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

2024 年 2 月 11 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，根结点对应全集对应的区间，叶子结点对应单点对应的区间。一般 leafy tree 要求没有结点仅有一个儿子。

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，根结点对应全集对应的区间，叶子结点对应单点对应的区间。一般 leafy tree 要求没有结点仅有一个儿子。

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 $\frac{\mathrm{size}(v)}{\mathrm{size}(u)} = \Theta(1)$。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 为根结点或 $\frac{\mathrm{size}(a)}{\mathrm{size}(u)} \geqslant c_1$。

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，不区分一个结点及其对应区间。

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，不区分一个结点及其对应区间。

我们称一个区间在一棵线段树上定位的结点集为最小的对应区间不交并为给定区间的结点集，称一个非叶结点的区间分界点为其左儿子结点对应区间的右端点。

我们先对广义重量平衡的树进行考察：

我们先对广义重量平衡的树进行考察：

1. 一棵广义重量平衡的树的任意子树均为广义重量平衡的。

我们先对广义重量平衡的树进行考察：

1. 一棵广义重量平衡的树的任意子树均为广义重量平衡的。
2. 一棵 n 个结点的树的树高为 $O(\log n)$，叶子结点个数为 $\Theta(n)$。

我们先对广义重量平衡的树进行考察：

1. 一棵广义重量平衡的树的任意子树均为广义重量平衡的。
2. 一棵 n 个结点的树的树高为 $O(\log n)$，叶子结点个数为 $\Theta(n)$。
3. 我们考察 $g(T) = \sum_{u \in T} f(\text{size}_T(u))$，其中 $\text{size}_T(u)$ 表示在树 T 中结点 u 的子树大小：

我们先对广义重量平衡的树进行考察：

1. 一棵广义重量平衡的树的任意子树均为广义重量平衡的。
2. 一棵 n 个结点的树的树高为 $O(\log n)$，叶子结点个数为 $\Theta(n)$。
3. 我们考察 $g(T) = \sum_{u \in T} f(\text{size}_T(u))$，其中 $\text{size}_T(u)$ 表示在树 T 中结点 u 的子树大小：
   1. 若 $f(n) = O(n^c)$，其中 $c < 1$，则 $g(n) = O(n)$。

我们先对广义重量平衡的树进行考察：

1. 一棵广义重量平衡的树的任意子树均为广义重量平衡的。
2. 一棵 n 个结点的树的树高为 $O(\log n)$，叶子结点个数为 $\Theta(n)$。
3. 我们考察 $g(T)=\sum_{u\in T} f(\mathrm{size}_T(u))$，其中 $\mathrm{size}_T(u)$ 表示在树 T 中结点 u 的子树大小：
   1. 若 $f(n)=O(n^c)$，其中 $c<1$，则 $g(n)=O(n)$。
   2. 其余情况同样类似主定理，大家可以自行扩展，除 $f(n)=\Theta(n)$ 时 $g(n)=O(n\log n)$ 这一显然情况外应用较少。

基于线段树的定义（weight balanced binary leafy tree），我们可以额外证明以下性质（以下性质的证明无需用到广义重量平衡的性质）：

基于线段树的定义（weight balanced binary leafy tree），我们可以额外证明以下性质（以下性质的证明无需用到广义重量平衡的性质）：

1. 线段树的任意子树均为线段树。

基于线段树的定义（weight balanced binary leafy tree），我们可以额外证明以下性质（以下性质的证明无需用到广义重量平衡的性质）：

1. 线段树的任意子树均为线段树。
2. 线段树中一个前缀定位的结点集唯一存在：根结点或若干左儿子结点 $u_1, u_2, \ldots, u_k$，满足 u_1 在根结点左儿子的左链中，$u_{i+1}(i \in [1, k) \cap \mathbb{N})$ 在 u_i 的右兄弟的左链中（即 u_i 是 u_{i+1} 所在的极长左链中深度最小的结点的左兄弟）。同时满足条件的左儿子结点集均作为一个非根结点对应区间的前缀定位的结点集。后缀与前缀有左右对称的类似性质。

基于线段树的定义（weight balanced binary leafy tree），我们可以额外证明以下性质（以下性质的证明无需用到广义重量平衡的性质）：

1. 线段树的任意子树均为线段树。
2. 线段树中一个前缀定位的结点集唯一存在：根结点或若干左儿子结点 $u_1, u_2, \ldots, u_k$，满足 u_1 在根结点左儿子的左链中，$u_{i+1}(i \in [1, k) \cap \mathbb{N})$ 在 u_i 的右兄弟的左链中（即 u_i 是 u_{i+1} 所在的极长左链中深度最小的结点的左兄弟）。同时满足条件的左儿子结点集均作为一个非根结点对应区间的前缀定位的结点集。后缀与前缀有左右对称的类似性质。
3. 线段树中任意区间 $[l, r]$ 定位的结点集为 l 和 r 对应叶子结点的最近公共祖先或 $[l, m]$（设其区间分界点为 m）在其左子树内定位的结点集与 $(m, r]$ 在其右子树内定位的结点集的不交并。

基于线段树的定义（weight balanced binary leafy tree），我们可以额外证明以下性质（以下性质的证明无需用到广义重量平衡的性质）：

1. 线段树的任意子树均为线段树。
2. 线段树中一个前缀定位的结点集唯一存在：根结点或若干左儿子结点 $u_1, u_2, \ldots, u_k$，满足 u_1 在根结点左儿子的左链中，$u_{i+1}(i \in [1, k) \cap \mathbb{N})$ 在 u_i 的右兄弟的左链中（即 u_i 是 u_{i+1} 所在的极长左链中深度最小的结点的左兄弟）。同时满足条件的左儿子结点集均作为一个非根结点对应区间的前缀定位的结点集。后缀与前缀有左右对称的类似性质。
3. 线段树中任意区间 $[l, r]$ 定位的结点集为 l 和 r 对应叶子结点的最近公共祖先或 $[l, m]$（设其区间分界点为 m）在其左子树内定位的结点集与 $(m, r]$ 在其右子树内定位的结点集的不交并。
4. 线段树有 $n-1$ 个左儿子结点，其对应区间右端点集为 $[1, n) \cap \mathbb{N}$；有 $n-1$ 个右儿子结点，其对应区间左端点集为 $(1, n] \cap \mathbb{N}$。

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。

第二个性质可以推出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结点数不超过两倍的这些结点深度最大值。

第一个性质确保了我们可以递归地设计算法。

第二个性质可以推出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结点数不超过两倍的这些结点深度最大值。

第二个性质中的 u_k 的对应区间右端点为前缀右端点（不妨记为 r），容易说明对于 $r \in [1, n) \cap \mathbb{N}$，$u_k$ 为 r 与 $r+1$ 的对应叶结点最近公共祖先的左儿子，容易推出其深度不超过 r 与 $r+1$ 任意一个的对应叶结点，可以在带权线段树等一些变种中分析复杂度。

第一个性质确保了我们可以递归地设计算法。

第二个性质可以推出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结点数不超过两倍的这些结点深度最大值。

第二个性质中的 u_k 的对应区间右端点为前缀右端点（不妨记为 r），容易说明对于 $r \in [1, n) \cap \mathbb{N}$，$u_k$ 为 r 与 $r+1$ 的对应叶结点最近公共祖先的左儿子，容易推出其深度不超过 r 与 $r+1$ 任意一个的对应叶结点，可以在带权线段树等一些变种中分析复杂度。

第二个与第三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。而根据重量平衡的性质我们可以知道树高为 $\Theta(\log n)$。

第一个性质确保了我们可以递归地设计算法。

第二个性质可以推出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结点数不超过两倍的这些结点深度最大值。

第二个性质中的 u_k 的对应区间右端点为前缀右端点（不妨记为 r），容易说明对于 $r \in [1, n) \cap \mathbb{N}$，$u_k$ 为 r 与 $r+1$ 的对应叶结点最近公共祖先的左儿子，容易推出其深度不超过 r 与 $r+1$ 任意一个的对应叶结点，可以在带权线段树等一些变种中分析复杂度。

第二个与第三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。而根据重量平衡的性质我们可以知道树高为 $\Theta(\log n)$。

第四个性质可以用来给线段树的结点编号，如将根结点编号为 1，对于一个结点，设其区间分界点为 m，则左儿子结点编号为 $2m$，右儿子结点编号为 $2m+1$。

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：对于线段树不是单个叶结点的情况递归修改单点所在的子线段树，修改根结点的信息。时间复杂度为递归至单点所在深度加上对不超过该深度的每一深度的修改一个结点信息的时间之和，当每个结点可以 $\Theta(1)$ 修改信息且递归没有伴随额外操作时时间复杂度即 $\Theta(\log n)$。

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：对于线段树不是单个叶结点的情况递归修改单点所在的子线段树，修改根结点的信息。时间复杂度为递归至单点所在深度加上对不超过该深度的每一深度的修改一个结点信息的时间之和，当每个结点可以 $\Theta(1)$ 修改信息且递归没有伴随额外操作时时间复杂度即 $\Theta(\log n)$。
3. 单点查询：对于线段树是单个叶结点的情况直接返回结点维护的信息，否则递归查询单点所在的子线段树。时间复杂度为递归至单点所在深度加上查询一个叶结点信息的时间之和，当可以 $O(\log n)$ 查询叶子结点信息且递归没有伴随额外操作时时间复杂度即 $\Theta(\log n)$。

为了方便区间查询，我们通常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，我们通常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。

4 区间查询：我们可以先用 $\Theta(\log n)$ 的时间求出给定区间定位的结点集（判断给定区间是否包含根结点对应区间，如果否则递归左子树和右子树中对应区间与给定区间有交的一者或两者），再依次查询这些结点，时间复杂度为两者之和。对于一般给定区间，当我们可以 $\Theta(1)$ 查询一个结点且递归没有伴随额外操作时时间复杂度即 $\Theta(\log n)$。特别地，全局查询只需查询根结点即可，当我们可以 $\Theta(1)$ 查询一个结点时时间复杂度仅有 $\Theta(1)$。

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递归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递归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。

当可以 $\Theta(1)$ 下放懒标记时，递归过程中下放懒标记的时间复杂度不超过递归自身时间复杂度。

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递归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。

当可以 $\Theta(1)$ 下放懒标记时，递归过程中下放懒标记的时间复杂度不超过递归自身时间复杂度。

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递归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。

当可以 $\Theta(1)$ 下放懒标记时，递归过程中下放懒标记的时间复杂度不超过递归自身时间复杂度。

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。

在修改满足结合律时我们可以通过懒标记实现通过 1 次修改的复合修改一个结点的信息。

5 区间修改：我们可以先用 $\Theta(\log n)$ 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，再依次修改这些结点的信息，时间复杂度为两者之和。对于一般给定区间，当我们可以 $\Theta(1)$ 修改一个结点的信息且懒标记可以 $\Theta(1)$ 下放时时间复杂度即 $\Theta(\log n)$。特别地，全局修改只需修改根结点的信息，当我们可以 $\Theta(1)$ 修改一个结点的信息时时间复杂度仅有 $\Theta(1)$。

5 区间修改：我们可以先用 $\Theta(\log n)$ 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，再依次修改这些结点的信息，时间复杂度为两者之和。对于一般给定区间，当我们可以 $\Theta(1)$ 修改一个结点的信息且懒标记可以 $\Theta(1)$ 下放时时间复杂度即 $\Theta(\log n)$。特别地，全局修改只需修改根结点的信息，当我们可以 $\Theta(1)$ 修改一个结点的信息时时间复杂度仅有 $\Theta(1)$。

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、区间求最大值及其出现次数。

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、区间求最大值及其出现次数。

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。

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、区间求最大值及其出现次数。

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。

注意满足一些性质的问题通常用后面介绍的时空常数更小且代码难度更小的树状数组实现，如离线静态二维数点问题。

在我们无需下传信息时（注意标记永久化也无需下传），我们可以直接自底向上对线段树维护的信息进行上传。

在我们无需下传信息时（注意标记永久化也无需下传），我们可以直接自底向上对线段树维护的信息进行上传。

为了方便实现，我们可以将 n 补齐为 2 的幂，并采用根结点标号为 1，标号为 i 的结点左右儿子结点标号分别为 $2i$ 和 $2i+1$。容易说明，对应区间为 $[l,r]$ 的结点标号为 $(n+l-1)/(r-l+1)$，标号为 i 的结点的深度 $d=\lfloor \log_2 i \rfloor$（根结点深度为 1），对应区间为 $\left[in/2^d-n+1,(i+1)n/2^d\right]\cap\mathbb{N}$。尤其在无需维护对应区间左右端点的情况下可以更方便地根据当前结点编号求出相邻结点编号。

在我们无需下传信息时（注意标记永久化也无需下传），我们可以直接自底向上对线段树维护的信息进行上传。

为了方便实现，我们可以将 n 补齐为 2 的幂，并采用根结点标号为 1，标号为 i 的结点左右儿子结点标号分别为 $2i$ 和 $2i+1$。容易说明，对应区间为 $[l,r]$ 的结点标号为 $(n+l-1)/(r-l+1)$，标号为 i 的结点的深度 $d=\lfloor \log_2 i \rfloor$（根结点深度为 1），对应区间为 $\left[in/2^d-n+1,(i+1)n/2^d\right]\cap\mathbb{N}$。尤其在无需维护对应区间左右端点的情况下可以更方便地根据当前结点编号求出相邻结点编号。

我们可以使用暴力求最近公共祖先的方法定位结点集：一开始令 $u=n+l-2, v=n+r$，若 $v=u+1$ 则算法结束，若 $u \bmod 2=0$ 则 $u+1$ 在定位结点集中，若 $v \bmod 2=1$ 则 $v-1$ 在定位结点集中，再将 u,v 均除以二重复流程，可以说明定位结点集中的每个结点均能通过上述算法求出。注意到该实现方法可以方便地同时支持上传信息，类似的实现技巧有时被称为 zkw 线段树。

我们可以对线段树的最经典问题抽象化如下：

我们可以对线段树的最经典问题抽象化如下：

Problem (区间修改区间询问的线段树经典问题)

有两个集合 A, B，对于 $a \in A, b \in B$，可以在 $\Theta(1)$ 的复杂度内计算出 $b * a \in A$，B 上存在可以在 $\Theta(1)$ 复杂度内计算的二元运算 $\cdot$，满足 $\forall a \in A, b_1, b_2 \in B, b_2 * (b_1 * a) = (b_2 \cdot b_1) * a$，$A$ 上存在可以在 $\Theta(1)$ 复杂度计算的运算 $\oplus$，满足 $\forall a_1, a_2, a_3 \in A, (a_1 \oplus a_2) \oplus a_3 = a_1 \oplus (a_2 \oplus a_3)$ 且 $\forall a_1, a_2 \in A, b \in B, (b * a_1) \oplus (b * a_2) = b * (a_1 \oplus a_2)$。

给定一个长度为 n 的序列 a，满足 $a_i \in A$，q 次操作：

1. 给定 $[l, r] \subseteq [1, n], x \in B$，对 $i \in [l, r] \cap \mathbb{N}$ 执行 $a_i \leftarrow x * a_i$。
2. 给定 $[l, r] \subseteq [1, n], l, r \in \mathbb{N}$，询问 $a_l \oplus a_{l+1} \oplus \cdots \oplus a_r$。

强制在线，要求复杂度 $O(n + q \log n)$。

注意复杂度可以不仅限于时空，也可以用于统计矩阵乘法次数等。

注意复杂度可以不仅限于时空，也可以用于统计矩阵乘法次数等。

Solution

我们对每个结点维护子树和 $a \in A$ 与懒标记 $b \in B$ 即可，下面分析 $*, \cdot, \oplus$ 三种运算次数的常数（设 h 为树高，且只考虑 n 足够大的情况）：

注意复杂度可以不仅限于时空，也可以用于统计矩阵乘法次数等。

Solution

我们对每个结点维护子树和 $a \in A$ 与懒标记 $b \in B$ 即可，下面分析 $*, \cdot, \oplus$ 三种运算次数的常数（设 h 为树高，且只考虑 n 足够大的情况）：

我们先考虑定位结点集：我们会对定位的结点集的到根链并去掉自身的其余结点进行一次下传懒标记的操作，即进行至多 $2h-3$ 次下传懒标记操作。

注意复杂度可以不仅限于时空，也可以用于统计矩阵乘法次数等。

Solution

我们对每个结点维护子树和 $a \in A$ 与懒标记 $b \in B$ 即可，下面分析 $*, \cdot, \oplus$ 三种运算次数的常数（设 h 为树高，且只考虑 n 足够大的情况）：

我们先考虑定位结点集：我们会对定位的结点集的到根链并去掉自身的其余结点进行一次下传懒标记的操作，即进行至多 $2h-3$ 次下传懒标记操作。

对于构造线段树，我们会进行 $n-1$ 次上传信息操作。

注意复杂度可以不仅限于时空，也可以用于统计矩阵乘法次数等。

Solution

我们对每个结点维护子树和 $a \in A$ 与懒标记 $b \in B$ 即可，下面分析 $*, \cdot, \oplus$ 三种运算次数的常数（设 h 为树高，且只考虑 n 足够大的情况）：

我们先考虑定位结点集：我们会对定位的结点集的到根链并去掉自身的其余结点进行一次下传懒标记的操作，即进行至多 $2h-3$ 次下传懒标记操作。

对于构造线段树，我们会进行 $n-1$ 次上传信息操作。

对于区间修改，我们会先定位结点集，再对定位的结点集打上标记，最后对定位的结点集到根链去掉自身的其余结点进行上传信息操作，即至多 $2h-3$ 次下传懒标记与上传信息，至多 $2h-4$ 次打标记操作。

注意复杂度可以不仅限于时空，也可以用于统计矩阵乘法次数等。

Solution

我们对每个结点维护子树和 $a \in A$ 与懒标记 $b \in B$ 即可，下面分析 $*, \cdot, \oplus$ 三种运算次数的常数（设 h 为树高，且只考虑 n 足够大的情况）：

我们先考虑定位结点集：我们会对定位的结点集的到根链并去掉自身的其余结点进行一次下传懒标记的操作，即进行至多 $2h-3$ 次下传懒标记操作。

对于构造线段树，我们会进行 $n-1$ 次上传信息操作。

对于区间修改，我们会先定位结点集，再对定位的结点集打上标记，最后对定位的结点集到根链去掉自身的其余结点进行上传信息操作，即至多 $2h-3$ 次下传懒标记与上传信息，至多 $2h-4$ 次打标记操作。

对于区间查询，我们会先定位结点集，再对定位的结点集维护的信息求和，即至多 $2h-3$ 次下传懒标记与 $2h-5$ 次 $\oplus$ 运算。

Solution

上传信息操作即一次 $\oplus$ 运算，下放懒标记操作进行两次打标记操作并将当前结点懒标记重置为单位元，一次打标记操作即一次 $*$ 与一次 $\cdot$ 运算。

Solution

上传信息操作即一次 $\oplus$ 运算，下放懒标记操作进行两次打标记操作并将当前结点懒标记重置为单位元，一次打标记操作即一次 $*$ 与一次 $\cdot$ 运算。

注意到对于单位元懒标记我们无需执行下传，对于打标记而当前懒标记为单位元时我们无需进行 $\cdot$ 可以直接复制，若在同次操作时还将会被上传信息则无需进行 $*$，在一次操作内对一个结点连续打两次标记可以使用 2 次 $\cdot$ 与 1 次 $*$。我们考察 $\cdot$ 运算次数与非单位元懒标记数量之和，我们发现一次下放懒标记操作和一次打标记操作均至多加一。

Solution

上传信息操作即一次 $\oplus$ 运算，下放懒标记操作进行两次打标记操作并将当前结点懒标记重置为单位元，一次打标记操作即一次 $*$ 与一次 $\cdot$ 运算。

注意到对于单位元懒标记我们无需执行下传，对于打标记而当前懒标记为单位元时我们无需进行 $\cdot$ 可以直接复制，若在同次操作时还将会被上传信息则无需进行 $*$，在一次操作内对一个结点连续打两次标记可以使用 2 次 $\cdot$ 与 1 次 $*$。我们考察 $\cdot$ 运算次数与非单位元懒标记数量之和，我们发现一次下放懒标记操作和一次打标记操作均至多加一。

因此对于构造线段树，我们会进行 $n-1$ 次 $\oplus$；对于单次区间修改，我们会进行严格至多 $2h-3$ 次 $\oplus$，$2h-1$ 次 $*$ 与 $6h-10$ 次 $\cdot$，其中 $\cdot$ 均摊至多 $4h-7$ 次；对于单次区间查询，我们会进行严格至多 $2h-5$ 次 $\oplus$，$4h-6$ 次 $*$ 与 $\cdot$，其中 $\cdot$ 均摊至多 $2h-3$ 次。

上述运算次数分析无法取到所有上界，存在 $o(h)$ 的误差，区间查询中其中至多 $2h-3$ 次的 $*$ 也可用等量 $\oplus$ 替代。

上述运算次数分析无法取到所有上界，存在 $o(h)$ 的误差，区间查询中其中至多 $2h-3$ 次的 $*$ 也可用等量 $\oplus$ 替代。

在区间左乘矩阵、区间向量和等这三种运算较慢时运算次数的倍数变化可以在程序运行时间上体现出差距。

上述运算次数分析无法取到所有上界，存在 $o(h)$ 的误差，区间查询中其中至多 $2h-3$ 次的 $*$ 也可用等量 $\oplus$ 替代。

在区间左乘矩阵、区间向量和等这三种运算较慢时运算次数的倍数变化可以在程序运行时间上体现出差距。

注意到一般情况下懒标记与信息的运算相比条件转跳指令并不慢，部分上述优化 h 上常数的判断往往并不需要。

上述运算次数分析无法取到所有上界，存在 $o(h)$ 的误差，区间查询中其中至多 $2h-3$ 次的 $*$ 也可用等量 $\oplus$ 替代。

在区间左乘矩阵、区间向量和等这三种运算较慢时运算次数的倍数变化可以在程序运行时间上体现出差距。

注意到一般情况下懒标记与信息的运算相比条件转跳指令并不慢，部分上述优化 h 上常数的判断往往并不需要。

具体实现过程可以参考文件夹下代码（主要在官方还未使用 -O2 时实现，因此有一些在当下不必要的常数优化），平时 OI 无需对线段树进行封装，可以对线段树结点的上下传操作、打标记操作进行适当封装，注意用结构体或类存储结点信息可以比多个数组更好地利用空间连续性。注意严格满足前面抽象叙述条件的集合 A, B 中的元素往往需要携带下标、长度等线段树递归过程中可以直接使用的信息，会造成较大的常数开销，建议在具体题目中使用更贴合题目需求的实现。

Problem (单点修改求前缀和)

有一个群 G。给定 $g_1, g_2, \ldots, g_n \in G$，$q$ 次操作：

1. 单点修改：给定 $p \in [1, n] \cap \mathbb{N}, h \in G$，执行 $g_p \leftarrow h$。
2. 求前缀和：给定 $p \in [1, n] \cap \mathbb{N}$，求 $g_1 g_2 \cdots g_p$。

强制在线，q 与 n 同阶。

Problem (单点修改求前缀和)

有一个群 G。给定 $g_1, g_2, \ldots, g_n \in G$，$q$ 次操作：

1. 单点修改：给定 $p \in [1, n] \cap \mathbb{N}, h \in G$，执行 $g_p \leftarrow h$。
2. 求前缀和：给定 $p \in [1, n] \cap \mathbb{N}$，求 $g_1 g_2 \cdots g_p$。

强制在线，q 与 n 同阶。

根据 Mihai Pătraşcu 的结果，上述问题在 cell probe 模型下的时间复杂度为 $\Omega(n \log n \log |G|/w)$。由于严谨的证明需要一定的信息论内容，我们下面根据直觉不加证明地使用一些结论进行简略的证明。

Problem (单点修改求前缀和)

有一个群 G。给定 $g_1, g_2, \ldots, g_n \in G$，$q$ 次操作：

1. 单点修改：给定 $p \in [1, n] \cap \mathbb{N}, h \in G$，执行 $g_p \leftarrow h$。
2. 求前缀和：给定 $p \in [1, n] \cap \mathbb{N}$，求 $g_1 g_2 \cdots g_p$。

强制在线，q 与 n 同阶。

根据 Mihai Pătraşcu 的结果，上述问题在 cell probe 模型下的时间复杂度为 $\Omega(n \log n \log |G|/w)$。由于严谨的证明需要一定的信息论内容，我们下面根据直觉不加证明地使用一些结论进行简略的证明。

Proof

我们不妨证明 $q = 2n, n = 2^k, k \in \mathbb{N}$ 的情况，容易说明其余情况的结论均可以通过这种情况的结论得到。

Proof

我们考虑一类特殊构造的输入证明最坏时间复杂度的下界，我们可以用概率论说明一般情况下往往有常数略小的相似表现：

Proof

我们考虑一类特殊构造的输入证明最坏时间复杂度的下界，我们可以用概率论说明一般情况下往往有常数略小的相似表现：

初始每个元素为单位元，对于 $i \in [0, n) \cap \mathbb{N}$，我们在第 $2i+1$ 次操作对位置 $\pi(i)+1$ 进行一次单点修改，在第 $2i+2$ 次操作查询位置 $\pi(i)+1$ 的前缀和。$\pi(i)$ 为 i 在 k 位二进制表示下高低位反转后得到的结果，显然是一个 $[0, n) \cap \mathbb{N}$ 上的排列。

Proof

我们考虑一类特殊构造的输入证明最坏时间复杂度的下界，我们可以用概率论说明一般情况下往往有常数略小的相似表现：

初始每个元素为单位元，对于 $i \in [0, n) \cap \mathbb{N}$，我们在第 $2i+1$ 次操作对位置 $\pi(i)+1$ 进行一次单点修改，在第 $2i+2$ 次操作查询位置 $\pi(i)+1$ 的前缀和。$\pi(i)$ 为 i 在 k 位二进制表示下高低位反转后得到的结果，显然是一个 $[0, n) \cap \mathbb{N}$ 上的排列。

注意到强制在线，我们令第 i 个操作后的存储器的存储情况为状态 i，所有操作前为状态 0。

Proof

我们考虑一类特殊构造的输入证明最坏时间复杂度的下界，我们可以用概率论说明一般情况下往往有常数略小的相似表现：

初始每个元素为单位元，对于 $i\in[0,n)\cap\mathbb{N}$，我们在第 $2i+1$ 次操作对位置 $\pi(i)+1$ 进行一次单点修改，在第 $2i+2$ 次操作查询位置 $\pi(i)+1$ 的前缀和。$\pi(i)$ 为 i 在 k 位二进制表示下高低位反转后得到的结果，显然是一个 $[0,n)\cap\mathbb{N}$ 上的排列。

注意到强制在线，我们令第 i 个操作后的存储器的存储情况为状态 i，所有操作前为状态 0。

对于 $j\in[0,k]\cap\mathbb{N},a\in[0,2^{k-j})$，我们考察第 $a\cdot 2^{j+1}+b(0<b\leqslant 2^j)$ 次操作中的修改对第 $a\cdot 2^{j+1}+b(2^j<b\leqslant 2^{j+1})$ 次操作中的询问的影响，可以说明状态 $(2a+1)\cdot 2^j$ 存储了这些操作中修改的值按位置的前缀和的信息，且在 $(2a+1)\cdot 2^j+b(0<b\leqslant 2^j)$ 的操作中进行了访问。

Proof

由于群中元素关于群的运算具有消去律，我们可以用差分说明这些前缀和的信息量与这些修改操作的值的信息量是一样的，为 $\max\left(2^{j-1},1\right)\log|G|$ 位。

Proof

由于群中元素关于群的运算具有消去律，我们可以用差分说明这些前缀和的信息量与这些修改操作的值的信息量是一样的，为 $\max\left(2^{j-1},1\right)\log|G|$ 位。

注意到不同修改操作的值独立任意给定，同时可能存取的信息互不干扰，我们可以直接进行累加，得到至少存取 $(k+2)n\log|G|/2$ 位信息。

Proof

由于群中元素关于群的运算具有消去律，我们可以用差分说明这些前缀和的信息量与这些修改操作的值的信息量是一样的，为 $\max\left(2^{j-1},1\right)\log|G|$ 位。

注意到不同修改操作的值独立任意给定，同时可能存取的信息互不干扰，我们可以直接进行累加，得到至少存取 $(k+2)n\log|G|/2$ 位信息。

因此在 cell probe 模型下时间复杂度为 $\Omega(n\log n\log|G|/w)$，且相邻 q' 次操作的总时间复杂度为 $\Omega(q'\log q'\log|G|/w)$。

Proof

由于群中元素关于群的运算具有消去律，我们可以用差分说明这些前缀和的信息量与这些修改操作的值的信息量是一样的，为 $\max\left(2^{j-1},1\right)\log|G|$ 位。

注意到不同修改操作的值独立任意给定，同时可能存取的信息互不干扰，我们可以直接进行累加，得到至少存取 $(k+2)n\log|G|/2$ 位信息。

因此在 cell probe 模型下时间复杂度为 $\Omega(n\log n\log|G|/w)$，且相邻 q' 次操作的总时间复杂度为 $\Omega(q'\log q'\log|G|/w)$。

上述证明也可以看作对操作建立完全二叉树，考虑左儿子需要给右儿子传递的信息量。

对于 $q_1 = o(q_2)$ 的情况，我们可以将二叉改为 $\Theta(1 + q_2/q_1)$ 叉，考虑前面叉的修改给每一叉的询问传递的信息量，可以说明 cell probe 模型下复杂度为 $\Omega\left(q_2 \log n / \log(1 + q_2/q_1) \log |G|/w\right)$。

对于 $q_1 = o(q_2)$ 的情况，我们可以将二叉改为 $\Theta(1 + q_2/q_1)$ 叉，考虑前面叉的修改给每一叉的询问传递的信息量，可以说明 cell probe 模型下复杂度为 $\Omega\left(q_2 \log n / \log(1 + q_2/q_1) \log |G|/w\right)$。

对于 $q_2 = o(q_1)$ 的情况，我们类似地将二叉改为 $\Theta(1 + q_1/q_2)$ 叉，考虑每一叉的修改给后面叉的询问传递的信息量，可以说明 cell probe 模型下复杂度为 $\Omega\left(q_1 \log n / \log(1 + q_1/q_2) \log |G|/w\right)$。

对于 $q_1 = o(q_2)$ 的情况，我们可以将二叉改为 $\Theta(1 + q_2/q_1)$ 叉，考虑前面叉的修改给每一叉的询问传递的信息量，可以说明 cell probe 模型下复杂度为 $\Omega\left(q_2 \log n / \log(1 + q_2/q_1) \log |G|/w\right)$。

对于 $q_2 = o(q_1)$ 的情况，我们类似地将二叉改为 $\Theta(1 + q_1/q_2)$ 叉，考虑每一叉的修改给后面叉的询问传递的信息量，可以说明 cell probe 模型下复杂度为 $\Omega\left(q_1 \log n / \log(1 + q_1/q_2) \log |G|/w\right)$。

我们可以把线段树同样改成多叉线段树进行时间复杂度优化（该做法在 $q_2 = o(q_1)$ 的情况达到该下界可能需交换律），使用较少，我们直接在多叉树状数组处介绍。

在重链剖分等一些应用场景中，将叶结点的深度按照一定参数调整深度大小，使线段树的复杂度根据操作区间的一些性质进行平衡，可能有助于改善时间复杂度。

在重链剖分等一些应用场景中，将叶结点的深度按照一定参数调整深度大小，使线段树的复杂度根据操作区间的一些性质进行平衡，可能有助于改善时间复杂度。

具体地，设所有叶子结点的集合为 S，结点 u 子树内叶子结点的集合为 $S(u)$，$\forall u \in S, w_u \in \mathbb{R}_+$，则我们可以使结点 u 的深度为 $O\left(1+\log\left(\sum_{i\in S} w_i \Big/ \sum_{i\in S(u)} w_i\right)\right)$。

在重链剖分等一些应用场景中，将叶结点的深度按照一定参数调整深度大小，使线段树的复杂度根据操作区间的一些性质进行平衡，可能有助于改善时间复杂度。

具体地，设所有叶子结点的集合为 S，结点 u 子树内叶子结点的集合为 $S(u)$，$\forall u \in S, w_u \in \mathbb{R}_+$，则我们可以使结点 u 的深度为 $O\left(1+\log\left(\sum_{i\in S} w_i \Big/ \sum_{i\in S(u)} w_i\right)\right)$。

我们甚至可以保证一个更强的性质：带权广义重量平衡，即将广义重量平衡的条件中的子树大小改为子树的带权大小即权值和。

在重链剖分等一些应用场景中，将叶结点的深度按照一定参数调整深度大小，使线段树的复杂度根据操作区间的一些性质进行平衡，可能有助于改善时间复杂度。

具体地，设所有叶子结点的集合为 S，结点 u 子树内叶子结点的集合为 $S(u)$，$\forall u \in S, w_u \in \mathbb{R}_+$，则我们可以使结点 u 的深度为 $O\left(1+\log\left(\sum_{i\in S} w_i \Big/ \sum_{i\in S(u)} w_i\right)\right)$。

我们甚至可以保证一个更强的性质：带权广义重量平衡，即将广义重量平衡的条件中的子树大小改为子树的带权大小即权值和。

为了便于叙述，我们依然假设叶子结点编号为 $[1,n]\cap\mathbb{N}$ 且按自然数序排列。

在重链剖分等一些应用场景中，将叶结点的深度按照一定参数调整深度大小，使线段树的复杂度根据操作区间的一些性质进行平衡，可能有助于改善时间复杂度。

具体地，设所有叶子结点的集合为 S，结点 u 子树内叶子结点的集合为 $S(u)$，$\forall u \in S, w_u \in \mathbb{R}_+$，则我们可以使结点 u 的深度为 $O\left(1+\log\left(\sum_{i\in S} w_i \Big/ \sum_{i\in S(u)} w_i\right)\right)$。

我们甚至可以保证一个更强的性质：带权广义重量平衡，即将广义重量平衡的条件中的子树大小改为子树的带权大小即权值和。

为了便于叙述，我们依然假设叶子结点编号为 $[1,n]\cap\mathbb{N}$ 且按自然数序排列。

在实现时，我们可以在建树时先求出每个结点的区间分界点并存储在这个结点上。注意到区间分界点互不相同的性质依然成立，我们依然可以直接根据区间分界点得到左右儿子编号，其余也与一般线段树一致即可。

我们考虑一种简单的求区间分界点的方法：对应区间 $[l,r]\cap\mathbb{N}$ 的结点的区间分界点为

$\min\left(\{r-1\}\cup\left\{i\in[l,r)\cap\mathbb{N}\middle|\sum_{j=l}^{i}w_j\geqslant\sum_{j=l}^{r}w_j/2\right\}\right)$。

我们考虑一种简单的求区间分界点的方法：对应区间 $[l,r]\cap\mathbb{N}$ 的结点的区间分界点为

$\min\left(\{r-1\}\cup\left\{i\in[l,r)\cap\mathbb{N}\middle|\sum_{j=l}^{i}w_j\geqslant\sum_{j=l}^{r}w_j/2\right\}\right)$。

我们可以容易地通过分类讨论说明一个结点 u 的距离为 2 的后代 v 若不是叶结点，则其对应区间权值和小于 u 的一半。换言之，对于任意一个结点 v，若 v 的三级祖先 u 存在则 u 的对应区间权值和大于 v 的两倍，即带权广义重量平衡成立。

动态 dp，即动态修改 dp 中需要的参数，查询某一状态的 dp 结果。

动态 dp，即动态修改 dp 中需要的参数，查询某一状态的 dp 结果。

注意到函数的复合自然满足结合律，如果复合内容的信息量有限，我们可以方便地使用数据结构维护转移的信息。

动态 dp，即动态修改 dp 中需要的参数，查询某一状态的 dp 结果。

注意到函数的复合自然满足结合律，如果复合内容的信息量有限，我们可以方便地使用数据结构维护转移的信息。

由于我们在讨论线段树，这里仅对 dp 状态呈链状结构的情况进行介绍。

Problem

有状态集合 A 与转移集合 B，存在一个 $B \times A \to A$ 的可以 $\Theta(1)$ 复杂度计算的二元运算 $*$，另要求 B 上存在 $\Theta(1)$ 复杂度计算的二元运算 $\cdot$，满足 $\forall b_1, b_2 \in B, a \in A, (b_2 \cdot b_1) * a = b_2 * (b_1 * a)$。

给定初始状态 f_0 与 n 个其它状态 $f_1, f_2, \ldots, f_n$，满足 $\forall i \in [0, n] \cap \mathbb{N}$, $f_i \in A$。给定转移 $b_1, \ldots, b_n$，满足 $\forall i \in [1, n] \cap \mathbb{N}, b_i \in B, a_i = b_i * a_{i-1}$。$q$ 次操作：

1. 给定 $[l, r] \subseteq [1, n], x \in B$，对于 $i \in [l, r] \cap \mathbb{N}$，执行修改 $b_i \leftarrow x$。
2. 给定初始状态 $x \in A$，执行 $f_0 \leftarrow x$。
3. 给定位置 $p \in [1, n] \cap \mathbb{N}$，查询 f_p。

要求复杂度 $O(n + q \log n)$。

对于每个转移 $b \in B$ 对应了一个状态集合 A 到自身的映射 $\forall a \in A, g_b(a) = b * a$，$\cdot$ 即 $g_{b_2 \cdot b_1} = g_{b_2} \circ g_{b_1}$，其中 $\circ$ 为函数复合，结合律是自然的，其直观意义为通过从 a 到 $b_1 * a$ 的转移 b_1 与从 $b_1 * a$ 到 $b_2 * (b_1 * a)$ 的转移 b_2 得到从 a 到 $b_2 * (b_1 * a)$ 的转移 $b_2 \cdot b_1$。

对于每个转移 $b \in B$ 对应了一个状态集合 A 到自身的映射 $\forall a \in A, g_b(a) = b * a$，$\cdot$ 即 $g_{b_2 \cdot b_1} = g_{b_2} \circ g_{b_1}$，其中 $\circ$ 为函数复合，结合律是自然的，其直观意义为通过从 a 到 $b_1 * a$ 的转移 b_1 与从 $b_1 * a$ 到 $b_2 * (b_1 * a)$ 的转移 b_2 得到从 a 到 $b_2 * (b_1 * a)$ 的转移 $b_2 \cdot b_1$。

我们考虑对于对应区间为 $[l, r]$ 的线段树结点维护 $b_r \cdot (b_{r-1} \cdot (\cdots b_l \cdots))$，区间赋值可以方便地通过懒标记解决，对可能的结点对应区间长度 l（$O(\log n)$ 个）预处理 x^l，查询求出前缀定位的结点集，依次做转移即可。为了方便我们可以将 n 补全为 2 的幂，使每个结点的对应区间长度为 2 的幂。

对于每个转移 $b \in B$ 对应了一个状态集合 A 到自身的映射 $\forall a \in A, g_b(a) = b * a$，$\cdot$ 即 $g_{b_2 \cdot b_1} = g_{b_2} \circ g_{b_1}$，其中 $\circ$ 为函数复合，结合律是自然的，其直观意义为通过从 a 到 $b_1 * a$ 的转移 b_1 与从 $b_1 * a$ 到 $b_2 * (b_1 * a)$ 的转移 b_2 得到从 a 到 $b_2 * (b_1 * a)$ 的转移 $b_2 \cdot b_1$。

我们考虑对于对应区间为 $[l, r]$ 的线段树结点维护 $b_r \cdot (b_{r-1} \cdot (\cdots b_l \cdots))$，区间赋值可以方便地通过懒标记解决，对可能的结点对应区间长度 l（$O(\log n)$ 个）预处理 x^l，查询求出前缀定位的结点集，依次做转移即可。为了方便我们可以将 n 补全为 2 的幂，使每个结点的对应区间长度为 2 的幂。

对于单次区间赋值，严格至多 $\lfloor \log_2(r-l+1) \rfloor + 2\lceil \log_2 n \rceil - 1$ 次 $\cdot$ 运算（对于单点修改，可以减少 $\lceil \log_2 n \rceil - 1$ 次）。对于单次查询，严格至多 $\lceil \log_2 n \rceil$ 次 $*$ 运算。

Problem (区间最大子段和)

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$，q 次操作：

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 $\max\limits_{x \leqslant l \leqslant r \leqslant y} \sum\limits_{i=l}^{r} a_i$。

q 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 $a_1, a_2, \ldots, a_n$，q 次操作：

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 $\max\limits_{x \leqslant l \leqslant r \leqslant y} \sum\limits_{i=l}^{r} a_i$。

q 和 n 同阶，要求时间复杂度 $O(n \log n)$。

Solution

作为一个经典做法，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 (区间最大子段和)

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$，q 次操作：

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 $\max\limits_{x \leqslant l \leqslant r \leqslant y} \sum\limits_{i=l}^{r} a_i$。

q 和 n 同阶，要求时间复杂度 $O(n \log n)$。

Solution

作为一个经典做法，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。

时间复杂度 $\Theta(n + q \log n)$。

Solution

我们考虑全局最大子段和的经典做法：从小到大枚举 $i \in [1, n] \cap \mathbb{N}$，维护以 i 结束的前缀的最大子段（不可为空）和 b_i 与最大后缀（可为空）和 c_i，特别地，$b_0 = -\infty, c_0 = 0$。我们有转移 $b_i = \max(b_{i-1}, c_{i-1} + a_i), c_i = \max(c_{i-1} + a_i, 0)$。

Solution

我们考虑全局最大子段和的经典做法：从小到大枚举 $i \in [1, n] \cap \mathbb{N}$，维护以 i 结束的前缀的最大子段（不可为空）和 b_i 与最大后缀（可为空）和 c_i，特别地，$b_0 = -\infty, c_0 = 0$。我们有转移 $b_i = \max(b_{i-1}, c_{i-1} + a_i), c_i = \max(c_{i-1} + a_i, 0)$。

我们考虑动态维护这个 dp，虽然我们可以用在转移中加上一种重置状态的转移强行规约到之前的形式，但注意到维护的是区间转移的复合，我们只需将转移的查询从前缀改成任意区间即可。

Solution

我们令状态为二元序偶 (b, c)，注意到转移只用一个数 a 在复合意义下是不封闭的：$b_1 = \max(b_0, c_0 + a_1), c_1 = \max(c_0 + a_1, 0), b_2 = \max(b_1, c_1 + a_2), c_2 = \max(c_1 + a_2, 0)$ 则
$b_2 = \max(b_0, c_0 + a_1, c_0 + a_1 + a_2, a_2), c_2 = \max(c_0 + a_1 + a_2, a_2, 0)$ 不能同样用一个数 a 表示，但我们通过人工迭代得出了一个对封闭复合的包含需要的转移的集合（即需要的转移集合在复合运算下生成的幺半群）：(d_0, d_1, d_2, d_3)，表示 $b' = \max(b, c + d_0, d_1), c' = \max(c + d_2, d_3)$。

Solution

我们令状态为二元序偶 (b, c)，注意到转移只用一个数 a 在复合意义下是不封闭的：$b_1 = \max(b_0, c_0 + a_1), c_1 = \max(c_0 + a_1, 0), b_2 = \max(b_1, c_1 + a_2), c_2 = \max(c_1 + a_2, 0)$ 则 $b_2 = \max(b_0, c_0 + a_1, c_0 + a_1 + a_2, a_2), c_2 = \max(c_0 + a_1 + a_2, a_2, 0)$ 不能同样用一个数 a 表示，但我们通过人工迭代得出了一个对封闭复合的包含需要的转移的集合（即需要的转移集合在复合运算下生成的幺半群）：(d_0, d_1, d_2, d_3)，表示 $b' = \max(b, c + d_0, d_1), c' = \max(c + d_2, d_3)$。

因此我们可以使用动态 dp，时间复杂度 $\Theta(n + q \log n)$。

Solution

我们令状态为二元序偶 (b, c)，注意到转移只用一个数 a 在复合意义下是不封闭的：$b_1 = \max(b_0, c_0 + a_1), c_1 = \max(c_0 + a_1, 0), b_2 = \max(b_1, c_1 + a_2), c_2 = \max(c_1 + a_2, 0)$ 则 $b_2 = \max(b_0, c_0 + a_1, c_0 + a_1 + a_2, a_2), c_2 = \max(c_0 + a_1 + a_2, a_2, 0)$ 不能同样用一个数 a 表示，但我们通过人工迭代得出了一个对封闭复合的包含需要的转移的集合（即需要的转移集合在复合运算下生成的幺半群）：(d_0, d_1, d_2, d_3)，表示 $b' = \max(b, c + d_0, d_1), c' = \max(c + d_2, d_3)$。

因此我们可以使用动态 dp，时间复杂度 $\Theta(n + q \log n)$。

观察 d_0, d_1, d_2, d_3 的直观意义，可以发现这与前面的经典做法本质相同。

Solution

注意到人工迭代费时费力，这里介绍一种常用的写法：我们将状态加上幺元写作向量，转移写作矩阵。

Solution

注意到人工迭代费时费力，这里介绍一种常用的写法：我们将状态加上幺元写作向量，转移写作矩阵。

例如这题使用 $(\max,+)$ 的矩阵，状态对应的向量为 $(b,c,0)^{\mathrm{T}}$，转移对应的矩阵为 $\begin{pmatrix} 0 & a & -\infty \\ -\infty & a & 0 \\ -\infty & -\infty & 0 \end{pmatrix}$，注意到这是一个上三角矩阵且主对角线上首尾两项恒为 0，我们只需要维护其余四项的值，与前面的做法本质相同。

Problem (区间加区间正弦值和)

给定一个长度为 n 的数组，q 次操作：区间加、求区间正弦值之和。q 与 n 同阶，要求时间复杂度 $O(n \log n)$。

Problem (区间加区间正弦值和)

给定一个长度为 n 的数组，q 次操作：区间加、求区间正弦值之和。q 与 n 同阶，要求时间复杂度 $O(n \log n)$。

Solution

可以注意到 $e^{ix} = \cos x + i \sin x$，转化为区间乘，求区间和。

Problem (区间加区间正弦值和)

给定一个长度为 n 的数组，q 次操作：区间加、求区间正弦值之和。q 与 n 同阶，要求时间复杂度 $O(n\log n)$。

Solution

可以注意到 $e^{ix}=\cos x+i\sin x$，转化为区间乘，求区间和。

对每个结点维护对应区间正弦值和、余弦值和并利用和角公式修改结点信息也是等效的。

Problem (区间加区间正弦值和)

给定一个长度为 n 的数组，q 次操作：区间加、求区间正弦值之和。q 与 n 同阶，要求时间复杂度 $O(n\log n)$。

Solution

可以注意到 $e^{ix}=\cos x+i\sin x$，转化为区间乘，求区间和。

对每个结点维护对应区间正弦值和、余弦值和并利用和角公式修改结点信息也是等效的。

时间复杂度 $\Theta(n+q\log n)$。

Problem (区间位运算)

给定 n 个位运算 $f_1(x), \ldots, f_n(x)$，其中 $f_i(x) = a_i *_i x$。q 次操作：

1. 给定 $p \in [1, n] \cap \mathbb{N}, \cdot, b$，执行 $f_i(x) \leftarrow b \cdot x$。
2. 给定 $[l, r] \subseteq [1, n], l, r \in \mathbb{N}, x$，求 $f_r(f_{r-1}(\cdots f_l(x) \cdots))$。

$1 \leqslant n \leqslant 10^6, 0 \leqslant q \leqslant 10^6, 0 \leqslant a_i, b \leqslant 10^9, *_i, \cdot \in \{\text{and}, \text{or}, \text{xor}\}$。

Problem (区间位运算)

给定 n 个位运算 $f_1(x), \ldots, f_n(x)$，其中 $f_i(x) = a_i *_i x$。q 次操作：

1. 给定 $p \in [1, n] \cap \mathbb{N}, \cdot, b$，执行 $f_i(x) \leftarrow b \cdot x$。
2. 给定 $[l, r] \subseteq [1, n], l, r \in \mathbb{N}, x$，求 $f_r(f_{r-1}(\cdots f_l(x) \cdots))$。

$1 \leqslant n \leqslant 10^6, 0 \leqslant q \leqslant 10^6, 0 \leqslant a_i, b \leqslant 10^9, *_i, \cdot \in \{\text{and}, \text{or}, \text{xor}\}$。

Solution

对于每一位，每个位运算都可以看做是 $\text{mod}2$ 意义下的一次函数。

Problem (区间位运算)

给定 n 个位运算 $f_1(x), \ldots, f_n(x)$，其中 $f_i(x) = a_i *_i x$。q 次操作：

1. 给定 $p \in [1, n] \cap \mathbb{N}, \cdot, b$，执行 $f_i(x) \leftarrow b \cdot x$。
2. 给定 $[l, r] \subseteq [1, n], l, r \in \mathbb{N}, x$，求 $f_r(f_{r-1}(\cdots f_l(x) \cdots))$。

$1 \leqslant n \leqslant 10^6, 0 \leqslant q \leqslant 10^6, 0 \leqslant a_i, b \leqslant 10^9, *_i, \cdot \in \{\text{and}, \text{or}, \text{xor}\}$。

Solution

对于每一位，每个位运算都可以看做是 mod2 意义下的一次函数。

于是问题相当于区间一次函数复合，直接用线段树进行维护，利用压位同时处理所有位，通过位运算上传信息。

Problem (区间位运算)

给定 n 个位运算 $f_1(x), \ldots, f_n(x)$，其中 $f_i(x) = a_i *_i x$。q 次操作：

1. 给定 $p \in [1, n] \cap \mathbb{N}, \cdot, b$，执行 $f_i(x) \leftarrow b \cdot x$。
2. 给定 $[l, r] \subseteq [1, n], l, r \in \mathbb{N}, x$，求 $f_r(f_{r-1}(\cdots f_l(x) \cdots))$。

$1 \leqslant n \leqslant 10^6, 0 \leqslant q \leqslant 10^6, 0 \leqslant a_i, b \leqslant 10^9, *_i, \cdot \in \{\text{and}, \text{or}, \text{xor}\}$。

Solution

对于每一位，每个位运算都可以看做是 mod2 意义下的一次函数。

于是问题相当于区间一次函数复合，直接用线段树进行维护，利用压位同时处理所有位，通过位运算上传信息。

时间复杂度 $\Theta(n + q \log n)$。

Problem (【清华集训 2015】V)

给定一个长度为 n 的数组 a，q 次操作：区间加、区间减并对 0 取最大值、区间赋值、单点求值、单点求历史最大值。

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

Problem (【清华集训 2015】V)

给定一个长度为 n 的数组 a，q 次操作：区间加、区间减并对 0 取最大值、区间赋值、单点求值、单点求历史最大值。

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

Solution

我们维护懒标记：值变为当前值加上某个数与某个数的最大值，历史最大值与当前值加上某个数与某个数的最大值取最大值。显然懒标记是便于复合的。

Problem (【清华集训 2015】V)

给定一个长度为 n 的数组 a，q 次操作：区间加、区间减并对 0 取最大值、区间赋值、单点求值、单点求历史最大值。

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

Solution

我们维护懒标记：值变为当前值加上某个数与某个数的最大值，历史最大值与当前值加上某个数与某个数的最大值取最大值。显然懒标记是便于复合的。

时间复杂度 $\Theta(n + q \log n)$。

Problem（【清华集训 2015】V）

给定一个长度为 n 的数组 a，q 次操作：区间加、区间减并对 0 取最大值、区间赋值、单点求值、单点求历史最大值。

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

Solution

我们维护懒标记：值变为当前值加上某个数与某个数的最大值，历史最大值与当前值加上某个数与某个数的最大值取最大值。显然懒标记是便于复合的。

时间复杂度 $\Theta(n + q \log n)$。

更复杂的历史最值操作会在 Segment Tree Beats 处介绍。

Problem (【SHOI2008】堵塞的交通)

给定一张 $2 \times n$ 的网格图的子图，q 次操作：修改网格图上相邻两点在维护的子图中是否有边、询问两点间连通性。

$1 \leqslant n, q \leqslant 10^5$。

Problem (【SHOI2008】堵塞的交通)

给定一张 $2 \times n$ 的网格图的子图，q 次操作：修改网格图上相邻两点在维护的子图中是否有边、询问两点间连通性。

$1 \leqslant n, q \leqslant 10^5$。

Solution

注意到两个点之间的每一列点都是它们的一组割点，用线段树维护只保留区间内边时区间两侧的 4 个点的连通性。

Problem (【SHOI2008】堵塞的交通)

给定一张 $2 \times n$ 的网格图的子图，q 次操作：修改网格图上相邻两点在维护的子图中是否有边、询问两点间连通性。

$1 \leqslant n, q \leqslant 10^5$。

Solution

注意到两个点之间的每一列点都是它们的一组割点，用线段树维护只保留区间内边时区间两侧的 4 个点的连通性。

询问时注意可以通过两边绕回来，对前后缀也进行询问。

Problem (【SHOI2008】堵塞的交通)

给定一张 $2 \times n$ 的网格图的子图，q 次操作：修改网格图上相邻两点在维护的子图中是否有边、询问两点间连通性。

$1 \leqslant n, q \leqslant 10^5$。

Solution

注意到两个点之间的每一列点都是它们的一组割点，用线段树维护只保留区间内边时区间两侧的 4 个点的连通性。

询问时注意可以通过两边绕回来，对前后缀也进行询问。

时间复杂度 $\Theta(n + q \log n)$。

Problem (SOJ563 斐波那契数列)

给定一个长度为 n 的数列 a，初始均为 0。q 次操作：

1. 给定 $[l,r] \subseteq [1,n]$，对于 $i \in [l,r] \cap \mathbb{N}$，执行 $a_i \leftarrow a_i + f_{i-l+1}$。其中 $f_0 = 0, f_1 = 1, f_i = f_{i-1} + f_{i-2} (i \geqslant 2, i \in \mathbb{N})$。
2. 给定 $[l,r] \subseteq [1,n]$，求 $\sum_{i \in [l,r] \cap \mathbb{N}} a_i^2$。

$0 \leqslant n, q \leqslant 2 \times 10^5$。

Problem (SOJ563 斐波那契数列)

给定一个长度为 n 的数列 a，初始均为 0。q 次操作：

1. 给定 $[l,r] \subseteq [1,n]$，对于 $i \in [l,r] \cap \mathbb{N}$，执行 $a_i \leftarrow a_i + f_{i-l+1}$。其中 $f_0 = 0, f_1 = 1, f_i = f_{i-1} + f_{i-2} (i \geqslant 2, i \in \mathbb{N})$。
2. 给定 $[l,r] \subseteq [1,n]$，求 $\sum_{i \in [l,r] \cap \mathbb{N}} a_i^2$。

$0 \leqslant n, q \leqslant 2 \times 10^5$。

Solution

注意到斐波那契数列是一个常系数线性递推数列。

Problem (SOJ563 斐波那契数列)

给定一个长度为 n 的数列 a，初始均为 0。q 次操作：

1. 给定 $[l,r] \subseteq [1,n]$，对于 $i \in [l,r] \cap \mathbb{N}$，执行 $a_i \leftarrow a_i + f_{i-l+1}$。其中 $f_0 = 0, f_1 = 1, f_i = f_{i-1} + f_{i-2} (i \geqslant 2, i \in \mathbb{N})$。
2. 给定 $[l,r] \subseteq [1,n]$，求 $\sum_{i \in [l,r] \cap \mathbb{N}} a_i^2$。

$0 \leqslant n, q \leqslant 2 \times 10^5$。

Solution

注意到斐波那契数列是一个常系数线性递推数列。

我们可以把操作写成将 $i \in [l,r] \cap \mathbb{N}, a_i \leftarrow a_i + \overrightarrow{t_2} A^{i-l} \overrightarrow{v_a}$，其中 $\overrightarrow{t_2}$ 是固定的行向量，A 是固定的方阵，$\overrightarrow{v_a}$ 是列向量。每个结点先递归左儿子，在修改一个结点信息后将 $\overrightarrow{v_a}$ 左乘上 A^l 即可，其中 l 为对应区间长度。

Solution

令 $b_i = a_i^2$，我们考虑设计懒标记与维护的信息使得我们可以在 $\Theta(1)$ 的时间复杂度内修改或查询一个结点的信息。

Solution

令 $b_i = a_i^2$，我们考虑设计懒标记与维护的信息使得我们可以在 $\Theta(1)$ 的时间复杂度内修改或查询一个结点的信息。

注意到 $b'_i = a'^2_i = \left(a_i + \overrightarrow{t_2} A^{i-l} \overrightarrow{v_a}\right)^2 = b_i + \left(\overrightarrow{t_2} A^{i-l} \overrightarrow{v_a}\right)^2 + 2a_i \overrightarrow{t_2} A^{i-l} \overrightarrow{v_a}$，我们可以对每个结点维护 b_i 的区间和与 $\sum_{i=l}^{r} a_i \overrightarrow{t_2} A^{i-l}$，即可 $\Theta(1)$ 查询一个结点的信息。

Solution

令 $b_i = a_i^2$，我们考虑设计懒标记与维护的信息使得我们可以在 $\Theta(1)$ 的时间复杂度内修改或查询一个结点的信息。

注意到 $b'_i = a'^2_i = \left(a_i + \overrightarrow{t_2} A^{i-l} \overrightarrow{v_a}\right)^2 = b_i + \left(\overrightarrow{t_2} A^{i-l} \overrightarrow{v_a}\right)^2 + 2a_i \overrightarrow{t_2} A^{i-l} \overrightarrow{v_a}$，我们可以对每个结点维护 b_i 的区间和与 $\sum_{i=l}^{r} a_i \overrightarrow{t_2} A^{i-l}$，即可 $\Theta(1)$ 查询一个结点的信息。

由矩阵乘法的分配律，我们在修改一个结点的信息时会使后者增加 $\sum_{i=0}^{r-l} \overrightarrow{t_2} A^i \overrightarrow{v_a} \overrightarrow{t_2} A^i$，注意到这个向量在 $r-l+1$ 一定时可以表示为一个常系数矩阵乘 $\overrightarrow{v_a}$，我们可以进行预处理。同样地，我们可以预处理 $\sum_{i=0}^{r-l} \left(\overrightarrow{t_2} A^i \overrightarrow{v_a}\right)^2$ 关于 $\overrightarrow{v_a}$ 两个分量的二次型。维护 $\overrightarrow{v_a}$ 的懒标记即可 $\Theta(1)$ 修改一个结点的信息。

Solution

令 $b_i = a_i^2$，我们考虑设计懒标记与维护的信息使得我们可以在 $\Theta(1)$ 的时间复杂度内修改或查询一个结点的信息。

注意到 $b'_i = a'^2_i = \left(a_i + \overrightarrow{t_2} A^{i-l} \overrightarrow{v_a}\right)^2 = b_i + \left(\overrightarrow{t_2} A^{i-l} \overrightarrow{v_a}\right)^2 + 2a_i \overrightarrow{t_2} A^{i-l} \overrightarrow{v_a}$，我们可以对每个结点维护 b_i 的区间和与 $\sum_{i=l}^{r} a_i \overrightarrow{t_2} A^{i-l}$，即可 $\Theta(1)$ 查询一个结点的信息。

由矩阵乘法的分配律，我们在修改一个结点的信息时会使后者增加 $\sum_{i=0}^{r-l} \overrightarrow{t_2} A^i \overrightarrow{v_a} \overrightarrow{t_2} A^i$，注意到这个向量在 $r-l+1$ 一定时可以表示为一个常系数矩阵乘 $\overrightarrow{v_a}$，我们可以进行预处理。同样地，我们可以预处理 $\sum_{i=0}^{r-l} \left(\overrightarrow{t_2} A^i \overrightarrow{v_a}\right)^2$ 关于 $\overrightarrow{v_a}$ 两个分量的二次型。维护 $\overrightarrow{v_a}$ 的懒标记即可 $\Theta(1)$ 修改一个结点的信息。

时间复杂度 $\Theta(n + q \log n)$。

Problem (【ZJOI2019】线段树)

初始时有一棵线段树，叶子结点编号按 dfs 序为 $[1,n]\cap\mathbb{N}$，非叶子结点区间分界点为当前对应区间左右端点平均值下取整，每个结点均没有标记。m 次操作：

1. 区间修改：给定一个区间，将当前所有线段树复制为两份，再对其中一份从根结点开始进行修改：若当前结点对应区间与给定区间无交则直接返回，若当前结点对应区间包含于给定区间则当前结点变为有标记状态，否则下传标记（若当前结点有标记则将两个儿子变为有标记状态并将当前结点变为无标记状态）并递归对两个儿子结点进行操作。
2. 询问所有线段树上的标记数量。

$1\leqslant n,m\leqslant 10^5$。

Solution

对于每个结点，维护 f 表示当前结点在所有线段树中被打标记的频率，g 表示当前结点到根链存在结点被打标记的频率。

Solution

对于每个结点，维护 f 表示当前结点在所有线段树中被打标记的频率，g 表示当前结点到根链存在结点被打标记的频率。

一次修改操作会使区间定位的结点集的 f 变为 $(f+1)/2$，g 变为 $(g+1)/2$，结点集的祖先结点（不含自身）的 f 与 g 变为 0 结点集的祖先结点的儿子结点且非前两者的结点的 f 变为 $(f+g)/2$，结点集子树内非集合内结点的 g 变为 $(g+1)/2$，其余结点的 f 与 g 不变，维护 f 之和即可。

Solution

对于每个结点，维护 f 表示当前结点在所有线段树中被打标记的频率，g 表示当前结点到根链存在结点被打标记的频率。

一次修改操作会使区间定位的结点集的 f 变为 $(f+1)/2$，g 变为 $(g+1)/2$，结点集的祖先结点（不含自身）的 f 与 g 变为 0 结点集的祖先结点的儿子结点且非前两者的结点的 f 变为 $(f+g)/2$，结点集子树内非集合内结点的 g 变为 $(g+1)/2$，其余结点的 f 与 g 不变，维护 f 之和即可。

时间复杂度 $\Theta(n+m+m_1 \log n)$。

Problem (矩形覆盖问题)

给定 n 个边与坐标轴平行的矩形，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 个边与坐标轴平行的矩形，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 $O(n \log n)$。

注意到问题答案只与同一维坐标的相对大小有关，当 $m > n$ 时，这类问题可以通过离散化使得值域 $m \leqslant n$。（部分问题可能会为 $\Theta(n)$。）

Problem (矩形覆盖问题)

给定 n 个边与坐标轴平行的矩形，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 $O(n \log n)$。

注意到问题答案只与同一维坐标的相对大小有关，当 $m > n$ 时，这类问题可以通过离散化使得值域 $m \leqslant n$。（部分问题可能会为 $\Theta(n)$。）

Solution

对于静态的二维问题，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个结点维护区间最小值及其个数与区间加的懒标记。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个结点维护区间最小值及其个数与区间加的懒标记。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。

时间复杂度为 $\Theta(m+n\log m)$。

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个结点维护区间最小值及其个数与区间加的懒标记。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。

时间复杂度为 $\Theta(m+n\log m)$。

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。

时间复杂度 $\Theta(m+(n+q)\log m)$。

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。

时间复杂度 $\Theta(m+(n+q)\log m)$。

事实上，这个问题在离散化后有更加优秀的算法（如时间复杂度 $O\left((n+q)\sqrt{\log(n+q)}\right)$），但这里不加介绍。

Problem (区间颜色数)

给定一个长度为 n 的数组 a，q 次询问给定 $[l,r] \subseteq [1,n]$ 求 $|\{a_i | i \in [l,r] \cap \mathbb{N}\}|$。

n, q 同阶，要求时间复杂度 $O(n \log n)$。

离线版本：【SDOI2009】HH 的项链 /【HEOI2012】采花，后者略有区别。在线需使用课件后面的内容。

Problem (区间颜色数)

给定一个长度为 n 的数组 a，q 次询问给定 $[l,r] \subseteq [1,n]$ 求 $|\{a_i | i \in [l,r] \cap \mathbb{N}\}|$。

n, q 同阶，要求时间复杂度 $O(n \log n)$。

离线版本：【SDOI2009】HH 的项链 / 【HEOI2012】采花，后者略有区别。在线需使用课件后面的内容。

Solution

考虑区间 $[l,r]$ 中每种值第一次出现的位置，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)

给定一个长度为 n 的数组 a，q 次询问给定 $[l,r] \subseteq [1,n]$ 求 $|\{a_i | i \in [l,r] \cap \mathbb{N}\}|$。

n, q 同阶，要求时间复杂度 $O(n \log n)$。

离线版本：【SDOI2009】HH 的项链 / 【HEOI2012】采花，后者略有区别。在线需使用课件后面的内容。

Solution

考虑区间 $[l,r]$ 中每种值第一次出现的位置，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 $\mathrm{pre}_x = \max(\{0\} \cup \{i \in [1,x) \cap \mathbb{N} | a_i = a_x\})$，那么对于 $x \in [l,r] \cap \mathbb{N}$，$a_x$ 在 $[l,r]$ 中第一次出现等价于 $\mathrm{pre}_x < l$。

Problem (区间颜色数)

给定一个长度为 n 的数组 a，q 次询问给定 $[l,r] \subseteq [1,n]$ 求 $|\{a_i | i \in [l,r] \cap \mathbb{N}\}|$。

n, q 同阶，要求时间复杂度 $O(n \log n)$。

离线版本：【SDOI2009】HH 的项链 /【HEOI2012】采花，后者略有区别。在线需使用课件后面的内容。

Solution

考虑区间 $[l,r]$ 中每种值第一次出现的位置，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 $\mathrm{pre}_x = \max(\{0\} \cup \{i \in [1,x) \cap \mathbb{N} | a_i = a_x\})$，那么对于 $x \in [l,r] \cap \mathbb{N}$，$a_x$ 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。

Solution

我们可以从另一种较为套路的思路解决这个问题。

Solution

我们可以从另一种较为套路的思路解决这个问题。

我们从小到大枚举右端点 r，用线段树维护每个左端点 l 的答案 $f_{r,l}$。

Solution

我们可以从另一种较为套路的思路解决这个问题。

我们从小到大枚举右端点 r，用线段树维护每个左端点 l 的答案 $f_{r,l}$。

对于 $\mathrm{pre}_r < l \leqslant r$，$f_{r,l} = f_{r-1,l} + 1$；对于其它 l，$f_{r,l} = f_{r-1,l}$。相当于 f_r 由 f_{r-1} 通过一次区间加操作得到。

Solution

我们可以从另一种较为套路的思路解决这个问题。

我们从小到大枚举右端点 r，用线段树维护每个左端点 l 的答案 $f_{r,l}$。

对于 $\mathrm{pre}_r < l \leqslant r$，$f_{r,l} = f_{r-1,l} + 1$；对于其它 l，$f_{r,l} = f_{r-1,l}$。相当于 f_r 由 f_{r-1} 通过一次区间加操作得到。

时间复杂度 $\Theta((n+q)\log n)$，在线需要持久化相关内容。

Solution

我们可以从另一种较为套路的思路解决这个问题。

我们从小到大枚举右端点 r，用线段树维护每个左端点 l 的答案 $f_{r,l}$。

对于 $\mathrm{pre}_r < l \leqslant r$，$f_{r,l} = f_{r-1,l} + 1$；对于其它 l，$f_{r,l} = f_{r-1,l}$。相当于 f_r 由 f_{r-1} 通过一次区间加操作得到。

时间复杂度 $\Theta((n+q)\log n)$，在线需要持久化相关内容。

可以与二维数点问题的解法对比得到与上一页相同的结果。

Problem (区间 mex)

给定一个长度为 n 的数组 a，q 次询问给定 $[l, r] \subseteq [1, n]$ 求 $\operatorname{mex}\{a_i | i \in [l, r] \cap \mathbb{N}\}$。

n, q 同阶，要求时间复杂度 $O(n \log n)$。

Problem (区间 mex)

给定一个长度为 n 的数组 a，q 次询问给定 $[l,r] \subseteq [1,n]$ 求 $\operatorname{mex}\{a_i | i \in [l,r] \cap \mathbb{N}\}$。

n, q 同阶，要求时间复杂度 $O(n \log n)$。

Solution

我们从大到小枚举右端点 r，用线段树维护每个左端点的答案。每次左移相当于 $(\max\{i \in [1,n] \cap \mathbb{N} | a_i = a_r\}, r]$ 对 a_r 取最小值。

Problem (区间 mex)

给定一个长度为 n 的数组 a，q 次询问给定 $[l,r] \subseteq [1,n]$ 求 $\operatorname{mex}\{a_i | i \in [l,r] \cap \mathbb{N}\}$。

n, q 同阶，要求时间复杂度 $O(n \log n)$。

Solution

我们从大到小枚举右端点 r，用线段树维护每个左端点的答案。每次左移相当于 $(\max\{i \in [1,n] \cap \mathbb{N} | a_i = a_r\}, r]$ 对 a_r 取最小值。

注意到相同右端点的答案随左端点增大单调不增，因此该操作相当于找到第一个更小的位置再进行区间赋值，也可用 set 维护连续段。

Problem (区间 mex)

给定一个长度为 n 的数组 a，q 次询问给定 $[l,r] \subseteq [1,n]$ 求 $\operatorname{mex}\{a_i | i \in [l,r] \cap \mathbb{N}\}$。

n, q 同阶，要求时间复杂度 $O(n \log n)$。

Solution

我们从大到小枚举右端点 r，用线段树维护每个左端点的答案。每次左移相当于 $(\max\{i \in [1,n] \cap \mathbb{N} | a_i = a_r\}, r]$ 对 a_r 取最小值。

注意到相同右端点的答案随左端点增大单调不增，因此该操作相当于找到第一个更小的位置再进行区间赋值，也可用 set 维护连续段。

也可以从左往右枚举右端点用线段树维护每个值的最后出现位置，使用后面的线段树上二分。

Problem (区间 mex)

给定一个长度为 n 的数组 a，q 次询问给定 $[l,r] \subseteq [1,n]$ 求 $\operatorname{mex}\{a_i | i \in [l,r] \cap \mathbb{N}\}$。

n, q 同阶，要求时间复杂度 $O(n \log n)$。

Solution

我们从大到小枚举右端点 r，用线段树维护每个左端点的答案。每次左移相当于 $(\max\{i \in [1,n] \cap \mathbb{N} | a_i = a_r\}, r]$ 对 a_r 取最小值。

注意到相同右端点的答案随左端点增大单调不增，因此该操作相当于找到第一个更小的位置再进行区间赋值，也可用 set 维护连续段。

也可以从左往右枚举右端点用线段树维护每个值的最后出现位置，使用后面的线段树上二分。

时间复杂度 $\Theta(n \log n)$。

Problem (LOJ6698 一键挖矿)

给定一个 $n \times m$ 的矩阵 A，满足元素两两不同。求有多少子矩形值域连续，即 $|\{(l_1, r_1, l_2, r_2) | l_1, r_1 \in [1, n] \cap \mathbb{N}, l_2, r_2 \in [1, m] \cap \mathbb{N}, \exists l_3, r_3, \{A_{i,j} | (l_1 \leqslant i \leqslant r_1) \wedge (l_2 \leqslant j \leqslant r_2)\} = [l_3, r_3] \cap \mathbb{N} \neq \varnothing\}|$。

$1 \leqslant A_{i,j} \leqslant nm \leqslant 2 \times 10^5, n, m \geqslant 1$。

Problem (LOJ6698 一键挖矿)

给定一个 $n \times m$ 的矩阵 A，满足元素两两不同。求有多少子矩形值域连续，即 $|\{(l_1, r_1, l_2, r_2) | l_1, r_1 \in [1, n] \cap \mathbb{N}, l_2, r_2 \in [1, m] \cap \mathbb{N}, \exists l_3, r_3, \{A_{i,j} | (l_1 \leqslant i \leqslant r_1) \wedge (l_2 \leqslant j \leqslant r_2)\} = [l_3, r_3] \cap \mathbb{N} \neq \varnothing\}|$。

$1 \leqslant A_{i,j} \leqslant nm \leqslant 2 \times 10^5, n, m \geqslant 1$。

Solution

相当于一个 $n \times m$ 的矩阵与一个长度为 nm 的序列有一个一一对应关系，求有多少元素集合在序列上为区间，在矩阵内为子矩形。

Problem (LOJ6698 一键挖矿)

给定一个 $n \times m$ 的矩阵 A，满足元素两两不同。求有多少子矩形值域连续，即 $|\{(l_1, r_1, l_2, r_2) | l_1, r_1 \in [1, n] \cap \mathbb{N}, l_2, r_2 \in [1, m] \cap \mathbb{N}, \exists l_3, r_3, \{A_{i,j} | (l_1 \leqslant i \leqslant r_1) \wedge (l_2 \leqslant j \leqslant r_2)\} = [l_3, r_3] \cap \mathbb{N} \neq \varnothing\}|$。

$1 \leqslant A_{i,j} \leqslant nm \leqslant 2 \times 10^5, n, m \geqslant 1$。

Solution

相当于一个 $n \times m$ 的矩阵与一个长度为 nm 的序列有一个一一对应关系，求有多少元素集合在序列上为区间，在矩阵内为子矩形。

显然一维的更容易枚举，我们考虑枚举序列区间的右端点，计数有多少左端点满足区间对应在矩阵中是一个子矩形。

Solution

我们考虑构造一个线段树易于维护的指标，可以判定一个矩阵中的点集是否为一个矩形。

Solution

我们考虑构造一个线段树易于维护的指标，可以判定一个矩阵中的点集是否为一个矩形。

一定形式下的连续往往可以表示为局部的性质，我们考虑令指标为某类对象的个数，在待计数对象满足计数条件时取到可能的某上界或下界。

Solution

我们考虑构造一个线段树易于维护的指标，可以判定一个矩阵中的点集是否为一个矩形。

一定形式下的连续往往可以表示为局部的性质，我们考虑令指标为某类对象的个数，在待计数对象满足计数条件时取到可能的某上界或下界。

一个矩形有四个角。

Solution

我们考虑构造一个线段树易于维护的指标，可以判定一个矩阵中的点集是否为一个矩形。

一定形式下的连续往往可以表示为局部的性质，我们考虑令指标为某类对象的个数，在待计数对象满足计数条件时取到可能的某上界或下界。

一个矩形有四个角。

为了便于处理边界情况，我们在矩阵外包裹一层在点集外的元素，然后我们考察整个矩阵的 2×2 的子矩形。

Solution

我们考虑构造一个线段树易于维护的指标，可以判定一个矩阵中的点集是否为一个矩形。

一定形式下的连续往往可以表示为局部的性质，我们考虑令指标为某类对象的个数，在待计数对象满足计数条件时取到可能的某上界或下界。

一个矩形有四个角。

为了便于处理边界情况，我们在矩阵外包裹一层在点集外的元素，然后我们考察整个矩阵的 2×2 的子矩形。

我们对有且仅有只有一个点集内元素的 2×2 的子矩形进行计数，注意到按行标与列标正序和倒序共四种排序的第一个在点集内的元素一定对应了一个以它为四个角的 2×2 的子矩形，且矩形恰好有四个。

Solution

我们逐行进行考虑，发现满足只有四个 2×2 的子矩形中有且仅有一个点集内元素的点集是一个矩形当且仅当没有 2×2 的子矩形中有且仅有三个点集内元素。

Solution

我们逐行进行考虑，发现满足只有四个 2×2 的子矩形中有且仅有一个点集内元素的点集是一个矩形当且仅当没有 2×2 的子矩形中有且仅有三个点集内元素。

我们将点集内元素个数为 1 或 3 个的 2×2 的子矩形个数视为指标，在枚举值域区间右端点时用线段树维护每个左端点的指标最小值及出现次数即可。

Solution

我们逐行进行考虑，发现满足只有四个 2×2 的子矩形中有且仅有一个点集内元素的点集是一个矩形当且仅当没有 2×2 的子矩形中有且仅有三个点集内元素。

我们将点集内元素个数为 1 或 3 个的 2×2 的子矩形个数视为指标，在枚举值域区间右端点时用线段树维护每个左端点的指标最小值及出现次数即可。

时间复杂度 $\Theta(nm(\log n+\log m))$。

Problem (某知名广义线段树)

我们考虑区间分界点为任意独立给定的线段树（同时不保证广义重量平衡的条件）。给定一棵 n 个叶子的上述的广义线段树，每个点带点权，q 次在线询问一个区间定位的结点集的点权和。

要求时间复杂度 $O(n+q\log n)$。

Problem (某知名广义线段树)

我们考虑区间分界点为任意独立给定的线段树（同时不保证广义重量平衡的条件）。给定一棵 n 个叶子的上述的广义线段树，每个点带点权，q 次在线询问一个区间定位的结点集的点权和。

要求时间复杂度 $O(n+q\log n)$。

Solution

对于询问 $[l,r]$，我们可以求出 l 和 r 对应叶子结点的最近公共祖先 u。对于 u 的对应区间恰好为 $[l,r]$ 的情况很好处理，除此情况之外我们可以对 u 左儿子对应区间的一个后缀在其子树内的定位结点集权值和与右儿子对应区间的一个前缀在其子树内的定位结点集权值和分别计算。

Problem (某知名广义线段树)

我们考虑区间分界点为任意独立给定的线段树（同时不保证广义重量平衡的条件）。给定一棵 n 个叶子的上述的广义线段树，每个点带点权，q 次在线询问一个区间定位的结点集的点权和。

要求时间复杂度 $O(n+q\log n)$。

Solution

对于询问 $[l,r]$，我们可以求出 l 和 r 对应叶子结点的最近公共祖先 u。对于 u 的对应区间恰好为 $[l,r]$ 的情况很好处理，除此情况之外我们可以对 u 左儿子对应区间的一个后缀在其子树内的定位结点集权值和与右儿子对应区间的一个前缀在其子树内的定位结点集权值和分别计算。

两者显然是对称的，我们考虑如何计算一个右儿子结点 u 对应区间的一个前缀 $(m,r]$ 定位的结点集权值和。

Solution

对于前缀 $(m, r]$ 恰好是 u 的对应区间的情况显然权值和即 u 的点权，否则定位的结点集为一个左儿子结点集，其中从左到右排列的第一个左儿子结点对应区间左端点为 $m+1$，最后一个左儿子结点对应区间右端点为 r，第 i 个左儿子结点为第 $i+1$ 个左儿子结点所在的极长左链中深度最小的结点的左兄弟。

Solution

对于前缀 $(m, r]$ 恰好是 u 的对应区间的情况显然权值和即 u 的点权，否则定位的结点集为一个左儿子结点集，其中从左到右排列的第一个左儿子结点对应区间左端点为 $m+1$，最后一个左儿子结点对应区间右端点为 r，第 i 个左儿子结点为第 $i+1$ 个左儿子结点所在的极长左链中深度最小的结点的左兄弟。

我们考虑对所有左儿子结点建立一个辅助森林，每个左儿子的父亲为其所在的极长左链中深度最小的结点的左兄弟，并对森林中每个结点求出其祖先结点（含自身）的权值和。所求的答案即 r 对应叶子结点所在的极长右链中深度最小的结点的祖先权值和减去 u 的左兄弟的祖先权值和。

Solution

对于前缀 $(m, r]$ 恰好是 u 的对应区间的情况显然权值和即 u 的点权，否则定位的结点集为一个左儿子结点集，其中从左到右排列的第一个左儿子结点对应区间左端点为 $m+1$，最后一个左儿子结点对应区间右端点为 r，第 i 个左儿子结点为第 $i+1$ 个左儿子结点所在的极长左链中深度最小的结点的左兄弟。

我们考虑对所有左儿子结点建立一个辅助森林，每个左儿子的父亲为其所在的极长左链中深度最小的结点的左兄弟，并对森林中每个结点求出其祖先结点（含自身）的权值和。所求的答案即 r 对应叶子结点所在的极长右链中深度最小的结点的祖先权值和减去 u 的左兄弟的祖先权值和。

时间复杂度与 $\Theta(n)$ 个结点，q 次询问最近公共祖先的时间复杂度相同。

Problem (【APIO2023】序列)

对一个长度为 m 的序列，定义其权值为其第 $\lfloor (m+1)/2 \rfloor$ 大或第 $\lceil (m+1)/2 \rceil$ 大的数的出现次数的最大值。给定一个长度为 n 的序列 a，求其所有区间的权值最大值。

$1 \leqslant a_i \leqslant n \leqslant 5 \times 10^5$。

Problem (【APIO2023】序列)

对一个长度为 m 的序列，定义其权值为其第 $\lfloor (m+1)/2 \rfloor$ 大或第 $\lceil (m+1)/2 \rceil$ 大的数的出现次数的最大值。给定一个长度为 n 的序列 a，求其所有区间的权值最大值。

$1 \leqslant a_i \leqslant n \leqslant 5 \times 10^5$。

Solution

我们考虑改变枚举次序，对每个数 x 求出以其为中位数之一的区间的最长长度。

Problem (【APIO2023】序列)

对一个长度为 m 的序列，定义其权值为其第 $\lfloor (m+1)/2 \rfloor$ 大或第 $\lceil (m+1)/2 \rceil$ 大的数的出现次数的最大值。给定一个长度为 n 的序列 a，求其所有区间的权值最大值。

$1 \leqslant a_i \leqslant n \leqslant 5 \times 10^5$。

Solution

我们考虑改变枚举次序，对每个数 x 求出以其为中位数之一的区间的最长长度。

我们考虑构造关于 x 的另一个序列 b_i，当 $a_i < x$ 时 $b_i = -1$，当 $a_i > x$ 时 $b_i = 1$，当 $b_i = x$ 时 $b_i \in \{-1, 0, 1\}$。x 为 a 的一个区间的中位数之一当且仅当 b 的对应区间和可能为 0。

Solution

我们考虑如何检验左端点在 $[l_1, l_2]$ 中，右端点 $\geqslant r$ 的区间是否存在一个满足其中位数之一是 x，其中 $[l_1, l_2)$ 之中没有 x。

Solution

我们考虑如何检验左端点在 $[l_1, l_2]$ 中，右端点 $\geqslant r$ 的区间是否存在一个满足其中位数之一是 x，其中 $[l_1, l_2)$ 之中没有 x。

我们将 $< l_1$ 的位置全部确定值，然后线段树维护 b 的前缀和可能的最大值和最小值，我们只需要查看 $[l_1 - 1, l_2)$ 中可能的前缀取值与 $[r, n]$ 可能的前缀取值是否有交。（事实上我们也可以将 $[l_1 - 1, l_2)$ 换成 $[1, l_2)$，因为找到一个左端点 $< l_1$ 的解往往没有影响。）

Solution

我们考虑如何检验左端点在 $[l_1, l_2]$ 中，右端点 $\geqslant r$ 的区间是否存在一个满足其中位数之一是 x，其中 $[l_1, l_2)$ 之中没有 x。

我们将 $< l_1$ 的位置全部确定值，然后线段树维护 b 的前缀和可能的最大值和最小值，我们只需要查看 $[l_1 - 1, l_2)$ 中可能的前缀取值与 $[r, n]$ 可能的前缀取值是否有交。（事实上我们也可以将 $[l_1 - 1, l_2)$ 换成 $[1, l_2)$，因为找到一个左端点 $< l_1$ 的解往往没有影响。）

我们从小到大枚举 x，先将值为 x 的位置的最小可能值改为 -1，再从左往右枚举 x 出现的位置，不断判断 l_2 为当前位置是否能得到比当前答案大的解，再将当前位置的最大可能值从 1 改为 -1。

Solution

我们考虑如何检验左端点在 $[l_1, l_2]$ 中，右端点 $\geqslant r$ 的区间是否存在一个满足其中位数之一是 x，其中 $[l_1, l_2)$ 之中没有 x。

我们将 $< l_1$ 的位置全部确定值，然后线段树维护 b 的前缀和可能的最大值和最小值，我们只需要查看 $[l_1 - 1, l_2)$ 中可能的前缀取值与 $[r, n]$ 可能的前缀取值是否有交。（事实上我们也可以将 $[l_1 - 1, l_2)$ 换成 $[1, l_2)$，因为找到一个左端点 $< l_1$ 的解往往没有影响。）

我们从小到大枚举 x，先将值为 x 的位置的最小可能值改为 -1，再从左往右枚举 x 出现的位置，不断判断 l_2 为当前位置是否能得到比当前答案大的解，再将当前位置的最大可能值从 1 改为 -1。

时间复杂度 $\Theta(n \log n)$。

Problem (SOJ1814 游戏)

依次发生 q 个事件，你在决策前知道全部的事件信息：

1. 获得一瓶药水，可重复使用任意次数：给定能力值 v_i、神秘属性 c_i。
2. 一瓶之前获得的药水过期，不能再使用：给定其编号。
3. 遇到一个关卡：给定难度值 h_i，你需要使用一瓶未过期的 $v_j \geqslant h_i$ 的药水 j 使用。

你需要判定是否存在一种方案，如果存在，求出一个方案使用过的药水神秘属性最大值减最小值的最小值。

$1 \leqslant q_1, q_2, q_3 \leqslant 10^6, 1 \leqslant c_i, v_i, h_i \leqslant 10^9$。

Solution

我们需要最小化某一属性的极差，考虑使用双指针：将所有药水按 c_i 排序，枚举使用过的第一瓶，求使用过的最后一瓶的最小位置。

Solution

我们需要最小化某一属性的极差，考虑使用双指针：将所有药水按 c_i 排序，枚举使用过的第一瓶，求使用过的最后一瓶的最小位置。

注意到满足条件当且仅当每个关卡都能选择一瓶药水。

Solution

我们需要最小化某一属性的极差，考虑使用双指针：将所有药水按 c_i 排序，枚举使用过的第一瓶，求使用过的最后一瓶的最小位置。

注意到满足条件当且仅当每个关卡都能选择一瓶药水。

我们考虑双指针的过程，对问题进行重新叙述：

Solution

我们需要最小化某一属性的极差，考虑使用双指针：将所有药水按 c_i 排序，枚举使用过的第一瓶，求使用过的最后一瓶的最小位置。

注意到满足条件当且仅当每个关卡都能选择一瓶药水。

我们考虑双指针的过程，对问题进行重新叙述：

Problem

给定长度为 q_3 的序列 $\{h_i\}$ 的初值，$\Theta(q_1)$ 次操作：

Solution

我们需要最小化某一属性的极差，考虑使用双指针：将所有药水按 c_i 排序，枚举使用过的第一瓶，求使用过的最后一瓶的最小位置。

注意到满足条件当且仅当每个关卡都能选择一瓶药水。

我们考虑双指针的过程，对问题进行重新叙述：

Problem

给定长度为 q_3 的序列 $\{h_i\}$ 的初值，$\Theta(q_1)$ 次操作：

1. 加入一个操作：给定 $[l,r]\subseteq[1,n],v$，对于 $i\in[l,r]\cap\mathbb{N}$，若 $h_i\leqslant v$ 则将 h_i 赋值为 0。

Solution

我们需要最小化某一属性的极差，考虑使用双指针：将所有药水按 c_i 排序，枚举使用过的第一瓶，求使用过的最后一瓶的最小位置。

注意到满足条件当且仅当每个关卡都能选择一瓶药水。

我们考虑双指针的过程，对问题进行重新叙述：

Problem

给定长度为 q_3 的序列 $\{h_i\}$ 的初值，$\Theta(q_1)$ 次操作：

1. 加入一个操作：给定 $[l,r]\subseteq[1,n],v$，对于 $i\in[l,r]\cap\mathbb{N}$，若 $h_i\leqslant v$ 则将 h_i 赋值为 0。
2. 查询当前操作集合是否满足条件：查询 $\max h_i$ 是否为 0。

Solution

我们需要最小化某一属性的极差，考虑使用双指针：将所有药水按 c_i 排序，枚举使用过的第一瓶，求使用过的最后一瓶的最小位置。

注意到满足条件当且仅当每个关卡都能选择一瓶药水。

我们考虑双指针的过程，对问题进行重新叙述：

Problem

给定长度为 q_3 的序列 $\{h_i\}$ 的初值，$\Theta(q_1)$ 次操作：

1. 加入一个操作：给定 $[l,r]\subseteq[1,n],v$，对于 $i\in[l,r]\cap\mathbb{N}$，若 $h_i\leqslant v$ 则将 h_i 赋值为 0。
2. 查询当前操作集合是否满足条件：查询 $\max h_i$ 是否为 0。
3. 删除最早加入的一个操作。

Solution

对于没有删除操作的情况，我们可以用线段树对每个结点维护对应区间 h_i 的最大值与将对应区间内所有 $\geqslant v$ 的 h 赋值为 0 的懒标记。

Solution

对于没有删除操作的情况，我们可以用线段树对每个结点维护对应区间 h_i 的最大值与将对应区间内所有 $\geqslant v$ 的 h 赋值为 0 的懒标记。

对于有删除的情况，注意到懒标记可以标记永久化。

Solution

对于没有删除操作的情况，我们可以用线段树对每个结点维护对应区间 h_i 的最大值与将对应区间内所有 $\geqslant v$ 的 h 赋值为 0 的懒标记。

对于有删除的情况，注意到懒标记可以标记永久化。

我们观察懒标记的变化情况，发现可以视作每个结点有一个懒标记集合，每次操作可能会插入一个值或删除最早插入的值，实际懒标记是当前集合的最大值。

Solution

对于没有删除操作的情况，我们可以用线段树对每个结点维护对应区间 h_i 的最大值与将对应区间内所有 $\geqslant v$ 的 h 赋值为 0 的懒标记。

对于有删除的情况，注意到懒标记可以标记永久化。

我们观察懒标记的变化情况，发现可以视作每个结点有一个懒标记集合，每次操作可能会插入一个值或删除最早插入的值，实际懒标记是当前集合的最大值。

这是单调队列的经典问题，我们可以在每个结点用一个单调队列维护懒标记。

Solution

对于没有删除操作的情况，我们可以用线段树对每个结点维护对应区间 h_i 的最大值与将对应区间内所有 $\geqslant v$ 的 h 赋值为 0 的懒标记。

对于有删除的情况，注意到懒标记可以标记永久化。

我们观察懒标记的变化情况，发现可以视作每个结点有一个懒标记集合，每次操作可能会插入一个值或删除最早插入的值，实际懒标记是当前集合的最大值。

这是单调队列的经典问题，我们可以在每个结点用一个单调队列维护懒标记。

时间复杂度 $\Theta(q_3 + q_1 \log q_3)$。

Problem (大 sz 的游戏)

有一张 n 个点的有向图，对每个点给定参数 l_i, r_i，另给定一参数 L，j 到 i 有有向边当且仅当 $0 \leqslant j - i \leqslant L$ 且 $[l_i, r_i] \cap [l_j, r_j] \neq \varnothing$。对 $i \in [1, n] \cap \mathbb{N}$ 求点 i 到点 1 的最短路长度 d_i。

$1 \leqslant n \leqslant 5 \times 10^5$。

Problem (大 sz 的游戏)

有一张 n 个点的有向图，对每个点给定参数 l_i, r_i，另给定一参数 L，j 到 i 有有向边当且仅当 $0 \leqslant j - i \leqslant L$ 且 $[l_i, r_i] \cap [l_j, r_j] \neq \varnothing$。对 $i \in [1, n] \cap \mathbb{N}$ 求点 i 到点 1 的最短路长度 d_i。

$1 \leqslant n \leqslant 5 \times 10^5$。

Solution

我们转化一下问题，点 i 能在时间 $(i, i + L]$ 对 $[l_i, r_i]$ 产生 d_i 的贡献，d_i 即为时间 i 时 $[l_i, r_i]$ 贡献的最小值 $+1$。等价性是容易验证的。

Problem (大 sz 的游戏)

有一张 n 个点的有向图，对每个点给定参数 l_i, r_i，另给定一参数 L，j 到 i 有有向边当且仅当 $0 \leqslant j - i \leqslant L$ 且 $[l_i, r_i] \cap [l_j, r_j] \neq \varnothing$。对 $i \in [1, n] \cap \mathbb{N}$ 求点 i 到点 1 的最短路长度 d_i。

$1 \leqslant n \leqslant 5 \times 10^5$。

Solution

我们转化一下问题，点 i 能在时间 $(i, i + L]$ 对 $[l_i, r_i]$ 产生 d_i 的贡献，d_i 即为时间 i 时 $[l_i, r_i]$ 贡献的最小值 $+1$。等价性是容易验证的。

考虑标记永久化，对线段树的每个结点维护一个单调队列即可。

Problem (大 sz 的游戏)

有一张 n 个点的有向图，对每个点给定参数 l_i, r_i，另给定一参数 L，j 到 i 有有向边当且仅当 $0 \leqslant j - i \leqslant L$ 且 $[l_i, r_i] \cap [l_j, r_j] \neq \varnothing$。对 $i \in [1, n] \cap \mathbb{N}$ 求点 i 到点 1 的最短路长度 d_i。

$1 \leqslant n \leqslant 5 \times 10^5$。

Solution

我们转化一下问题，点 i 能在时间 $(i, i + L]$ 对 $[l_i, r_i]$ 产生 d_i 的贡献，d_i 即为时间 i 时 $[l_i, r_i]$ 贡献的最小值 $+1$。等价性是容易验证的。

考虑标记永久化，对线段树的每个结点维护一个单调队列即可。

时间复杂度 $\Theta(n \log n)$。

Problem (【Ynoi2015】世上最幸福的女孩)

给定一个长度为 n 的数组 a，q 次操作：全局加、求区间最大子段和。

$1 \leqslant n \leqslant 3 \times 10^5, 1 \leqslant q \leqslant 6 \times 10^5$，内存限制 128 MiB。

Problem (【Ynoi2015】世上最幸福的女孩)

给定一个长度为 n 的数组 a，q 次操作：全局加、求区间最大子段和。

$1 \leqslant n \leqslant 3 \times 10^5, 1 \leqslant q \leqslant 6 \times 10^5$，内存限制 128 MiB。

Solution

用线段树维护数组 a 在对应区间加上 x 后的区间和、最大前后缀和、最大子段和的上包络线，根据儿子结点的信息需要在关于对应区间长度成线性的时间复杂度内得到当前结点的信息，建树时空复杂度 $\Theta(n \log n)$。

Problem (【Ynoi2015】世上最幸福的女孩)

给定一个长度为 n 的数组 a，q 次操作：全局加、求区间最大子段和。

$1 \leqslant n \leqslant 3 \times 10^5, 1 \leqslant q \leqslant 6 \times 10^5$，内存限制 128 MiB。

Solution

用线段树维护数组 a 在对应区间加上 x 后的区间和、最大前后缀和、最大子段和的上包络线，根据儿子结点的信息需要在关于对应区间长度成线性的时间复杂度内得到当前结点的信息，建树时空复杂度 $\Theta(n \log n)$。

我们维护一个变量表示当前数组相对于初始数组全局加的值，每次查询相当于查询给定区间在加上某个值后的最大子段和。

Solution

我们在每次询问直接在给定区间定位的结点集维护的上包络线上二分查询当前增量的区间和、最大前后缀和、最大子段和，然后与区间最大子段和一致将信息合并，单次查询时间复杂度 $\Theta(\log^2 n)$。

Solution

我们在每次询问直接在给定区间定位的结点集维护的上包络线上二分查询当前增量的区间和、最大前后缀和、最大子段和，然后与区间最大子段和一致将信息合并，单次查询时间复杂度 $\Theta(\log^2 n)$。

我们考虑离线将询问按增量排序，注意到每个结点二分找到的位置单调不降，我们每次从上一次的位置顺序查找即可，在查找到前遍历的位置不会重复，因此查询总时间复杂度 $\Theta((n+q)\log n)$。（强制在线可以用分散层叠做到单次严格 $\Theta(\log n)$，但空间难以进一步优化。）

Solution

我们在每次询问直接在给定区间定位的结点集维护的上包络线上二分查询当前增量的区间和、最大前后缀和、最大子段和，然后与区间最大子段和一致将信息合并，单次查询时间复杂度 $\Theta(\log^2 n)$。

我们考虑离线将询问按增量排序，注意到每个结点二分找到的位置单调不降，我们每次从上一次的位置顺序查找即可，在查找到前遍历的位置不会重复，因此查询总时间复杂度 $\Theta((n+q)\log n)$。（强制在线可以用分散层叠做到单次严格 $\Theta(\log n)$，但空间难以进一步优化。）

我们同时处理所有询问，我们对每个排序后的询问编号记录原编号与左右端点，用链式前向星对每个结点维护一个新编号的不相交集合，我们可以直接对每个结点用一个数记录最小的编号，对每个编号记录它的后继，共 $2n-1+q$ 个 `int` 即可。

Solution

初始时将所有询问编号放入根结点的集合。我们遍历整棵线段树，递归进入一个结点时，将其维护的集合根据询问区间分成在当前结点查询、需递归左子树、无需递归左子树需递归右子树三类，分别作为当前结点、左儿子、右儿子的维护集合。接着递归左子树，将左儿子的集合归并入右儿子，递归右子树，合并两个儿子信息得到当前结点的上包络线，遍历当前结点的集合与上包络线进行查询，删除右端点与当前结点对应区间右端点相同的询问后将右儿子维护的集合归并入当前结点。

Solution

初始时将所有询问编号放入根结点的集合。我们遍历整棵线段树，递归进入一个结点时，将其维护的集合根据询问区间分成在当前结点查询、需递归左子树、无需递归左子树需递归右子树三类，分别作为当前结点、左儿子、右儿子的维护集合。接着递归左子树，将左儿子的集合归并入右儿子，递归右子树，合并两个儿子信息得到当前结点的上包络线，遍历当前结点的集合与上包络线进行查询，删除右端点与当前结点对应区间右端点相同的询问后将右儿子维护的集合归并入当前结点。

每个深度最多一个结点需要存储上包络线，总空间复杂度 $\Theta(n+q)$。

Solution

初始时将所有询问编号放入根结点的集合。我们遍历整棵线段树，递归进入一个结点时，将其维护的集合根据询问区间分成在当前结点查询、需递归左子树、无需递归左子树需递归右子树三类，分别作为当前结点、左儿子、右儿子的维护集合。接着递归左子树，将左儿子的集合归并入右儿子，递归右子树，合并两个儿子信息得到当前结点的上包络线，遍历当前结点的集合与上包络线进行查询，删除右端点与当前结点对应区间右端点相同的询问后将右儿子维护的集合归并入当前结点。

每个深度最多一个结点需要存储上包络线，总空间复杂度 $\Theta(n+q)$。

可以说明总时间复杂度仍为 $\Theta((n+q)\log n)$。

Solution

初始时将所有询问编号放入根结点的集合。我们遍历整棵线段树，递归进入一个结点时，将其维护的集合根据询问区间分成在当前结点查询、需递归左子树、无需递归左子树需递归右子树三类，分别作为当前结点、左儿子、右儿子的维护集合。接着递归左子树，将左儿子的集合归并入右儿子，递归右子树，合并两个儿子信息得到当前结点的上包络线，遍历当前结点的集合与上包络线进行查询，删除右端点与当前结点对应区间右端点相同的询问后将右儿子维护的集合归并入当前结点。

每个深度最多一个结点需要存储上包络线，总空间复杂度 $\Theta(n+q)$。

可以说明总时间复杂度仍为 $\Theta((n+q)\log n)$。

一些类似问题将在 KTT 小节介绍。

Problem (Souvenirs)

给定一个长度为 n 的序列 a，q 次询问：求 $\min_{i=l}^{r} \min_{j=i+1}^{r} |a_i - a_j|$。$1 \leqslant n, q \leqslant 10^5, 1 \leqslant a_i \leqslant V \leqslant 10^5$。

数据随机版本：【JSOI2009】面试的考验。

Problem (Souvenirs)

给定一个长度为 n 的序列 a，q 次询问：求 $\min_{i=l}^{r} \min_{j=i+1}^{r} |a_i - a_j|$。$1 \leqslant n, q \leqslant 10^5, 1 \leqslant a_i \leqslant V \leqslant 10^5$。

数据随机版本：【JSOI2009】面试的考验。

Solution

我们考虑离线，枚举右端点，用线段树维护每个左端点对应的答案。

Problem (Souvenirs)

给定一个长度为 n 的序列 a，q 次询问：求 $\min_{i=l}^{r} \min_{j=i+1}^{r} |a_i - a_j|$。$1 \leqslant n, q \leqslant 10^5, 1 \leqslant a_i \leqslant V \leqslant 10^5$。

数据随机版本：【JSOI2009】面试的考验。

Solution

我们考虑离线，枚举右端点，用线段树维护每个左端点对应的答案。

考虑右端点向右移动一位的影响：j 为新的右端点，$i \in [1, j) \cap \mathbb{N}$，左端点 $\leqslant i$ 的答案会对 $|a_i - a_j|$ 取最小值。

Problem (Souvenirs)

给定一个长度为 n 的序列 a，q 次询问：求 $\min_{i=l}^{r}\min_{j=i+1}^{r}|a_i-a_j|$。

$1\leqslant n,q\leqslant 10^5,1\leqslant a_i\leqslant V\leqslant 10^5$。

数据随机版本：【JSOI2009】面试的考验。

Solution

我们考虑离线，枚举右端点，用线段树维护每个左端点对应的答案。

考虑右端点向右移动一位的影响：j 为新的右端点，$i\in[1,j)\cap\mathbb{N}$，左端点 $\leqslant i$ 的答案会对 $|a_i-a_j|$ 取最小值。

事实上有效的 i 的取值很少，我们不妨分 $a_i\geqslant a_j$ 和 $a_i<a_j$ 两种情况讨论，这两种情况基本对称，这里我们以 $a_i\geqslant a_j$ 为例。

Solution

注意到若 $x < y < z$ 且 $a_y \geqslant a_z, a_x \geqslant \frac{a_y + a_z}{2}$ 则由于 (x, y) 的存在 (x, z) 不会对答案产生任何影响。

Solution

注意到若 $x < y < z$ 且 $a_y \geqslant a_z, a_x \geqslant \frac{a_y + a_z}{2}$ 则由于 (x, y) 的存在 (x, z) 不会对答案产生任何影响。

我们令 i_0 为满足 $i < j$ 且 $a_i \geqslant a_j$ 的最大的 i，显然 $i \geqslant i_0$ 且 $i \neq i_0$ 的 i 都不会产生影响。

Solution

注意到若 $x < y < z$ 且 $a_y \geqslant a_z, a_x \geqslant \frac{a_y + a_z}{2}$ 则由于 (x, y) 的存在 (x, z) 不会对答案产生任何影响。

我们令 i_0 为满足 $i < j$ 且 $a_i \geqslant a_j$ 的最大的 i，显然 $i \geqslant i_0$ 且 $i \neq i_0$ 的 i 都不会产生影响。

令 $i_k (k \in \mathbb{N}^*)$ 为满足 $i < i_{k-1}$ 且 $a_j \leqslant a_i < \frac{a_j + a_{i_{k-1}}}{2}$ 的最大的 i。

Solution

注意到若 $x < y < z$ 且 $a_y \geqslant a_z, a_x \geqslant \frac{a_y + a_z}{2}$ 则由于 (x, y) 的存在 (x, z) 不会对答案产生任何影响。

我们令 i_0 为满足 $i < j$ 且 $a_i \geqslant a_j$ 的最大的 i，显然 $i \geqslant i_0$ 且 $i \neq i_0$ 的 i 都不会产生影响。

令 $i_k (k \in \mathbb{N}^*)$ 为满足 $i < i_{k-1}$ 且 $a_j \leqslant a_i < \frac{a_j + a_{i_{k-1}}}{2}$ 的最大的 i。

显然若 i_k 不存在，则 $i < i_{k-1}$ 的 i 没有任何影响，否则 $i_k < i < i_{k-1}$ 的 i 没有任何影响，根据数学归纳法可知 $i \geqslant i_k$ 且 $\forall p \in [0, k] \cap \mathbb{N}, i \neq i_p$ 的 i 没有任何影响。

Solution

注意到若 $x < y < z$ 且 $a_y \geqslant a_z, a_x \geqslant \frac{a_y + a_z}{2}$ 则由于 (x, y) 的存在 (x, z) 不会对答案产生任何影响。

我们令 i_0 为满足 $i < j$ 且 $a_i \geqslant a_j$ 的最大的 i，显然 $i \geqslant i_0$ 且 $i \neq i_0$ 的 i 都不会产生影响。

令 $i_k (k \in \mathbb{N}^*)$ 为满足 $i < i_{k-1}$ 且 $a_j \leqslant a_i < \frac{a_j + a_{i_{k-1}}}{2}$ 的最大的 i。

显然若 i_k 不存在，则 $i < i_{k-1}$ 的 i 没有任何影响，否则 $i_k < i < i_{k-1}$ 的 i 没有任何影响，根据数学归纳法可知 $i \geqslant i_k$ 且 $\forall p \in [0, k] \cap \mathbb{N}, i \neq i_p$ 的 i 没有任何影响。

由 $a_j \leqslant a_{i_k} < \frac{a_j + a_{i_{k-1}}}{2}$ 且 $a_i \in \mathbb{Z}$ 可知 k 为 $O(\log V)$ 个，我们维护前缀的权值线段树每次暴力查询区间最大值即可做到 $O(V + n \log^2 V + q \log V)$ 的时间复杂度。

Problem

维护一个整数集合，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

维护一个整数集合，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 $O(q \log q)$。

为避免用到课件后面的内容，我们不妨先假设每次给定的数在 $[1, q]$ 之间，如果允许离线我们也可以通过离散化得到。

Problem

维护一个整数集合，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 $O(q \log q)$。

为避免用到课件后面的内容，我们不妨先假设每次给定的数在 $[1, q]$ 之间，如果允许离线我们也可以通过离散化得到。

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。我们考虑在线段树上直接二分。

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。我们考虑在线段树上直接二分。

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有一些运算比较冗余，首先我们可以以线段树中包含查询区间的结点的区间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，如这题将之前求出的小于当前结点对应区间左端点的前缀和加上左儿子结点的对应区间和即可得到小于等于左儿子结点对应区间右端点的前缀和。

线段树上二分往往给定一个区间端点，利用单调性二分出另一区间端点。区间操作的区间在多数情况下较小且无需区间定位结点集最近公共祖先的不含自身的祖先的结点信息时，我们可以使用 finger search 的技巧优化常数或时间复杂度。

线段树上二分往往给定一个区间端点，利用单调性二分出另一区间端点。区间操作的区间在多数情况下较小且无需区间定位结点集最近公共祖先的不含自身的祖先的结点信息时，我们可以使用 finger search 的技巧优化常数或时间复杂度。

我们不再从根结点开始递归二分，转而从给定的某一端点开始，每次将当前结点改为父亲结点直到区间的另一端点在当前结点的子树内，之后再按之前所述的方法进行二分。

线段树上二分往往给定一个区间端点，利用单调性二分出另一区间端点。区间操作的区间在多数情况下较小且无需区间定位结点集最近公共祖先的不含自身的祖先的结点信息时，我们可以使用 finger search 的技巧优化常数或时间复杂度。

我们不再从根结点开始递归二分，转而从给定的某一端点开始，每次将当前结点改为父亲结点直到区间的另一端点在当前结点的子树内，之后再按之前所述的方法进行二分。

令 $\Phi(i)$ 表示位置 i 对应叶结点的到根链中的右儿子结点数量，最终得到的区间为 $[l,r]\cap\mathbb{N}$，时间复杂度为 $\Theta(\Phi(l)-\Phi(r)+\log(r-l+2))$。

线段树上二分往往给定一个区间端点，利用单调性二分出另一区间端点。区间操作的区间在多数情况下较小且无需区间定位结点集最近公共祖先的不含自身的祖先的结点信息时，我们可以使用 finger search 的技巧优化常数或时间复杂度。

我们不再从根结点开始递归二分，转而从给定的某一端点开始，每次将当前结点改为父亲结点直到区间的另一端点在当前结点的子树内，之后再按之前所述的方法进行二分。

令 $\Phi(i)$ 表示位置 i 对应叶结点的到根链中的右儿子结点数量，最终得到的区间为 $[l,r]\cap\mathbb{N}$，时间复杂度为 $\Theta(\Phi(l)-\Phi(r)+\log(r-l+2))$。

部分区间较小的输入给定区间的区间操作也可使用类似方法优化，对于多次进行线段树上二分并对二分出的区间进行操作的部分问题我们也可以使用后述的条件递归进行常数优化甚至时间复杂度优化。

Problem (【APIO2018】新家)

给定 n 个商店，第 i 个商店类型为 $t_i \in [1,k] \cap \mathbb{N}$，在时间 $[a_i,b_i] \subseteq [1,T]$ 在位置 $x_i \in [1,V] \cap \mathbb{N}$ 出现。q 次询问：给定 $y_i \in [1,T] \cap \mathbb{N}$，$l_i \in [1,V] \cap \mathbb{N}$，设 $f(y,l,j)$ 为时间 y 所有类型为 j 的商店与位置 l 的距离最小值，若不存在该类商店则为 $+\infty$，求 $\max_{j=1}^{k} f(y_i,l_i,j)$。

$1 \leqslant n,k,q \leqslant 3 \times 10^5$，$T = V = 10^8$。

Problem (【APIO2018】新家)

给定 n 个商店，第 i 个商店类型为 $t_i \in [1,k] \cap \mathbb{N}$，在时间 $[a_i,b_i] \subseteq [1,T]$ 在位置 $x_i \in [1,V] \cap \mathbb{N}$ 出现。q 次询问：给定 $y_i \in [1,T] \cap \mathbb{N}$，$l_i \in [1,V] \cap \mathbb{N}$，设 $f(y,l,j)$ 为时间 y 所有类型为 j 的商店与位置 l 的距离最小值，若不存在该类商店则为 $+\infty$，求 $\max_{j=1}^{k} f(y_i,l_i,j)$。

$1 \leqslant n,k,q \leqslant 3 \times 10^5$，$T = V = 10^8$。

Solution

我们考虑如何用线段树维护一个区间 $[l,r]$ 是否包含所有类型的商店，支持在某个位置加入某个类型的商店或删除某个存在的商店。

Problem (【APIO2018】新家)

给定 n 个商店，第 i 个商店类型为 $t_i \in [1,k] \cap \mathbb{N}$，在时间 $[a_i,b_i] \subseteq [1,T]$ 在位置 $x_i \in [1,V] \cap \mathbb{N}$ 出现。q 次询问：给定 $y_i \in [1,T] \cap \mathbb{N}$，$l_i \in [1,V] \cap \mathbb{N}$，设 $f(y,l,j)$ 为时间 y 所有类型为 j 的商店与位置 l 的距离最小值，若不存在该类商店则为 $+\infty$，求 $\max_{j=1}^{k} f(y_i,l_i,j)$。

$1 \leqslant n,k,q \leqslant 3 \times 10^5$，$T = V = 10^8$。

Solution

我们考虑如何用线段树维护一个区间 $[l,r]$ 是否包含所有类型的商店，支持在某个位置加入某个类型的商店或删除某个存在的商店。

注意到区间种类数并不方便直接维护，我们可以对每个商店在 $-\infty$ 与 $+\infty$ 两个位置各增加一个，查询位置 $> r$ 的商店的同类前驱的位置是否都 $\geqslant l$，其正确性很方便验证。

Solution

注意到用 set 可以很方便地维护同一类型的商店，用 set 或 priority_queue 可以很方便地维护每个位置的所有商店的同类前驱的位置的最小值，我们可以将转化后的问题重新以以下形式化叙述：

Solution

注意到用 set 可以很方便地维护同一类型的商店，用 set 或 priority_queue 可以很方便地维护每个位置的所有商店的同类前驱的位置的最小值，我们可以将转化后的问题重新以以下形式化叙述：

维护一个长度不超过 $n+1$ 的序列，共 $4n$ 次单点修改序列一个位置的值以及 q 次查询给定一个 x 求最小的一个位置 i 使得以 i 为左端点的后缀最大值 $> 2x - p_i$。

Solution

注意到用 set 可以很方便地维护同一类型的商店，用 set 或 priority_queue 可以很方便地维护每个位置的所有商店的同类前驱的位置的最小值，我们可以将转化后的问题重新以以下形式化叙述：

维护一个长度不超过 $n+1$ 的序列，共 $4n$ 次单点修改序列一个位置的值以及 q 次查询给定一个 x 求最小的一个位置 i 使得以 i 为左端点的后缀最大值 $> 2x - p_i$。

显然对于一个询问若 i 满足条件则 $[i, n] \cap \mathbb{Z}$ 中的数均满足条件，因此可以在线段树上二分，每次判断答案位置在左子树还是右子树。

Solution

注意到用 set 可以很方便地维护同一类型的商店，用 set 或 priority_queue 可以很方便地维护每个位置的所有商店的同类前驱的位置的最小值，我们可以将转化后的问题重新以以下形式化叙述：

维护一个长度不超过 $n+1$ 的序列，共 $4n$ 次单点修改序列一个位置的值以及 q 次查询给定一个 x 求最小的一个位置 i 使得以 i 为左端点的后缀最大值 $> 2x - p_i$。

显然对于一个询问若 i 满足条件则 $[i, n] \cap \mathbb{Z}$ 中的数均满足条件，因此可以在线段树上二分，每次判断答案位置在左子树还是右子树。

总时间复杂度 $\Theta((n+q)\log n)$。

Solution

注意到用 set 可以很方便地维护同一类型的商店，用 set 或 priority_queue 可以很方便地维护每个位置的所有商店的同类前驱的位置的最小值，我们可以将转化后的问题重新以以下形式化叙述：

维护一个长度不超过 $n+1$ 的序列，共 $4n$ 次单点修改序列一个位置的值以及 q 次查询给定一个 x 求最小的一个位置 i 使得以 i 为左端点的后缀最大值 $> 2x - p_i$。

显然对于一个询问若 i 满足条件则 $[i, n] \cap \mathbb{Z}$ 中的数均满足条件，因此可以在线段树上二分，每次判断答案位置在左子树还是右子树。

总时间复杂度 $\Theta((n+q)\log n)$。

也可将相同位置的商店按一定顺序排列（可等价地视作位置有微小差距），使得每个位置最多一个商店，$+\infty$ 处使用 k 个商店，序列长度 $n+k$。

这里讨论比较广义的李超树，一般指的是修改或查询时先定位到区间，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。

这里讨论比较广义的李超树，一般指的是修改或查询时先定位到区间，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。

对于几乎所有线段树，若上传信息或修改单个结点的信息明显慢于递归，则我们可以在修改时将其设置为一个表示待求值的状态，在查询单个结点信息时对待求值的信息进行递归求值，容易说明其时间复杂度不会变劣。注意这种技巧由于时间复杂度是均摊的，不可持久化。

Problem (带插入半平面交 /【HEOI2013】Segment)

维护一个函数集合，q 次操作：插入一个定义域为一个区间的一次函数，询问某个自变量取值下所有定义域包含该点的函数值最大值。

强制在线，要求时间复杂度 $O(q\log^2 q)$。

弱化版本：【JSOI2008】Blue Mary 开公司。

Problem (带插入半平面交 /【HEOI2013】Segment)

维护一个函数集合，q 次操作：插入一个定义域为一个区间的一次函数，询问某个自变量取值下所有定义域包含该点的函数值最大值。

强制在线，要求时间复杂度 $O(q\log^2 q)$。

弱化版本：【JSOI2008】Blue Mary 开公司。

Solution

相当于插入线段求给定横坐标最高点，我们考虑用线段树维护 n 个点的答案。

Problem (带插入半平面交 /【HEOI2013】Segment)

维护一个函数集合，q 次操作：插入一个定义域为一个区间的一次函数，询问某个自变量取值下所有定义域包含该点的函数值最大值。

强制在线，要求时间复杂度 $O(q\log^2 q)$。

弱化版本：【JSOI2008】Blue Mary 开公司。

Solution

相当于插入线段求给定横坐标最高点，我们考虑用线段树维护 n 个点的答案。

我们试图在每个结点维护至多一条线段，通过恰当的操作使得覆盖每个点的 $\Theta(\log n)$ 个线段一定有一个是答案。

Problem (带插入半平面交 /【HEOI2013】Segment)

维护一个函数集合，q 次操作：插入一个定义域为一个区间的一次函数，询问某个自变量取值下所有定义域包含该点的函数值最大值。

强制在线，要求时间复杂度 $O(q\log^2 q)$。

弱化版本：【JSOI2008】Blue Mary 开公司。

Solution

相当于插入线段求给定横坐标最高点，我们考虑用线段树维护 n 个点的答案。

我们试图在每个结点维护至多一条线段，通过恰当的操作使得覆盖每个点的 $\Theta(\log n)$ 个线段一定有一个是答案。

插入一个线段时，根据定义域区间在线段树上的定位结果分成 $\Theta(\log n)$ 段，并在这些子树内分别进行插入操作。

Solution

对于往根结点没有维护线段的线段树进行插入的情况，显然可以将插入的线段放在根结点维护。

Solution

对于往根结点没有维护线段的线段树进行插入的情况，显然可以将插入的线段放在根结点维护。

否则我们观察在区间分界点的横坐标插入的线段与根结点维护的线段的大小关系，显然较小的一者至多对左子树与右子树的其中之一中的横坐标产生贡献，可以根据交点位置或左右端点横坐标的大小关系求得，我们将较大的一者作为根结点新的维护线段，较小的一者在对应子树内进行递归插入即可。

Solution

对于往根结点没有维护线段的线段树进行插入的情况，显然可以将插入的线段放在根结点维护。

否则我们观察在区间分界点的横坐标插入的线段与根结点维护的线段的大小关系，显然较小的一者至多对左子树与右子树的其中之一中的横坐标产生贡献，可以根据交点位置或左右端点横坐标的大小关系求得，我们将较大的一者作为根结点新的维护线段，较小的一者在对应子树内进行递归插入即可。

注意到往一棵子树插入单个线段的时间复杂度为 $\Theta(\log n)$，单次全局修改 $\Theta(\log n)$，单次任意区间的修改为 $O(\log^2 n)$，单次查询 $\Theta(\log q)$。

Solution

对于往根结点没有维护线段的线段树进行插入的情况，显然可以将插入的线段放在根结点维护。

否则我们观察在区间分界点的横坐标插入的线段与根结点维护的线段的大小关系，显然较小的一者至多对左子树与右子树的其中之一中的横坐标产生贡献，可以根据交点位置或左右端点横坐标的大小关系求得，我们将较大的一者作为根结点新的维护线段，较小的一者在对应子树内进行递归插入即可。

注意到往一棵子树插入单个线段的时间复杂度为 $\Theta(\log n)$，单次全局修改 $\Theta(\log n)$，单次任意区间的修改为 $O(\log^2 n)$，单次查询 $\Theta(\log q)$。

在线段树分治处有复杂度更优的更一般版本。

Problem (楼房重建)

一个长度为 n 的数组 a，初始均为 0，q 次操作：单点修改后求全局前缀最大值有多少种不同的取值。

$1 \leqslant n, q \leqslant 10^5$。

Problem (楼房重建)

一个长度为 n 的数组 a，初始均为 0，q 次操作：单点修改后求全局前缀最大值有多少种不同的取值。

$1 \leqslant n, q \leqslant 10^5$。

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)

一个长度为 n 的数组 a，初始均为 0，q 次操作：单点修改后求全局前缀最大值有多少种不同的取值。

$1 \leqslant n, q \leqslant 10^5$。

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。

对于修改时上传信息和询问，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)

一个长度为 n 的数组 a，初始均为 0，q 次操作：单点修改后求全局前缀最大值有多少种不同的取值。

$1 \leqslant n, q \leqslant 10^5$。

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。

对于修改时上传信息和询问，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。

单次修改时间复杂度 $O(\log^2 n)$，单次全局查询 $\Theta(\log n)$。

Problem (前缀 min 求和)

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

强制在线，要求时间复杂度 $O(n + q \log^2 n)$。

Problem (前缀 min 求和)

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

强制在线，要求时间复杂度 $O(n + q\log^2 n)$。

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 a，q 次操作：单点修改、求一个数列的 $\sum_{i=l}^{r} \min\{a_l, a_{l+1}, \ldots, a_i\}$。

强制在线，要求时间复杂度 $O(n + q\log^2 n)$。

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。

Problem (前缀 min 求和)

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

强制在线，要求时间复杂度 $O(n + q\log^2 n)$。

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。

时间复杂度 $O(n + q\log^2 n)$。

Problem (前缀 min 求和)

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

强制在线，要求时间复杂度 $O(n + q\log^2 n)$。

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。

时间复杂度 $O(n + q\log^2 n)$。

本题与上一题用下一小节内容可得到复杂度更优的离线算法。

Problem (【CEOI2017】Building Bridges)

一张无向图，初始时点集 V 为 $[1,n]\cap\mathbb{N}$，无边，可以以 w_i 的代价将 i 从点集中删除，若对于 $i<j,i,j\in V$ 满足 $\forall k\in(i,j),k\notin V$，可以以 $(h_i-h_j)^2$ 的代价将 $\{i,j\}$ 加入边集，问使 1 与 n 连通的最小代价。

$1\leqslant n\leqslant 10^5,0\leqslant h_i,|w_i|\leqslant 10^6$。

Problem (【CEOI2017】Building Bridges)

一张无向图，初始时点集 V 为 $[1,n]\cap\mathbb{N}$，无边，可以以 w_i 的代价将 i 从点集中删除，若对于 $i<j,i,j\in V$ 满足 $\forall k\in(i,j),k\notin V$，可以以 $(h_i-h_j)^2$ 的代价将 $\{i,j\}$ 加入边集，问使 1 与 n 连通的最小代价。

$1\leqslant n\leqslant 10^5,0\leqslant h_i,|w_i|\leqslant 10^6$。

Solution

我们设 $s_i=\sum_{j=1}^{i}w_i$，f_i 为点 1 与点 i 连通的最小代价，则 $f_1=0$ 且有递推式 $f_i=\min_{j=1}^{i-1}(f_j+(h_i-h_j)^2+s_{i-1}-s_j)$。

Problem (【CEOI2017】Building Bridges)

一张无向图，初始时点集 V 为 $[1,n]\cap\mathbb{N}$，无边，可以以 w_i 的代价将 i 从点集中删除，若对于 $i<j,i,j\in V$ 满足 $\forall k\in(i,j),k\notin V$，可以以 $(h_i-h_j)^2$ 的代价将 $\{i,j\}$ 加入边集，问使 1 与 n 连通的最小代价。

$1\leqslant n\leqslant 10^5,0\leqslant h_i,|w_i|\leqslant 10^6$。

Solution

我们设 $s_i=\sum_{j=1}^{i}w_i$，f_i 为点 1 与点 i 连通的最小代价，则 $f_1=0$ 且有递推式 $f_i=\min_{j=1}^{i-1}(f_j+(h_i-h_j)^2+s_{i-1}-s_j)$。

我们令 $k_i=-2h_i,b_i=f_i+h_i^2-s_i$，则 $f_i=h_i^2+s_{i-1}+\min_{j=1}^{i-1}(k_jh_i+b_j)$。

Problem (【CEOI2017】Building Bridges)

一张无向图，初始时点集 V 为 $[1,n]\cap\mathbb{N}$，无边，可以以 w_i 的代价将 i 从点集中删除，若对于 $i<j,i,j\in V$ 满足 $\forall k\in(i,j),k\notin V$，可以以 $(h_i-h_j)^2$ 的代价将 $\{i,j\}$ 加入边集，问使 1 与 n 连通的最小代价。

$1\leqslant n\leqslant 10^5,0\leqslant h_i,|w_i|\leqslant 10^6$。

Solution

我们设 $s_i=\sum_{j=1}^{i}w_i$，f_i 为点 1 与点 i 连通的最小代价，则 $f_1=0$ 且有递推式 $f_i=\min_{j=1}^{i-1}(f_j+(h_i-h_j)^2+s_{i-1}-s_j)$。

我们令 $k_i=-2h_i,b_i=f_i+h_i^2-s_i$，则 $f_i=h_i^2+s_{i-1}+\min_{j=1}^{i-1}(k_jh_i+b_j)$。

注意到是全局修改，时间复杂度为 $\Theta(n\log n)$。

单侧递归方法无用论，李欣隆在 NOIWC2024 讲课课件中提及的 TCS 理论，指树形数据结构中单侧递归的方法可以在相同的时空复杂度内被在每个结点维护可合并的持久化同类数据结构替代，且后者能维护严格更多的信息，能解决一些单侧递归难以解决的问题。

单侧递归方法无用论，李欣隆在 NOIWC2024 讲课课件中提及的 TCS 理论，指树形数据结构中单侧递归的方法可以在相同的时空复杂度内被在每个结点维护可合并的持久化同类数据结构替代，且后者能维护严格更多的信息，能解决一些单侧递归难以解决的问题。

但在 OI 中，单侧递归在一些问题上具有更小的时空常数与代码难度，值得被学习使用。

Segment Tree Beats，又称吉如一线段树，普及于吉如一的集训队论文，用于解决区间最值操作的相关问题，通过在部分条件下继续递归操作维护信息，使用势能分析保证时间复杂度。

Segment Tree Beats，又称吉如一线段树，普及于吉如一的集训队论文，用于解决区间最值操作的相关问题，通过在部分条件下继续递归操作维护信息，使用势能分析保证时间复杂度。

势能分析，即引入势能函数 Φ 以辅助总时间复杂度分析。例如有 q 次操作，第 i 次操作用时 T_i，结束后势能函数的值为 $\Phi(i)$，特别地，初始势能函数值为 $\Phi(0)$，那么总用时为 $\Phi(0)-\Phi(q)+\sum_{i=1}^{q}(T_i+\Phi(i)-\Phi(i-1))$。在具体问题中往往有更为直观的叙述方式。

Problem (Gorgeous Sequence)

给定一个长度为 n 的数组 a，q 次操作：

1. 给定 $[l,r] \subseteq [1,n], x$，对 $i \in [l,r] \cap \mathbb{N}$ 执行 $a_i \leftarrow \min(a_i, x)$。
2. 给定 $[l,r] \subseteq [1,n]$，求 $\max_{i \in [l,r] \cap \mathbb{N}} a_i$。
3. 给定 $[l,r] \subseteq [1,n]$，求 $\sum_{i \in [l,r] \cap \mathbb{N}} a_i$。

$1 \leqslant n, q \leqslant 10^6$。

Problem (Gorgeous Sequence)

给定一个长度为 n 的数组 a，q 次操作：

1. 给定 $[l,r]\subseteq[1,n],x$，对 $i\in[l,r]\cap\mathbb{N}$ 执行 $a_i\leftarrow\min(a_i,x)$。
2. 给定 $[l,r]\subseteq[1,n]$，求 $\max_{i\in[l,r]\cap\mathbb{N}}a_i$。
3. 给定 $[l,r]\subseteq[1,n]$，求 $\sum_{i\in[l,r]\cap\mathbb{N}}a_i$。

$1\leqslant n,q\leqslant 10^6$。

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 $O(n \log n)$。（即我们将所有结点子树内不同数的个数之和视为势能函数，这部分代价通过势能函数预支了。）

Solution

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 $O(n \log n)$。（即我们将所有结点子树内不同数的个数之和视为势能函数，这部分代价通过势能函数预支了。）

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 $\Theta(n)$，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 $\Theta(n)$，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 $O(n\log n+q\log^2 n)$。

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 $\Theta(n)$，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 $O(n\log n+q\log^2 n)$。

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 $O(n\log n+q\log^2 n)$。

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 $\Theta(n)$，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 $O(n\log n+q\log^2 n)$。

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 $O(n\log n+q\log^2 n)$。

值得注意的是，据说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 $\Theta(n\log n+q\log^2 n)$ 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。

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

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。

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

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。

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

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

注意到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，区间取最大值同理，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

注意到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，区间取最大值同理，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。

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

注意到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，区间取最大值同理，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 $O(n \log n + q \log^2 n)$。

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。

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 $O(n \log n + q \log^2 n)$。

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 $\Theta(n \log v + q(\log n + \log v))$。

我们对每个结点维护对应区间的最大值、次大值、除最大值以外的数与次大值差的最大公约数。

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 $\Theta(n \log v + q(\log n + \log v))$。

我们对每个结点维护对应区间的最大值、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 $a_i (i \in [l, r] \cap \mathbb{N})$ 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 $c_i (i \in [l, r] \cap \mathbb{N})$ 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 $O(n\log^2 n+q\log^3 n)$，但据说没有构造出 $\omega((n+q)\log^2 n)$ 的数据。

Problem (区间后缀最值个数)

给定一个长度为 n 的数列 a，q 次操作：

1. 给定 $p \in [1,n] \cap \mathbb{N}, x \in [1,V] \cap \mathbb{N}$，执行 $a_p \leftarrow x$。
2. 给定 $[l,r] \subseteq [1,n]$，求 $|\{\min\{a_j | j \in [i,r] \cap \mathbb{N}\} | i \in [l,r] \cap \mathbb{N}\}|$。

$1 \leqslant n,q \leqslant 10^6, 1 \leqslant a_i \leqslant V = 10^9$。

后缀查询版本：【UR #19】前进四。

Problem (区间后缀最值个数)

给定一个长度为 n 的数列 a，q 次操作：

1. 给定 $p \in [1,n] \cap \mathbb{N}, x \in [1,V] \cap \mathbb{N}$，执行 $a_p \leftarrow x$。
2. 给定 $[l,r] \subseteq [1,n]$，求 $|\{\min\{a_j | j \in [i,r] \cap \mathbb{N}\} | i \in [l,r] \cap \mathbb{N}\}|$。

$1 \leqslant n,q \leqslant 10^6, 1 \leqslant a_i \leqslant V = 10^9$。

后缀查询版本：【UR #19】前进四。

Solution

我们考虑离线。

Problem (区间后缀最值个数)

给定一个长度为 n 的数列 a，q 次操作：

1. 给定 $p \in [1,n] \cap \mathbb{N}, x \in [1,V] \cap \mathbb{N}$，执行 $a_p \leftarrow x$。
2. 给定 $[l,r] \subseteq [1,n]$，求 $|\{\min\{a_j | j \in [i,r] \cap \mathbb{N}\} | i \in [l,r] \cap \mathbb{N}\}|$。

$1 \leqslant n,q \leqslant 10^6, 1 \leqslant a_i \leqslant V = 10^9$。

后缀查询版本：【UR #19】前进四。

Solution

我们考虑离线。

对于一次询问 $[l,r]$，我们可以用两个变量与一次循环解决询问：初始时 $b = a_r, c = 1$，从大到小枚举 $[l,r) \cap \mathbb{N}$ 中的元素 i，若 $a_i < b$ 则执行 $b \leftarrow a_i, c \leftarrow c+1$。

Solution

我们考虑同时维护 q_2 对 (b_i, c_i)。

Solution

我们考虑同时维护 q_2 对 (b_i, c_i)。

注意到我们可以预处理出每个位置在询问时的值，由若干区间与对应的值组成，每个区间的询问值相同，且所有位置的区间总数不超过 $n + q_1$。

Solution

我们考虑同时维护 q_2 对 (b_i, c_i)。

注意到我们可以预处理出每个位置在询问时的值，由若干区间与对应的值组成，每个区间的询问值相同，且所有位置的区间总数不超过 $n + q_1$。

我们从大到小枚举 $[1, n) \cap \mathbb{N}$ 中的位置 i，对所有 (b_j, c_j) 根据 a_i 的变化进行若干次区间操作。对于 $i \in [1, n] \cap \mathbb{N}$，我们可以在枚举 i 后（对于 $i = n$ 则为枚举 $n - 1$ 前）将 $r_j = i$ 的 (b_j, c_j) 赋值为 $(a_i, 1)$，再查询 $l_j = i$ 的 c_j。

Solution

我们考虑同时维护 q_2 对 (b_i, c_i)。

注意到我们可以预处理出每个位置在询问时的值，由若干区间与对应的值组成，每个区间的询问值相同，且所有位置的区间总数不超过 $n + q_1$。

我们从大到小枚举 $[1, n) \cap \mathbb{N}$ 中的位置 i，对所有 (b_j, c_j) 根据 a_i 的变化进行若干次区间操作。对于 $i \in [1, n] \cap \mathbb{N}$，我们可以在枚举 i 后（对于 $i = n$ 则为枚举 $n - 1$ 前）将 $r_j = i$ 的 (b_j, c_j) 赋值为 $(a_i, 1)$，再查询 $l_j = i$ 的 c_j。

问题转化为至多 $n - 1 + q_1$ 次区间取最小值，q_2 次修改单点为初始情况与查询单点值的历史变化次数。

Solution

我们考虑同时维护 q_2 对 (b_i, c_i)。

注意到我们可以预处理出每个位置在询问时的值，由若干区间与对应的值组成，每个区间的询问值相同，且所有位置的区间总数不超过 $n + q_1$。

我们从大到小枚举 $[1, n) \cap \mathbb{N}$ 中的位置 i，对所有 (b_j, c_j) 根据 a_i 的变化进行若干次区间操作。对于 $i \in [1, n] \cap \mathbb{N}$，我们可以在枚举 i 后（对于 $i = n$ 则为枚举 $n - 1$ 前）将 $r_j = i$ 的 (b_j, c_j) 赋值为 $(a_i, 1)$，再查询 $l_j = i$ 的 c_j。

问题转化为至多 $n - 1 + q_1$ 次区间取最小值，q_2 次修改单点为初始情况与查询单点值的历史变化次数。

时间复杂度 $\Theta((n + q) \log q)$。

Definition 4.1 ((n,s) Davenport-Schinzel 序列)

记一个长度为 m 的序列 $\sigma_1,\sigma_2,\ldots,\sigma_m$ 是一个 (n,s) Davenport-Schinzel 序列（简记为 $\mathrm{DS}(n,s)$ 序列）当且仅当 σ_i 为 1 至 n 的整数，且满足：

1. σ 中相邻两项值不同。
2. 对于任意 $x\neq y$，任何 x,y 交替构成的序列如果是 σ 的子序列，则长度不超过 $s+1$。

注意这里的交替构成不一定要求周期完整出现，即序列长度可以为奇数。

根据 Micha Sharir 与 Pankaj K. Agarwal 的相关论文，有以下定理：

Theorem 4.1

记 $\lambda_s(n)$ 为 $\mathrm{DS}(n,s)$ 序列可能的最长长度，有

$$\lambda_s(n)=\begin{cases} n, & s=1 \\ 2n-1, & s=2 \\ 2n\alpha(n)+O(n), & s=3 \\ \Theta(n2^{\alpha(n)}), & s=4 \\ \Theta(n\alpha(n)2^{\alpha(n)}), & s=5 \\ n2^{\alpha(n)^t/t!+O(\alpha(n)^{t-1})}, & s\geqslant 6, t=\left\lfloor\frac{s-2}{2}\right\rfloor \end{cases}$$

其中 $\alpha(n)$ 是反 Ackermann 函数。

根据 Micha Sharir 与 Pankaj K. Agarwal 的相关论文，有以下定理：

Theorem 4.1

记 $\lambda_s(n)$ 为 $\mathrm{DS}(n,s)$ 序列可能的最长长度，有

$$\lambda_s(n)=\begin{cases} n, & s=1\\ 2n-1, & s=2\\ 2n\alpha(n)+O(n), & s=3\\ \Theta(n2^{\alpha(n)}), & s=4\\ \Theta(n\alpha(n)2^{\alpha(n)}), & s=5\\ n2^{\alpha(n)^t/t!+O(\alpha(n)^{t-1})}, & s\geqslant 6, t=\left\lfloor\frac{s-2}{2}\right\rfloor \end{cases}$$

其中 $\alpha(n)$ 是反 Ackermann 函数。

注意到 $\alpha(n)$ 的增长速度，$\lambda_s(n)$ 对于任意常数 s 是近乎线性的。

KTT，即 Kinetic Tournament Tree，是一种叶子结点信息为一个单自变量函数的线段树，对每个结点维护对应区间中的函数在当前自变量取值下值最大的一个函数的位置与这个位置发生变化的比当前取值大的最小自变量取值。

KTT，即 Kinetic Tournament Tree，是一种叶子结点信息为一个单自变量函数的线段树，对每个结点维护对应区间中的函数在当前自变量取值下值最大的一个函数的位置与这个位置发生变化的比当前取值大的最小自变量取值。

对于一个函数族 $\mathcal{F}$，满足其中的函数定义域与值域均相同，令 $g(\mathcal{F})$ 为 $\mathcal{F}$ 中任意两个函数 f_1, f_2 中 $f_1(x) - f_2(x)$ 在 x 从 $-\infty$ 到 ∞ 的变号次数的最大值（0 不算变号）。我们观察 n 个函数的最大值形成的函数即它们的上包络线，它可以通过一个分段函数的形式来表示，每个段为原先的某个函数，令一个序列表示每个段的原先函数的编号，则这是一个 $\mathrm{DS}(n, g(\mathcal{F}))$ 序列，因此其长度不超过 $\lambda_{g(\mathcal{F})}(n)$。

KTT，即 Kinetic Tournament Tree，是一种叶子结点信息为一个单自变量函数的线段树，对每个结点维护对应区间中的函数在当前自变量取值下值最大的一个函数的位置与这个位置发生变化的比当前取值大的最小自变量取值。

对于一个函数族 $\mathcal{F}$，满足其中的函数定义域与值域均相同，令 $g(\mathcal{F})$ 为 $\mathcal{F}$ 中任意两个函数 f_1, f_2 中 $f_1(x) - f_2(x)$ 在 x 从 $-\infty$ 到 ∞ 的变号次数的最大值（0 不算变号）。我们观察 n 个函数的最大值形成的函数即它们的上包络线，它可以通过一个分段函数的形式来表示，每个段为原先的某个函数，令一个序列表示每个段的原先函数的编号，则这是一个 $\mathrm{DS}(n, g(\mathcal{F}))$ 序列，因此其长度不超过 $\lambda_{g(\mathcal{F})}(n)$。

在该小节的后续部分我们会以 D 为定义域，要求自变量 x 时刻满足 $x \in D$ 并假设 D 与函数族中的函数值域上均定义了全序关系可以进行比较，此外在自变量上需要加法的题目中 D 还满足加法与序关系之间与加法本身的一些基本性质，如 $\mathbb{R}$ 的那些公理（充分但应该不必要）。

我们对一些常见情况进行考察。

我们对一些常见情况进行考察。

一个常见的情况是 $D \subseteq \mathbb{R}$ 上的最高 k 次的多项式。$g\left(\left\{\sum_{i=0}^{k} a_i x^i \middle| \forall i \in [0,k] \cap \mathbb{N}, a_i \in \mathbb{R}\right\}\right) = k$，如 n 个实数域上的一次函数的上包络线最多分为 $\lambda_1(n) = n$ 段。

我们对一些常见情况进行考察。

一个常见的情况是 $D \subseteq \mathbb{R}$ 上的最高 k 次的多项式。$g\left(\left\{\sum_{i=0}^{k} a_i x^i \middle| \forall i \in [0,k] \cap \mathbb{N}, a_i \in \mathbb{R}\right\}\right) = k$，如 n 个实数域上的一次函数的上包络线最多分为 $\lambda_1(n) = n$ 段。

在有的问题中，我们只在一段区间上有值。在该小节中，令 $\mathrm{R}\mathcal{F} = \left\{ \begin{cases} f & (l \leqslant x \leqslant r) \\ -\infty & \text{otherwise} \end{cases} \middle| f \in \mathcal{F}, l, r \in D \right\}$。$g(\mathrm{R}\mathcal{F}) \leqslant g(\mathcal{F}) + 2$，特别地，$D \subseteq \mathbb{R}$ 时 $g\left(\mathrm{R}\left\{\sum_{i=0}^{k} a_i x^i \middle| \forall i \in [0,k] \cap \mathbb{N}, a_i \in \mathbb{R}\right\}\right) = k+2$，如 n 条笛卡尔平面上的线段的上包络线分为不超过 $\lambda_3(n) = \Theta(n\alpha(n))$ 段。

Problem

有一个函数族 $\mathcal{F}$，满足对于 $f_1, f_2 \in \mathcal{F}, x_1, x_2 \in D$，可以在 $\Theta(A)$ 的时间复杂度内求出 $f_1(x_1)$，在 $\Theta(B)$ 的时间复杂度内求出 $\min\{x|\{-1,1\} \subseteq \{\mathrm{sgn}(f_1(x_1+a)-f_2(x_2+a))|0 \leqslant a \leqslant x\}\}$ 与 $a \geqslant 0$ 最小的满足值不为 0 的 $\mathrm{sgn}(f_1(x_1+a)-f_2(x_2+a))$，给定 n 个其中的函数 $f_1, f_2, \ldots, f_n$，与 n 个自变量 $x_1, x_2, \ldots, x_n$，q 次操作：

1. 单点修改：给定 $p \in [1,n] \cap \mathbb{N}, h \in \mathcal{F}$，执行 $f_p \leftarrow h$。
2. 全局加自变量：给定 $\Delta x \geqslant 0$，对于 $i \in [1,n] \cap \mathbb{N}$，$x_i \leftarrow x_i + \Delta x$。
3. 区间最大值：给定 $[l,r] \subseteq [1,n]$，查询 $\max_{i \in [l,r] \cap \mathbb{N}} f_i(x_i)$。

要求时间复杂度 $O(\lambda_{g(\mathrm{R}\,\mathcal{F})}(n+q_1)\log n(\log n + B) + q_2 + q_3 A \log n)$。

Solution

我们对每个结点维护对应区间中的函数在当前自变量时最大的一个函数的位置与该函数不是区间内最大值的最小自变量增量（如果有多个函数取得最大值，选取该增量最大的任意一个），可以在 $\Theta(B)$ 的时间复杂度内上传信息，因此建树时间复杂度 $\Theta(nB)$。

Solution

我们对每个结点维护对应区间中的函数在当前自变量时最大的一个函数的位置与该函数不是区间内最大值的最小自变量增量（如果有多个函数取得最大值，选取该增量最大的任意一个），可以在 $\Theta(B)$ 的时间复杂度内上传信息，因此建树时间复杂度 $\Theta(nB)$。

查询时我们直接对定位的结点集中维护的函数取最大值即可，单次时间复杂度严格 $\Theta(A \log n)$，如果比较两个函数的值比求值复杂度优，可以先比较出最大的一个再求值。

Solution

我们对每个结点维护对应区间中的函数在当前自变量时最大的一个函数的位置与该函数不是区间内最大值的最小自变量增量（如果有多个函数取得最大值，选取该增量最大的任意一个），可以在 $\Theta(B)$ 的时间复杂度内上传信息，因此建树时间复杂度 $\Theta(nB)$。

查询时我们直接对定位的结点集中维护的函数取最大值即可，单次时间复杂度严格 $\Theta(A\log n)$，如果比较两个函数的值比求值复杂度优，可以先比较出最大的一个再求值。

对于单点修改，我们可以递归到叶子进行暴力修改并上传信息，单次时间复杂度严格 $\Theta(B\log n)$。

Solution

对于全局加自变量，我们对每个结点维护子树内加自变量的懒标记。考虑对当前结点的子树加自变量的过程，如果取到最大值的函数不变，则直接打上懒标记返回；如果两个儿子结点维护的函数不变，则重新上传信息并打上懒标记返回；否则递归两个儿子并重新上传信息。

Solution

对于全局加自变量，我们对每个结点维护子树内加自变量的懒标记。考虑对当前结点的子树加自变量的过程，如果取到最大值的函数不变，则直接打上懒标记返回；如果两个儿子结点维护的函数不变，则重新上传信息并打上懒标记返回；否则递归两个儿子并重新上传信息。

我们考虑一个结点会被递归的次数。

Solution

对于全局加自变量，我们对每个结点维护子树内加自变量的懒标记。考虑对当前结点的子树加自变量的过程，如果取到最大值的函数不变，则直接打上懒标记返回；如果两个儿子结点维护的函数不变，则重新上传信息并打上懒标记返回；否则递归两个儿子并重新上传信息。

我们考虑一个结点会被递归的次数。

对于不进行单点修改的情况，显然每个结点会作为递归的终点时当前自变量增量使该结点维护的函数发生了变化，因此最多 $\lambda_{g(\mathcal{F})}(l_u)$ 次，其中 l_u 为结点 u 的对应区间长度。此外，只有该节点维护的函数发生了变化，回溯时父亲结点上传信息需要 $\Theta(B)$ 重新计算两个函数差值下次变号的自变量增量。

Solution

因此在不进行单点修改的情况，全局加自变量操作的总时间复杂度为 $O(q_2 + \sum(\log n - \log l_u + B)\lambda_{g(\mathcal{F})}(l_u))$，由主定理可知为 $O(q_2 + \lambda_{g(\mathcal{F})}(n)\log n(\log n + B))$。

Solution

因此在不进行单点修改的情况，全局加自变量操作的总时间复杂度为 $O(q_2 + \sum(\log n - \log l_u + B)\lambda_{g(\mathcal{F})}(l_u))$，由主定理可知为 $O(q_2 + \lambda_{g(\mathcal{F})}(n)\log n(\log n + B))$。

注意到我们可以维护每个结点需要改变维护函数的最小自变量增量，每次直接选取该值最小的结点从下到上上传信息，此时每个结点只需维护一个函数无需维护子树内改变维护函数的最小自变量增量，因此某次上传信息时发现维护函数不变即可直接终止。如果我们可以方便地建立一个可以 $\Theta(C)$ 时间计算的 D 到 $[1, V] \cap \mathbb{N}$ 的保持序关系的映射，我们可以使用后面的 vEB 树将该操作总时间复杂度优化至 $O(q_2 + \lambda_{g(\mathcal{F})}(n)\log n(\log\log V + B + C))$。（虽然由于只维护区间当前最优函数可能不算 KTT。）

Solution

因此在不进行单点修改的情况，全局加自变量操作的总时间复杂度为 $O(q_2 + \sum(\log n - \log l_u + B)\lambda_{g(\mathcal{F})}(l_u))$，由主定理可知为 $O(q_2 + \lambda_{g(\mathcal{F})}(n)\log n(\log n + B))$。

注意到我们可以维护每个结点需要改变维护函数的最小自变量增量，每次直接选取该值最小的结点从下到上上传信息，此时每个结点只需维护一个函数无需维护子树内改变维护函数的最小自变量增量，因此某次上传信息时发现维护函数不变即可直接终止。如果我们可以方便地建立一个可以 $\Theta(C)$ 时间计算的 D 到 $[1, V] \cap \mathbb{N}$ 的保持序关系的映射，我们可以使用后面的 vEB 树将该操作总时间复杂度优化至 $O(q_2 + \lambda_{g(\mathcal{F})}(n)\log n(\log\log V + B + C))$。（虽然由于只维护区间当前最优函数可能不算 KTT。）

注意不单点修改可以做得更好，见线段树分治小节。

Solution

对于有单点修改的情况，我们观察一个位置的函数在每次加自变量时的取值，为之前一次修改或初始给出（之前没有修改）的函数的值，因此我们在分析该复杂度时可以将一个位置的函数视为一个分段次数最多为修改次数加一的分段函数，这也与至多修改次数加一个 $\mathrm{R}\mathcal{F}$ 上的函数的最大值等价。与上面的复杂度分析类似，时间复杂度为 $O(q_2+\sum(\log n-\log l_u+B)\lambda_{g(\mathrm{R}\mathcal{F})}(l_u+k_u))$，其中 k_u 为结点 u 子树内的单点修改次数。我们可以先对深度相同的结点求和，结果为 $O(q_2+\lambda_{g(\mathrm{R}\mathcal{F})}(n+q_1)\log n(\log n+B))$。

Solution

对于有单点修改的情况，我们观察一个位置的函数在每次加自变量时的取值，为之前一次修改或初始给出（之前没有修改）的函数的值，因此我们在分析该复杂度时可以将一个位置的函数视为一个分段次数最多为修改次数加一的分段函数，这也与至多修改次数加一个 $\mathrm{R}\mathcal{F}$ 上的函数的最大值等价。与上面的复杂度分析类似，时间复杂度为 $O(q_2+\sum(\log n-\log l_u+B)\lambda_{g(\mathrm{R}\mathcal{F})}(l_u+k_u))$，其中 k_u 为结点 u 子树内的单点修改次数。我们可以先对深度相同的结点求和，结果为 $O(q_2+\lambda_{g(\mathrm{R}\mathcal{F})}(n+q_1)\log n(\log n+B))$。

总时间复杂度 $O(\lambda_{g(\mathrm{R}\mathcal{F})}(n+q_1)\log n(\log n+B)+q_2+q_3A\log n)$。

Solution

对于有单点修改的情况，我们观察一个位置的函数在每次加自变量时的取值，为之前一次修改或初始给出（之前没有修改）的函数的值，因此我们在分析该复杂度时可以将一个位置的函数视为一个分段次数最多为修改次数加一的分段函数，这也与至多修改次数加一个 $\mathrm{R}\mathcal{F}$ 上的函数的最大值等价。与上面的复杂度分析类似，时间复杂度为 $O(q_2+\sum(\log n-\log l_u+B)\lambda_{g(\mathrm{R}\mathcal{F})}(l_u+k_u))$，其中 k_u 为结点 u 子树内的单点修改次数。我们可以先对深度相同的结点求和，结果为 $O(q_2+\lambda_{g(\mathrm{R}\mathcal{F})}(n+q_1)\log n(\log n+B))$。

总时间复杂度 $O(\lambda_{g(\mathrm{R}\mathcal{F})}(n+q_1)\log n(\log n+B)+q_2+q_3A\log n)$。

同样可以用 vEB 树维护每个结点维护函数改变的最小自变量增量，时间复杂度 $O(\lambda_{g(\mathrm{R}\mathcal{F})}(n+q_1)\log n(\log\log V+B+C)+q_2+q_3A\log n)$。

Problem

有一个满足 $g(\mathcal{F})=1$ 的函数族 $\mathcal{F}$，满足 $\forall f(x)\in\mathcal{F},\Delta x\in D,f(x-\Delta x)\in\mathcal{F}$ 且对于 $f_1,f_2\in\mathcal{F},x_1,x_2\in D$，可以在 $\Theta(A)$ 的时间复杂度内求出 $f_1(x_1)$，在 $\Theta(B)$ 的时间复杂度内求出 $\min\{x|\{-1,1\}\subseteq\{\operatorname{sgn}(f_1(x_1+a)-f_2(x_2+a))|0\leqslant a\leqslant x\}\}$ 与 $a\geqslant 0$ 最小的满足值不为 0 的 $\operatorname{sgn}(f_1(x_1+a)-f_2(x_2+a))$，给定 n 个其中的函数 $f_1,f_2,\ldots,f_n$，与 n 个自变量 $x_1,x_2,\ldots,x_n$，q 次操作：

1. 单点修改：给定 $p\in[1,n]\cap\mathbb{N},h\in\mathcal{F}$，执行 $f_p\leftarrow h$。
2. 区间加自变量：给定 $[l,r]\subseteq[1,n],\Delta x\geqslant 0$，对于 $i\in[l,r]\cap\mathbb{N}$，$x_i\leftarrow x_i+\Delta x$。
3. 区间最大值：给定 $[l,r]\subseteq[1,n]$，查询 $\max_{i\in[l,r]\cap\mathbb{N}}f_i(x_i)$。

要求时间复杂度 $O((n+(q_1+q_2)\log n)\log n(\log n+B)+q_3A\log n)$。

Solution

除区间加自变量先定位给定区间的结点集再从结点集中结点而非根结点开始递归以外算法流程均与全局加的版本一致，我们只需分析新的区间加自变量操作的复杂度即可。

Solution

除区间加自变量先定位给定区间的结点集再从结点集中结点而非根结点开始递归以外算法流程均与全局加的版本一致，我们只需分析新的区间加自变量操作的复杂度即可。

我们在每个非叶结点上放一个势能函数，有势能当且仅当该结点的左右儿子维护的函数差值会在自变量继续增大时发生变号。我们考虑先考察一次区间加自变量操作势能发生的变化。

Solution

除区间加自变量先定位给定区间的结点集再从结点集中结点而非根结点开始递归以外算法流程均与全局加的版本一致，我们只需分析新的区间加自变量操作的复杂度即可。

我们在每个非叶结点上放一个势能函数，有势能当且仅当该结点的左右儿子维护的函数差值会在自变量继续增大时发生变号。我们考虑先考察一次区间加自变量操作势能发生的变化。

$\mathcal{F}$ 对左右平移封闭保证了任意区间自变量增加操作时两个函数在之后等量增加自变量时差值至多变号一次。

Solution

除区间加自变量先定位给定区间的结点集再从结点集中结点而非根结点开始递归以外算法流程均与全局加的版本一致，我们只需分析新的区间加自变量操作的复杂度即可。

我们在每个非叶结点上放一个势能函数，有势能当且仅当该结点的左右儿子维护的函数差值会在自变量继续增大时发生变号。我们考虑先考察一次区间加自变量操作势能发生的变化。

$\mathcal{F}$ 对左右平移封闭保证了任意区间自变量增加操作时两个函数在之后等量增加自变量时差值至多变号一次。

定位结点集的祖先结点（不含自身）的势能可能会由无变有，我们接下来考虑每个定位的结点集中的结点向下递归时的变化。

Solution

每次递归会以若干有势能的结点作为势能终点，并将其变为无势能，我们观察其祖先结点势能的变高情况。

Solution

每次递归会以若干有势能的结点作为势能终点，并将其变为无势能，我们观察其祖先结点势能的变高情况。

我们可以按后序遍历线段树的顺序对每个 $[u \in P]$（其中 P 为与左儿子维护函数相同的结点的集合）发生变化的结点 u 求它到它与下一个结点的最近公共祖先（不含）的链上（对于最后一个结点为到根链）的势能变化，显然其余结点势能不变化，一个结点的势能变化不会被计算两次。即只需在分析父亲结点的势能变化时考虑当前结点的兄弟未被递归且父亲结点的左右大小关系不发生变化的情况。

Solution

每次递归会以若干有势能的结点作为势能终点，并将其变为无势能，我们观察其祖先结点势能的变高情况。

我们可以按后序遍历线段树的顺序对每个 $[u \in P]$（其中 P 为与左儿子维护函数相同的结点的集合）发生变化的结点 u 求它到它与下一个结点的最近公共祖先（不含）的链上（对于最后一个结点为到根链）的势能变化，显然其余结点势能不变化，一个结点的势能变化不会被计算两次。即只需在分析父亲结点的势能变化时考虑当前结点的兄弟未被递归且父亲结点的左右大小关系不发生变化的情况。

我们从下往上考察若当前结点维护的函数发生了变化，其父亲结点维护函数与势能的变化，若维护函数发生变化将父亲结点作为新的当前结点继续考察即可。若父亲结点有势能则势能不会变高，我们下面只需考虑父亲结点无势能的情况。

Solution

对于当前结点与父亲结点维护函数相同的情况，当前结点原先维护的函数从原先的自变量任意增加都不会低于兄弟结点维护的函数，注意到当前结点新的函数从当前自变量任意增加不低于原先的函数，由全序关系的传递性可知势能也不会变高。

Solution

对于当前结点与父亲结点维护函数相同的情况，当前结点原先维护的函数从原先的自变量任意增加都不会低于兄弟结点维护的函数，注意到当前结点新的函数从当前自变量任意增加不低于原先的函数，由全序关系的传递性可知势能也不会变高。

对于兄弟结点与父亲结点维护函数相同的情况，由传递性我们可知当前结点新的函数比兄弟结点的函数在原先的自变量下小，若当前结点新的函数作为父亲结点新的函数则在中间发生了变号，父亲结点势能不会增加，否则父亲结点势能恢复但由于父亲结点维护的函数没有变化，其深度更小的祖先结点的势能不会变化。

Solution

对于当前结点与父亲结点维护函数相同的情况，当前结点原先维护的函数从原先的自变量任意增加都不会低于兄弟结点维护的函数，注意到当前结点新的函数从当前自变量任意增加不低于原先的函数，由全序关系的传递性可知势能也不会变高。

对于兄弟结点与父亲结点维护函数相同的情况，由传递性我们可知当前结点新的函数比兄弟结点的函数在原先的自变量下小，若当前结点新的函数作为父亲结点新的函数则在中间发生了变号，父亲结点势能不会增加，否则父亲结点势能恢复但由于父亲结点维护的函数没有变化，其深度更小的祖先结点的势能不会变化。

对于 $[u \in P]$ 发生变化且原先没有势能的结点，其至少有一个儿子结点维护的函数发生了变化，我们可以视作先放回再取出。

Solution

综上所述，每个 $[u \in P]$ 发生变化的点会取出势能，在其最多一个祖先结点会放回势能，且只有这两个结点之间的链需要 $\Theta(B)$ 上传信息，其余势能不会变高。

Solution

综上所述，每个 $[u \in P]$ 发生变化的点会取出势能，在其最多一个祖先结点会放回势能，且只有这两个结点之间的链需要 $\Theta(B)$ 上传信息，其余势能不会变高。

我们令每个结点的势能大小为 $d(d+B)$，其中 d 为结点深度，则单次操作的均摊时间复杂度为 $O(\log^2 n(\log n + B))$。

Solution

综上所述，每个 $[u \in P]$ 发生变化的点会取出势能，在其最多一个祖先结点会放回势能，且只有这两个结点之间的链需要 $\Theta(B)$ 上传信息，其余势能不会变高。

我们令每个结点的势能大小为 $d(d+B)$，其中 d 为结点深度，则单次操作的均摊时间复杂度为 $O(\log^2 n(\log n + B))$。

对于查询操作，势能不会变化。对于单点修改操作，修改的叶子结点到根链可能会恢复势能，额外需要 $O(\log^2 n(\log n + B))$ 的时间复杂度。

Solution

综上所述，每个 $[u \in P]$ 发生变化的点会取出势能，在其最多一个祖先结点会放回势能，且只有这两个结点之间的链需要 $\Theta(B)$ 上传信息，其余势能不会变高。

我们令每个结点的势能大小为 $d(d+B)$，其中 d 为结点深度，则单次操作的均摊时间复杂度为 $O(\log^2 n(\log n + B))$。

对于查询操作，势能不会变化。对于单点修改操作，修改的叶子结点到根链可能会恢复势能，额外需要 $O(\log^2 n(\log n + B))$ 的时间复杂度。

初始势能之和为 $\Theta(n \log n(\log n + B))$，总时间复杂度 $O((n + (q_1 + q_2)\log n)\log n(\log n + B) + q_3 A \log n)$。

Problem (包含两类区间修改的序列最值问题)

给定长度为 n 的数列 k, b，q 次操作：

1. 给定 $[l, r] \subseteq [1, n], x \in \mathbb{R}$，对于 $i \in [l, r] \cap \mathbb{N}$，执行 $b_i \leftarrow k_i x + b_i$。
2. 给定 $[l, r] \subseteq [1, n], c, k', b' \in \mathbb{R}$，对 $i \in [l, r] \cap \mathbb{N}$，执行 $k_i \leftarrow ck_i + k', b_i \leftarrow cb_i + b'$。
3. 给定 $[l, r] \subseteq [1, n]$，求 $\max_{i=l}^{r} b_i$。

$x \geqslant 0$，要求时间复杂度 $O(n \log^2 n + (q_1 + q_2) \log^3 n + q_3 \log n)$。

Problem (包含两类区间修改的序列最值问题)

给定长度为 n 的数列 k, b，q 次操作：

1. 给定 $[l, r] \subseteq [1, n], x \in \mathbb{R}$，对于 $i \in [l, r] \cap \mathbb{N}$，执行 $b_i \leftarrow k_i x + b_i$。
2. 给定 $[l, r] \subseteq [1, n], c, k', b' \in \mathbb{R}$，对 $i \in [l, r] \cap \mathbb{N}$，执行 $k_i \leftarrow ck_i + k', b_i \leftarrow cb_i + b'$。
3. 给定 $[l, r] \subseteq [1, n]$，求 $\max_{i=l}^{r} b_i$。

$x \geqslant 0$，要求时间复杂度 $O(n \log^2 n + (q_1 + q_2) \log^3 n + q_3 \log n)$。

Solution

令 $f_i(x) = k_i x + b_i$，我们考虑用 KTT 维护 $f_i(x)$ 的区间上下包络线，操作一即区间加自变量，操作三即区间查询最大值。

Solution

我们现在维护了 $f_i(x)$ 的区间最大值最小值与其发生改变的最小自变量增量，以及区间加自变量的懒标记。

Solution

我们现在维护了 $f_i(x)$ 的区间最大值最小值与其发生改变的最小自变量增量，以及区间加自变量的懒标记。

对于操作二我们先定位结点集，再对结点集中的结点进行修改操作。

Solution

我们现在维护了 $f_i(x)$ 的区间最大值最小值与其发生改变的最小自变量增量，以及区间加自变量的懒标记。

对于操作二我们先定位结点集，再对结点集中的结点进行修改操作。

经过简单的代数整理，我们可以发现两个函数经过相同的修改操作后交点不变，即下次最大最小值发生改变的最小自变量增量不变（除 $c = 0$ 这一简单情况）。当 $c < 0$ 时，我们需要对取到最大最小值的函数进行互换，否则取到最大最小值的函数不变。我们发现这些变化都是易于用懒标记维护的，注意与区间加自变量的懒标记的联系。

Solution

我们现在维护了 $f_i(x)$ 的区间最大值最小值与其发生改变的最小自变量增量，以及区间加自变量的懒标记。

对于操作二我们先定位结点集，再对结点集中的结点进行修改操作。

经过简单的代数整理，我们可以发现两个函数经过相同的修改操作后交点不变，即下次最大最小值发生改变的最小自变量增量不变（除 $c = 0$ 这一简单情况）。当 $c < 0$ 时，我们需要对取到最大最小值的函数进行互换，否则取到最大最小值的函数不变。我们发现这些变化都是易于用懒标记维护的，注意与区间加自变量的懒标记的联系。

沿用区间加自变量的势能分析，总时间复杂度 $O(n \log^2 n + (q_1 + q_2) \log^3 n + q_3 \log n)$。

Problem (EI 的第六分块)

给定一个长度为 n 的序列 a，q 次操作：

1. 给定 $[l,r]\subseteq[1,n],x\geqslant 0$，对于 $i\in[l,r]\cap\mathbb{N}$，执行 $a_i\leftarrow a_i+x$。
2. 给定 $[l,r]\subseteq[1,n]$，查询 $\max\left(0,\max_{[i,j]\subseteq[l,r]}\sum_{k\in[i,j]\cap\mathbb{N}}a_k\right)$。

$1\leqslant n,q\leqslant 4\times 10^5,|a_i|\leqslant 10^9,0\leqslant x\leqslant 10^6$。

相比第六分块【Ynoi2018】末日时在做什么？有没有空？可以来拯救吗？增加了 $x\geqslant 0$ 的限制并扩大了数据范围。

Solution

对于线段树的一个结点 u，令 lsn_u 为其左儿子，rsn_u 为其右儿子，$\mathrm{sum}_u(x) = \mathrm{sum}_{\mathrm{lsn}_u}(x) + \mathrm{sum}_{\mathrm{rsn}_u}(x)$ 表示它对应区间加 x 后的对应区间和，$\mathrm{pre}_u(x) = \max\left(\mathrm{pre}_{\mathrm{lsn}_u}(x), \mathrm{pre}_{\mathrm{rsn}_u}(x) + \mathrm{sum}_{\mathrm{lsn}_u}(x)\right)$ 表示它对应区间加 x 后的最大前缀和，对称地，$\mathrm{suf}_u(x)$ 表示其加 x 后的最大后缀和，$\mathrm{ans}_u(x) = \max\left(\mathrm{ans}_{\mathrm{lsn}_u}(x), \mathrm{ans}_{\mathrm{rsn}_u}(x), \mathrm{suf}_{\mathrm{lsn}_u}(x) + \mathrm{pre}_{\mathrm{rsn}_u}(x)\right)$ 表示它对应区间加 x 后最大子段和。

Solution

对于线段树的一个结点 u，令 lsn_u 为其左儿子，rsn_u 为其右儿子，$\mathrm{sum}_u(x) = \mathrm{sum}_{\mathrm{lsn}_u}(x) + \mathrm{sum}_{\mathrm{rsn}_u}(x)$ 表示它对应区间加 x 后的对应区间和，$\mathrm{pre}_u(x) = \max\left(\mathrm{pre}_{\mathrm{lsn}_u}(x), \mathrm{pre}_{\mathrm{rsn}_u}(x) + \mathrm{sum}_{\mathrm{lsn}_u}(x)\right)$ 表示它对应区间加 x 后的最大前缀和，对称地，$\mathrm{suf}_u(x)$ 表示其加 x 后的最大后缀和，$\mathrm{ans}_u(x) = \max\left(\mathrm{ans}_{\mathrm{lsn}_u}(x), \mathrm{ans}_{\mathrm{rsn}_u}(x), \mathrm{suf}_{\mathrm{lsn}_u}(x) + \mathrm{pre}_{\mathrm{rsn}_u}(x)\right)$ 表示它对应区间加 x 后最大子段和。

考虑 KTT，我们维护 $\mathrm{sum}_u(x)$ 与 $\mathrm{pre}_u(x), \mathrm{suf}_u(x), \mathrm{ans}_u(x)$ 在 $x = 0$ 时取最大值的一项及这个哪一项发生改变的最小的 x。每次定位结点集后若维护的函数发生改变则递归否则直接打上懒标记。

Solution

考虑分析递归的均摊复杂度，我们在每个结点上放两个势能函数表示 $\mathrm{ans}(x)$ 当前取到最大值的一项与另外两项若 x 继续增大差值是否会发生变号。

Solution

考虑分析递归的均摊复杂度，我们在每个结点上放两个势能函数表示 $\mathrm{ans}(x)$ 当前取到最大值的一项与另外两项若 x 继续增大差值是否会发生变号。

与维护函数序列的区间加自变量类似，我们可以证明令结点有势能时的势能大小为深度的平方，递归修改 $\mathrm{ans}(x)$ 函数过程可以均摊分析，区间加均摊时间复杂度 $O(\log^3 n)$。

Solution

考虑分析递归的均摊复杂度，我们在每个结点上放两个势能函数表示 $\mathrm{ans}(x)$ 当前取到最大值的一项与另外两项若 x 继续增大差值是否会发生变号。

与维护函数序列的区间加自变量类似，我们可以证明令结点有势能时的势能大小为深度的平方，递归修改 $\mathrm{ans}(x)$ 函数过程可以均摊分析，区间加均摊时间复杂度 $O(\log^3 n)$。

注意到最大前缀和决策点只会从左往右的单调性，每个结点维护的最大前缀和函数改为与其右儿子结点一致后不会再改为与其左儿子一致，最大后缀和左右对称同理。

Solution

考虑分析递归的均摊复杂度，我们在每个结点上放两个势能函数表示 $\mathrm{ans}(x)$ 当前取到最大值的一项与另外两项若 x 继续增大差值是否会发生变号。

与维护函数序列的区间加自变量类似，我们可以证明令结点有势能时的势能大小为深度的平方，递归修改 $\mathrm{ans}(x)$ 函数过程可以均摊分析，区间加均摊时间复杂度 $O(\log^3 n)$。

注意到最大前缀和决策点只会从左往右的单调性，每个结点维护的最大前缀和函数改为与其右儿子结点一致后不会再改为与其左儿子一致，最大后缀和左右对称同理。

因此修改最大前缀和与最大后缀和递归时的时间复杂度至多为 $O(n\log^2 n)$，会增加至多 $O(n\log^3 n)$ 的势能。

Solution

考虑分析递归的均摊复杂度，我们在每个结点上放两个势能函数表示 $\mathrm{ans}(x)$ 当前取到最大值的一项与另外两项若 x 继续增大差值是否会发生变号。

与维护函数序列的区间加自变量类似，我们可以证明令结点有势能时的势能大小为深度的平方，递归修改 $\mathrm{ans}(x)$ 函数过程可以均摊分析，区间加均摊时间复杂度 $O(\log^3 n)$。

注意到最大前缀和决策点只会从左往右的单调性，每个结点维护的最大前缀和函数改为与其右儿子结点一致后不会再改为与其左儿子一致，最大后缀和左右对称同理。

因此修改最大前缀和与最大后缀和递归时的时间复杂度至多为 $O(n\log^2 n)$，会增加至多 $O(n\log^3 n)$ 的势能。

总时间复杂度 $O((n+q_1)\log^3 n + q_2 \log n)$。

事实上，区间加的数非负对于算法时间复杂度为满足 $c < 1.5$ 的 $O(n^c)$ 是相当重要的，李欣隆给出了 $(\max, +)$ 矩阵乘法到第六分块的一个规约。

事实上，区间加的数非负对于算法时间复杂度为满足 $c < 1.5$ 的 $O(n^c)$ 是相当重要的，李欣隆给出了 $(\max, +)$ 矩阵乘法到第六分块的一个规约。

Theorem 4.2

给定一个 $p \times r$ 的实矩阵 B，q 次询问给定一个 $1 \times p$ 的实向量 A，对于 $k \in [1, r] \cap \mathbb{N}$，查询 $\max_{j=1}^{p}(A_{1,j} + B_{j,k})$，可以通过一个长度为 $\Theta(pr)$ 的序列进行 $\Theta(pq)$ 次单点修改、全局加非负数、撤销前一次修改、求全局最大子段和解决。

事实上，区间加的数非负对于算法时间复杂度为满足 $c < 1.5$ 的 $O(n^c)$ 是相当重要的，李欣隆给出了 $(\max, +)$ 矩阵乘法到第六分块的一个规约。

Theorem 4.2

给定一个 $p \times r$ 的实矩阵 B，q 次询问给定一个 $1 \times p$ 的实向量 A，对于 $k \in [1, r] \cap \mathbb{N}$，查询 $\max_{j=1}^{p}(A_{1,j} + B_{j,k})$，可以通过一个长度为 $\Theta(pr)$ 的序列进行 $\Theta(pq)$ 次单点修改、全局加非负数、撤销前一次修改、求全局最大子段和解决。

Proof

我们不妨设 A 的元素值非负，B 的元素属于 $[0, V]$。元素非负显然可以通过 A 的一行或 B 的一列同时加上相同数再在答案中减去得到。

Proof

我们考虑建立一个由 p 个块构成的序列 a，相邻块之间用 $-\infty$ 分隔，每个块长度为 r。

Proof

我们考虑建立一个由 p 个块构成的序列 a，相邻块之间用 $-\infty$ 分隔，每个块长度为 r。

我们令第 j 个块的第 k 个元素初始为 $b_{j,k}-[k>1]b_{j,k-1}-(2k-3)\,V$，容易说明在进行 $k-1$ 次全局加 $2V$ 后前 k 个元素为非负数，其余元素为非正数，这个块的最大子段和即为前 k 个元素之和。

Proof

我们考虑建立一个由 p 个块构成的序列 a，相邻块之间用 $-\infty$ 分隔，每个块长度为 r。

我们令第 j 个块的第 k 个元素初始为 $b_{j,k}-[k>1]b_{j,k-1}-(2k-3)\,V$，容易说明在进行 $k-1$ 次全局加 $2V$ 后前 k 个元素为非负数，其余元素为非正数，这个块的最大子段和即为前 k 个元素之和。

由于块间由 $-\infty$ 分隔，我们可以通过一次询问求出所有块的最大前缀和。

Proof

我们考虑建立一个由 p 个块构成的序列 a，相邻块之间用 $-\infty$ 分隔，每个块长度为 r。

我们令第 j 个块的第 k 个元素初始为 $b_{j,k}-[k>1]b_{j,k-1}-(2k-3)\,V$，容易说明在进行 $k-1$ 次全局加 $2V$ 后前 k 个元素为非负数，其余元素为非正数，这个块的最大子段和即为前 k 个元素之和。

由于块间由 $-\infty$ 分隔，我们可以通过一次询问求出所有块的最大前缀和。

我们只需在每次询问的开头将第 j 个块的第一个元素加上 $A_{1,j}$，进行 $k-1$ 次全局加 $2V$ 并进行 k 次全局最大子段和的查询即可。在询问结束后撤销所有操作。

注意到单点修改是容易撤销的，区间加正数可以通过区间加相反数进行撤销，矩阵 A, B 乘积的第 i 行与 A 的第 i 行与 B 的乘积相同，APSP 假设给出了 $(\max, +)$ 矩阵乘法的下界，我们有以下推论：

注意到单点修改是容易撤销的，区间加正数可以通过区间加相反数进行撤销，矩阵 A, B 乘积的第 i 行与 A 的第 i 行与 B 的乘积相同，APSP 假设给出了 $(\max, +)$ 矩阵乘法的下界，我们有以下推论：

Corollary 4.1

若 APSP 假设成立，则不存在常数 $c < 1.5$ 使得以下问题可以在 $O(n^c)$ 的时间复杂度内解决：

Problem

给定一个长度为 n 的数列 a，$q = \Theta(n)$ 次操作：

1. 给定 x，对 $i \in [1, n] \cap \mathbb{N}$ 执行 $a_i \leftarrow a_i + x$。
2. 给定 $p \in [1, n] \cap \mathbb{N}, x$，执行 $a_p \leftarrow x$。
3. 查询 $\max_{[l,r] \subseteq [1,n]} \sum_{i \in [l,r] \cap \mathbb{N}} a_i$。

Problem (【SNOI2020】区间和)

给定一个长度为 n 的数组 a，q 次操作：区间取最大值、求区间最大子段和（子段可为空）。

$1 \leqslant n \leqslant 10^5, 1 \leqslant q \leqslant 2 \times 10^5$，在任意时刻 $|a_i| \leqslant 10^9$。

Problem (【SNOI2020】区间和)

给定一个长度为 n 的数组 a，q 次操作：区间取最大值、求区间最大子段和（子段可为空）。

$1 \leqslant n \leqslant 10^5, 1 \leqslant q \leqslant 2 \times 10^5$，在任意时刻 $|a_i| \leqslant 10^9$。

Solution

与 EI 的第六分块类似，我们对每个结点维护对应区间最小值加上 x 后对应区间和、最大前后缀和与最大子段和的函数并采取同样的势能分析。用 Segment Tree Beats 的技巧对每个结点维护最小值与次小值。

Problem (【SNOI2020】区间和)

给定一个长度为 n 的数组 a，q 次操作：区间取最大值、求区间最大子段和（子段可为空）。

$1 \leqslant n \leqslant 10^5, 1 \leqslant q \leqslant 2 \times 10^5$，在任意时刻 $|a_i| \leqslant 10^9$。

Solution

与 EI 的第六分块类似，我们对每个结点维护对应区间最小值加上 x 后对应区间和、最大前后缀和与最大子段和的函数并采取同样的势能分析。用 Segment Tree Beats 的技巧对每个结点维护最小值与次小值。

注意到一次修改的均摊时间复杂度为严格的 $\Theta(\log n)$ 加上区间取最值递归的结点与定位结点集到根链并恢复的势能，共 $\Theta((n+q_1)\log n)$ 个结点。

Problem (【SNOI2020】区间和)

给定一个长度为 n 的数组 a，q 次操作：区间取最大值、求区间最大子段和（子段可为空）。

$1 \leqslant n \leqslant 10^5, 1 \leqslant q \leqslant 2 \times 10^5$，在任意时刻 $|a_i| \leqslant 10^9$。

Solution

与 EI 的第六分块类似，我们对每个结点维护对应区间最小值加上 x 后对应区间和、最大前后缀和与最大子段和的函数并采取同样的势能分析。用 Segment Tree Beats 的技巧对每个结点维护最小值与次小值。

注意到一次修改的均摊时间复杂度为严格的 $\Theta(\log n)$ 加上区间取最值递归的结点与定位结点集到根链并恢复的势能，共 $\Theta((n+q_1)\log n)$ 个结点。

总时间复杂度 $\Theta((n+q_1)\log^3 n + q_2 \log n)$。

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

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

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 (Lucky Array)

定义仅含数码 4 和数码 7 的数为幸运数，给定一个长度为 n 的数组 a，q 次操作：区间加正整数、求区间内有多少数是幸运数。

$1 \leqslant n, m \leqslant 10^5$，保证过程中 a_i 不超过 $v = 10^4$。

Problem (Lucky Array)

定义仅含数码 4 和数码 7 的数为幸运数，给定一个长度为 n 的数组 a，q 次操作：区间加正整数、求区间内有多少数是幸运数。

$1 \leqslant n, m \leqslant 10^5$，保证过程中 a_i 不超过 $v = 10^4$。

Solution

令 $\leqslant v$ 的幸运数个数为 k，注意到 $k = 30$。

Problem (Lucky Array)

定义仅含数码 4 和数码 7 的数为幸运数，给定一个长度为 n 的数组 a，q 次操作：区间加正整数、求区间内有多少数是幸运数。

$1 \leqslant n, m \leqslant 10^5$，保证过程中 a_i 不超过 $v = 10^4$。

Solution

令 $\leqslant v$ 的幸运数个数为 k，注意到 $k = 30$。

我们对线段树记录区间内有多少个幸运数以及区间内每个数与大于它的最小幸运数的差的最小值。

Problem (Lucky Array)

定义仅含数码 4 和数码 7 的数为幸运数，给定一个长度为 n 的数组 a，q 次操作：区间加正整数、求区间内有多少数是幸运数。

$1 \leqslant n, m \leqslant 10^5$，保证过程中 a_i 不超过 $v = 10^4$。

Solution

令 $\leqslant v$ 的幸运数个数为 k，注意到 $k = 30$。

我们对线段树记录区间内有多少个幸运数以及区间内每个数与大于它的最小幸运数的差的最小值。

定位后的每次递归当且仅当区间内有位置 i 满足 $\leqslant a_i$ 的幸运数个数增加，显然这一指标不会减小。

Problem (Lucky Array)

定义仅含数码 4 和数码 7 的数为幸运数，给定一个长度为 n 的数组 a，q 次操作：区间加正整数、求区间内有多少数是幸运数。

$1 \leqslant n, m \leqslant 10^5$，保证过程中 a_i 不超过 $v = 10^4$。

Solution

令 $\leqslant v$ 的幸运数个数为 k，注意到 $k = 30$。

我们对线段树记录区间内有多少个幸运数以及区间内每个数与大于它的最小幸运数的差的最小值。

定位后的每次递归当且仅当区间内有位置 i 满足 $\leqslant a_i$ 的幸运数个数增加，显然这一指标不会减小。

总时间复杂度 $\Theta(n \min(k \log n, m) + m \log n)$。

Problem (Lucky Array)

定义仅含数码 4 和数码 7 的数为幸运数，给定一个长度为 n 的数组 a，q 次操作：区间加正整数、求区间内有多少数是幸运数。

$1 \leqslant n, m \leqslant 10^5$，保证过程中 a_i 不超过 $v = 10^4$。

Solution

令 $\leqslant v$ 的幸运数个数为 k，注意到 $k = 30$。

我们对线段树记录区间内有多少个幸运数以及区间内每个数与大于它的最小幸运数的差的最小值。

定位后的每次递归当且仅当区间内有位置 i 满足 $\leqslant a_i$ 的幸运数个数增加，显然这一指标不会减小。

总时间复杂度 $\Theta(n \min(k \log n, m) + m \log n)$。

有非均摊的 $\Theta\left(n + m\sqrt{kn}\right)$ 的分块做法，这里不展开。

Problem (SOJ1969 黑鸭的序列)

给定一个长度为 n 的序列 a，q 次操作：区间加非负数、区间赋值、求区间需要至少多少次单点开根下取整操作使得区间值全相同。

$1 \leqslant n, q \leqslant 2.5 \times 10^5$，保证 a_i 时刻满足 $1 \leqslant a_i \leqslant V = 2 \times 10^9$。

Problem (SOJ1969 黑鸭的序列)

给定一个长度为 n 的序列 a，q 次操作：区间加非负数、区间赋值、求区间需要至少多少次单点开根下取整操作使得区间值全相同。

$1 \leqslant n, q \leqslant 2.5 \times 10^5$，保证 a_i 时刻满足 $1 \leqslant a_i \leqslant V = 2 \times 10^9$。

Solution

我们考虑一棵结点集为 $[1, V] \cap \mathbb{N}$ 的树，i 的父亲结点为 $\lfloor \sqrt{i} \rfloor$。容易说明问题所求即一个区间内的结点到它们的最近公共祖先的距离之和。

Problem (SOJ1969 黑鸭的序列)

给定一个长度为 n 的序列 a，q 次操作：区间加非负数、区间赋值、求区间需要至少多少次单点开根下取整操作使得区间值全相同。

$1 \leqslant n, q \leqslant 2.5 \times 10^5$，保证 a_i 时刻满足 $1 \leqslant a_i \leqslant V = 2 \times 10^9$。

Solution

我们考虑一棵结点集为 $[1, V] \cap \mathbb{N}$ 的树，i 的父亲结点为 $\lfloor \sqrt{i} \rfloor$。容易说明问题所求即一个区间内的结点到它们的最近公共祖先的距离之和。

注意到该树的 bfs 序即为 $\mathbb{N}$ 上一般定义的序关系，且树高为 $\lfloor \log_2(1 + 2\log_2 V) \rfloor$（单个结点树高为 0）。

Problem (SOJ1969 黑鸭的序列)

给定一个长度为 n 的序列 a，q 次操作：区间加非负数、区间赋值、求区间需要至少多少次单点开根下取整操作使得区间值全相同。

$1 \leqslant n, q \leqslant 2.5 \times 10^5$，保证 a_i 时刻满足 $1 \leqslant a_i \leqslant V = 2 \times 10^9$。

Solution

我们考虑一棵结点集为 $[1, V] \cap \mathbb{N}$ 的树，i 的父亲结点为 $\lfloor \sqrt{i} \rfloor$。容易说明问题所求即一个区间内的结点到它们的最近公共祖先的距离之和。

注意到该树的 bfs 序即为 $\mathbb{N}$ 上一般定义的序关系，且树高为 $\lfloor \log_2(1 + 2\log_2 V) \rfloor$（单个结点树高为 0）。

注意到一个结点集的最近公共祖先为其 dfs 序最小和最大的两个结点的最近公共祖先，而相同深度的 dfs 序大小关系与 bfs 序一致。

Solution

我们考虑用线段树维护区间的数在树上的深度和与每个深度的数的最大最小值。

Solution

我们考虑用线段树维护区间的数在树上的深度和与每个深度的数的最大最小值。

我们考虑维护区间加和区间赋值的懒标记与需要维护的值，一次区间加如果使一个深度的数的最大值深度变化则暴力递归。

Solution

我们考虑用线段树维护区间的数在树上的深度和与每个深度的数的最大最小值。

我们考虑维护区间加和区间赋值的懒标记与需要维护的值，一次区间加如果使一个深度的数的最大值深度变化则暴力递归。

我们考虑递归的时间复杂度，一开始序列由 n 个连续段组成，一次区间赋值会将端点处的连续段分裂，再把这个区间作为一个新的连续段替换掉原先这个范围的所有连续段。

Solution

我们考虑用线段树维护区间的数在树上的深度和与每个深度的数的最大最小值。

我们考虑维护区间加和区间赋值的懒标记与需要维护的值，一次区间加如果使一个深度的数的最大值深度变化则暴力递归。

我们考虑递归的时间复杂度，一开始序列由 n 个连续段组成，一次区间赋值会将端点处的连续段分裂，再把这个区间作为一个新的连续段替换掉原先这个范围的所有连续段。

显然每个连续段最多递归 $\Theta(\log\log V)$ 次，只需上传 $\Theta(1)$ 个深度的信息，且会一同递归至定位的结点集，故递归总时间复杂度 $\Theta((n+q)\log n\log\log V)$。注意全部上传复杂度乘上 $\Theta(\log\log V)$ 但常数较小。

Solution

我们考虑用线段树维护区间的数在树上的深度和与每个深度的数的最大最小值。

我们考虑维护区间加和区间赋值的懒标记与需要维护的值，一次区间加如果使一个深度的数的最大值深度变化则暴力递归。

我们考虑递归的时间复杂度，一开始序列由 n 个连续段组成，一次区间赋值会将端点处的连续段分裂，再把这个区间作为一个新的连续段替换掉原先这个范围的所有连续段。

显然每个连续段最多递归 $\Theta(\log\log V)$ 次，只需上传 $\Theta(1)$ 个深度的信息，且会一同递归至定位的结点集，故递归总时间复杂度 $\Theta((n+q)\log n\log\log V)$。注意全部上传复杂度乘上 $\Theta(\log\log V)$ 但常数较小。

我们其实可以在查询时再对定位结点的子树进行递归以减小常数。

Solution

我们考虑用线段树维护区间的数在树上的深度和与每个深度的数的最大最小值。

我们考虑维护区间加和区间赋值的懒标记与需要维护的值，一次区间加如果使一个深度的数的最大值深度变化则暴力递归。

我们考虑递归的时间复杂度，一开始序列由 n 个连续段组成，一次区间赋值会将端点处的连续段分裂，再把这个区间作为一个新的连续段替换掉原先这个范围的所有连续段。

显然每个连续段最多递归 $\Theta(\log\log V)$ 次，只需上传 $\Theta(1)$ 个深度的信息，且会一同递归至定位的结点集，故递归总时间复杂度 $\Theta((n+q)\log n\log\log V)$。注意全部上传复杂度乘上 $\Theta(\log\log V)$ 但常数较小。

我们其实可以在查询时再对定位结点的子树进行递归以减小常数。

总时间复杂度 $\Theta((n+q)\log n\log\log V)$。

Problem (【UNR #5】诡异操作)

给定一个长度为 n 的序列 a，q 次操作：

1. 给定 $[l,r]\subseteq[1,n],v$，对 $i\in[l,r]\cap\mathbb{N}$ 执行 $a_i\leftarrow\lfloor a_i/v\rfloor$。
2. 给定 $[l,r]\subseteq[1,n],v$，对 $i\in[l,r]\cap\mathbb{N}$ 执行 $a_i\leftarrow a_i\operatorname{and}v$。
3. 给定 $[l,r]\subseteq[1,n]$，求 $\sum_{i\in[l,r]\cap\mathbb{N}}a_i$。

$1\leqslant n\leqslant 3\times10^5,1\leqslant q\leqslant 2\times10^5,0\leqslant a_i,v<2^{128}$。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，q 次操作：

1. 给定 $[l,r]\subseteq[1,n],v$，对 $i\in[l,r]\cap\mathbb{N}$ 执行 $a_i\leftarrow\lfloor a_i/v\rfloor$。
2. 给定 $[l,r]\subseteq[1,n],v$，对 $i\in[l,r]\cap\mathbb{N}$ 执行 $a_i\leftarrow a_i\operatorname{and}v$。
3. 给定 $[l,r]\subseteq[1,n]$，求 $\sum_{i\in[l,r]\cap\mathbb{N}}a_i$。

$1\leqslant n\leqslant 3\times10^5,1\leqslant q\leqslant 2\times10^5,0\leqslant a_i,v<2^{128}$。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，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，相当于对维护内容进行了转置。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，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，相当于对维护内容进行了转置。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 $\Theta(n \log \max a_i + q \log^2 n)$。

Problem (【UR #8】决战圆锥曲线)

给定 n 个点，第 i 个点初始时的横坐标 $x_i = i$，纵坐标 y_i 在 $[0, m] \cap \mathbb{N}$ 中均匀随机生成，你需要支持 q 次操作：

1. p 在 $[1, n] \cap \mathbb{N}$ 中均匀随机生成，y 在 $[0, m]$ 中均匀随机生成，将第 p 个点的纵坐标修改为 y。
2. $\{l, r\}$ 通过两次在 $[1, n] \cap \mathbb{N}$ 中均匀随机生成得到且 $l \leqslant r$，将 $i \in [l, r] \cap \mathbb{N}$ 的 y_i 修改为 $m - y_i$。
3. 给定 a, b, c，$\{l, r\}$ 通过两次在 $[1, n] \cap \mathbb{N}$ 中均匀随机生成得到且 $l \leqslant r$，查询 $i \in [l, r] \cap \mathbb{N}$ 的点 (x_i, y_i) 中 $f(x_i, y_i) = ax_i + by_i + cx_iy_i$ 的最大值。

$1 \leqslant n, q_3 \leqslant 10^5, 1 \leqslant q \leqslant 10^6, m = 10^5, 0 \leqslant a, b < 10^6, 0 \leqslant c < 40$，其中 q_3 为查询操作的次数。

Solution

显然若 $x_j \geqslant x_i, y_j \geqslant y_i$ 则第 i 个点的贡献肯定不超过第 j 个点的贡献。

Solution

显然若 $x_j \geqslant x_i, y_j \geqslant y_i$ 则第 i 个点的贡献肯定不超过第 j 个点的贡献。

我们考虑查询一个长度为 l 区间时期望有多少个点可能产生贡献。

Solution

显然若 $x_j \geqslant x_i, y_j \geqslant y_i$ 则第 i 个点的贡献肯定不超过第 j 个点的贡献。

我们考虑查询一个长度为 l 区间时期望有多少个点可能产生贡献。

由于纵坐标随机，区间从后往前第 i 个结点可能产生的概率是 $1/i$，根据期望的线性性，长度为 l 的区间期望有 $\Theta(\log l)$ 个点可能产生贡献。

Solution

显然若 $x_j \geqslant x_i, y_j \geqslant y_i$ 则第 i 个点的贡献肯定不超过第 j 个点的贡献。

我们考虑查询一个长度为 l 区间时期望有多少个点可能产生贡献。

由于纵坐标随机，区间从后往前第 i 个结点可能产生的概率是 $1/i$，根据期望的线性性，长度为 l 的区间期望有 $\Theta(\log l)$ 个点可能产生贡献。

对于一个对应区间为 $[l, r]$ 的结点，设 $> r$ 的部分产生的最大贡献为 t，若 $f(r, \max_{i=l}^{r} y_i) \leqslant t$ 则区间内的点无需进行计算，否则区间内一定有至少一个不被上述偏序的点。

Solution

显然若 $x_j \geqslant x_i, y_j \geqslant y_i$ 则第 i 个点的贡献肯定不超过第 j 个点的贡献。

我们考虑查询一个长度为 l 区间时期望有多少个点可能产生贡献。

由于纵坐标随机，区间从后往前第 i 个结点可能产生的概率是 $1/i$，根据期望的线性性，长度为 l 的区间期望有 $\Theta(\log l)$ 个点可能产生贡献。

对于一个对应区间为 $[l, r]$ 的结点，设 $> r$ 的部分产生的最大贡献为 t，若 $f(r, \max_{i=l}^{r} y_i) \leqslant t$ 则区间内的点无需进行计算，否则区间内一定有至少一个不被上述偏序的点。

因此我们在递归的过程中维护当前答案，优先右子树用上述条件判断是否需要递归，单次查询时间复杂度 $O(\log^2 n)$。

Solution

显然若 $x_j \geqslant x_i, y_j \geqslant y_i$ 则第 i 个点的贡献肯定不超过第 j 个点的贡献。

我们考虑查询一个长度为 l 区间时期望有多少个点可能产生贡献。

由于纵坐标随机，区间从后往前第 i 个结点可能产生的概率是 $1/i$，根据期望的线性性，长度为 l 的区间期望有 $\Theta(\log l)$ 个点可能产生贡献。

对于一个对应区间为 $[l, r]$ 的结点，设 $> r$ 的部分产生的最大贡献为 t，若 $f(r, \max_{i=l}^{r} y_i) \leqslant t$ 则区间内的点无需进行计算，否则区间内一定有至少一个不被上述偏序的点。

因此我们在递归的过程中维护当前答案，优先右子树用上述条件判断是否需要递归，单次查询时间复杂度 $O(\log^2 n)$。

总时间复杂度 $O(q \log n + q_3 \log^2 n)$。

Problem (SOJ1631 鸽子的彩灯)

给定 n 盏灯，第 i 盏灯有属性 s_i, c_i，在位置 a_i。m 次询问每次给定区间 $[l, r]$，表示仅打开位置在该区间的灯，令 v_1 为打开的灯的 c_i 之和，若第 i 盏灯被打开则 $v_{i+1} = v_i - c_i$，否则 $v_{i+1} = v_i$，问 $\max\{i | v_i > s_i\}$。

$1 \leqslant n, m \leqslant 5 \times 10^5, 1 \leqslant c_i \leqslant s_i \leqslant 10^9, \sum c_i \leqslant 10^9$，$a$ 为 n 阶排列。

Problem (SOJ1631 鸽子的彩灯)

给定 n 盏灯，第 i 盏灯有属性 s_i, c_i，在位置 a_i。m 次询问每次给定区间 $[l, r]$，表示仅打开位置在该区间的灯，令 v_1 为打开的灯的 c_i 之和，若第 i 盏灯被打开则 $v_{i+1} = v_i - c_i$，否则 $v_{i+1} = v_i$，问 $\max\{i | v_i > s_i\}$。

$1 \leqslant n, m \leqslant 5 \times 10^5, 1 \leqslant c_i \leqslant s_i \leqslant 10^9, \sum c_i \leqslant 10^9$，$a$ 为 n 阶排列。

Solution

我们考虑离线求解。

Problem (SOJ1631 鸽子的彩灯)

给定 n 盏灯，第 i 盏灯有属性 s_i, c_i，在位置 a_i。m 次询问每次给定区间 $[l, r]$，表示仅打开位置在该区间的灯，令 v_1 为打开的灯的 c_i 之和，若第 i 盏灯被打开则 $v_{i+1} = v_i - c_i$，否则 $v_{i+1} = v_i$，问 $\max\{i | v_i > s_i\}$。

$1 \leqslant n, m \leqslant 5 \times 10^5, 1 \leqslant c_i \leqslant s_i \leqslant 10^9, \sum c_i \leqslant 10^9$，$a$ 为 n 阶排列。

Solution

我们考虑离线求解。

我们按编号从大到小枚举灯 i，维护每个询问的 v，相当于将区间包含给定位置的询问的 v 增加 c_i，将 $v > s_i$ 的询问记录答案并将 v 赋为 $-\infty$。

Solution

我们将询问按左端点从小到大排序，左端点相同按右端点从大到小排序。

Solution

我们将询问按左端点从小到大排序，左端点相同按右端点从大到小排序。

注意到若区间 X 包含于区间 Y，则区间 X 的答案不超过区间 Y，我们维护不被任何没求出答案的询问区间包含的询问区间（构成的集合称为当前集合）的 v，显然一个询问只会插入一次、删除一次。

Solution

我们将询问按左端点从小到大排序，左端点相同按右端点从大到小排序。

注意到若区间 X 包含于区间 Y，则区间 X 的答案不超过区间 Y，我们维护不被任何没求出答案的询问区间包含的询问区间（构成的集合称为当前集合）的 v，显然一个询问只会插入一次、删除一次。

我们注意到一个位置在两两不包含的区间中被一段区间的区间包含，因此每枚举一个灯先做一次区间加再做一次全局超过一个值的所有位置的查询，最后把删除了一些查询后不再被包含的查询插入即可。

Solution

注意到若当前集合中一个区间不被当前集合中的其它区间包含，则下一个当前集合中不被集合中其它区间包含的区间为其之后第一个右端点比它大的区间。

Solution

注意到若当前集合中一个区间不被当前集合中的其它区间包含，则下一个当前集合中不被集合中其它区间包含的区间为其之后第一个右端点比它大的区间。

我们用线段树维护排序后的询问，对每个结点维护当前集合中 v 的最大值，未求出答案的询问的右端点最大值即可。我们对位置维护当前编号后缀的 c 的区间和，每次插入进行查询即可。

Solution

注意到若当前集合中一个区间不被当前集合中的其它区间包含，则下一个当前集合中不被集合中其它区间包含的区间为其之后第一个右端点比它大的区间。

我们用线段树维护排序后的询问，对每个结点维护当前集合中 v 的最大值，未求出答案的询问的右端点最大值即可。我们对位置维护当前编号后缀的 c 的区间和，每次插入进行查询即可。

时间复杂度 $\Theta((n+m)\log m + n\log n)$。

Problem

给定一个大小为 V 的数组 a，初始值均为 0，q 次操作：

1. 给定 $[l,r] \subseteq [1,V]$，对 $i \in [l,r] \cap \mathbb{N}$ 执行 $a_i \leftarrow a_i + x$。
2. 给定 $[l,r] \subseteq [1,V]$，求 $\sum_{i \in [l,r] \cap \mathbb{N}} a_i$。

强制在线，$V = 10^9, 1 \leqslant q \leqslant 5 \times 10^5$。

Problem

给定一个大小为 V 的数组 a，初始值均为 0，q 次操作：

1. 给定 $[l,r] \subseteq [1,V]$，对 $i \in [l,r] \cap \mathbb{N}$ 执行 $a_i \leftarrow a_i + x$。
2. 给定 $[l,r] \subseteq [1,V]$，求 $\sum_{i \in [l,r] \cap \mathbb{N}} a_i$。

强制在线，$V = 10^9, 1 \leqslant q \leqslant 5 \times 10^5$。

注意到虽然直接建立线段树结点很多，但是绝大多数结点都和初始状态一样可以无需维护直接得到信息。这个时候我们可以动态开点。

具体地，对一个线段树结点，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结点。

具体地，对一个线段树结点，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结点。

只有当一个结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。

具体地，对一个线段树结点，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结点。

只有当一个结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。

每次定位结点集会在每个深度增加至多四个结点。

具体地，对一个线段树结点，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结点。

只有当一个结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。

每次定位结点集会在每个深度增加至多四个结点。

询问时正常进行，遇到空结点就当成一个正常的，区间和为 0 的结点对待即可。

具体地，对一个线段树结点，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结点。

只有当一个结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。

每次定位结点集会在每个深度增加至多四个结点。

询问时正常进行，遇到空结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 (二维线段树)

维护一个 $n_1 \times n_2$ 的矩阵，初始元素均为 0，q 次操作：单点修改，矩形求和，矩形求最大值。

要求时间复杂度 $O(q \log n \log m)$。

Problem (二维线段树)

维护一个 $n_1 \times n_2$ 的矩阵，初始元素均为 0，q 次操作：单点修改，矩形求和，矩形求最大值。

要求时间复杂度 $O(q \log n \log m)$。

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的信息，内层线段树需要动态开点。

Problem (二维线段树)

维护一个 $n_1 \times n_2$ 的矩阵，初始元素均为 0，q 次操作：单点修改，矩形求和，矩形求最大值。

要求时间复杂度 $O(q \log n \log m)$。

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的信息，内层线段树需要动态开点。

时空复杂度为 $\Theta(q \log n \log m)$。

Problem (二维线段树)

维护一个 $n_1 \times n_2$ 的矩阵，初始元素均为 0，q 次操作：单点修改，矩形求和，矩形求最大值。

要求时间复杂度 $O(q \log n \log m)$。

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的信息，内层线段树需要动态开点。

时空复杂度为 $\Theta(q \log n \log m)$。

更高维的情况类似，注意在 n, m, q 同阶时矩形加矩形最值同样可以通过 $(\min, +)$ 矩阵乘法规约得到在 APSP 假设下没有 $c < 1.5$ 的 $O(n^c)$ 的做法。

Problem (动态逆序数 /【国家集训队 2011】排队)

一个序列 a 的逆序对为满足 $(i < j) \wedge (a_i > a_j)$ 的 (i, j)。

给定一个长度为 n 的序列 a，q 次修改：给定 x, y，交换 a_x 和 a_y，每次修改后求 a 的逆序对数。

要求时间复杂度 $O(q \log^2 n)$。

Problem (动态逆序数 /【国家集训队 2011】排队)

一个序列 a 的逆序对为满足 $(i < j) \wedge (a_i > a_j)$ 的 (i, j)。

给定一个长度为 n 的序列 a，q 次修改：给定 x, y，交换 a_x 和 a_y，每次修改后求 a 的逆序对数。

要求时间复杂度 $O(q \log^2 n)$。

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 (动态逆序数 /【国家集训队 2011】排队)

一个序列 a 的逆序对为满足 $(i < j) \wedge (a_i > a_j)$ 的 (i, j)。

给定一个长度为 n 的序列 a，q 次修改：给定 x, y，交换 a_x 和 a_y，每次修改后求 a 的逆序对数。

要求时间复杂度 $O(q \log^2 n)$。

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)。

因此我们可以将问题转化为带修改的二维数点问题，直接套用二维线段树解决。

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。

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

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。

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

可以发现与时间线段树和 cdq 分治是实质等效的。

Problem (【Ynoi2016】镜中的昆虫)

给定一个长度为 n 的整数数组 a，q 次操作：

1. 给定 $[l,r] \subseteq [1,n], x$，对 $i \in [l,r] \cap \mathbb{N}$ 执行 $a_i \leftarrow x$。
2. 给定 $[l,r] \subseteq [1,n]$，求 $|\{a_i | i \in [l,r] \cap \mathbb{N}\}|$。

$1 \leqslant n, q \leqslant 10^5, 1 \leqslant a_i \leqslant 10^9$。

Problem (【Ynoi2016】镜中的昆虫)

给定一个长度为 n 的整数数组 a，q 次操作：

1. 给定 $[l,r] \subseteq [1,n], x$，对 $i \in [l,r] \cap \mathbb{N}$ 执行 $a_i \leftarrow x$。
2. 给定 $[l,r] \subseteq [1,n]$，求 $|\{a_i | i \in [l,r] \cap \mathbb{N}\}|$。

$1 \leqslant n, q \leqslant 10^5, 1 \leqslant a_i \leqslant 10^9$。

Solution

令 $\mathrm{pre}_x = \max(\{0\} \cup \{i \in [1,x) \cap \mathbb{N} | a_i = a_x\})$。询问相当于 (pre_i, i) 的二维数点。

Solution

我们观察一次操作可能会带来的 pre_i 的变化：我们将 a 用若干值相同的连续段表示，区间赋值 l, r, x 时先按 l, r 将其所在的连续段分裂。显然 $i < l$ 的 pre_i 不会变化，pre_l 变为 $\max(\{0\} \cup \{i \in [1, l) \cap \mathbb{N} | a_i = x\})$，区间内的所有连续段的左端点除 l 外 pre 均变为其减一，区间内原先出现过的值与 x 在区间后第一次出现位置也会变化，其余 pre 均不变。

Solution

我们观察一次操作可能会带来的 pre_i 的变化：我们将 a 用若干值相同的连续段表示，区间赋值 l, r, x 时先按 l, r 将其所在的连续段分裂。显然 $i < l$ 的 pre_i 不会变化，pre_l 变为 $\max(\{0\} \cup \{i \in [1, l) \cap \mathbb{N} | a_i = x\})$，区间内的所有连续段的左端点除 l 外 pre 均变为其减一，区间内原先出现过的值与 x 在区间后第一次出现位置也会变化，其余 pre 均不变。

注意到一次区间赋值完可以将整个区间合并为一个连续段，若区间内有 k 个连续段则区间赋值最多改变 $\Theta(k)$ 个 pre，且初始至多 n 个连续段，每次修改最多分裂两个，总变化量为 $\Theta(n+q)$，即我们将问题转化为了初始 n 个点，$\Theta(n+q)$ 次修改，至多 q 次查询的带修二维数点问题。连续段与每个数的出现位置可以直接使用 `set`，也可以用线段树和动态开点线段树。

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

我们建立树状数组，对每个结点维护对应区间的权值线段树。

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

我们建立树状数组，对每个结点维护对应区间的权值线段树。

查询时给定区间的权值线段树的信息可用 $\Theta(\log n)$ 个树状数组结点对应区间的权值线段树线性表示，直接进行线段树二分即可。

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

我们建立树状数组，对每个结点维护对应区间的权值线段树。

查询时给定区间的权值线段树的信息可用 $\Theta(\log n)$ 个树状数组结点对应区间的权值线段树线性表示，直接进行线段树二分即可。

时间复杂度 $\Theta(n\log n+q\log^2 n)$。

我们以单点修改区间和为例对持久化线段树进行介绍。

我们以单点修改区间和为例对持久化线段树进行介绍。

Problem

给定一个长度为 n 的序列 a_0，初始 $m = 0$，q 次操作：

1. 给定 $k \in [0, m] \cap \mathbb{N}, p \in [1, n] \cap \mathbb{N}, x$，将 m 增加 1 再将 a_k 复制至 a_m，执行 $a_{m,p} \leftarrow x$。
2. 给定 $k \in [0, m] \cap \mathbb{N}, [l, r] \subseteq [1, n]$，求 $\sum_{i \in [l,r] \cap \mathbb{N}} a_{k,i}$。

要求时间复杂度 $O(n + q \log n)$。

我们以单点修改区间和为例对持久化线段树进行介绍。

Problem

给定一个长度为 n 的序列 a_0，初始 $m = 0$，q 次操作：

1. 给定 $k \in [0, m] \cap \mathbb{N}, p \in [1, n] \cap \mathbb{N}, x$，将 m 增加 1 再将 a_k 复制至 a_m，执行 $a_{m,p} \leftarrow x$。
2. 给定 $k \in [0, m] \cap \mathbb{N}, [l, r] \subseteq [1, n]$，求 $\sum_{i \in [l,r] \cap \mathbb{N}} a_{k,i}$。

要求时间复杂度 $O(n + q \log n)$。

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，即 Path Copy。

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结点的状态，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结点的状态，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。

OI 中较高需求的持久化数组常用持久化线段树实现，部分持久化在一些具体问题下也可以分析出比暴力实现 Fat Node 优的时间复杂度。

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结点的状态，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。

OI 中较高需求的持久化数组常用持久化线段树实现，部分持久化在一些具体问题下也可以分析出比暴力实现 Fat Node 优的时间复杂度。

由于这种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。（据说“主席树”另要求线段树维护值域。）

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 $[1,n]\cap\mathbb{Z}$。

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）。

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）。

查询时给定区间的权值线段树中每个结点的信息可表示为两个前缀的权值线段树对应位置信息的差，在两棵树上一起进行线段树二分即可。

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）。

查询时给定区间的权值线段树中每个结点的信息可表示为两个前缀的权值线段树对应位置信息的差，在两棵树上一起进行线段树二分即可。

时间复杂度 $\Theta((n+q)\log n)$。

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）。

查询时给定区间的权值线段树中每个结点的信息可表示为两个前缀的权值线段树对应位置信息的差，在两棵树上一起进行线段树二分即可。

时间复杂度 $\Theta((n+q)\log n)$。

注意到该做法相当于将带修区间第 k 大的前述做法中的树状数组改为前缀和。

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 (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。

时间复杂度 $\Theta((n+q)\log n)$。

Problem (【国家集训队 2012】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 $a<b<c<d$。

要求时间复杂度 $O(n\log n+q\log^2 n)$。

Solution

我们考虑询问，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

我们考虑询问，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。

检验答案是否 $\geqslant x$ 只需知道是否存在符合题意的区间满足区间和 > 0。

Solution

我们考虑询问，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。

检验答案是否 $\geqslant x$ 只需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 的线段树上维护区间最大前、后缀和即可。

Solution

我们考虑询问，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。

检验答案是否 $\geqslant x$ 只需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 的线段树上维护区间最大前、后缀和即可。

注意到在关于 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，使用部分持久化使得我们可以查询任意 x 的线段树。

Solution

我们考虑询问，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。

检验答案是否 $\geqslant x$ 只需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 的线段树上维护区间最大前、后缀和即可。

注意到在关于 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，使用部分持久化使得我们可以查询任意 x 的线段树。

时间复杂度 $\Theta(n\log n+q\log^2 n)$。

Problem (【ICPC2023 杭州】K. 卡牌游戏)

给定一个长度为 n 的数列 a，q 次询问：给定一个区间，初始栈为空（询问间相互独立），将区间内的数依次枚举做以下操作，问最终栈中的元素个数。

1. 若栈中没有相同数，则将该数压入至栈顶。
2. 若栈中有相同数，则不断弹出栈顶直至弹出了相同数。

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

Problem (【ICPC2023 杭州】K. 卡牌游戏)

给定一个长度为 n 的数列 a，q 次询问：给定一个区间，初始栈为空（询问间相互独立），将区间内的数依次枚举做以下操作，问最终栈中的元素个数。

1. 若栈中没有相同数，则将该数压入至栈顶。
2. 若栈中有相同数，则不断弹出栈顶直至弹出了相同数。

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

Solution

令 $p_i = \min\{j | (j > i) \wedge (a_j = a_i)\}$，$f_{l,r}$ 为给定区间为 $[l, r]$ 的答案。

Problem (【ICPC2023 杭州】K. 卡牌游戏)

给定一个长度为 n 的数列 a，q 次询问：给定一个区间，初始栈为空（询问间相互独立），将区间内的数依次枚举做以下操作，问最终栈中的元素个数。

1. 若栈中没有相同数，则将该数压入至栈顶。
2. 若栈中有相同数，则不断弹出栈顶直至弹出了相同数。

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

Solution

令 $p_i = \min\{j | (j > i) \wedge (a_j = a_i)\}$，$f_{l,r}$ 为给定区间为 $[l, r]$ 的答案。

我们有 $f_{l,r} = \begin{cases} f_{l+1,r} + 1 & (l \leqslant r < p_l) \\ f_{p_l+1,r} & (r \geqslant p_l) \end{cases}$，考虑用线段树进行维护。

Solution

由于询问强制在线，显然我们需要持久化线段树。

Solution

由于询问强制在线，显然我们需要持久化线段树。

注意到 f_l 可以由 f_{l+1} 与 f_{p_l+1} 以 p_l 为分界点拼接后进行区间加一得到，我们可以在一个持久化线段树上做区间赋值为另一棵持久化线段树的结点的操作，区间加是简单的。

Solution

由于询问强制在线，显然我们需要持久化线段树。

注意到 f_l 可以由 f_{l+1} 与 f_{p_l+1} 以 p_l 为分界点拼接后进行区间加一得到，我们可以在一个持久化线段树上做区间赋值为另一棵持久化线段树的结点的操作，区间加是简单的。

我们可以在每个结点维护一个赋值标记指向另一个结点，但更简单的实现方式是注意到我们已经动态开点可持久化，我们可以直接在儿子结点整个被赋值时直接将儿子结点的编号改为要赋值的结点。与后面的线段树分裂与合并类似，但单次操作时间复杂度为严格 $\Theta(\log n)$。

Solution

由于询问强制在线，显然我们需要持久化线段树。

注意到 f_l 可以由 f_{l+1} 与 f_{p_l+1} 以 p_l 为分界点拼接后进行区间加一得到，我们可以在一个持久化线段树上做区间赋值为另一棵持久化线段树的结点的操作，区间加是简单的。

我们可以在每个结点维护一个赋值标记指向另一个结点，但更简单的实现方式是注意到我们已经动态开点可持久化，我们可以直接在儿子结点整个被赋值时直接将儿子结点的编号改为要赋值的结点。与后面的线段树分裂与合并类似，但单次操作时间复杂度为严格 $\Theta(\log n)$。

总时间复杂度 $\Theta((n+q)\log n)$。

Problem (【UNR #1】火车管理)

n 个栈，初始均为空，q 次操作：

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。

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

Problem (【UNR #1】火车管理)

n 个栈，初始均为空，q 次操作：

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。

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

Solution

用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持久化线段树做区间赋值即可。

Solution

区间压入一个数直接对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 $\Theta(q \log n)$。

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：

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，即根结点是空结点，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，即根结点是空结点，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，即根结点是空结点，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结点数，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，即根结点是空结点，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结点数，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时间复杂度不超过总空间复杂度。

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，即根结点是空结点，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结点数，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时间复杂度不超过总空间复杂度。

本质上我们在新建结点时放了 $\Theta(1)$ 的时间复杂度势能。

对于时间复杂度分析中带有势能分析的线段树，势能函数往往会在合并时发生改变，我们需要更细致的权衡。

对于时间复杂度分析中带有势能分析的线段树，势能函数往往会在合并时发生改变，我们需要更细致的权衡。

对于会合并两个叶子的情况，我们可以在叶子结点上放一次单点修改需要的势能，因此我们下面只考虑叶子结点位置不重复只会修改非叶子结点势能的情况。注意显然单点修改可以用线段树合并实现，给出了一个复杂度下界。

对于时间复杂度分析中带有势能分析的线段树，势能函数往往会在合并时发生改变，我们需要更细致的权衡。

对于会合并两个叶子的情况，我们可以在叶子结点上放一次单点修改需要的势能，因此我们下面只考虑叶子结点位置不重复只会修改非叶子结点势能的情况。注意显然单点修改可以用线段树合并实现，给出了一个复杂度下界。

对于 Segment Tree Beats 每个结点势能函数为对应区间内数种类数的情况，合并两个结点后势能恰为原先势能之和，不会额外增加势能。

对于时间复杂度分析中带有势能分析的线段树，势能函数往往会在合并时发生改变，我们需要更细致的权衡。

对于会合并两个叶子的情况，我们可以在叶子结点上放一次单点修改需要的势能，因此我们下面只考虑叶子结点位置不重复只会修改非叶子结点势能的情况。注意显然单点修改可以用线段树合并实现，给出了一个复杂度下界。

对于 Segment Tree Beats 每个结点势能函数为对应区间内数种类数的情况，合并两个结点后势能恰为原先势能之和，不会额外增加势能。

对于 Segment Tree Beats 每个结点势能函数为当其对应区间最大值与父亲结点不同时为深度否则（含空结点）为 0 的情况，合并两个结点后只有自身结点和兄弟结点可能会增加势能，每个结点相比于不线段树合并依然是 $\Theta(1)$ 的额外势能。

对于时间复杂度分析中带有势能分析的线段树，势能函数往往会在合并时发生改变，我们需要更细致的权衡。

对于会合并两个叶子的情况，我们可以在叶子结点上放一次单点修改需要的势能，因此我们下面只考虑叶子结点位置不重复只会修改非叶子结点势能的情况。注意显然单点修改可以用线段树合并实现，给出了一个复杂度下界。

对于 Segment Tree Beats 每个结点势能函数为对应区间内数种类数的情况，合并两个结点后势能恰为原先势能之和，不会额外增加势能。

对于 Segment Tree Beats 每个结点势能函数为当其对应区间最大值与父亲结点不同时为深度否则（含空结点）为 0 的情况，合并两个结点后只有自身结点和兄弟结点可能会增加势能，每个结点相比于不线段树合并依然是 $\Theta(1)$ 的额外势能。

对于 KTT 函数族中差值至多一次变号的情况，我们在每个结点存储两份原先的势能，一份用于原先的操作，另一份用于合并时重置势能。

对于一些更一般的分析，我们可以用启发式合并的思想以多乘上一个 log 的代价去证明时间复杂度的上界。

对于一些更一般的分析，我们可以用启发式合并的思想以多乘上一个 log 的代价去证明时间复杂度的上界。

即我们在合并两棵子树时，时间复杂度不劣于将结点个数小的一者依次进行单点修改。

Problem (子树内众数)

给定一棵 n 个结点的树与每个结点的点权，求每个子树内出现次数最多的点权的出现次数。

要求时间复杂度 $O(n \log n)$。

Problem (子树内众数)

给定一棵 n 个结点的树与每个结点的点权，求每个子树内出现次数最多的点权的出现次数。

要求时间复杂度 $O(n \log n)$。

Solution

对每个结点用一棵权值线段树维护子树内每个点权的出现次数，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。

Problem (子树内众数)

给定一棵 n 个结点的树与每个结点的点权，求每个子树内出现次数最多的点权的出现次数。

要求时间复杂度 $O(n \log n)$。

Solution

对每个结点用一棵权值线段树维护子树内每个点权的出现次数，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。

每个树结点只会新建 $\Theta(\log n)$ 个线段树结点，因此总时空复杂度为 $\Theta(n \log n)$。

Problem (【HNOI2012】永无乡)

给定一张无向带点权图 $G=(V,E)$，q 次操作：

1. 给定 $u,v\in V$，执行 $E\leftarrow E\cup\{\{u,v\}\}$。
2. 给定 $u\in V,k\in\mathbb{N}$，求点 u 所在连通块的第 k 小点权。

$V=[1,n]\cap\mathbb{N},1\leqslant|E_0|\leqslant n\leqslant 10^5,1\leqslant q\leqslant 3\times 10^5$，其中 E_0 为初始的 E。

Problem (【HNOI2012】永无乡)

给定一张无向带点权图 $G=(V,E)$，q 次操作：

1. 给定 $u,v\in V$，执行 $E\leftarrow E\cup\{\{u,v\}\}$。
2. 给定 $u\in V,k\in\mathbb{N}$，求点 u 所在连通块的第 k 小点权。

$V=[1,n]\cap\mathbb{N},1\leqslant|E_0|\leqslant n\leqslant 10^5,1\leqslant q\leqslant 3\times 10^5$，其中 E_0 为初始的 E。

Solution

用并查集维护连通块情况，对每个连通块维护一棵权值线段树。

Problem (【HNOI2012】永无乡)

给定一张无向带点权图 $G=(V,E)$，q 次操作：

1. 给定 $u,v\in V$，执行 $E\leftarrow E\cup\{\{u,v\}\}$。
2. 给定 $u\in V,k\in\mathbb{N}$，求点 u 所在连通块的第 k 小点权。

$V=[1,n]\cap\mathbb{N},1\leqslant|E_0|\leqslant n\leqslant 10^5,1\leqslant q\leqslant 3\times 10^5$，其中 E_0 为初始的 E。

Solution

用并查集维护连通块情况，对每个连通块维护一棵权值线段树。
合并连通块时进行线段树合并，询问时在线段树上二分。

Problem (【HNOI2012】永无乡)

给定一张无向带点权图 $G=(V,E)$，q 次操作：

1. 给定 $u,v\in V$，执行 $E\leftarrow E\cup\{\{u,v\}\}$。
2. 给定 $u\in V,k\in\mathbb{N}$，求点 u 所在连通块的第 k 小点权。

$V=[1,n]\cap\mathbb{N},1\leqslant|E_0|\leqslant n\leqslant 10^5,1\leqslant q\leqslant 3\times 10^5$，其中 E_0 为初始的 E。

Solution

用并查集维护连通块情况，对每个连通块维护一棵权值线段树。
合并连通块时进行线段树合并，询问时在线段树上二分。
时间复杂度 $\Theta((n+q)\log n)$。

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，对每个结点 i 给定值 w_i，叶子结点 u 的权值 $v_u = w_u$，非叶结点 u 的权值 $v_u = |v_{\text{lson}(u)} - v_{\text{rson}(u)}| + w_u$。$q$ 次操作：单点修改 w_i 后问根的权值。

$1 \leqslant n, q \leqslant 2 \times 10^5, 0 \leqslant w_i < 20$。

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，对每个结点 i 给定值 w_i，叶子结点 u 的权值 $v_u = w_u$，非叶结点 u 的权值 $v_u = |v_{\text{lson}(u)} - v_{\text{rson}(u)}| + w_u$。$q$ 次操作：单点修改 w_i 后问根的权值。

$1 \leqslant n, q \leqslant 2 \times 10^5, 0 \leqslant w_i < 20$。

Solution

我们考虑离线。

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，对每个结点 i 给定值 w_i，叶子结点 u 的权值 $v_u = w_u$，非叶结点 u 的权值 $v_u = |v_{\text{lson}(u)} - v_{\text{rson}(u)}| + w_u$。$q$ 次操作：单点修改 w_i 后问根的权值。

$1 \leqslant n, q \leqslant 2 \times 10^5, 0 \leqslant w_i < 20$。

Solution

我们考虑离线。

我们对每个结点用动态开点线段树维护每个时间的权值，非叶结点的线段树通过儿子结点的线段树合并后进行若干次区间加操作得到。

Solution

我们现在只需要考虑动态开点线段树的叶子结点（即对应区间为同一个值）与另一棵树如何进行快速合并。其余部分沿用之前的分析方式可以得到时间复杂度为 $O(n + q\log q)$。

Solution

我们现在只需要考虑动态开点线段树的叶子结点（即对应区间为同一个值）与另一棵树如何进行快速合并。其余部分沿用之前的分析方式可以得到时间复杂度为 $O(n+q\log q)$。

我们考虑对每个结点维护对应区间的最大值、最小值、加法标记、取相反数标记。

Solution

我们现在只需要考虑动态开点线段树的叶子结点（即对应区间为同一个值）与另一棵树如何进行快速合并。其余部分沿用之前的分析方式可以得到时间复杂度为 $O(n + q\log q)$。

我们考虑对每个结点维护对应区间的最大值、最小值、加法标记、取相反数标记。

如果合并一个值与一棵树不会减小树的区间极差，那么我们可以通过打标记 $\Theta(1)$ 解决，否则进行递归合并且树的根结点维护的极差至少减少 1。

Solution

我们现在只需要考虑动态开点线段树的叶子结点（即对应区间为同一个值）与另一棵树如何进行快速合并。其余部分沿用之前的分析方式可以得到时间复杂度为 $O(n+q\log q)$。

我们考虑对每个结点维护对应区间的最大值、最小值、加法标记、取相反数标记。

如果合并一个值与一棵树不会减小树的区间极差，那么我们可以通过打标记 $\Theta(1)$ 解决，否则进行递归合并且树的根结点维护的极差至少减少 1。

共 q 次区间加最多将所有结点的极差之和增加 $O(q\log q\max w_i)$，且此外没有增加，最小减小至 0。

Solution

我们现在只需要考虑动态开点线段树的叶子结点（即对应区间为同一个值）与另一棵树如何进行快速合并。其余部分沿用之前的分析方式可以得到时间复杂度为 $O(n+q\log q)$。

我们考虑对每个结点维护对应区间的最大值、最小值、加法标记、取相反数标记。

如果合并一个值与一棵树不会减小树的区间极差，那么我们可以通过打标记 $\Theta(1)$ 解决，否则进行递归合并且树的根结点维护的极差至少减少 1。

共 q 次区间加最多将所有结点的极差之和增加 $O(q\log q\max w_i)$，且此外没有增加，最小减小至 0。

因此总时间复杂度为 $O(n+q\log q\max w_i)$，类似 Segment Tree Beats 常数极小或复杂度难以取满。

Problem (Escape Through Leaf)

给定一棵 n 个结点的树，你可以用 $a_x b_y$ 的代价从结点 x 跳到其子树内与自身不同的结点 y，对每个结点询问从它跳到叶子结点的代价和的最小值。

$2 \leqslant n \leqslant 10^5, 0 \leqslant |a_i|, |b_i| \leqslant 10^5$。

Problem (Escape Through Leaf)

给定一棵 n 个结点的树，你可以用 $a_x b_y$ 的代价从结点 x 跳到其子树内与自身不同的结点 y，对每个结点询问从它跳到叶子结点的代价和的最小值。

$2 \leqslant n \leqslant 10^5, 0 \leqslant |a_i|, |b_i| \leqslant 10^5$。

Solution

我们对每个结点用一棵李超线段树维护从其祖先结点跳到子树内结点再跳到叶子结点的最小代价和。

Solution

非叶子结点的线段树通过儿子结点的线段树合并后进行一次全局修改得到。

Solution

非叶子结点的线段树通过儿子结点的线段树合并后进行一次全局修改得到。

两棵李超线段树合并时我们将一个根结点上的线段在另一棵树上进行全局操作，再将子树合并。

Solution

非叶子结点的线段树通过儿子结点的线段树合并后进行一次全局修改得到。

两棵李超线段树合并时我们将一个根结点上的线段在另一棵树上进行全局操作，再将子树合并。

注意到每个线段所处结点的深度只增大不减小，总时间复杂度 $\Theta(n \log n)$。

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 $\Theta(\log n)$。

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 $\Theta(\log n)$。

对于每个结点存储了势能的情况，我们分裂每个结点时额外会有两个新的势能减去原先势能的均摊复杂度。

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 $\Theta(\log n)$。

对于每个结点存储了势能的情况，我们分裂每个结点时额外会有两个新的势能减去原先势能的均摊复杂度。

注意我们可以将区间操作规约到两次分裂加上全局操作加上两次合并。

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。

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

类似版本：洛谷【模板】线段树分裂。

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结点，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 $\Theta(\log v)$ 的结点。

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结点，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 $\Theta(\log v)$ 的结点。

然后看合并，如果至少存在一个空结点直接返回，否则递归合并左右儿子。由于每次会删除一个结点，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结点，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 $\Theta(\log v)$ 的结点。

然后看合并，如果至少存在一个空结点直接返回，否则递归合并左右儿子。由于每次会删除一个结点，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结点，可以写一个内存回收来保证总的结点个数不会太多。

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串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串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 $O(n+q\log^2 n)$ 的时间复杂度。

Problem (【Ynoi2013】Ynoi)

给定一个长度为 n 的数组 a，q 次操作：

1. 给定 $[l,r] \subseteq [1,n], x$，对于 $i \in [l,r] \cap \mathbb{N}$，$a_i \leftarrow a_i \operatorname{xor} x$。
2. 给定 $[l,r] \subseteq [1,n], l, r \in \mathbb{N}$，将 $a_l, a_{l+1}, \ldots, a_r$ 赋值为其排序后的结果。
3. 给定 $[l,r] \subseteq [1,n]$，查询 $\operatorname{xor}_{i \in [l,r] \cap \mathbb{N}} a_i$。

$1 \leqslant n, q \leqslant 10^5, 0 \leqslant a_i, x < V = 10^8$。

Problem (【Ynoi2013】Ynoi)

给定一个长度为 n 的数组 a，q 次操作：

1. 给定 $[l,r] \subseteq [1,n], x$，对于 $i \in [l,r] \cap \mathbb{N}$，$a_i \leftarrow a_i \operatorname{xor} x$。
2. 给定 $[l,r] \subseteq [1,n], l, r \in \mathbb{N}$，将 $a_l, a_{l+1}, \ldots, a_r$ 赋值为其排序后的结果。
3. 给定 $[l,r] \subseteq [1,n]$，查询 $\operatorname{xor}_{i \in [l,r] \cap \mathbb{N}} a_i$。

$1 \leqslant n, q \leqslant 10^5, 0 \leqslant a_i, x < V = 10^8$。

Solution

我们考虑用 (S,x) 刻画一个长度为 $|S|$ 的连续段 $s_1 \operatorname{xor} x, s_2 \operatorname{xor} x, \ldots, s_{|S|} \operatorname{xor} x$，其中 S 为用后面的 01 Trie 维护的可重集，$s_1, s_2, \ldots, s_{|S|}$ 为 S 从小到大排序的结果。

Solution

我们可以将 a 通过若干连续段的形式用线段树维护。具体而言，我们在连续段起点的对应叶子结点存放其信息，其余叶子结点不存放信息，我们可以方便地维护一个位置所在的与下一个连续段的存放位置，与完全包含于每个结点对应区间的连续段的区间异或和（此题由于查询信息的可减性也可维护起点在对应区间的连续段的区间查询信息和）。

Solution

我们可以将 a 通过若干连续段的形式用线段树维护。具体而言，我们在连续段起点的对应叶子结点存放其信息，其余叶子结点不存放信息，我们可以方便地维护一个位置所在的与下一个连续段的存放位置，与完全包含于每个结点对应区间的连续段的区间异或和（此题由于查询信息的可减性也可维护起点在对应区间的连续段的区间查询信息和）。

区间修改时，我们先找到 l, r 所在的连续段，若不为连续段端点则将连续段分裂成两个连续段。对于区间异或操作，我们直接在线段树上维护懒标记表示对应区间中的连续段异或上懒标记。对于区间排序操作，我们遍历区间内的所有连续段，将这些连续段合并为一个连续段作为排序结果记录在左端点上。

Solution

我们可以将 a 通过若干连续段的形式用线段树维护。具体而言，我们在连续段起点的对应叶子结点存放其信息，其余叶子结点不存放信息，我们可以方便地维护一个位置所在的与下一个连续段的存放位置，与完全包含于每个结点对应区间的连续段的区间异或和（此题由于查询信息的可减性也可维护起点在对应区间的连续段的区间查询信息和）。

区间修改时，我们先找到 l, r 所在的连续段，若不为连续段端点则将连续段分裂成两个连续段。对于区间异或操作，我们直接在线段树上维护懒标记表示对应区间中的连续段异或上懒标记。对于区间排序操作，我们遍历区间内的所有连续段，将这些连续段合并为一个连续段作为排序结果记录在左端点上。

区间查询时，我们可以用线段树直接求出完整连续段信息，再对区间端点所在连续段进行区间查询即可。

Solution

注意到合并的次数为 $O(n+q)$，查找遍历连续段等线段树操作的总时间复杂度为 $\Theta((n+q)\log n)$。

Solution

注意到合并的次数为 $O(n+q)$，查找遍历连续段等线段树操作的总时间复杂度为 $\Theta((n+q)\log n)$。

我们观察连续段需要支持的操作：初始化 n 个长度为 1 的连续段、$\Theta(n+q)$ 次将一个连续段排序、$\Theta(n+q)$ 次将两个排序好的连续段合并为它们排序的结果、$\Theta(q)$ 次将一个连续段按位置分裂成两个连续段、$\Theta(q)$ 次区间查询并对连续段持续维护全局查询的结果。

Solution

注意到合并的次数为 $O(n+q)$，查找遍历连续段等线段树操作的总时间复杂度为 $\Theta((n+q)\log n)$。

我们观察连续段需要支持的操作：初始化 n 个长度为 1 的连续段、$\Theta(n+q)$ 次将一个连续段排序、$\Theta(n+q)$ 次将两个排序好的连续段合并为它们排序的结果、$\Theta(q)$ 次将一个连续段按位置分裂成两个连续段、$\Theta(q)$ 次区间查询并对连续段持续维护全局查询的结果。

我们可以用 01 Trie 维护 S，将 S 的所有元素异或上 x 可以直接在根结点上打懒标记，递归时若当前位需交换则交换左右儿子，下传后续位的懒标记信息。将两个集合合并与按位置分裂直接使用 01 Trie 合并与分裂即可。

Solution

注意到合并的次数为 $O(n+q)$，查找遍历连续段等线段树操作的总时间复杂度为 $\Theta((n+q)\log n)$。

我们观察连续段需要支持的操作：初始化 n 个长度为 1 的连续段、$\Theta(n+q)$ 次将一个连续段排序、$\Theta(n+q)$ 次将两个排序好的连续段合并为它们排序的结果、$\Theta(q)$ 次将一个连续段按位置分裂成两个连续段、$\Theta(q)$ 次区间查询并对连续段持续维护全局查询的结果。

我们可以用 01 Trie 维护 S，将 S 的所有元素异或上 x 可以直接在根结点上打懒标记，递归时若当前位需交换则交换左右儿子，下传后续位的懒标记信息。将两个集合合并与按位置分裂直接使用 01 Trie 合并与分裂即可。

总时间复杂度 $\Theta((n+q)\log n+(n+q)\log V)$。

一些复杂的连续段高效维护也常用平衡树。例如 SOJ1378 誓约胜利之剑中，线段树按照此题实现每次遍历一个区间内的连续段每一个连续段均需 $O(\log n)$ 的时间复杂度，而平衡树在 $O(\log n)$ 时间复杂度定位区间的基础上每一个遍历的连续段可以仅需 $\Theta(1)$ 完成。我们需要只对维护了连续段信息的线段树结点集的子集最近公共祖先的结点维护线段树信息并动态维护这棵虚树，实质上实现了支持快速合并、分裂、遍历区间的连续段的每个非叶结点至少两个儿子且其树高不超过 $\log n + 1$ 的 leafy tree，可以视为一种特殊功能的平衡树。

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。

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。

注意到之前的很多问题可以用树状数组优化时空常数及代码难度。

```
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]);
    return res;
}
```

```
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]);
    return res;
}
```

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

注意前面提供的也是最常见的树状数组实现方式需要能将问题转化为单点修改，前缀查询且修改或查询一个结点时无需读取或写入其它结点的信息（如 `push_up` 或 `push_down`）。

注意前面提供的也是最常见的树状数组实现方式需要能将问题转化为单点修改，前缀查询且修改或查询一个结点时无需读取或写入其它结点的信息（如 `push_up` 或 `push_down`）。

Problem

几个问题的例子：

1. 单点加，求前缀和。

注意前面提供的也是最常见的树状数组实现方式需要能将问题转化为单点修改，前缀查询且修改或查询一个结点时无需读取或写入其它结点的信息（如 `push_up` 或 `push_down`）。

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。

注意前面提供的也是最常见的树状数组实现方式需要能将问题转化为单点修改，前缀查询且修改或查询一个结点时无需读取或写入其它结点的信息（如 `push_up` 或 `push_down`）。

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。

注意前面提供的也是最常见的树状数组实现方式需要能将问题转化为单点修改，前缀查询且修改或查询一个结点时无需读取或写入其它结点的信息（如 `push_up` 或 `push_down`）。

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，求前缀和。

注意前面提供的也是最常见的树状数组实现方式需要能将问题转化为单点修改，前缀查询且修改或查询一个结点时无需读取或写入其它结点的信息（如 `push_up` 或 `push_down`）。

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，求前缀和。
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。（注意下述做法用到了信息可逆的性质。）

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。（注意下述做法用到了信息可逆的性质。）

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 $k = \Theta(1 + q_2/q_1)$，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。（注意下述做法用到了信息可逆的性质。）

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 $k = \Theta(1 + q_2/q_1)$，总时间复杂度 $\Theta(q_2 \log n / \log(1 + q_2/q_1))$。

若 $q_1 > q_2$，在满足交换律的情况下，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 $k = \Theta(1 + q_1/q_2)$，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

给定一个 $n \times m$ 的矩阵 a，q 次操作：

1. 给定 $x \in [1, n] \cap \mathbb{N}, y \in [1, m] \cap \mathbb{N}, z$，执行 $a_{x,y} \leftarrow z$。
2. 给定 $x \in [1, n] \cap \mathbb{N}, y \in [1, m] \cap \mathbb{N}$，求 $\sum_{i=1}^{x} \sum_{j=1}^{y} a_{i,j}$。

$1 \leqslant n, m \leqslant 5000, 1 \leqslant q \leqslant 10^5$。

Problem

给定一个 $n \times m$ 的矩阵 a，q 次操作：

1. 给定 $x \in [1, n] \cap \mathbb{N}, y \in [1, m] \cap \mathbb{N}, z$，执行 $a_{x,y} \leftarrow z$。
2. 给定 $x \in [1, n] \cap \mathbb{N}, y \in [1, m] \cap \mathbb{N}$，求 $\sum_{i=1}^{x} \sum_{j=1}^{y} a_{i,j}$。

$1 \leqslant n, m \leqslant 5000, 1 \leqslant q \leqslant 10^5$。

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。

Problem

给定一个 $n \times m$ 的矩阵 a，q 次操作：

1. 给定 $x \in [1, n] \cap \mathbb{N}, y \in [1, m] \cap \mathbb{N}, z$，执行 $a_{x,y} \leftarrow z$。
2. 给定 $x \in [1, n] \cap \mathbb{N}, y \in [1, m] \cap \mathbb{N}$，求 $\sum_{i=1}^{x} \sum_{j=1}^{y} a_{i,j}$。

$1 \leqslant n, m \leqslant 5000, 1 \leqslant q \leqslant 10^5$。

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。

时间复杂度 $\Theta(nm + q \log n \log m)$。

“猫树”其实是个俗称，得名于 ImmortalCO（陈俊锟），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，得名于 ImmortalCO（陈俊锟），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。

我们考虑广义的 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。

Property 2.1

S 为一个区间的集合。若对于任意的 $[l,r] \subseteq [1,n]$ 满足 $l,r \in \mathbb{N}$，均满足 $[l,r] \in S$ 或 $\exists m \in [l,r) \cap \mathbb{N}, ([l,m] \in S) \wedge ([m+1,r] \in S)$，则 $|S| = \Omega(n \log n)$。

Property 2.1

S 为一个区间的集合。若对于任意的 $[l,r]\subseteq[1,n]$ 满足 $l,r\in\mathbb{N}$，均满足 $[l,r]\in S$ 或 $\exists m\in[l,r)\cap\mathbb{N},([l,m]\in S)\wedge([m+1,r]\in S)$，则 $|S|=\Omega(n\log n)$。

Proof

对于 $i\in\mathbb{N}^+$，令 $S_i=\{[l,r]\in S|2^{i-1}<r-l+2\leqslant 2^i\}$。则满足 $l_0\in([1,n+2-2^i]\cap\mathbb{N})\backslash(\{l|[l,r]\in S_i\}\cup\{r+2-2^i|[l,r]\in S_i\}),r_0=l_0+2^i-2$ 的区间 $[l_0,r_0]$ 不满足条件，因此 $n-2^i+2\leqslant 2|S_i|$。

Property 2.1

S 为一个区间的集合。若对于任意的 $[l,r]\subseteq[1,n]$ 满足 $l,r\in\mathbb{N}$，均满足 $[l,r]\in S$ 或 $\exists m\in[l,r)\cap\mathbb{N},([l,m]\in S)\wedge([m+1,r]\in S)$，则 $|S|=\Omega(n\log n)$。

Proof

对于 $i\in\mathbb{N}^+$，令 $S_i=\{[l,r]\in S|2^{i-1}<r-l+2\leqslant 2^i\}$。则满足 $l_0\in([1,n+2-2^i]\cap\mathbb{N})\setminus(\{l|[l,r]\in S_i\}\cup\{r+2-2^i|[l,r]\in S_i\}),r_0=l_0+2^i-2$ 的区间 $[l_0,r_0]$ 不满足条件，因此 $n-2^i+2\leqslant 2|S_i|$。

因此 $|S|=\sum_{i\geqslant 1}|S_i|\geqslant\frac{1}{2}\sum_{i\geqslant 1}\max(n-2^i+2,0)=\frac{1}{2}(\lfloor\log_2(n+2)\rfloor(n+2)+1)-2^{\lfloor\log_2(n+2)\rfloor}=\Omega(n\log n)$。

注意前述证明在常数上并不是一个确界，你可以通过【Ynoi2010】Exponential tree 了解类似问题。

注意前述证明在常数上并不是一个确界，你可以通过【Ynoi2010】Exponential tree 了解类似问题。

根据姚期智的论文，长度为 n 的序列预处理 $m(m \geqslant n)$ 个区间需要 $\Theta(\alpha(m,n)+n/(m-n+1))$ 个区间合并为任意给定区间，其中 $\alpha(m,n)$ 为反 Ackermann 函数。

注意前述证明在常数上并不是一个确界，你可以通过【Ynoi2010】Exponential tree 了解类似问题。

根据姚期智的论文，长度为 n 的序列预处理 $m(m \geqslant n)$ 个区间需要 $\Theta(\alpha(m,n)+n/(m-n+1))$ 个区间合并为任意给定区间，其中 $\alpha(m,n)$ 为反 Ackermann 函数。

因此在 q 与 n 同阶时，我们可以在 $\Theta(n\alpha(n))$ 的时间复杂度内完成静态区间半群询问。

注意前述证明在常数上并不是一个确界，你可以通过【Ynoi2010】Exponential tree 了解类似问题。

根据姚期智的论文，长度为 n 的序列预处理 $m(m \geqslant n)$ 个区间需要 $\Theta(\alpha(m,n)+n/(m-n+1))$ 个区间合并为任意给定区间，其中 $\alpha(m,n)$ 为反 Ackermann 函数。

因此在 q 与 n 同阶时，我们可以在 $\Theta(n\alpha(n))$ 的时间复杂度内完成静态区间半群询问。

三区间合并将在后面介绍。

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对应区间后缀和右儿子对应区间前缀维护区间和、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前缀。

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前缀。

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。

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。

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。

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。

可以注意到笛卡尔树上的区间问题和任意给定区间分界点的广义线段树是类似的。

Problem (【Ynoi2009】rprmq1)

一个 $n \times n$ 的矩阵 A，初始为 0。先进行 m 次修改操作：矩形加。再进行 q 次查询操作：求矩形最大值。

$1 \leqslant n, m \leqslant 5 \times 10^4, 1 \leqslant q \leqslant 5 \times 10^5$。

Problem (【Ynoi2009】rprmq1)

一个 $n \times n$ 的矩阵 A，初始为 0。先进行 m 次修改操作：矩形加。再进行 q 次查询操作：求矩形最大值。

$1 \leqslant n, m \leqslant 5 \times 10^4, 1 \leqslant q \leqslant 5 \times 10^5$。

Solution

注意到所有修改均在所有询问前，可以视为静态的二维问题。

Problem (【Ynoi2009】rprmq1)

一个 $n \times n$ 的矩阵 A，初始为 0。先进行 m 次修改操作：矩形加。再进行 q 次查询操作：求矩形最大值。

$1 \leqslant n, m \leqslant 5 \times 10^4, 1 \leqslant q \leqslant 5 \times 10^5$。

Solution

注意到所有修改均在所有询问前，可以视为静态的二维问题。

我们选择一维进行枚举，相当于区间加求区间从某一版本开始的历史最大值。

Problem (【Ynoi2009】rprmq1)

一个 $n \times n$ 的矩阵 A，初始为 0。先进行 m 次修改操作：矩形加。再进行 q 次查询操作：求矩形最大值。

$1 \leqslant n, m \leqslant 5 \times 10^4, 1 \leqslant q \leqslant 5 \times 10^5$。

Solution

注意到所有修改均在所有询问前，可以视为静态的二维问题。

我们选择一维进行枚举，相当于区间加求区间从某一版本开始的历史最大值。

注意到从某一版本开始的最大值导致懒标记较难维护，可能需要持久化平衡树的分裂与合并。

Solution

我们考虑对时间即枚举的一维建立一棵猫树，对所有左儿子结点从后往前做区间加区间历史最大值，对所有右儿子结点从前往后做区间加区间历史最大值，查询操作用猫树直接定位并做两次查询区间历史最大值即可。

Solution

我们考虑对时间即枚举的一维建立一棵猫树，对所有左儿子结点从后往前做区间加区间历史最大值，对所有右儿子结点从前往后做区间加区间历史最大值，查询操作用猫树直接定位并做两次查询区间历史最大值即可。

注意到可以离线，我们可以先将询问存储在猫树上，然后遍历猫树的同时求解，这样可以做到 $\Theta(n+m+q)$ 的空间复杂度，同时避免了线段树需要部分持久化。

Solution

我们考虑对时间即枚举的一维建立一棵猫树，对所有左儿子结点从后往前做区间加区间历史最大值，对所有右儿子结点从前往后做区间加区间历史最大值，查询操作用猫树直接定位并做两次查询区间历史最大值即可。

注意到可以离线，我们可以先将询问存储在猫树上，然后遍历猫树的同时求解，这样可以做到 $\Theta(n+m+q)$ 的空间复杂度，同时避免了线段树需要部分持久化。

时间复杂度 $\Theta(m\log^2 n+q\log n)$。

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。
$1 \leqslant n \leqslant 10^6, 0 \leqslant a_i < V = 2^{32}$。

Problem (最大异或和)

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

Solution

维护前缀的 01-trie，每次在上面查询最大异或值，贪心递归即可。

Problem (最大异或和)

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

Solution

维护前缀的 01-trie，每次在上面查询最大异或值，贪心递归即可。
时间复杂度 $\Theta(n \log V)$。

Problem (区间最大异或和)

给定一个长度为 n 的数列 a。
每次给定三个 $[l,r] \subseteq [1,n], x \in \mathbb{N}$，询问 $\max\{x \operatorname{xor} a_i | i \in [l,r] \cap \mathbb{N}\}$。
$1 \leqslant n,q \leqslant 5 \times 10^5, 0 \leqslant a_i, x < V = 2^{32}$。

Problem (区间最大异或和)

给定一个长度为 n 的数列 a。
每次给定三个 $[l,r] \subseteq [1,n], x \in \mathbb{N}$，询问 $\max\{x \operatorname{xor} a_i | i \in [l,r] \cap \mathbb{N}\}$。
$1 \leqslant n, q \leqslant 5 \times 10^5, 0 \leqslant a_i, x < V = 2^{32}$。

Solution

类似前面一题，我们建立前缀的持久化 01-trie。

Problem (区间最大异或和)

给定一个长度为 n 的数列 a。
每次给定三个 $[l,r] \subseteq [1,n], x \in \mathbb{N}$，询问 $\max\{x \operatorname{xor} a_i | i \in [l,r] \cap \mathbb{N}\}$。
$1 \leqslant n, q \leqslant 5 \times 10^5, 0 \leqslant a_i, x < V = 2^{32}$。

Solution

类似前面一题，我们建立前缀的持久化 01-trie。
时间复杂度 $\Theta((n+q)\log V)$。

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 的最大值右移 k 位的结果。

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

Solution

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

Solution

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

全局异或容易用打全局标记解决，全局增加 1 可以通过模拟二进制运算，不断处理最低位 1 的数量从低到高的情况。实现上为每次交换两个子树后走 0 对应的子树，再将最后变为后 k 位全 0 的位置右移 k 位的最大值增加 1。

Solution

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

全局异或容易用打全局标记解决，全局增加 1 可以通过模拟二进制运算，不断处理最低位 1 的数量从低到高的情况。实现上为每次交换两个子树后走 0 对应的子树，再将最后变为后 k 位全 0 的位置右移 k 位的最大值增加 1。

时间复杂度 $\Theta((n+q_1)k+q)$。

Problem (【模板】普通 Trie 树 2)

维护若干个集合 $A_1, A_2, \ldots,$ q 次操作：

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。

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

Solution

和线段树的版本没什么区别，就是改成 01-Trie，每个结点维护一下异或标记就行了。

Solution

和线段树的版本没什么区别，就是改成 01-Trie，每个结点维护一下异或标记就行了。

时间复杂度 $\Theta(q \log V)$。

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。

$1 \leqslant q \leqslant 5 \times 10^6, 1 \leqslant x \leqslant V \leqslant 5 \times 10^6$，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！

我们考虑一些更快的算法，现在我们要求单次操作时间复杂度为 $o(\log V)$。

我会压位分块再套树状数组！

我们考虑一些更快的算法，现在我们要求单次操作时间复杂度为 $o(\log V)$。

压位 Trie，就是把 Trie 的每个结点改为 k 叉，这样插入删除可以在 $\Theta(\log_k V)$ 的时间内完成。

我会压位分块再套树状数组！

我们考虑一些更快的算法，现在我们要求单次操作时间复杂度为 $o(\log V)$。

压位 Trie，就是把 Trie 的每个结点改为 k 叉，这样插入删除可以在 $\Theta(\log_k V)$ 的时间内完成。

对于前驱后继，若 $k = O(w)$，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！

我们考虑一些更快的算法，现在我们要求单次操作时间复杂度为 $o(\log V)$。

压位 Trie，就是把 Trie 的每个结点改为 k 叉，这样插入删除可以在 $\Theta(\log_k V)$ 的时间内完成。

对于前驱后继，若 $k = O(w)$，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 $\Theta(\log V/\log\log V)$ 或 $\Theta(\log V/\log w)$，空间复杂度为 $\Theta(V/w)$ 或 $\Theta((V\log V)/(w\log w))$。

我会压位分块再套树状数组！

我们考虑一些更快的算法，现在我们要求单次操作时间复杂度为 $o(\log V)$。

压位 Trie，就是把 Trie 的每个结点改为 k 叉，这样插入删除可以在 $\Theta(\log_k V)$ 的时间内完成。

对于前驱后继，若 $k = O(w)$，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 $\Theta(\log V/\log\log V)$ 或 $\Theta(\log V/\log w)$，空间复杂度为 $\Theta(V/w)$ 或 $\Theta((V\log V)/(w\log w))$。

如果 V 过大我们可以使用动态开点，空间复杂度 $\Theta(qw\log V/\log w)$，但会造成更大的时空常数。（你当然可以用哈希进一步做到 $\Theta(q\log V/\log w)$。）

我会压位分块再套树状数组！

我们考虑一些更快的算法，现在我们要求单次操作时间复杂度为 $o(\log V)$。

压位 Trie，就是把 Trie 的每个结点改为 k 叉，这样插入删除可以在 $\Theta(\log_k V)$ 的时间内完成。

对于前驱后继，若 $k = O(w)$，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 $\Theta(\log V/\log\log V)$ 或 $\Theta(\log V/\log w)$，空间复杂度为 $\Theta(V/w)$ 或 $\Theta((V\log V)/(w\log w))$。

如果 V 过大我们可以使用动态开点，空间复杂度 $\Theta(qw\log V/\log w)$，但会造成更大的时空常数。（你当然可以用哈希进一步做到 $\Theta(q\log V/\log w)$。）

压位 trie 和 vEB 树的代码可以看钱哥的国家集训队论文。

钱哥的论文中还给出了压位 Trie 的一些拓展。

钱哥的论文中还给出了压位 Trie 的一些拓展。

Problem

维护一个集合，q 次操作：插入一个元素、删除一个元素、查询小于一个元素的元素个数。其中元素属于 $[1, V] \cap \mathbb{N}$。

钱哥的论文中还给出了压位 Trie 的一些拓展。

Problem

维护一个集合，q 次操作：插入一个元素、删除一个元素、查询小于一个元素的元素个数。其中元素属于 $[1, V] \cap \mathbb{N}$。

Solution

为了方便，这里我们称子树大小为对应区间内元素个数。

钱哥的论文中还给出了压位 Trie 的一些拓展。

Problem

维护一个集合，q 次操作：插入一个元素、删除一个元素、查询小于一个元素的元素个数。其中元素属于 $[1, V] \cap \mathbb{N}$。

Solution

为了方便，这里我们称子树大小为对应区间内元素个数。

我们使用压位 Trie，设叉数为 B，显然每次查询需要求 $\Theta(\log_B V)$ 次前若干个儿子的子树大小和。

钱哥的论文中还给出了压位 Trie 的一些拓展。

Problem

维护一个集合，q 次操作：插入一个元素、删除一个元素、查询小于一个元素的元素个数。其中元素属于 $[1, V] \cap \mathbb{N}$。

Solution

为了方便，这里我们称子树大小为对应区间内元素个数。

我们使用压位 Trie，设叉数为 B，显然每次查询需要求 $\Theta(\log_B V)$ 次前若干个儿子的子树大小和。

我们对每个结点维护子树大小。一个结点每经过 B 次操作重构其儿子结点子树大小的前缀和，那么我们需要对每个结点维护两个元素个数小于 B 值域为 $[0, B) \cap \mathbb{N}$ 的可重集，查询小于给定值的数的个数。

Solution

类似四毛子处理块内信息的手段，我们将所有可能的可重集用 $\Theta(q)$ 以内的整数编号。注意到一次操作和查询只有 $\Theta(B)$ 种参数，我们在 $\mathrm{poly}(B)$ 的时间复杂度内预处理出每个可重集经过每种操作的结果与每种查询的答案。

Solution

类似四毛子处理块内信息的手段，我们将所有可能的可重集用 $\Theta(q)$ 以内的整数编号。注意到一次操作和查询只有 $\Theta(B)$ 种参数，我们在 $\mathrm{poly}(B)$ 的时间复杂度内预处理出每个可重集经过每种操作的结果与每种查询的答案。

注意到预处理的时间复杂度为 $\mathrm{poly}(B)\sum_{i=0}^{B-1}\binom{B+i}{B}$，令 $B=\log q/3$ 则预处理时间复杂度为 $o(q)$。

Solution

类似四毛子处理块内信息的手段，我们将所有可能的可重集用 $\Theta(q)$ 以内的整数编号。注意到一次操作和查询只有 $\Theta(B)$ 种参数，我们在 $\mathrm{poly}(B)$ 的时间复杂度内预处理出每个可重集经过每种操作的结果与每种查询的答案。

注意到预处理的时间复杂度为 $\mathrm{poly}(B)\sum_{i=0}^{B-1}\binom{B+i}{B}$，令 $B=\log q/3$ 则预处理时间复杂度为 $o(q)$。

总时间复杂度 $\Theta(q\log q/\log\log q)$。

动态开点压位 Trie 也可以进行线段树合并。

动态开点压位 Trie 也可以进行线段树合并。

我们同样可以在每个结点上放一个势能来证明合并两个结点的次数。但注意到每个结点的儿子个数并非 $O(1)$，我们可能需要 $\omega(1)$ 的时间去合并两个结点。

动态开点压位 Trie 也可以进行线段树合并。

我们同样可以在每个结点上放一个势能来证明合并两个结点的次数。但注意到每个结点的儿子个数并非 $O(1)$，我们可能需要 $\omega(1)$ 的时间去合并两个结点。

我们可以在合并两个结点时将儿子结点个数较小者没在较大者相同位置出现的儿子逐个插入，这样每个结点会被插入至多 $\log_2 B$ 次，因此合并压位 Trie 在一次插入操作的均摊时间复杂度是 $\Theta(\log V)$，但瓶颈在枚举儿子和赋值上，常数较小，常数较大的合并两个结点在一次插入操作的均摊复杂度仅有 $\Theta(\log V/\log B)$。

动态开点压位 Trie 也可以进行线段树合并。

我们同样可以在每个结点上放一个势能来证明合并两个结点的次数。但注意到每个结点的儿子个数并非 $O(1)$，我们可能需要 $\omega(1)$ 的时间去合并两个结点。

我们可以在合并两个结点时将儿子结点个数较小者没在较大者相同位置出现的儿子逐个插入，这样每个结点会被插入至多 $\log_2 B$ 次，因此合并压位 Trie 在一次插入操作的均摊时间复杂度是 $\Theta(\log V)$，但瓶颈在枚举儿子和赋值上，常数较小，常数较大的合并两个结点在一次插入操作的均摊复杂度仅有 $\Theta(\log V/\log B)$。

具体的一些细节同样可以看钱哥论文。

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 $\left[0, 2^{2^U}\right)$ 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。

对于每个结点，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 $[0, 2^{2^U})$ 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。

对于每个结点，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 $[0, 2^{2^U})$ 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。

对于每个结点，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 $w = 64$）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 $w = 64$）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 $w = 64$）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，在元素不重复的情况下空间复杂度为 $O(n/w)$。10^9 的值域也只需要约 200MB 内存，足够大多数情况下的使用。

当前的权值范围足够小（例如 $w = 64$）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，在元素不重复的情况下空间复杂度为 $O(n/w)$。10^9 的值域也只需要约 200MB 内存，足够大多数情况下的使用。

我们也可以通过动态开点甚至在其基础上使用哈希等方式牺牲一定的时间常数换取更好的空间复杂度。

Problem (前驱查找问题 Predecessor Search Problem)

给定一个大小为 n 的 l 位非负整数集合 Y，多次给定 x 查询 $\max\{y \in Y | y \leqslant x\}$。

$\log n \leqslant l \leqslant w$，强制在线。

Problem (前驱查找问题 Predecessor Search Problem)

给定一个大小为 n 的 l 位非负整数集合 Y，多次给定 x 查询 $\max\{y \in Y | y \leqslant x\}$。

$\log n \leqslant l \leqslant w$，强制在线。

根据 Mihai Pătraşcu 和 Mikkel Thorup 的结果，如果我们使用 S 个 w 位非负整数预处理 Y 的信息。令 $a = \log \frac{S}{n} + \log w$，则在 cell probe 模型下（仅考虑内存访问的时间复杂度），单次询问时间复杂度为 $\Omega\left(1 + \min\left(\log_w n, \log \frac{l - \log n}{a}, \frac{\log(l/a)}{\log(a/\log n \cdot \log(l/a))}, \frac{\log(l/a)}{\log(\log(l/a)/\log(\log n/a))}\right)\right)$，这里 $\log x = \lceil \log_2(x+2) \rceil$。你可以在他们的论文中找到包括 vEB 树在内的某些情况下达到下界的算法。

Problem (前驱查找问题 Predecessor Search Problem)

给定一个大小为 n 的 l 位非负整数集合 Y，多次给定 x 查询 $\max\{y \in Y | y \leqslant x\}$。

$\log n \leqslant l \leqslant w$，强制在线。

根据 Mihai Pătraşcu 和 Mikkel Thorup 的结果，如果我们使用 S 个 w 位非负整数预处理 Y 的信息。令 $a = \log \frac{S}{n} + \log w$，则在 cell probe 模型下（仅考虑内存访问的时间复杂度），单次询问时间复杂度为 $\Omega\left(1 + \min\left(\log_w n, \log \frac{l - \log n}{a}, \frac{\log(l/a)}{\log(a/\log n \cdot \log(l/a))}, \frac{\log(l/a)}{\log(\log(l/a)/\log(\log n/a))}\right)\right)$，这里 $\log x = \lceil \log_2(x+2) \rceil$。你可以在他们的论文中找到包括 vEB 树在内的某些情况下达到下界的算法。

一个常见的情况是当 $l = \gamma_1 \log n, w = \gamma_2 \log n$ 其中 $\gamma_2 \geqslant \gamma_1 > 1$ 为常数且 $S = n \log^{O(1)} n$ 时，时间复杂度为 $\Omega(\log l)$。

Problem (部分持久化数组)

给定一个长度为 n 的数组 a，满足 $\forall i \in [1,n] \cap \mathbb{N}, a_i \in A$，$q$ 次操作：

1. 给定 $p \in [1,n] \cap \mathbb{N}, x \in A$，执行操作 $a_p \leftarrow x$。
2. 给定 $v \in [1,q'] \cap \mathbb{N}, p \in [1,n] \cap \mathbb{N}$，查询 a_p 在第 v 次操作前的值，当前为第 q' 次操作。

强制在线。

Problem (部分持久化数组)

给定一个长度为 n 的数组 a，满足 $\forall i \in [1,n] \cap \mathbb{N}, a_i \in A$，$q$ 次操作：

1. 给定 $p \in [1,n] \cap \mathbb{N}, x \in A$，执行操作 $a_p \leftarrow x$。
2. 给定 $v \in [1,q'] \cap \mathbb{N}, p \in [1,n] \cap \mathbb{N}$，查询 a_p 在第 v 次操作前的值，当前为第 q' 次操作。

强制在线。

我们考虑使用部分持久化数组解决前驱查找问题进行规约。（以下叙述优先使用前驱查找问题中定义的变量名称。）

我们建立一个长度为 N 的整数的部分持久化数组 a，我们需要记录 $b = \lceil 2^l/N \rceil$ 个版本编号，数组初始元素为 -1，第 $i(i \in [0, b) \cap \mathbb{N})$ 个记录的版本的位置 p 的值从前一个记录的版本中修改为 i 当且仅当 $(p-1)b + i \in Y$。我们另外用长度为 N 的数组记录 $\max\{y \in Y | y \leqslant ib\}$。容易说明需要长度为 N 的部分持久化数组其值域 $|A| > b$，进行 n 次修改且需 $\Theta(N + b)$ 的额外空间。

我们建立一个长度为 N 的整数的部分持久化数组 a，我们需要记录 $b = \lceil 2^l/N \rceil$ 个版本编号，数组初始元素为 -1，第 $i(i \in [0, b) \cap \mathbb{N})$ 个记录的版本的位置 p 的值从前一个记录的版本中修改为 i 当且仅当 $(p-1)b + i \in Y$。我们另外用长度为 N 的数组记录 $\max\{y \in Y | y \leqslant ib\}$。容易说明需要长度为 N 的部分持久化数组其值域 $|A| > b$，进行 n 次修改且需 $\Theta(N + b)$ 的额外空间。

对于每次询问 $\max\{y \in Y | y \leqslant x\}$，我们先询问 $a_{\lfloor x/b \rfloor + 1}$ 在第 $x \bmod b$ 个记录的版本时的值，设为 z，若 $z = -1$，则直接查询记录的 $\max\{y \in Y | y \leqslant \lfloor x/b \rfloor\ b\}$ 的值，否则直接返回 $\lfloor x/b \rfloor\ b + z$。显然只需要一次部分持久化数组上的查询以及 $\Theta(1)$ 的额外时间。

我们考虑到前述当 $l = \gamma_1 \log n, w = \gamma_2 \log n, S = n \log^{O(1)} n$，其中 $\gamma_2 \geqslant \gamma_1 > 1$ 为常数时单次查询时间复杂度为 $\Omega(\log l) = \Omega(\log \log n)$，长度为 N，值域为 $\Omega(n)$ 的持久化数组进行 n 次修改后单次查询的时间复杂度为 $\Omega(\log \log n)$，其中空间复杂度为 $O\left(n \log^{O(1)} n\right)$，$N = n^{\Omega(1)}, w = n^{O(1)}$。

我们考虑到前述当 $l = \gamma_1 \log n, w = \gamma_2 \log n, S = n \log^{O(1)} n$，其中 $\gamma_2 \geqslant \gamma_1 > 1$ 为常数时单次查询时间复杂度为 $\Omega(\log l) = \Omega(\log \log n)$，长度为 N，值域为 $\Omega(n)$ 的持久化数组进行 n 次修改后单次查询的时间复杂度为 $\Omega(\log \log n)$，其中空间复杂度为 $O\left(n \log^{O(1)} n\right)$，$N = n^{\Omega(1)}, w = n^{O(1)}$。

换言之，长度为 n 的持久化数组进行 q 次修改且值域 $|A| > q$，$q = n^{O(1)}, w = n^{O(1)}$ 时若空间复杂度为 $O\left(q \log^{O(1)} q\right)$ 则在 cell probe 模型下单次查询时间复杂度为 $\Omega(\log \log q)$。

我们考虑到前述当 $l=\gamma_1\log n, w=\gamma_2\log n, S=n\log^{O(1)} n$，其中 $\gamma_2\geqslant\gamma_1>1$ 为常数时单次查询时间复杂度为 $\Omega(\log l)=\Omega(\log\log n)$，长度为 N，值域为 $\Omega(n)$ 的持久化数组进行 n 次修改后单次查询的时间复杂度为 $\Omega(\log\log n)$，其中空间复杂度为 $O\left(n\log^{O(1)} n\right)$，$N=n^{\Omega(1)}, w=n^{O(1)}$。

换言之，长度为 n 的持久化数组进行 q 次修改且值域 $|A|>q$，$q=n^{O(1)}, w=n^{O(1)}$ 时若空间复杂度为 $O\left(q\log^{O(1)} q\right)$ 则在 cell probe 模型下单次查询时间复杂度为 $\Omega(\log\log q)$。

在规约过程中我们也可以感受到持久化与前驱查找之间的一定联系。

对于完全持久化数组，Dietz 提供了 $O(m+n)$ 空间，修改均摊 $O(\log\log m)$ 查询严格 $O(\log\log m)$ 的算法。Milan Straka 提供了每次操作严格 $O((\log\log m)^2/\log\log\log m)$ 的 $O(m+n)$ 空间的算法与单次严格 $O(\log\log m)$ 的空间 $O\left(n+m^{1+1/(c\log\log m)}\right)$（$c$ 为常数）的算法并将以上所有时间复杂度中的 m 改为了 $\min(n,m)$。Milan Straka 另提供了一种当每个元素被修改不超过 k 次或平均每 n 个操作不超过 k 次时单次询问 $O\left(\sqrt{\log k/\log\log k}\right)$ 的算法。

类似我们基于二叉线段树结构的二区间合并，我们现在给出基于根号叉线段树结构的三区间合并与其相关数据结构 Sqrt Tree。

类似我们基于二叉线段树结构的二区间合并，我们现在给出基于根号叉线段树结构的三区间合并与其相关数据结构 Sqrt Tree。

我们同样要求维护的信息在运算上有结合律。

类似我们基于二叉线段树结构的二区间合并，我们现在给出基于根号叉线段树结构的三区间合并与其相关数据结构 Sqrt Tree。

我们同样要求维护的信息在运算上有结合律。

不妨先考虑完全静态的情况，即三区间合并。

类似我们基于二叉线段树结构的二区间合并，我们现在给出基于根号叉线段树结构的三区间合并与其相关数据结构 Sqrt Tree。

我们同样要求维护的信息在运算上有结合律。

不妨先考虑完全静态的情况，即三区间合并。

我们考察一个区间在 vEB 树上的定位情况：一个非空区间如果不是根结点或叶结点的对应区间，则它可以表示为一个结点的一个儿子 l 的对应区间的一个后缀，在 l 右边的一个儿子 r 的对应区间的一个前缀，与它们之间的儿子结点的对应区间的三者的不交并。

类似我们基于二叉线段树结构的二区间合并，我们现在给出基于根号叉线段树结构的三区间合并与其相关数据结构 Sqrt Tree。

我们同样要求维护的信息在运算上有结合律。

不妨先考虑完全静态的情况，即三区间合并。

我们考察一个区间在 vEB 树上的定位情况：一个非空区间如果不是根结点或叶结点的对应区间，则它可以表示为一个结点的一个儿子 l 的对应区间的一个后缀，在 l 右边的一个儿子 r 的对应区间的一个前缀，与它们之间的儿子结点的对应区间的三者的不交并。

注意到完全静态，我们可以把 summary structure 直接改成预处理所有区间的信息和，处理时间关于原结点的对应区间长度成线性。我们对每个结点维护其所有前缀和与后缀和。

类似我们基于二叉线段树结构的二区间合并，我们现在给出基于根号叉线段树结构的三区间合并与其相关数据结构 Sqrt Tree。

我们同样要求维护的信息在运算上有结合律。

不妨先考虑完全静态的情况，即三区间合并。

我们考察一个区间在 vEB 树上的定位情况：一个非空区间如果不是根结点或叶结点的对应区间，则它可以表示为一个结点的一个儿子 l 的对应区间的一个后缀，在 l 右边的一个儿子 r 的对应区间的一个前缀，与它们之间的儿子结点的对应区间的三者的不交并。

注意到完全静态，我们可以把 summary structure 直接改成预处理所有区间的信息和，处理时间关于原结点的对应区间长度成线性。我们对每个结点维护其所有前缀和与后缀和。

我们将每个儿子结点预处理前后缀和，对于每个结点用 summary structure 维护区间儿子结点的对应区间和的数据结构称为 Sqrt Tree。

我们将 summary structure 直接暴力预处理所有区间信息和的 Sqrt Tree 称为没有索引的 Sqrt Tree。

我们将 summary structure 直接暴力预处理所有区间信息和的 Sqrt Tree 称为没有索引的 Sqrt Tree。

预处理时空复杂度为 $T(n) = \sqrt{n}T(\sqrt{n}) + \Theta(n) = \Theta(n \log \log n)$。

我们将 summary structure 直接暴力预处理所有区间信息和的 Sqrt Tree 称为没有索引的 Sqrt Tree。

预处理时空复杂度为 $T(n) = \sqrt{n}T(\sqrt{n}) + \Theta(n) = \Theta(n \log \log n)$。

我们可以将 n 补全为 2 的幂，并将叉数也限制为 2 的幂，则我们可以使用位运算在 $\Theta(1)$ 的时间复杂度内完成查询。

我们将 summary structure 直接暴力预处理所有区间信息和的 Sqrt Tree 称为没有索引的 Sqrt Tree。

预处理时空复杂度为 $T(n) = \sqrt{n}T(\sqrt{n}) + \Theta(n) = \Theta(n \log \log n)$。

我们可以将 n 补全为 2 的幂，并将叉数也限制为 2 的幂，则我们可以使用位运算在 $\Theta(1)$ 的时间复杂度内完成查询。

Property 5.1

S 为一个区间的集合。若对于任意的 $[l, r] \subseteq [1, n]$，均满足 $\exists k \in \{1, 2, 3\}, [l_1, r_1], [l_2, r_2], \ldots, [l_k, r_k] \in S, (\forall i, j \in [1, k] \cap \mathbb{N}, (i = j) \vee ([l_i, r_i] \cap [l_j, r_j] = \varnothing)) \wedge \left(\bigcup_{i=1}^{k} [l_i, r_i] \cap \mathbb{N} = [l, r] \cap \mathbb{N}\right)$，则 $|S| = \Omega(n \log \log n)$。

Proof

我们设阈值 $B = \Theta(\sqrt{n})$，令 $L = [1, (n-B)/2) \cap \mathbb{N}$，$R = ((n+B)/2, n] \cap \mathbb{N}$，$T = \{[l, r] \in S | r - l \geqslant B\}$，$L_T = \{l \in L | \exists r, [l, r] \in T\}$，$R_T = \{r \in R | \exists l, [l, r] \in T\}$。下面我们证明 $|T| = \Omega(n)$。

Proof

我们设阈值 $B = \Theta(\sqrt{n})$，令 $L = [1, (n-B)/2) \cap \mathbb{N}$，$R = ((n+B)/2, n] \cap \mathbb{N}$，$T = \{[l, r] \in S | r - l \geqslant B\}$，$L_T = \{l \in L | \exists r, [l, r] \in T\}$，$R_T = \{r \in R | \exists l, [l, r] \in T\}$。下面我们证明 $|T| = \Omega(n)$。

对于 $|L| = \Theta(n)$ 或 $|R| = \Theta(n)$，显然成立。

Proof

我们设阈值 $B=\Theta\left(\sqrt{n}\right)$，令 $L=[1,(n-B)/2)\cap\mathbb{N}$，$R=((n+B)/2,n]\cap\mathbb{N}$，$T=\{[l,r]\in S|r-l\geqslant B\}$，$L_T=\{l\in L|\exists r,[l,r]\in T\}$，$R_T=\{r\in R|\exists l,[l,r]\in T\}$。下面我们证明 $|T|=\Omega(n)$。

对于 $|L|=\Theta(n)$ 或 $|R|=\Theta(n)$，显然成立。

对于 $l\in L\setminus L_T$，$\exists l'\in L_T, 0<l'-l<B$。$(R,R_T)$ 有对称结论。

Proof

我们设阈值 $B=\Theta\left(\sqrt{n}\right)$，令 $L=[1,(n-B)/2)\cap\mathbb{N}$，$R=((n+B)/2,n]\cap\mathbb{N}$，$T=\{[l,r]\in S|r-l\geqslant B\}$，$L_T=\{l\in L|\exists r,[l,r]\in T\}$，$R_T=\{r\in R|\exists l,[l,r]\in T\}$。下面我们证明 $|T|=\Omega(n)$。

对于 $|L|=\Theta(n)$ 或 $|R|=\Theta(n)$，显然成立。

对于 $l\in L\setminus L_T$，$\exists l'\in L_T, 0<l'-l<B$。$(R,R_T)$ 有对称结论。

令 s_l 为 $L\setminus L_T$ 划分成若干每个集合能在上式中取相同 l' 的最小集合个数，对称地定义 s_r。我们有 $s_lB+|L_T|\geqslant|L|, s_rB+|R_T|\geqslant|R|$。

Proof

我们设阈值 $B=\Theta\left(\sqrt{n}\right)$，令 $L=[1,(n-B)/2)\cap\mathbb{N}$，$R=((n+B)/2,n]\cap\mathbb{N}$，$T=\{[l,r]\in S|r-l\geqslant B\}$，$L_T=\{l\in L|\exists r,[l,r]\in T\}$，$R_T=\{r\in R|\exists l,[l,r]\in T\}$。下面我们证明 $|T|=\Omega(n)$。

对于 $|L|=\Theta(n)$ 或 $|R|=\Theta(n)$，显然成立。

对于 $l\in L\setminus L_T$，$\exists l'\in L_T, 0<l'-l<B$。$(R,R_T)$ 有对称结论。

令 s_l 为 $L\setminus L_T$ 划分成若干每个集合能在上式中取相同 l' 的最小集合个数，对称地定义 s_r。我们有 $s_lB+|L_T|\geqslant|L|, s_rB+|R_T|\geqslant|R|$。

若 $|L|=o(n),|R|=o(n)$ 我们有 $s_l=\Omega(n/B), s_r=\Omega(n/B)$，观察左端点在 $L\setminus L_T$ 中右端点在 $R\setminus R_T$ 的区间，我们可以说明 $|T|\geqslant s_l\cdot s_r=\Omega(n^2/B^2)=\Omega(n)$。

Proof

我们设阈值 $B=\Theta\left(\sqrt{n}\right)$，令 $L=[1,(n-B)/2)\cap\mathbb{N}$，$R=((n+B)/2,n]\cap\mathbb{N}$，$T=\{[l,r]\in S|r-l\geqslant B\}$，$L_T=\{l\in L|\exists r,[l,r]\in T\}$，$R_T=\{r\in R|\exists l,[l,r]\in T\}$。下面我们证明 $|T|=\Omega(n)$。

对于 $|L|=\Theta(n)$ 或 $|R|=\Theta(n)$，显然成立。

对于 $l\in L\setminus L_T$，$\exists l'\in L_T, 0<l'-l<B$。$(R,R_T)$ 有对称结论。

令 s_l 为 $L\setminus L_T$ 划分成若干每个集合能在上式中取相同 l' 的最小集合个数，对称地定义 s_r。我们有 $s_lB+|L_T|\geqslant|L|, s_rB+|R_T|\geqslant|R|$。

若 $|L|=o(n),|R|=o(n)$ 我们有 $s_l=\Omega(n/B), s_r=\Omega(n/B)$，观察左端点在 $L\setminus L_T$ 中右端点在 $R\setminus R_T$ 的区间，我们可以说明 $|T|\geqslant s_l\cdot s_r=\Omega(n^2/B^2)=\Omega(n)$。

注意到 T 对 $r-l<B$ 的区间 $[l,r]$ 没有贡献，解递推式 $f(n)=\Omega\left(\sqrt{n}f(n/\sqrt{n})\right)+\Omega(n)$ 可知 $|S|=\Omega(n\log\log n)$。

对于单点修改，我们考虑需要改变的维护的信息，除了根结点的 summary structure 需要 $\Theta(n)$ 的时间以外都可以在 $O(\sqrt{n})$ 的时间内完成修改。我们可以把根结点的 summary structure 也改成一棵没有索引的 Sqrt Tree。

对于单点修改，我们考虑需要改变的维护的信息，除了根结点的 summary structure 需要 $\Theta(n)$ 的时间以外都可以在 $O(\sqrt{n})$ 的时间内完成修改。我们可以把根结点的 summary structure 也改成一棵没有索引的 Sqrt Tree。

这样我们做到了查询至多五个区间合并，修改时间复杂度 $\Theta(\sqrt{n})$。

对于单点修改，我们考虑需要改变的维护的信息，除了根结点的 summary structure 需要 $\Theta(n)$ 的时间以外都可以在 $O(\sqrt{n})$ 的时间内完成修改。我们可以把根结点的 summary structure 也改成一棵没有索引的 Sqrt Tree。

这样我们做到了查询至多五个区间合并，修改时间复杂度 $\Theta(\sqrt{n})$。

对于区间修改（满足线段树经典问题形式化叙述中的条件），一种方法是我们像线段树一样从上向下递归，如果结点对应区间包含于给定区间则打上懒标记返回，否则更新 summary structure 与前后缀和并递归。单次修改时间复杂度 $\Theta(\sqrt{n})$，查询时由于懒标记的存在单次时间复杂度 $\Theta(\log \log n)$。

对于单点修改，我们考虑需要改变的维护的信息，除了根结点的 summary structure 需要 $\Theta(n)$ 的时间以外都可以在 $O(\sqrt{n})$ 的时间内完成修改。我们可以把根结点的 summary structure 也改成一棵没有索引的 Sqrt Tree。

这样我们做到了查询至多五个区间合并，修改时间复杂度 $\Theta(\sqrt{n})$。

对于区间修改（满足线段树经典问题形式化叙述中的条件），一种方法是我们像线段树一样从上向下递归，如果结点对应区间包含于给定区间则打上懒标记返回，否则更新 summary structure 与前后缀和并递归。单次修改时间复杂度 $\Theta(\sqrt{n})$，查询时由于懒标记的存在单次时间复杂度 $\Theta(\log\log n)$。

另一种方法是我们只对根结点的儿子打懒标记，对于其它情况我们暴力重构。单次修改时间复杂度 $\Theta(\sqrt{n}\log\log n)$，单次查询时间复杂度 $\Theta(1)$。

作为后两小节经常用于去除复杂度中 log 的一个常用技巧，我们先介绍一下分散层叠。

作为后两小节经常用于去除复杂度中 log 的一个常用技巧，我们先介绍一下分散层叠。

分块时我们经常对 m 组长度为 n 的数组（索引）进行同一个数的二分，我们直接做的时间复杂度为 $\Theta(m \log n)$，但是根据信息论我们经过足够的预处理在 nm 个数中进行二分的复杂度下界只有 $\Theta(\log(nm)) = \Theta(\log n + \log m)$。

作为后两小节经常用于去除复杂度中 log 的一个常用技巧，我们先介绍一下分散层叠。

分块时我们经常对 m 组长度为 n 的数组（索引）进行同一个数的二分，我们直接做的时间复杂度为 $\Theta(m \log n)$，但是根据信息论我们经过足够的预处理在 nm 个数中进行二分的复杂度下界只有 $\Theta(\log(nm)) = \Theta(\log n + \log m)$。

分散层叠的核心思路是：对于两个从小到大排序的数列 a 和 b，且对于 a 的每一项已经记录了在 b 中的相对位置，且 b 任意一个长度为 c 的区间中均包含了一个 a 中记录的位置，则我们可以在 $O(\log c)$ 的时间复杂度内根据一个数及其在 a 中的相对位置二分得出其在 b 中的相对位置。

我们通常以二叉树的形式每次合并两个索引，满足原先的索引长度为 c 的区间中均包含了一个合并后形成的索引记录的相对位置，当 c 为常数时我们显然可以在 $\Theta(1)$ 的时间复杂度内根据一个数 x 与其在合并后索引的相对位置得出原先索引中的相对位置。

我们通常以二叉树的形式每次合并两个索引，满足原先的索引长度为 c 的区间中均包含了一个合并后形成的索引记录的相对位置，当 c 为常数时我们显然可以在 $\Theta(1)$ 的时间复杂度内根据一个数 x 与其在合并后索引的相对位置得出原先索引中的相对位置。

一种常见的实现是从左右儿子结点的索引每相邻 c 个数中选取一个数进行归并得到当前结点的索引。构建一个索引的时间复杂度为关于索引长度成线性。当 c 是常数时我们在关于结点集到根链并大小成线性的时间复杂度内可以根据一个数及其在根结点索引的相对位置得出其在它们索引的相对位置。当 $c = 2$ 时父亲结点的索引大小不超过儿子结点的索引大小较大值，当 $c = 3$ 时父亲结点索引大小不超过儿子结点索引大小较大值的 $\frac{2}{3}$。

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，如果父结点子树大小 $\leqslant B$ 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。

Problem (魔法少女网站·改 I)

给定长度为 n 的序列 a。q 次操作：

1. 给定 $[l,r]\subseteq[1,n],x$，对 $i\in[l,r]\cap\mathbb{N}$ 执行 $a_i\leftarrow a_i+x$。
2. 给定 $[l,r]\subseteq[1,n],x$，求 $|\{[b,c]\cap\mathbb{N}|([b,c]\subseteq[l,r])\wedge(\forall i\in[b,c]\cap\mathbb{N},a_i\leqslant x)\}\setminus\{\varnothing\}|$。

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

Solution

我们先考虑如何维护一个块，使得我们可以 $\Theta(k)$ 求出 k 个块区间的不交并的区间的答案。

Solution

我们先考虑如何维护一个块，使得我们可以 $\Theta(k)$ 求出 k 个块区间的不交并的区间的答案。

我们对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、块内与前两者不交的满足条件的子区间数。

Solution

我们先考虑如何维护一个块，使得我们可以 $\Theta(k)$ 求出 k 个块区间的不交并的区间的答案。

我们对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、块内与前两者不交的满足条件的子区间数。

我们定义一个块的索引为块内元素从小到大排序构成的数列。显然在已知索引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；已知查询的 x 在索引中的位置后可以在 $\Theta(1)$ 的时间复杂度内询问一个块，并在 $\Theta(1)$ 的时间复杂度内合并两个块询问得到的信息作为它们依序连接形成的块询问得到的信息，在 $\Theta(1)$ 的时间复杂度内根据询问得到的信息求出答案。

Solution

我们对于对应区间大小 $\leqslant B$ 的线段树结点直接维护块信息，对于其它结点我们维护一个索引保证索引长度不超过任意一个儿子结点的某一小于 1 的常数倍且可以通过分散层叠在 $\Theta(1)$ 的时间复杂度内根据 x 在当前结点的索引中的位置求出在左右儿子结点的索引中的位置，如在之前二叉树分散层叠的描述中取 $c = 3$。

Solution

构建块信息的时间复杂度为 $\Theta(n + n\log B)$，构建索引的时间复杂度为 $\Theta(n)$。建树总时间复杂度为 $\Theta(n + n\log B)$。

Solution

构建块信息的时间复杂度为 $\Theta(n + n\log B)$，构建索引的时间复杂度为 $\Theta(n)$。建树总时间复杂度为 $\Theta(n + n\log B)$。

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 $\Theta(\log n + B)$。

Solution

构建块信息的时间复杂度为 $\Theta(n + n\log B)$，构建索引的时间复杂度为 $\Theta(n)$。建树总时间复杂度为 $\Theta(n + n\log B)$。

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 $\Theta(\log n + n/B)$。

Solution

构建块信息的时间复杂度为 $\Theta(n + n\log B)$，构建索引的时间复杂度为 $\Theta(n)$。建树总时间复杂度为 $\Theta(n + n\log B)$。

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 $\Theta(\log n + n/B)$。

总时间复杂度为 $\Theta\left((n+q)\log n + \sqrt{q_1 q_2 n}\right)$。

Solution

构建块信息的时间复杂度为 $\Theta(n + n\log B)$，构建索引的时间复杂度为 $\Theta(n)$。建树总时间复杂度为 $\Theta(n + n\log B)$。

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 $\Theta(\log n + n/B)$。

总时间复杂度为 $\Theta\left((n+q)\log n + \sqrt{q_1 q_2 n}\right)$。

同样可以延迟上传结点信息以减小常数。

Problem (【Ynoi2014】置身天上之森)

给定一个长度为 n 的序列 $a_1, a_2, \ldots, a_n$，有一棵叶子结点个数为 n 的线段树，对应区间为 $[l, r]$ 的非叶结点的区间分界点为 $\lfloor \frac{l+r}{2} \rfloor$。$q$ 次操作：

1. 给定 $[l, r] \subseteq [1, n], x$，对于 $i \in [l, r] \cap \mathbb{N}$，$a_i \leftarrow a_i + x$。
2. 给定 l, r, x，询问有多少个线段树结点满足对应区间包含于 $[l, r]$ 且区间和不超过 x。

Solution

我们考虑在线段树上对于对应区间长度不同的结点分别维护，令 n_i 为对应区间长度为 i 的线段树结点数量。

Solution

我们考虑在线段树上对于对应区间长度不同的结点分别维护，令 n_i 为对应区间长度为 i 的线段树结点数量。

线段树所有结点对应区间长度构成的可重集与 $\left\{\left\lfloor\frac{n+j}{2^i}\right\rfloor\middle|i\in\mathbb{N},j\in[0,2^i)\cap\mathbb{N}\right\}$ 去除 0 后的结果相同。可以说明对于 $i\in[0,\log n]\cap\mathbb{N}$，$\left\lfloor n2^{-i}\right\rfloor$ 与 $\left\lceil n2^{-i}\right\rceil$ 的出现次数为 $O(2^i)$，其余长度不可能出现。

Solution

我们考虑在线段树上对于对应区间长度不同的结点分别维护，令 n_i 为对应区间长度为 i 的线段树结点数量。

线段树所有结点对应区间长度构成的可重集与 $\left\{\left\lfloor\frac{n+j}{2^i}\right\rfloor\middle|i\in\mathbb{N},j\in[0,2^i)\cap\mathbb{N}\right\}$ 去除 0 后的结果相同。可以说明对于 $i\in[0,\log n]\cap\mathbb{N}$，$\left\lfloor n2^{-i}\right\rfloor$ 与 $\left\lceil n2^{-i}\right\rceil$ 的出现次数为 $O(2^i)$，其余长度不可能出现。

我们对于子树内结点对应区间长度为 i 的数量 $\leqslant B_i(B_i\leqslant n_i)$ 的线段树结点维护其从小到大排序的结果并作为索引。对于 $>B_i$ 的线段树结点维护索引的分散层叠。

Solution

注意到由于共用线段树定位线段树的时间不必累加，单次修改的时间复杂度为 $\Theta(\log n + \sum B_i)$，单次查询的时间复杂度为 $\Theta(\log n + \sum n_i/B_i)$，我们可以取 $B_i = \min\left(n_i, \max\left(1, \sqrt{q_2 n_i / q_1}\right)\right)$。

Solution

注意到由于共用线段树定位线段树的时间不必累加，单次修改的时间复杂度为 $\Theta(\log n+\sum B_i)$，单次查询的时间复杂度为 $\Theta(\log n+\sum n_i/B_i)$，我们可以取 $B_i=\min\left(n_i,\max\left(1,\sqrt{q_2n_i/q_1}\right)\right)$。

根据等比数列求和的结果，总时间复杂度为 $\Theta\left((n+q)\log n+\sum\sqrt{q_1q_2n_i}\right)=\Theta\left((n+q)\log n+\sqrt{q_1q_2n}\right)$。

众所周知，如果我们每次分治时选取中点对左右分别递归，这种普遍的分治的分治树就是线段树，线段树的算法也对应了这种分治过程。

众所周知，如果我们每次分治时选取中点对左右分别递归，这种普遍的分治的分治树就是线段树，线段树的算法也对应了这种分治过程。

那么有一种分治叫线段树分治是怎么回事呢？

众所周知，如果我们每次分治时选取中点对左右分别递归，这种普遍的分治的分治树就是线段树，线段树的算法也对应了这种分治过程。

那么有一种分治叫线段树分治是怎么回事呢？

线段树分治，就是在分治时，需要利用分治树即线段树的结构，在上面显式地维护信息或查询分治前放上去的信息；从线段树的角度看，就是利用分治的思想把每次操作放到了时间线段树上，设计一个以遍历为主的算法。

Problem (离线动态图连通性)

维护一张无向简单图 $G=(V,E)$，其中 $V=[1,n]\cap\mathbb{N}$。q 次操作：

1. 给定 $u,v\in V$，执行 $E\leftarrow E\cup\{\{u,v\}\}$。
2. 给定 $u,v\in V$，执行 $E\leftarrow E\setminus\{\{u,v\}\}$。
3. 给定 $u,v\in V$，求 G 中 u,v 是否连通。

q 与 n 同阶，初始时 $|E|=0$，要求时间复杂度 $O(n\log n)$。

Problem (离线动态图连通性)

维护一张无向简单图 $G=(V,E)$，其中 $V=[1,n]\cap\mathbb{N}$。q 次操作：

1. 给定 $u,v\in V$，执行 $E\leftarrow E\cup\{\{u,v\}\}$。
2. 给定 $u,v\in V$，执行 $E\leftarrow E\setminus\{\{u,v\}\}$。
3. 给定 $u,v\in V$，求 G 中 u,v 是否连通。

q 与 n 同阶，初始时 $|E|=0$，要求时间复杂度 $O(n\log n)$。

Solution

我们将所有操作离线下来，用在一个时间区间加入一条边的操作替代加入或删除一条边的操作。

Solution

建立叶子结点个数为时间长度（查询次数减连续查询次数，$t \leqslant \min(q_1 + 1, q_3)$）的线段树，在一个时间区间插入一条边可以在其定位的结点集中插入这条边。

Solution

建立叶子结点个数为时间长度（查询次数减连续查询次数，$t \leqslant \min(q_1 + 1, q_3)$）的线段树，在一个时间区间插入一条边可以在其定位的结点集中插入这条边。

我们用可撤销并查集在 dfs 线段树的过程中维护插入当前结点到根链上结点维护的集合中的边后的图连通性。

Solution

建立叶子结点个数为时间长度（查询次数减连续查询次数，$t \leqslant \min(q_1 + 1, q_3)$）的线段树，在一个时间区间插入一条边可以在其定位的结点集中插入这条边。

我们用可撤销并查集在 dfs 线段树的过程中维护插入当前结点到根链上结点维护的集合中的边后的图连通性。

时间复杂度 $O((q_1 \log t + q_3) \log n)$。

Solution

建立叶子结点个数为时间长度（查询次数减连续查询次数，$t \leqslant \min(q_1 + 1, q_3)$）的线段树，在一个时间区间插入一条边可以在其定位的结点集中插入这条边。

我们用可撤销并查集在 dfs 线段树的过程中维护插入当前结点到根链上结点维护的集合中的边后的图连通性。

时间复杂度 $O((q_1 \log t + q_3) \log n)$。

注意到这与时间线段树上进行标记永久化的区间修改再遍历查询所有位置本质等价。

Solution

注意到在遍历线段树中结点 u 的子树时，我们只需要 u 子树内结点维护的集合中的边的顶点与查询的顶点的连通性信息。

Solution

注意到在遍历线段树中结点 u 的子树时，我们只需要 u 子树内结点维护的集合中的边的顶点与查询的顶点的连通性信息。

注意到只有每条边的出现时间的定位结点集的到根链并会需要两个顶点的连通性信息，每个询问的对应叶结点的到根链需要两个顶点的连通性信息，所有线段树结点需要连通性信息的顶点个数和至多 $\Theta(q \log \min(q_1, q_3))$ 个。

Solution

注意到在遍历线段树中结点 u 的子树时，我们只需要 u 子树内结点维护的集合中的边的顶点与查询的顶点的连通性信息。

注意到只有每条边的出现时间的定位结点集的到根链并会需要两个顶点的连通性信息，每个询问的对应叶结点的到根链需要两个顶点的连通性信息，所有线段树结点需要连通性信息的顶点个数和至多 $\Theta(q \log \min(q_1, q_3))$ 个。

我们可以通过暴力 dfs 或并查集连边并查询所有所需顶点根据遍历过程中父结点维护的连通性信息得到当前结点维护的连通性信息，前者时间复杂度 $\Theta(q \log t)$，后者时间复杂度 $\Theta(q \log t\alpha(q))$，其中 $\alpha(q)$ 为反阿克曼函数，但后者常数较小，实际运行时间可能更快。

Solution

注意到在遍历线段树中结点 u 的子树时，我们只需要 u 子树内结点维护的集合中的边的顶点与查询的顶点的连通性信息。

注意到只有每条边的出现时间的定位结点集的到根链并会需要两个顶点的连通性信息，每个询问的对应叶结点的到根链需要两个顶点的连通性信息，所有线段树结点需要连通性信息的顶点个数和至多 $\Theta(q \log \min(q_1, q_3))$ 个。

我们可以通过暴力 dfs 或并查集连边并查询所有所需顶点根据遍历过程中父结点维护的连通性信息得到当前结点维护的连通性信息，前者时间复杂度 $\Theta(q \log t)$，后者时间复杂度 $\Theta(q \log t\alpha(q))$，其中 $\alpha(q)$ 为反阿克曼函数，但后者常数较小，实际运行时间可能更快。

我们也可以使用动态树如 LCT 用其它方法做到 $\Theta(q \log n)$，常数较大且与线段树分治无关，这里不作叙述。

Problem (离线动态图点对间桥边数)

维护一张无向简单图 $G=(V,E)$，其中 $V=[1,n]\cap\mathbb{N}$。q 次操作：

1. 给定 $u,v\in V$，执行 $E\leftarrow E\cup\{\{u,v\}\}$。
2. 给定 $u,v\in V$，执行 $E\leftarrow E\setminus\{\{u,v\}\}$。
3. 给定 $u,v\in V$，求所有 u 到 v 路径边集的交的大小，特别地，若 u,v 不连通直接报告。

$1\leqslant n,q\leqslant 2\times 10^6$，初始时 $|E|=0$。

疑似来源：【ICPC2019 Winter Camp】Bridges。

Problem (离线动态图点对间桥边数)

维护一张无向简单图 $G=(V,E)$，其中 $V=[1,n]\cap\mathbb{N}$。q 次操作：

1. 给定 $u,v\in V$，执行 $E\leftarrow E\cup\{\{u,v\}\}$。
2. 给定 $u,v\in V$，执行 $E\leftarrow E\setminus\{\{u,v\}\}$。
3. 给定 $u,v\in V$，求所有 u 到 v 路径边集的交的大小，特别地，若 u,v 不连通直接报告。

$1\leqslant n,q\leqslant 2\times 10^6$，初始时 $|E|=0$。

疑似来源：【ICPC2019 Winter Camp】Bridges。

Solution

与动态图连通性相同离线后将边插入时间线段树的结点中，进行线段树分治，令 $t\leqslant\min(q_1+1,q_3)$ 为时间长度。

Solution

根据图论知识，对于图 G 的一个生成森林 T，先将所有边权赋值为 1，再对 G 中不在 T 中的边执行两个顶点间 T 中路径边权赋值为 0，最终边权为 1 的集合即为 G 的桥边集合，两个连通的结点间路径边集的交即为它们任意路径上的桥边之集。

Solution

根据图论知识，对于图 G 的一个生成森林 T，先将所有边权赋值为 1，再对 G 中不在 T 中的边执行两个顶点间 T 中路径边权赋值为 0，最终边权为 1 的集合即为 G 的桥边集合，两个连通的结点间路径边集的交即为它们任意路径上的桥边之集。

我们考虑在遍历线段树时维护一棵生成森林，若加入的边的两个顶点不连通则加入生成森林并将边权赋值为 1，否则将这两个顶点间的路径边权赋值为 0。

Solution

根据图论知识，对于图 G 的一个生成森林 T，先将所有边权赋值为 1，再对 G 中不在 T 中的边执行两个顶点间 T 中路径边权赋值为 0，最终边权为 1 的集合即为 G 的桥边集合，两个连通的结点间路径边集的交即为它们任意路径上的桥边之集。

我们考虑在遍历线段树时维护一棵生成森林，若加入的边的两个顶点不连通则加入生成森林并将边权赋值为 1，否则将这两个顶点间的路径边权赋值为 0。

使用 LCT 可以做到时间复杂度 $\Theta((q_1 \log t + q_3) \log n)$。

Solution

根据图论知识，对于图 G 的一个生成森林 T，先将所有边权赋值为 1，再对 G 中不在 T 中的边执行两个顶点间 T 中路径边权赋值为 0，最终边权为 1 的集合即为 G 的桥边集合，两个连通的结点间路径边集的交即为它们任意路径上的桥边之集。

我们考虑在遍历线段树时维护一棵生成森林，若加入的边的两个顶点不连通则加入生成森林并将边权赋值为 1，否则将这两个顶点间的路径边权赋值为 0。

使用 LCT 可以做到时间复杂度 $\Theta((q_1 \log t + q_3) \log n)$。

与离线动态图连通性相同，我们只需维护子树内所需顶点的虚树即可，对虚树的每条边维护原树中的路径中边权为 1 的边数。

Solution

根据图论知识，对于图 G 的一个生成森林 T，先将所有边权赋值为 1，再对 G 中不在 T 中的边执行两个顶点间 T 中路径边权赋值为 0，最终边权为 1 的集合即为 G 的桥边集合，两个连通的结点间路径边集的交即为它们任意路径上的桥边之集。

我们考虑在遍历线段树时维护一棵生成森林，若加入的边的两个顶点不连通则加入生成森林并将边权赋值为 1，否则将这两个顶点间的路径边权赋值为 0。

使用 LCT 可以做到时间复杂度 $\Theta((q_1 \log t + q_3) \log n)$。

与离线动态图连通性相同，我们只需维护子树内所需顶点的虚树即可，对虚树的每条边维护原树中的路径中边权为 1 的边数。

我们可以使用树上差分在关于子树内所需顶点个数成线性的时间复杂度内根据父亲结点的虚树得到当前结点的虚树。

Solution

根据图论知识，对于图 G 的一个生成森林 T，先将所有边权赋值为 1，再对 G 中不在 T 中的边执行两个顶点间 T 中路径边权赋值为 0，最终边权为 1 的集合即为 G 的桥边集合，两个连通的结点间路径边集的交即为它们任意路径上的桥边之集。

我们考虑在遍历线段树时维护一棵生成森林，若加入的边的两个顶点不连通则加入生成森林并将边权赋值为 1，否则将这两个顶点间的路径边权赋值为 0。

使用 LCT 可以做到时间复杂度 $\Theta((q_1 \log t + q_3) \log n)$。

与离线动态图连通性相同，我们只需维护子树内所需顶点的虚树即可，对虚树的每条边维护原树中的路径中边权为 1 的边数。

我们可以使用树上差分在关于子树内所需顶点个数成线性的时间复杂度内根据父亲结点的虚树得到当前结点的虚树。

时间复杂度 $\Theta(q \log t)$。

Problem (动态凸包)

维护一个点集 S，初始为空，q 次询问：

1. 插入一个点 $(x, y) \in \mathbb{R} \times \mathbb{R}$：执行 $S \leftarrow S \cup \{(x, y)\}$；
2. 删除一个点 $(x, y) \in S$：执行 $S \leftarrow S \setminus \{(x, y)\}$；
3. 给定 $a, b > 0$，求 $\max_{(x,y)\in S}(ax + by)$。

要求时间复杂度 $O(q \log q)$。

Problem (动态凸包)

维护一个点集 S，初始为空，q 次询问：

1. 插入一个点 $(x, y) \in \mathbb{R} \times \mathbb{R}$：执行 $S \leftarrow S \cup \{(x, y)\}$；
2. 删除一个点 $(x, y) \in S$：执行 $S \leftarrow S \setminus \{(x, y)\}$；
3. 给定 $a, b > 0$，求 $\max_{(x,y)\in S}(ax + by)$。

要求时间复杂度 $O(q \log q)$。

Solution

相当于维护凸包，然后询问时在凸包上二分找点。

Problem (动态凸包)

维护一个点集 S，初始为空，q 次询问：

1. 插入一个点 $(x, y) \in \mathbb{R} \times \mathbb{R}$：执行 $S \leftarrow S \cup \{(x, y)\}$；
2. 删除一个点 $(x, y) \in S$：执行 $S \leftarrow S \setminus \{(x, y)\}$；
3. 给定 $a, b > 0$，求 $\max_{(x,y)\in S}(ax + by)$。

要求时间复杂度 $O(q \log q)$。

Solution

相当于维护凸包，然后询问时在凸包上二分找点。
由于存在删除，凸包并不是很好维护。

Solution

我们将问题离线，那么我们建立一棵时间线段树，令 $n = \min(q_1, q_3)$，把每个点按照存在时间插入到不超过 $\Theta(\log n)$ 个线段树结点的对应集合中。

Solution

我们将问题离线，那么我们建立一棵时间线段树，令 $n = \min(q_1, q_3)$，把每个点按照存在时间插入到不超过 $\Theta(\log n)$ 个线段树结点的对应集合中。

注意到询问时可以在多个凸包上分别求最大值然后合并答案，我们可以直接在每个结点维护对应集合的凸包，我们在询问时从叶子到根的所有凸包都求一遍答案即可。暴力实现的时间复杂度是 $\Theta(\log^2 n)$。

Solution

我们将问题离线，那么我们建立一棵时间线段树，令 $n = \min(q_1, q_3)$，把每个点按照存在时间插入到不超过 $\Theta(\log n)$ 个线段树结点的对应集合中。

注意到询问时可以在多个凸包上分别求最大值然后合并答案，我们可以直接在每个结点维护对应集合的凸包，我们在询问时从叶子到根的所有凸包都求一遍答案即可。暴力实现的时间复杂度是 $\Theta(\log^2 n)$。

我们可以将所有询问的 b/a 排序，用双指针避免二分，做到总时间复杂度 $\Theta((q_1 + q_2)\log q_1 + (q_1 + q_3)\log n) = \Theta(q_1 \log q_1 + q_3 \log n)$。

Solution

我们将问题离线，那么我们建立一棵时间线段树，令 $n = \min(q_1, q_3)$，把每个点按照存在时间插入到不超过 $\Theta(\log n)$ 个线段树结点的对应集合中。

注意到询问时可以在多个凸包上分别求最大值然后合并答案，我们可以直接在每个结点维护对应集合的凸包，我们在询问时从叶子到根的所有凸包都求一遍答案即可。暴力实现的时间复杂度是 $\Theta(\log^2 n)$。

我们可以将所有询问的 b/a 排序，用双指针避免二分，做到总时间复杂度 $\Theta((q_1 + q_2)\log q_1 + (q_1 + q_3)\log n) = \Theta(q_1 \log q_1 + q_3 \log n)$。

注意到这与时间线段树上进行标记永久化的区间修改再进行单点查询本质等价。

Solution

我们将问题离线，那么我们建立一棵时间线段树，令 $n = \min(q_1, q_3)$，把每个点按照存在时间插入到不超过 $\Theta(\log n)$ 个线段树结点的对应集合中。

注意到询问时可以在多个凸包上分别求最大值然后合并答案，我们可以直接在每个结点维护对应集合的凸包，我们在询问时从叶子到根的所有凸包都求一遍答案即可。暴力实现的时间复杂度是 $\Theta(\log^2 n)$。

我们可以将所有询问的 b/a 排序，用双指针避免二分，做到总时间复杂度 $\Theta((q_1 + q_2)\log q_1 + (q_1 + q_3)\log n) = \Theta(q_1 \log q_1 + q_3 \log n)$。

注意到这与时间线段树上进行标记永久化的区间修改再进行单点查询本质等价。

我们可以使用将【Ynoi2015】世上最幸福的女孩中优化空间的技巧中查询与修改进行对偶，将这题空间复杂度优化至 $\Theta(q_1 + q_3)$。

Problem (给定区间操作序列区间单点查询)

有两个集合 A, B，对于 $a \in A, b \in B$，可以在 $\Theta(1)$ 的时空复杂度内计算出 $b * a \in A$，B 上存在可以在 $\Theta(1)$ 时空复杂度内计算的二元运算 $\cdot$，满足 $\forall a \in A, b_1, b_2 \in B, b_2 * (b_1 * a) = (b_2 \cdot b_1) * a$。给定一个长度为 n 值域为 A 的序列 a，q 次操作：

1. 给定 $[l, r] \subseteq [1, n], x \in B$，表示一个操作：对于 $i \in [l, r] \cap \mathbb{N}$，$a_i \leftarrow x * a_i$。
2. 给定 $[u, v] \subseteq [1, q'), k \in [1, n] \cap \mathbb{N}$，询问当只进行 $[u, v] \cap \mathbb{N}$ 内的操作时 a_k 的值，q' 表示当前的操作数量。

强制在线，要求时间复杂度 $O(q \log q_1)$。

特殊情况：【清华集训 2014】玄学、【Ynoi2058】《A Path Towards Autonomous Machine Intelligence》阅读报告（更新中...）。

Solution

显然我们要想办法用时间线段树来完成询问。

Solution

显然我们要想办法用时间线段树来完成询问。

首先考虑离线的做法，显然我们只要将修改插入到它对应叶子到根的所有结点，每个结点维护一个长度为 k 的序列 (p_i, v_i)，令 $p_0 = 0, p_k = n$，该序列表示经过时间线段树当前结点对应区间的操作对于 $i \in [1, k] \cap \mathbb{Z}$，所有 $j \in (p_{i-1}, p_i] \cap \mathbb{N}$ 的 a_j 被修改为 $v_i * a_j$，显然 k 不超过结点对应区间长度。

Solution

显然我们要想办法用时间线段树来完成询问。

首先考虑离线的做法，显然我们只要将修改插入到它对应叶子到根的所有结点，每个结点维护一个长度为 k 的序列 (p_i, v_i)，令 $p_0 = 0, p_k = n$，该序列表示经过时间线段树当前结点对应区间的操作对于 $i \in [1, k] \cap \mathbb{Z}$，所有 $j \in (p_{i-1}, p_i] \cap \mathbb{N}$ 的 a_j 被修改为 $v_i * a_j$，显然 k 不超过结点对应区间长度。

查询时我们可以定位 $[u, v] \cap \mathbb{N}$，在维护的序列上找到对应位置暴力更新即可。

Solution

如果在线的话，我们仍然维护这个序列：当一个结点的对应区间右端点为当前时间时，用它的两个儿子的序列进行归并即可。显然询问时定位的结点的对应区间右端点不会超过当前时间。

Solution

如果在线的话，我们仍然维护这个序列：当一个结点的对应区间右端点为当前时间时，用它的两个儿子的序列进行归并即可。显然询问时定位的结点的对应区间右端点不会超过当前时间。

构造线段树复杂度 $\Theta(q_1 \log q_1)$，用每个结点上二分暴力实现单次询问时间复杂度 $\Theta(\log q_1 \log \min(n, q_1))$。

Solution

如果在线的话，我们仍然维护这个序列：当一个结点的对应区间右端点为当前时间时，用它的两个儿子的序列进行归并即可。显然询问时定位的结点的对应区间右端点不会超过当前时间。

构造线段树复杂度 $\Theta(q_1 \log q_1)$，用每个结点上二分暴力实现单次询问时间复杂度 $\Theta(\log q_1 \log \min(n, q_1))$。

显然我们可以试图用分散层叠去优化找到一个位置在序列中的出现位置。

Solution

如果在线的话，我们仍然维护这个序列：当一个结点的对应区间右端点为当前时间时，用它的两个儿子的序列进行归并即可。显然询问时定位的结点的对应区间右端点不会超过当前时间。

构造线段树复杂度 $\Theta(q_1 \log q_1)$，用每个结点上二分暴力实现单次询问时间复杂度 $\Theta(\log q_1 \log \min(n, q_1))$。

显然我们可以试图用分散层叠去优化找到一个位置在序列中的出现位置。

与【Ynoi2019】魔法少女网站不同的是，我们不仅对超过一定深度的结点有需要支持查询的初始索引，对于任意一个线段树结点，我们都有一个不超过对应区间长度的初始索引需要支持查询。

Solution

我们可以对每个已经归并出序列的结点直接暴力维护子树内初始索引中的每个位置作为索引，维护其在左右子树索引中的位置，因此我们只需在每次询问中求出查询的位置在所有当前时间前缀定位的结点中的当前索引的位置即可。

Solution

我们可以对每个已经归并出序列的结点直接暴力维护子树内初始索引中的每个位置作为索引，维护其在左右子树索引中的位置，因此我们只需在每次询问中求出查询的位置在所有当前时间前缀定位的结点中的当前索引的位置即可。

注意到前缀定位的结点集的性质，我们在每个长度为 l_1 的左儿子结点将右端点为其左端点减一的左儿子结点（设长度为 l_2）的最终索引每 $\Theta(l_2/l_1)$ 个选取至少一个与当前结点的当前索引进行归并得到当前结点的最终索引。

Solution

我们可以对每个已经归并出序列的结点直接暴力维护子树内初始索引中的每个位置作为索引，维护其在左右子树索引中的位置，因此我们只需在每次询问中求出查询的位置在所有当前时间前缀定位的结点中的当前索引的位置即可。

注意到前缀定位的结点集的性质，我们在每个长度为 l_1 的左儿子结点将右端点为其左端点减一的左儿子结点（设长度为 l_2）的最终索引每 $\Theta(l_2/l_1)$ 个选取至少一个与当前结点的当前索引进行归并得到当前结点的最终索引。

可以容易地说明构造线段树时间复杂度不变，单次询问时间复杂度 $\Theta(\log q_1)$，即总时间复杂度 $\Theta(q \log q_1)$。

我们考虑枚举序列位置关于时间维护信息，可以将该问题转化为：

Problem

一个值域为 B 的序列 b，有 n 个版本，对于 $i \in (1, n] \cap \mathbb{N}$，第 i 个版本在第 $i-1$ 个版本上通过若干单点修改得到。初始时长度 $m = 0$，q 次操作：

1. 给定 $x \in B$ 与若干 $(p, y) \in ([1, n] \cap \mathbb{N}) \times B$，表示在 b 个版本尾部插入一个位置的初值与该位置的单点修改情况，操作后 m 会增加 1。
2. 给定 $p \in [1, n] \cap \mathbb{N}, [l, r] \subseteq [1, m]$，查询第 p 个版本中的区间和。

我们考虑枚举序列位置关于时间维护信息，可以将该问题转化为：

Problem

一个值域为 B 的序列 b，有 n 个版本，对于 $i\in(1,n]\cap\mathbb{N}$，第 i 个版本在第 $i-1$ 个版本上通过若干单点修改得到。初始时长度 $m=0$，q 次操作：

1. 给定 $x\in B$ 与若干 $(p,y)\in([1,n]\cap\mathbb{N})\times B$，表示在 b 个版本尾部插入一个位置的初值与该位置的单点修改情况，操作后 m 会增加 1。
2. 给定 $p\in[1,n]\cap\mathbb{N},[l,r]\subseteq[1,m]$，查询第 p 个版本中的区间和。

而该解法等价于维护对应区间包含于当前前缀的持久化线段树结点，并用分散层叠维护持久化线段树在当前前缀定位结点集（若干左儿子结点）的结点编号。

Problem (尾部插入区间函数最值)

有一个函数族 $\mathcal{F}$，满足对于 $f_1, f_2 \in \mathcal{F}, x_1, x_2 \in \mathbb{R}$，可以在 $\Theta(A)$ 的时间复杂度内求出 $f_1(x_1)$，在 $\Theta(B)$ 的时间复杂度内求出 $\min\{x | \{-1,1\} \subseteq \{\operatorname{sgn}(f_1(x_1+a)-f_2(x_2+a)) | 0 \leqslant a \leqslant x\}\}$ 与 $a \geqslant 0$ 最小的满足值不为 0 的 $\operatorname{sgn}(f_1(x_1+a)-f_2(x_2+a))$。初始时 $n=0$，q 次操作：

1. 令 $n \leftarrow n+1$，再给定函数 $f_n \in \mathcal{F}$。
2. 给定区间 $[l,r] \subseteq [1,n]$ 与自变量 x，求 $\max_{i \in [l,r] \cap \mathbb{N}} f_i(x)$。

强制在线。要求时间复杂度 $O((\lambda_{g(\mathcal{F})}(q_1)B + q_2 A)\log q_1)$，其中 $g(\mathcal{F})$ 与 $\lambda_s(n)$ 的定义见 KTT 小节。

Problem (尾部插入区间函数最值)

有一个函数族 $\mathcal{F}$，满足对于 $f_1, f_2 \in \mathcal{F}, x_1, x_2 \in \mathbb{R}$，可以在 $\Theta(A)$ 的时间复杂度内求出 $f_1(x_1)$，在 $\Theta(B)$ 的时间复杂度内求出 $\min\{x | \{-1,1\} \subseteq \{\operatorname{sgn}(f_1(x_1+a)-f_2(x_2+a)) | 0 \leqslant a \leqslant x\}\}$ 与 $a \geqslant 0$ 最小的满足值不为 0 的 $\operatorname{sgn}(f_1(x_1+a)-f_2(x_2+a))$。初始时 $n=0$，q 次操作：

1. 令 $n \leftarrow n+1$，再给定函数 $f_n \in \mathcal{F}$。
2. 给定区间 $[l,r] \subseteq [1,n]$ 与自变量 x，求 $\max_{i \in [l,r] \cap \mathbb{N}} f_i(x)$。

强制在线。要求时间复杂度 $O((\lambda_{g(\mathcal{F})}(q_1)B + q_2 A)\log q_1)$，其中 $g(\mathcal{F})$ 与 $\lambda_s(n)$ 的定义见 KTT 小节。

Solution

与给定区间操作序列区间单点查询类似建立时间线段树维护分散层叠。

Solution

在给定区间操作序列区间单点查询的问题中，我们对每个结点维护了 $\Theta(l)$ 段的经过当前结点对应区间中的操作后每个位置发生的变化，上传信息时间复杂度 $\Theta(l)$，在此题中我们对每个结点维护对应区间内函数的至多 $\lambda_{g(\mathcal{F})}(l)$ 段的上包络线，上传信息时间复杂度 $\Theta(\lambda_{g(\mathcal{F})}(l)B)$，其中 l 为结点对应区间长度。

Solution

在给定区间操作序列区间单点查询的问题中，我们对每个结点维护了 $\Theta(l)$ 段的经过当前结点对应区间中的操作后每个位置发生的变化，上传信息时间复杂度 $\Theta(l)$，在此题中我们对每个结点维护对应区间内函数的至多 $\lambda_{g(\mathcal{F})}(l)$ 段的上包络线，上传信息时间复杂度 $\Theta(\lambda_{g(\mathcal{F})}(l)B)$，其中 l 为结点对应区间长度。

类似地，我们可以用主定理证明其修改操作的时间复杂度为 $\Theta(\lambda_{g(\mathcal{F})}(q_1)B\log q_1)$，单次询问时间复杂度为严格 $\Theta(A\log q_1)$。

Solution

在给定区间操作序列区间单点查询的问题中，我们对每个结点维护了 $\Theta(l)$ 段的经过当前结点对应区间中的操作后每个位置发生的变化，上传信息时间复杂度 $\Theta(l)$，在此题中我们对每个结点维护对应区间内函数的至多 $\lambda_{g(\mathcal{F})}(l)$ 段的上包络线，上传信息时间复杂度 $\Theta(\lambda_{g(\mathcal{F})}(l)B)$，其中 l 为结点对应区间长度。

类似地，我们可以用主定理证明其修改操作的时间复杂度为 $\Theta(\lambda_{g(\mathcal{F})}(q_1)B\log q_1)$，单次询问时间复杂度为严格 $\Theta(A\log q_1)$。

总时间复杂度 $\Theta((\lambda_{g(\mathcal{F})}(q_1)B+q_2A)\log q_1)$。

Solution

在给定区间操作序列区间单点查询的问题中，我们对每个结点维护了 $\Theta(l)$ 段的经过当前结点对应区间中的操作后每个位置发生的变化，上传信息时间复杂度 $\Theta(l)$，在此题中我们对每个结点维护对应区间内函数的至多 $\lambda_{g(\mathcal{F})}(l)$ 段的上包络线，上传信息时间复杂度 $\Theta(\lambda_{g(\mathcal{F})}(l)B)$，其中 l 为结点对应区间长度。

类似地，我们可以用主定理证明其修改操作的时间复杂度为 $\Theta(\lambda_{g(\mathcal{F})}(q_1)B\log q_1)$，单次询问时间复杂度为严格 $\Theta(A\log q_1)$。

总时间复杂度 $\Theta((\lambda_{g(\mathcal{F})}(q_1)B+q_2A)\log q_1)$。

注意到在这个问题上复杂度比李超树优且放宽了某些限制。

在最短路或最大流建图的过程中，我们可能会遇到以下形式：m 次，每次对 u 到一个区间 $[l, r]$ 连上长度为 x 的边。如果我们暴力进行建边的话，边数可能会达到 $\Theta(nm)$。

在最短路或最大流建图的过程中，我们可能会遇到以下形式：m 次，每次对 u 到一个区间 $[l, r]$ 连上长度为 x 的边。如果我们暴力进行建边的话，边数可能会达到 $\Theta(nm)$。

我们考虑用线段树进行优化，我们对线段树的叶子结点维护每个位置对应的图中的结点，对每个非叶子结点在图中新建一个结点，向两个儿子对应的结点分别连一条边（例如在最短路问题中连一条长度为 0 的有向边）。然后我们每次连边都可以定位到至多 $\Theta(\log n)$ 个结点，然后向这些结点对应的点连权值为给定边权的边即可。

Problem

m 次对左侧点区间 $[l_i, r_i]$ 往右侧点区间 $[u_i, v_i]$ 连边，要求额外点数 $O(n+m)$，额外边数 $O(n+m\log n)$。

Problem

m 次对左侧点区间 $[l_i, r_i]$ 往右侧点区间 $[u_i, v_i]$ 连边，要求额外点数 $O(n+m)$，额外边数 $O(n+m\log n)$。

Solution

我们在左侧线段树建立时儿子结点对应点向父亲结点对应点连边，右侧线段树父亲结点对应点向儿子结点对应点连边，每次操作定位左侧区间对应结点向一个新点连边，再将新点向定位的右侧区间对应结点连边即可。

Problem

n 个方格，对于 $i \in [1,n] \cap \mathbb{N}$，给定第 i 个方格的 6 个属性：$a_i, b_i, w_i, l_i, r_i, p_i$。需要把每个方格染成黑色或白色。第 i 个格子被染成黑色会产生 b_i 的贡献，被染成白色会产生 w_i 的贡献。如果第 i 个方格被染成黑色且存在 $j < i$ 满足第 j 个方格被染成白色且 $l_i < a_j < r_i$，那么会产生 p_i 的代价。求贡献减代价的最大值。

$1 \leqslant n \leqslant 5000$。

Problem

n 个方格，对于 $i \in [1,n] \cap \mathbb{N}$，给定第 i 个方格的 6 个属性：$a_i, b_i, w_i, l_i, r_i, p_i$。需要把每个方格染成黑色或白色。第 i 个格子被染成黑色会产生 b_i 的贡献，被染成白色会产生 w_i 的贡献。如果第 i 个方格被染成黑色且存在 $j < i$ 满足第 j 个方格被染成白色且 $l_i < a_j < r_i$，那么会产生 p_i 的代价。求贡献减代价的最大值。

$1 \leqslant n \leqslant 5000$。

Solution

我们可以先将基础答案加上所有 b_i 与 w_i 的和，然后改成被染成黑色产生 w_i 的代价，被染成白色产生 b_i 的代价，满足题中条件产生 p_i 的代价。

Solution

这种染成两种颜色之一要求总代价和最小的问题我们考虑建立最小割模型。

Solution

这种染成两种颜色之一要求总代价和最小的问题我们考虑建立最小割模型。

我们对每个结点 i 建立一个三个结点 $u_{i,1}, u_{i,2}, u_{i,3}$，将源点向 $u_{i,1}$ 连一条 $w_i + x$ 的边，$u_{i,1}$ 向 $u_{i,2}$ 连一条 $+\infty$ 的边，$u_{i,2}$ 向汇点连一条 $b_i + x$ 的边，$u_{i,1}$ 向 $u_{i,3}$ 连一条 p_i 的边，并将 $u_{i,3}$ 向满足 $j < i$ 且 $l_i < a_j < r_i$ 的 $u_{j,2}$ 连一条 $+\infty$ 的边。其中边权为 ∞ 的边在实现中可以将边权设为一个足够大的数，这个数与 x 均需大于最终求出的答案。

Solution

这种染成两种颜色之一要求总代价和最小的问题我们考虑建立最小割模型。

我们对每个结点 i 建立一个三个结点 $u_{i,1}, u_{i,2}, u_{i,3}$，将源点向 $u_{i,1}$ 连一条 $w_i + x$ 的边，$u_{i,1}$ 向 $u_{i,2}$ 连一条 $+\infty$ 的边，$u_{i,2}$ 向汇点连一条 $b_i + x$ 的边，$u_{i,1}$ 向 $u_{i,3}$ 连一条 p_i 的边，并将 $u_{i,3}$ 向满足 $j < i$ 且 $l_i < a_j < r_i$ 的 $u_{j,2}$ 连一条 $+\infty$ 的边。其中边权为 ∞ 的边在实现中可以将边权设为一个足够大的数，这个数与 x 均需大于最终求出的答案。

这类似于一个二维数点问题，如果使用二维线段树可以做到边数 $\Theta(n \log^2 n)$，但很显然这作为一个静态问题，我们可以直接使用主席树做到 $\Theta(n \log n)$。

Solution

这种染成两种颜色之一要求总代价和最小的问题我们考虑建立最小割模型。

我们对每个结点 i 建立一个三个结点 $u_{i,1}, u_{i,2}, u_{i,3}$，将源点向 $u_{i,1}$ 连一条 $w_i + x$ 的边，$u_{i,1}$ 向 $u_{i,2}$ 连一条 $+\infty$ 的边，$u_{i,2}$ 向汇点连一条 $b_i + x$ 的边，$u_{i,1}$ 向 $u_{i,3}$ 连一条 p_i 的边，并将 $u_{i,3}$ 向满足 $j < i$ 且 $l_i < a_j < r_i$ 的 $u_{j,2}$ 连一条 $+\infty$ 的边。其中边权为 ∞ 的边在实现中可以将边权设为一个足够大的数，这个数与 x 均需大于最终求出的答案。

这类似于一个二维数点问题，如果使用二维线段树可以做到边数 $\Theta(n \log^2 n)$，但很显然这作为一个静态问题，我们可以直接使用主席树做到 $\Theta(n \log n)$。

时空复杂度与 n 个点 $\Theta(n \log n)$ 条边的有源汇最大流相同。