距。

钱哥给的代码：https://loj.ac/s/824942，并自称一般不必要，可能要 q_1, q_2 差 100 倍才能拉出一点差距。

容易发现：$\sqrt{n}$ 叉树状数组就是分块做法，当然一般的树形数据结构也可以视为多层分块。

Problem

维护一个二维数组 a，支持以下操作：

将 $a_{x,y}$ 增加 z；

求 $\sum_{i\leqslant x,j\leqslant y} a_{i,j}$ 的值。

$1 \leqslant x, y \leqslant 5000, 1 \leqslant z, q \leqslant 10^5$。

Problem

维护一个二维数组 a，支持以下操作：
将 $a_{x,y}$ 增加 z；
求 $\sum_{i\leqslant x,j\leqslant y} a_{i,j}$ 的值。
$1 \leqslant x, y \leqslant 5000, 1 \leqslant z, q \leqslant 10^5$。

Solution

二维树状数组只需要把一维时一层循环改成两层循环就行了。

“猫树”其实是个俗称，以 lxl 为代表的不喜欢俗称的人会将其称为基于线段树的二区间合并。

“猫树”其实是个俗称，以 lxl 为代表的不喜欢俗称的人会将其称为基于线段树的二区间合并。

我们考虑 cdq 分治：计算左边部分内答案，计算左边部分对右边部分的贡献，计算右边部分内答案。其中有一种常见的情形：贡献为类似区间的形式，我们可以先求出左边部分所有后缀的一些信息和右边部分所有前缀的一些信息，然后根据这些信息求左边部分对右边部分的贡献。

我们建出其分治树，那么一个 $[l,r]$ 区间如果 $l\neq r$ 那么就可以表示为一个树上结点的左儿子对应区间的一个后缀和右儿子对应区间的一个前缀的不交并。我们发现对于一个子树内有 n 个叶子的结点，我们构造和修改其信息的时间复杂度均为 $\Theta(n)$，因此 n 个叶子结点的线段树构造需要 $\Theta(n\log n)$ 的时间复杂度，单点修改需要 $\Theta(n)$ 的时间复杂度，但是区间查询我们可以通过位运算快速定位做到 $\Theta(1)$ 的时间复杂度。注意空间复杂度为 $\Theta(n\log n)$。

我们建出其分治树，那么一个 $[l,r]$ 区间如果 $l\neq r$ 那么就可以表示为一个树上结点的左儿子对应区间的一个后缀和右儿子对应区间的一个前缀的不交并。我们发现对于一个子树内有 n 个叶子的结点，我们构造和修改其信息的时间复杂度均为 $\Theta(n)$，因此 n 个叶子结点的线段树构造需要 $\Theta(n\log n)$ 的时间复杂度，单点修改需要 $\Theta(n)$ 的时间复杂度，但是区间查询我们可以通过位运算快速定位做到 $\Theta(1)$ 的时间复杂度。注意空间复杂度为 $\Theta(n\log n)$。

在 RMQ 问题中，我们用 ST 表处理四毛子中块间的答案，但是 ST 表要求维护信息的运算不仅需要满足结合律，而且需要一定的交换律以及对于任意 x，x 和 x 运算得到 x。不知道有没有热心人测一下猫树和 ST 表的速度比较。

Problem (静态区间最大子段和)

给定一个长度为 n 的序列，每次查询一个给定区间的最大子段和。要求预处理时空复杂度为 $O(n)$，每次查询时空复杂度为 $O(1)$。

Problem (静态区间最大子段和)

给定一个长度为 n 的序列，每次查询一个给定区间的最大子段和。要求预处理时空复杂度为 $O(n)$，每次查询时空复杂度为 $O(1)$。

Solution

考虑四毛子时块间的询问：我们对块的每个前后缀与猫树的每个左儿子后缀和右儿子前缀维护区间和、最大前后缀和以及最大子段和，我们就很容易地解决了块间的询问。

Solution

我们如果一直继续递归套娃的话复杂度感觉确实只能到 $\log^*$（迭代对数），我们重新考虑类似 RMQ 的压位方式，对于一个长度为 l 的块，我们很好说明其子区间的答案区间只和 $\frac{l(l+1)}{2}$ 个子区间的大小关系这 $(l(l+1)/2)!$ 的信息量有关（实际应该可以通过讨论做的更小一些），因此我们可以在 $\Theta(l^2(l(l+1)/2)!)$ 的时空复杂度内预处理出所有块的所有子区间作为询问区间时的答案区间，查询时只要查表然后用预处理好的前缀和就能 $\Theta(1)$ 算出答案。

Solution

我们如果一直继续递归套娃的话复杂度感觉确实只能到 $\log^*$（迭代对数），我们重新考虑类似 RMQ 的压位方式，对于一个长度为 l 的块，我们很好说明其子区间的答案区间只和 $\frac{l(l+1)}{2}$ 个子区间的大小关系这 $(l(l+1)/2)!$ 的信息量有关（实际应该可以通过讨论做的更小一些），因此我们可以在 $\Theta(l^2(l(l+1)/2)!)$ 的时空复杂度内预处理出所有块的所有子区间作为询问区间时的答案区间，查询时只要查表然后用预处理好的前缀和就能 $\Theta(1)$ 算出答案。

我们令 $l=(\log n)^{1/c}$，其中 c 是适当的常数，那么我们可以在 $o(n)$ 的时间复杂度内完成预处理。注意到四毛子要求块长为 $\Omega(\log n)$，我们可以在块内先套一层四毛子做到 $l=(\log n)^{1/c}=\Omega(\log\log n)$。于是我们用四毛子套四毛子将时间复杂度做到了线性。

我感觉实际效果可能还不如一层四毛子块内暴力卡卡常（有指令集更好）然后把块大小调平衡跑得快。好像有基于 RMQ 问题的其他做法，可能常数会更小，这个做法只是作为演示猫树作四毛子块间结构的例子。

Problem (【HNOI2016】序列)

给出一个长度为 n 的序列，q 次询问一个区间 $[l, r]$ 的所有子区间的区间最小值之和。要求时间复杂度 $O(n \log n + q)$，强制在线。

Problem (【HNOI2016】序列)

给出一个长度为 n 的序列，q 次询问一个区间 $[l,r]$ 的所有子区间的区间最小值之和。要求时间复杂度 $O(n\log n+q)$，强制在线。

Solution

对每个区间 $[l,r]$，我们在线段树上找到分界点 m 使得 $[l,m]$ 与 $(m,r]$ 分别为某个结点左儿子对应区间的后缀与右儿子对应区间的后缀。

Problem (【HNOI2016】序列)

给出一个长度为 n 的序列，q 次询问一个区间 $[l,r]$ 的所有子区间的区间最小值之和。要求时间复杂度 $O(n\log n+q)$，强制在线。

Solution

对每个区间 $[l,r]$，我们在线段树上找到分界点 m 使得 $[l,m]$ 与 $(m,r]$ 分别为某个结点左儿子对应区间的后缀与右儿子对应区间的后缀。

其中 $[l,m]$ 和 $(m,r]$ 内部的答案我们可以用单调栈线性预处理，就只需要考虑跨越部分了。对 $[l,m]$ 中记录每个数到 m 的区间最小值，考虑这个最小值贡献答案的区间的右端点 r，这个最右的右端点可以通过左右的归并得出。我们同时处理出每个右端点在左边左端点最右哪个位置可以取到方便询问。最小值在右边区间同理。这样预处理之后，就能做到 $\Theta(1)$ 的询问复杂度。

Solution

这里还是介绍一种比较套路的笛卡尔树做法。

我们建立笛卡尔树后，$[l,r]$ 的任意子区间的最小值有三种情况：l 和 r 的最近公共祖先，l 和 r 其中一个到最近公共祖先（不含）的链上的值，某个以 l 到 r 链上点的儿子为根且包含在 $[l,r]$ 内的子树中的点。

我们预处理出笛卡尔树每个结点的子树范围以及子树区间的查询结果，可以发现第二种情况和第三种情况可以对 l 的祖先中 $\geqslant l$ 的点预处理一个关于 l 的一次函数的前缀和，r 的祖先同理。复杂度瓶颈在于求笛卡尔树上的最近公共祖先即 RMQ 问题。

01-trie 就是字符集只有 01 的 trie。类似于权值线段树，01-trie 在和二进制有关的一些问题中可以用来维护权值的集合。

Problem (最大异或和)

给定一个序列 a，求 $a_i \operatorname{xor} a_j$ 的最大值。
$1 \leqslant n \leqslant 10^6, 0 \leqslant a_i < 2^{32}$。

Problem (最大异或和)

给定一个序列 a，求 $a_i \operatorname{xor} a_j$ 的最大值。
$1 \leqslant n \leqslant 10^6, 0 \leqslant a_i < 2^{32}$。

Solution

维护前缀的 01-trie，每次在上面查询最大异或值即可。

Problem (区间最大异或和)

给定一个序列 a。

每次给定三个整数 x, l, r，询问 $\max\{x \operatorname{xor} a_i \mid l \leqslant i \leqslant r\}$。

$1 \leqslant n, q \leqslant 5 \times 10^5, 0 \leqslant a_i, x < 2^{32}$。

Problem (区间最大异或和)

给定一个序列 a。
每次给定三个整数 x, l, r，询问 $\max\{x \operatorname{xor} a_i \mid l \leqslant i \leqslant r\}$。
$1 \leqslant n, q \leqslant 5 \times 10^5, 0 \leqslant a_i, x < 2^{32}$。

Solution

类似前面一题，我们建立前缀的可持久化 01-trie。

Problem (【模板】普通 Trie 树 1)

给定一个序列 a，有以下三种操作：

1. 将所有 a_i 增加 1。
2. 将所有 a_i 异或 x。
3. 求所有 a_i 与 x 异或的最大值右移 32 位后的结果。

$1 \leqslant n, q \leqslant 5 \times 10^5, 0 \leqslant a_i, x < 2^{32}$。

Solution

考虑维护所有 a_i 的 01-Trie，从低位到高位建。每个点维护子树中高位的最大值。

Solution

考虑维护所有 a_i 的 01-Trie，从低位到高位建。每个点维护子树中高位的最大值。

全局异或容易用打全局标记解决，全局增加 1 可以通过枚举最低位 1 的数量分别处理。实现上为每次交换两个子树后走左子树。全 0 的位置高位最大值增加 1。

Problem (【模板】普通 Trie 树 2)

维护若干个集合 a，支持以下操作：

1. 在 a_i 中插入 x。
2. 删除 a_i 中的 x。
3. 将 a_i 中的所有数异或 x。
4. 求 a_i 中比 x 小的元素个数。
5. 求 a_i 中第 y 小（比它小的有 $y-1$ 个）的元素。
6. 将 a_i 中比 x 小的所有元素插入 a_j，并在 a_i 中删除它们。

$1 \leqslant i, y, q \leqslant 10^5, 1 \leqslant x \leqslant 10^9$。强制在线。

Solution

和线段树的版本没什么区别，就是改成 Trie，每个结点维护一下异或标记就行了。

Problem (普通 vEB 树)

请你用一种数据结构（可以参考标题）维护一个集合 a，支持以下操作：

1. 插入 x（如果 x 存在则不插入）。
2. 删除 x（如果 x 不存在则不删除）。
3. 求集合中比 x 小的元素中最大的。
4. 求集合中比 x 大的元素中最小的。

$1 \leqslant x, q \leqslant 5 \times 10^6$，空间 20MB。

我会压位分块再套树状数组！

我会压位分块再套树状数组！
现在我们要求单次操作时间复杂度为 $o(\log n)$。

我会压位分块再套树状数组！

现在我们要求单次操作时间复杂度为 $o(\log n)$。

压位 Trie，就是每次改为 k 叉，这样插入删除可以在 $\Theta(\log_k n)$ 的时间内完成。

我会压位分块再套树状数组！

现在我们要求单次操作时间复杂度为 $o(\log n)$。

压位 Trie，就是每次改为 k 叉，这样插入删除可以在 $\Theta(\log_k n)$ 的时间内完成。

对于前驱后继，若 $k = O(\log n)$，则可以每次 $\Theta(1)$ 找到下一个或上一个儿子的位置。

我会压位分块再套树状数组！

现在我们要求单次操作时间复杂度为 $o(\log n)$。

压位 Trie，就是每次改为 k 叉，这样插入删除可以在 $\Theta(\log_k n)$ 的时间内完成。

对于前驱后继，若 $k = O(\log n)$，则可以每次 $\Theta(1)$ 找到下一个或上一个儿子的位置。

单次操作时间复杂度为 $\Theta(\log n / \log\log n)$ 或 $\Theta(\log n / \log \omega)$，空间复杂度为 $\Theta(n/\omega)$ 或 $\Theta((n \log n)/(\omega \log \omega))$。

我会压位分块再套树状数组！

现在我们要求单次操作时间复杂度为 $o(\log n)$。

压位 Trie，就是每次改为 k 叉，这样插入删除可以在 $\Theta(\log_k n)$ 的时间内完成。

对于前驱后继，若 $k = O(\log n)$，则可以每次 $\Theta(1)$ 找到下一个或上一个儿子的位置。

单次操作时间复杂度为 $\Theta(\log n/\log\log n)$ 或 $\Theta(\log n/\log\omega)$，空间复杂度为 $\Theta(n/\omega)$ 或 $\Theta((n\log n)/(\omega\log\omega))$。

压位 trie 和 vEB 树的代码可以看钱哥的国家集训队论文。

vEB 树是另一种亚 log 数据结构。考虑维护 $[0, 2^{2^U})$ 的权值，有 2^U 个 2^U 大小的子树，并用一个 2^U 大小的 summary structure 来维护这些子树。即对子树大小为 n 的子树有 $\sqrt{n}$ 叉。

vEB 树是另一种亚 log 数据结构。考虑维护 $[0, 2^{2^U})$ 的权值，有 2^U 个 2^U 大小的子树，并用一个 2^U 大小的 summary structure 来维护这些子树。即对子树大小为 n 的子树有 $\sqrt{n}$ 叉。

对于每个节点，额外维护最大值和最小值，且最小值不保存在子树中。于是在空树中插入和删除成空树复杂度为 $\Theta(1)$。

vEB 树是另一种亚 log 数据结构。考虑维护 $[0, 2^{2U})$ 的权值，有 2^U 个 2^U 大小的子树，并用一个 2^U 大小的 summary structure 来维护这些子树。即对子树大小为 n 的子树有 $\sqrt{n}$ 叉。

对于每个节点，额外维护最大值和最小值，且最小值不保存在子树中。于是在空树中插入和删除成空树复杂度为 $\Theta(1)$。

插入是如果子树中存在就只在子树中插入，否则在空子树和 summary structure 中插入。删除类似。

vEB 树是另一种亚 log 数据结构。考虑维护 $[0, 2^{2U})$ 的权值，有 2^U 个 2^U 大小的子树，并用一个 2^U 大小的 summary structure 来维护这些子树。即对子树大小为 n 的子树有 $\sqrt{n}$ 叉。

对于每个节点，额外维护最大值和最小值，且最小值不保存在子树中。于是在空树中插入和删除成空树复杂度为 $\Theta(1)$。

插入是如果子树中存在就只在子树中插入，否则在空子树和 summary structure 中插入。删除类似。

后继是时先看对应子树的最大值，如果大于这个最大值就是 summary structure 后继子树中的最小值，否则在子树中查询。前驱类似。

当前的权值范围足够小（例如 $\omega = 64$）时，可以直接用压位的方法计算。

当前的权值范围足够小（例如 $\omega = 64$）时，可以直接用压位的方法计算。

单次操作时间复杂度为 $T(n) = T(\sqrt{n}) + 1$，即 $T(n) = \Theta(\log \log n)$。

当前的权值范围足够小（例如 $\omega = 64$）时，可以直接用压位的方法计算。

单次操作时间复杂度为 $T(n) = T(\sqrt{n}) + 1$，即 $T(n) = \Theta(\log \log n)$。

实践中，对于不超过 2^{24}（或 10^7）的值域只需要递归两次，而且空间复杂度约为 $O(n/\omega)$。10^9 的值域也只需要约 200MB 内存，足够大多数情况下的使用。

这里给出一种序列分块的另类写法：我们建一棵线段树，再设置一个块大小 B，如果子树大小不超过 B，那么根据左儿子和右儿子的信息构建当前结点的信息。

这里给出一种序列分块的另类写法：我们建一棵线段树，再设置一个块大小 B，如果子树大小不超过 B，那么根据左儿子和右儿子的信息构建当前结点的信息。

若 n 个叶子的线段树当 $n \leqslant B$ 时结点信息修改时间复杂度 $\Theta(n)$，查询时间复杂度 $\Theta(1)$，否则无需进行修改，也无法进行直接查询，根据前面的线段树时间复杂度分析，区间修改时间复杂度 $\Theta(\log n + \min(B, n))$ 或 $\Theta(\log n + \min(B, n) + n/B)$（取决于能否维护懒标记），区间查询定位时间复杂度 $\Theta(\log n)$，查询结点时如果结点子树大小超过 B，需要递归求解，递归到不超过 B 的结点到定位结点的链并不超过 $\Theta(n/B)$，询问总时间复杂度 $\Theta(\log n + n/B)$。

事实上，这与直接分块再对每个块建立线段树并把每个块视作整体建立线段树是基本等效的，如果父节点子树大小 $\leqslant B$ 的非叶结点不维护子树信息且不维护子树信息的结点也不会维护一些标记时与一般的分块基本是等效的。与一般的分块相比，有时区间查询次数与区间修改次数不同阶或者问题有一定的特殊性可能可以有一定的优势。

分块时我们经常对 m 组长度为 n 的数组进行同一个数的二分，我们直接做的时间复杂度为 $\Theta(m \log n)$，但是理论上我们在 nm 个数中进行二分的复杂度可以做到 $\Theta(\log(nm)) = \Theta(\log n + \log m)$。

分块时我们经常对 m 组长度为 n 的数组进行同一个数的二分，我们直接做的时间复杂度为 $\Theta(m\log n)$，但是理论上我们在 nm 个数中进行二分的复杂度可以做到 $\Theta(\log(nm)) = \Theta(\log n + \log m)$。

我们在这里直接在线段树维护分块上进行叙述，虽然分散层叠本身可以以任何的二叉树形式进行合并。

我们对每个子树大小 $\leqslant B$ 的结点维护了一个索引，我们希望能快速的知道询问时给定的数在定位的每一个区间的索引中的位置，我们设置一个阈值 c，对于子树大小 $> B$ 的结点，我们分别每隔 c 个数选取一个左右儿子索引中的数，并将两者归并形成当前结点的索引，询问时我们只要知道了一个数在当前结点索引中的位置那么我们就可以在 $\Theta(c)$ 的时间复杂度内知道这个数在左右儿子结点索引中的位置。当子树大小 $\leqslant B$ 的结点的索引大小为子树大小时，若 $c \geqslant 2$ 则每个结点索引的大小 $\leqslant B$，若 $c \geqslant 3$ 则每个结点到根链的结点索引大小和 $= O(B)$。我们一般在带修问题中取 $c = 3$。

Problem (魔法少女网站·改 I)

给定长度为 n 的序列 a。m 次操作，每次操作为以下两种之一：

1. 区间加。
2. 给定 l, r, x，问区间 $[l, r]$ 有多少个子区间的最大值小于等于 x。

$1 \leqslant n, m \leqslant 3 \times 10^5$。

Solution

我们以 $B(B \leqslant n)$ 为大小进行分块，每个块维护块内元素从小到大排序的结果以及对每个元素 x 当前块值不超过 x 的最长前缀、最长后缀、与前两者不交的块内满足条件的子区间数，显然在块内元素大小顺序已知的情况下我们可以在关于块大小线性的时间复杂度构造一个块的信息。

Solution

对于修改操作，我们对定位的每一个结点打上懒标记，并对到根链并中的索引和块信息进行重构，时间复杂度 $\Theta(\log n + B)$。对于查询操作，我们直接根据定位的结点子树内的块信息直接求答案即可，时间复杂度 $\Theta(\log n + n/B)$。若修改操作次数为 q_1，查询操作次数为 q_2（满足 $q_1 + q_2 = m$），则总时间复杂度为 $\Theta(m \log n + \sqrt{q_1 q_2 n})$。

Solution

此外，对于这种更新结点信息较慢的情况，有一个技巧可以进行常数优化：我们需要更新信息时，只将信息标记为待更新状态而不直接进行更新，等到查询到待更新信息时再进行递归求值。注意这种技巧由于时间复杂度是均摊的，不能用于可持久化数据结构。