# 数据结构选讲（一）

crashed

必可（www.becoder.com.cn）

2077 年 8 月 32 日

# 目录

# 前言

近年的出题特点：

# 前言

近年的出题特点：

- 以常见树形 polylog 复杂度数据结构为主，尤其是**线段树。**
  (但是也会涉及到较为偏门的数据结构，例如 k-d tree)

# 前言

近年的出题特点：

- 以常见树形 polylog 复杂度数据结构为主，尤其是**线段树。**
  (但是也会涉及到较为偏门的数据结构，例如 k-d tree)
- 侧重于维护数据结构之算法，以及配套数据结构进行统计之算法的设计技巧。

# 前言

近年的出题特点：

- 以常见树形 polylog 复杂度数据结构为主，尤其是**线段树。**
  (但是也会涉及到较为偏门的数据结构，例如 k-d tree)
- 侧重于维护数据结构之算法，以及配套数据结构进行统计之算法的设计技巧。
- 同样强调“从具体情境中抽象出合适的数据及目标”的过程。

# 前言

数据结构本质上是要在数据和目标不变的情况下，优化算法复杂度，降低程序时间开销。

# 前言

数据结构本质上是要在数据和目标不变的情况下，优化算法复杂度，降低程序时间开销。

“从计算层面优化算法复杂度”问题的特点：
**情境复杂，没有 First Principle。**

# 前言

数据结构本质上是要在数据和目标不变的情况下，优化算法复杂度，降低程序时间开销。

“从计算层面优化算法复杂度”问题的特点：
**情境复杂，没有 First Principle。**

导致这样的实际情况：

- 需要细致地分析问题，见招拆招。
- “见多识广”、积累技巧可能有用。
- 既要充分调动经验，又忌轻易套用模型或经验。

# 前言

优化程序时间开销的一般思想：

# 前言

优化程序时间开销的一般思想：

- 分组处理：几乎一切数据结构（包括分块），二进制分组等。
- 减小计算量：记忆化，标记，采样；“支配”性质。
- 调度计算顺序：分治法，调换维度，预处理。
- 激发硬件性能：并行计算。

# 前言

如何优雅地写出 300 行代码?

- think twice, code once.
  在实现之前推敲各类讨论和关键细节，在实现过程中集中注意力。
- 在关键的地方留一些注释。
- 按照一定的顺序编写代码。
  我个人习惯的顺序是——先写程序的主要流程，再实现复杂数据结构，最后填写关键的方法。
- 每个函数的副作用尽量小，避免出现莫名其妙的相互作用（i.e. 少用全局变量)；数据结构最好可以做好封装。
- 用好 namespace。
- 注意清空！注意清空！注意清空!
- 可以有意识地积累错误和错因，避免犯第二次错。

# 线段树常见技术

线段树见得多，考得也多，所以我们先来单独看看线段树上的操作技巧。
这里主要涉及的有：

- zkw
- 历史信息
- 线段树二分
- 势能线段树
- 可持久化线段树
- 线段树合并和分裂

(排序和例题顺序无关)

# Warm up!

## 彩虹蛋糕

你需要维护两个初始为空的多重集 $L, C$，支持 $q$ 次操作，每次操作为：向 $L$ 或者 $C$ 中加入或删除**恰好一个元素**，元素形如 $(x_c, x_y)$。
每次操作结束后，你都需要计算以下值：

$$\min_{(a_c,a_y)\in L,(b_c,b_y)\in C}\{\max\{a_c+b_c, a_y+b_y\}\}$$

数据范围：$1 \le q \le 10^6, 1 \le c, y \le 10^9$。

# Warm up!

$\max$ 究竟该取哪一项可以由 $a_c - a_y$ 和 $b_y - b_c$ 的偏序来决定。
基于此，我们离线后将所有曾经加入过多重集的二元组都按照 $c - y$ 或 $y - c$ 进行排序，并构建线段树。
在自底向上合并的过程中，因为左右子树中二元组的 $c - y$ 是严格有序的，所以我们可以直接统计答案。
这样复杂度就是 $O(q \log q)$ 的。

# Warm up!

## 青蛙题

给定正整数 $n$ 和一个正整数序列 $a_1, \ldots, a_n$，需要解决 $q$ 次询问。
每次询问给出整数 $l, r$，你需要确定区间 $[l, r]$ 的满足以下条件的子段 $[l', r']$ 的最短长度：

$$\{a_{l'}, a_{l'+1}, \cdots, a_{r'}\} = \{a_l, a_{l+1}, \cdots, a_r\}$$

数据范围：$1 \le n, q \le 10^6$。

# Warm up!

既然问题不要求我们在线，那我们就离线。离线自然可以考虑扫描线解决问题。自然地，子段可以看作是“后缀的前缀”。这个想法引导我们，计算 $[l, r]$ 每个后缀的最短前缀，使得这个前缀与后缀自身的元素集合相同；那么询问基本被拆分成了一个区间 $\min$ 的形式。
但需要注意的是，我们需要保证后缀和 $[l, r]$ 的元素集合相同，因而需要先确定最短的后缀 $[l', r]$ 使得其元素集合与 $[l, r]$ 相同，再借助这个信息确定区间 $\min$ 的查询范围。
拆分成区间 $\min$ 的优点是，我们把问题以较小的代价弱化为了“维护多个最短前缀长度”。在对于右端点 $r$ 扫描线的同时，我们同时维护在每个左端点 $l$ 上维护 $[l, r]$ 的最短前缀长度。更新过程最终归结为“区间等差数列赋值”——相当容易使用标记解决问题。
最终复杂度为 $O((n+q)\log n)$。

# Warm up!

既然问题不要求我们在线，那我们就离线。离线自然可以考虑扫描线解决问题。自然地，子段可以看作是“后缀的前缀”。这个想法引导我们，计算 $[l, r]$ 每个后缀的最短前缀，使得这个前缀与后缀自身的元素集合相同；那么询问基本被拆分成了一个区间 $\min$ 的形式。
但需要注意的是，我们需要保证后缀和 $[l, r]$ 的元素集合相同，因而需要先确定最短的后缀 $[l', r]$ 使得其元素集合与 $[l, r]$ 相同，再借助这个信息确定区间 $\min$ 的查询范围。
拆分成区间 $\min$ 的优点是，我们把问题以较小的代价弱化为了“维护多个最短前缀长度”。在对于右端点 $r$ 扫描线的同时，我们同时维护在每个左端点 $l$ 上维护 $[l, r]$ 的最短前缀长度。更新过程最终归结为“区间等差数列赋值”——相当容易使用标记解决问题。
最终复杂度为 $O((n+q)\log n)$。

**Question.** 扫描线的另一种常见处理方法是，维护单点的历史最值。在这个问题中该思路可行吗？如果不行劣势在哪里？
**Question.** 序列问题常见解法还有序列分治和分块，它们在此可行吗？如果不行劣势在哪里？

# 例题一

## *LG P6773.* [NOI2020] 命运

给定一棵树 $T=(V,E)$ 和点对集合 $\mathcal{Q}\subset V\times V$，满足对于所有 $(u,v)\in\mathcal{Q}$，都有 $u\neq v$，并且 $u$ 是 $v$ 在树 $T$ 上的祖先。
求有多少个 $E$ 的子集 $F$，满足对于任何 $(u,v)\in\mathcal{Q}$，都存在 $u$ 到 $v$ 路径上的一条边 $e$ 使得 $e\in F$。输出答案对于 $998244353$ 取模后的结果。
数据范围：$1\leq|V|\leq 5\times 10^5, 1\leq|\mathcal{Q}|\leq 5\times 10^5$。

# 例题一

*LG P6773.* [NOI2020] 命运

给定一棵树 $T=(V,E)$ 和点对集合 $Q\subset V\times V$，满足对于所有 $(u,v)\in Q$，都有 $u\neq v$，并且 $u$ 是 $v$ 在树 $T$ 上的祖先。
求有多少个 $E$ 的子集 $F$，满足对于任何 $(u,v)\in Q$，都存在 $u$ 到 $v$ 路径上的一条边 $e$ 使得 $e\in F$。输出答案对于 $998244353$ 取模后的结果。
数据范围：$1\leq |V|\leq 5\times 10^5, 1\leq |Q|\leq 5\times 10^5$。

**Hint.** 尝试写一个 $O(n^2)$ 的 dp，并精准地写下转移。

# 例题一

*LG P6773.* [NOI2020] 命运

给定一棵树 $T=(V,E)$ 和点对集合 $\mathcal{Q}\subset V\times V$，满足对于所有 $(u,v)\in\mathcal{Q}$，都有 $u\neq v$，并且 $u$ 是 $v$ 在树 $T$ 上的祖先。
求有多少个 $E$ 的子集 $F$，满足对于任何 $(u,v)\in\mathcal{Q}$，都存在 $u$ 到 $v$ 路径上的一条边 $e$ 使得 $e\in F$。输出答案对于 $998244353$ 取模后的结果。
数据范围：$1\leq|V|\leq 5\times 10^5, 1\leq|\mathcal{Q}|\leq 5\times 10^5$。

**Hint.** 尝试写一个 $O(n^2)$ 的 dp，并精准地写下转移。
**Hint.** 尝试用线段树维护 dp 值。

# 例题一

*LG P6773.* [NOI2020] 命运

给定一棵树 $T=(V,E)$ 和点对集合 $\mathcal{Q}\subset V\times V$，满足对于所有 $(u,v)\in\mathcal{Q}$，都有 $u\neq v$，并且 $u$ 是 $v$ 在树 $T$ 上的祖先。
求有多少个 $E$ 的子集 $F$，满足对于任何 $(u,v)\in\mathcal{Q}$，都存在 $u$ 到 $v$ 路径上的一条边 $e$ 使得 $e\in F$。输出答案对于 $998244353$ 取模后的结果。
数据范围：$1\le|V|\le 5\times 10^5, 1\le|\mathcal{Q}|\le 5\times 10^5$。

**Hint.** 尝试写一个 $O(n^2)$ 的 dp，并精准地写下转移。
**Hint.** 尝试用线段树维护 dp 值。
**Hint.** 利用线段树合并在递归过程中产生的信息来模仿 dp 值的转移过程。

# 例题一（线段树合并）

我们假想将 $F$ 中的边染成黑色，将 $E \setminus F$ 中的边染成白色；将每个 $Q$ 中的二元组看作是一条返祖边。这样条件等价于，每条返祖边对应的路径上至少要有一条黑色边。

状态 $f_{u,d}$ 表示，确定 $u$ 的子树内所有边的颜色后，对于那些有一个点在 $u$ 子树内而另一个在外、并且对应路径上还没有黑色边的返祖边，$u$ 子树外端点最大深度为 $d$ 时的方案总数。

# 例题一（线段树合并）

计算过程描述如下：

## 状态转移

1. 首先将 $f_{u,i}$ 设为 $[i=0]$。
2. 假设我们需要将 $u$ 的一个儿子 $v$ 的信息合并上来，设 $u$ 的深度为 $d_u$。
   i. 首先，应当将所有 $i \geq d_u+1$ 的 $f_{v,i}$ 置为 $0$。
   ii. 考虑边 $(u,v)$ 的影响。引入辅助状态 $g_i$。
   如果将 $(u,v)$ 染成白色，则所有返祖边的状态不变，即 $g_i \xleftarrow{+} f_{v,i}$。
   如果将 $(u,v)$ 染成黑色，则“跨过”$(u,v)$ 边的所有返祖边路径上都会有一条黑边，从而“最大深度”更新为 $0$，即 $g_0 \xleftarrow{+} f_{v,i}$。
   iii. 考虑 $v$ 子树信息与 $u$ 原有信息进行合并。设合并后 $u$ 的状态为 $f'_u$。
   枚举 $u,v$ 两侧的最大深度 $i,j$，不难写出 $f'_{u,\max\{i,j\}} \xleftarrow{+} f_{u,i} \times g_j$。
3. 最后将较深一端在 $u$ 的返祖边纳入考虑。
   逐一枚举这样的返祖边，设当前枚举到的返祖边较浅一端深度为 $d'$，设考虑这条边后的 dp 为 $f'$；此边转移结束后即 $f \leftarrow f'$。
   枚举 $u$ 子树内原先的最大深度 $i$，加入这条边会导致 $f'_{u,\max\{d',i\}} \xleftarrow{+} f_{u,i}$，据此转移即可。

# 例题一（线段树合并）

如果没有 2.iii 这一步，那么整个运算可以用（动态开点）线段树很容易地解决，维护单点修改区间求和即可。

# 例题一（线段树合并）

如果没有 2.iii 这一步，那么整个运算可以用（动态开点）线段树很容易地解决，维护单点修改区间求和即可。

将 2.iii 进一步改写为：

$$f'_{u,i} = \sum_{j<i} (f_{u,j} \times g_i + f_{u,i} \times g_j) + f_{u,i} \times g_i$$

我们只需要计算好某一侧的 $f_u$ 或 $g$ 的和即可。注意到线段树合并的时候，外层有一个分治结构，我们自然可以在向右子树递归的时候，维护好路径上所有左儿子的加和。这样到递归终点的时候，问题就退化为了“区间乘法”，额外维护一个标记即可。

**Note.** 维护两种及以上标记时，需要注意标记下传顺序。

# 线段树分裂

既然都说到线段树合并了，不妨也来考虑一下线段树分裂。

线段树分裂的主要目标是：对于一棵值域上的线段树，按照某一个键值 $k$，将其分为两棵线段树；其中一棵仅保留键值 $\leq k$ 的叶子的信息，另一棵仅保留键值 $> k$ 的叶子的信息。

这类分裂问题和非旋 Treap 的分裂高度相似。我们考虑到键值为 $k$ 的叶子的路径，路径左子树并路径本身构成了 $\leq k$ 的信息，路径右子树构成了 $> k$ 的信息，所以我们实际只需要将到 $k$ 路径（不含叶子）复制一遍，然后将 $\leq k$ 和 $> k$ 的信息分别接上去即可。

# 例题二

## *Gym104976K.* Card Game

对于一个正整数序列 $b_1, b_2, \ldots, b_m$，我们称对它进行一次“接龙游戏”的结果为:

1. 初始时，令 $c = []$。
2. 枚举 $i$ 从 $1$ 到 $m$:
   - 如果 $c$ 中存在和 $b_i$ 相同的元素，则删去 $c$ 中第一个和 $b_i$ 相同的元素及该元素之后的所有元素。
   - 否则，将 $b_i$ 添加在 $c$ 的末尾。
3. 最终序列 $c$ 的长度即为游戏结果。

现在，给定一个正整数序列 $a_1, \ldots, a_n$，你需要解决 $q$ 次询问。
每次询问给出整数参数 $1 \le l \le r \le n$，回答对 $a_l, a_{l+1}, \ldots, a_r$ 进行一次“接龙游戏”的结果。
数据范围：$1 \le n, q \le 3 \times 10^5, 1 \le a_i \le n$，要求**强制在线**。

# 例题二

## *Gym104976K.* Card Game

对于一个正整数序列 $b_1, b_2, \ldots, b_m$，我们称对它进行一次“接龙游戏”的结果为：

1. 初始时，令 $c = []$。
2. 枚举 $i$ 从 1 到 $m$：
   - 如果 $c$ 中存在和 $b_i$ 相同的元素，则删去 $c$ 中第一个和 $b_i$ 相同的元素及该元素之后的所有元素。
   - 否则，将 $b_i$ 添加在 $c$ 的末尾。
3. 最终序列 $c$ 的长度即为游戏结果。

现在，给定一个正整数序列 $a_1, \ldots, a_n$，你需要解决 $q$ 次询问。
每次询问给出整数参数 $1 \le l \le r \le n$，回答对 $a_l, a_{l+1}, \ldots, a_r$ 进行一次“接龙游戏”的结果。
数据范围：$1 \le n, q \le 3 \times 10^5, 1 \le a_i \le n$，要求**强制在线**。

**Hint.** 从进行“接龙游戏”的正整数序列的最左侧元素入手考虑。

# 例题二（可持久化线段树）

我们从最左侧元素入手考虑，则可发现它能否留存下来仅仅取决于之后有没有元素和它相同。
更进一步地，对于序列 $b_1, \dots, b_m$，如果 $i > 1$ 为最小的满足 $b_i = b_1$ 的下标 (假定存在)，则这个序列进行“接龙”的结果和 $b_{i+1}, \dots, b_m$ 完全相同；如果这样的 $i$ 不存在，则结果等于 $b_2, \dots, b_m$ 进行“接龙”的结果 $+1$。

由此，我们不难想到：记 $f_{l,r}$ 为 $a_l, \dots, a_r$ 进行“接龙游戏”的结果，记 $g_i = \min(\{n+1\} \cup \{i < j \le n : b_j = b_i\})$，那么

$$f_{l,r} = \begin{cases} 0 & l > r \\ f_{l+1,r} + 1 & l \le r \wedge r < g_l \\ f_{g_l+1,r} & l \le r \wedge r \ge g_l \end{cases}$$

# 例题二（可持久化线段树）

考虑对于 $l$ 从右往左扫描线，并维护出每一个 $r \geq l$ 的 $f_{l,r}$。
则我们实际需要执行的操作是：

1. 将 $f_{l+1}$ 的一段区间 $+1$。
2. 将 $f_{l+1}$ 和 $f_{g_l+1}$ 各自的一段区间分离后拼接起来。

# 例题二（可持久化线段树）

考虑对于 $l$ 从右往左扫描线，并维护出每一个 $r \geq l$ 的 $f_{l,r}$。
则我们实际需要执行的操作是：

1. 将 $f_{l+1}$ 的一段区间 $+1$。
2. 将 $f_{l+1}$ 和 $f_{g_l+1}$ 各自的一段区间分离后拼接起来。

看起来需要访问历史信息，是否需要可持久化平衡树？

# 例题二（可持久化线段树）

考虑对于 $l$ 从右往左扫描线，并维护出每一个 $r \geq l$ 的 $f_{l,r}$。
则我们实际需要执行的操作是：

1. 将 $f_{l+1}$ 的一段区间 $+1$。
2. 将 $f_{l+1}$ 和 $f_{g_l+1}$ 各自的一段区间分离后拼接起来。

看起来需要访问历史信息，是否需要可持久化平衡树?

注意到**分离出来的子区间在重新拼接之后，相对顺序并没有发生改变**，所以可持久化线段树（静态结构）即可胜任该任务。
一种有助于理解的设想：在 $f_{l+1}$ 上定位区间 $[l+1, g_l-1]$，在 $f_{g_l+1}$ 上定位区间 $[g_l, n]$，这样我们得到了 $O(\log n)$ 棵子线段树。现在我们在这些结点的基础上重新构建线段树就得到了拼接结果。

# 例题二（可持久化线段树）

回头考虑 1，我们只需要支持可持久化线段树上的标记即可。
一般来说，如果标记可以永久化，则添加标记相对容易；相较于普通线段树，只需要在复制结点的时候把标记也一块复制上即可。本题就是这种情况。

# 例题二（可持久化线段树）

回头考虑 1，我们只需要支持可持久化线段树上的标记即可。
一般来说，如果标记可以永久化，则添加标记相对容易；相较于普通线段树，只需要在复制结点的时候把标记也一块复制上即可。本题就是这种情况。

如果标记不能永久化，则必须要在进行一切操作之前把标记下放；因为自己的子节点也可能作为其它结点的子节点，所以不能直接修改，只能复制子节点后修改信息。这种操作并不会导致复杂度被破坏，但是会导致常数显著增大。

# 例题三

## *LG P8868.* [NOIP2022] 比赛

给定正整数 $n$ 和两个正整数序列 $a_1, \ldots, a_n$ 和 $b_1, \ldots, b_n$，你需要回答 $q$ 次询问。每次询问给出参数 $1 \le p \le q \le n$，你需要回答：

$$\sum_{l=p}^{q} \sum_{r=l}^{q} \left( \max_{i=l}^{r} a_i \right) \left( \max_{i=l}^{r} b_i \right)$$

数据范围：$1 \le n, Q \le 2.5 \times 10^5$，$\{a\}, \{b\}$ 分别构成 $1 \sim n$ 的排列。

# 例题三（历史信息）

多个询问之间没有关系，所以我们可考虑将询问离线处理。
假如只需要回答多个区间的 $\max a$，则我们可以扫描 $+$ 单调栈解决问题。

# 例题三（历史信息）

多个询问之间没有关系，所以我们可考虑将询问离线处理。
假如只需要回答多个区间的 $\max a$，则我们可以扫描 $+$ 单调栈解决问题。

假如我们可以维护好 $(\max a)(\max b)$ 的区间和，我们就可以通过“历史和”的技术得到“子区间和”。
注意到，向序列末尾加入一个数，实际上是将后缀最大值的某一段赋值。从而被完整覆盖的区间上，$\sum(\max a)(\max b)$ 的新值可以由 $\sum(\max a)$ 或者 $\sum(\max b)$ 以及所赋的值确定。
所以，只需要维护好区间内的 $\sum(\max a)(\max b), \sum(\max a), \sum(\max b)$ 和一些赋值标记。

# 例题三（历史信息）

多个询问之间没有关系，所以我们可考虑将询问离线处理。
假如只需要回答多个区间的 $\max a$，则我们可以扫描 $+$ 单调栈解决问题。

假如我们可以维护好 $(\max a)(\max b)$ 的区间和，我们就可以通过“历史和”的技术得到“子区间和”。
注意到，向序列末尾加入一个数，实际上是将后缀最大值的某一段赋值。从而被完整覆盖的区间上，$\sum(\max a)(\max b)$ 的新值可以由 $\sum(\max a)$ 或者 $\sum(\max b)$ 以及所赋的值确定。
所以，只需要维护好区间内的 $\sum(\max a)(\max b), \sum(\max a), \sum(\max b)$ 和一些赋值标记。

延拓到“历史和”的尝试中，一个观察是，需要维护的 $\sum\sum(\max a)(\max b), \sum(\max a)(\max b), \sum(\max a), \sum(\max b)$ 之间的相互运算总是**线性**的，从而我们可以维护它们组成的向量，并使用矩阵刻画标记。
为了优化算法常数，一种方法是将矩阵拆开，并只维护可能非零的值。

# 例题四

*UOJ164.* 【清华集训 2015】V

给定正整数 $n$ 和非负整数序列 $a_1, \dots, a_n$，需要解决 $q$ 次询问/操作：

1. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 加上 $x$。
2. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 赋为 $\max\{a_i - x, 0\}$。
3. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 赋为 $x$。
4. 给定整数 $y$，询问 $a_y$。
5. 给定整数 $y$，询问从开始操作前到现在 $a_y$ 的历史最大值。

数据范围：$1 \le n, m \le 5 \times 10^5, 0 \le a_i, x \le 10^9, 1 \le l \le r \le n, 1 \le y \le n$。

*UOJ164.* 【清华集训 2015】V

给定正整数 $n$ 和非负整数序列 $a_1, \ldots, a_n$，需要解决 $q$ 次询问/操作：

1. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 加上 $x$。
2. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 赋为 $\max\{a_i - x, 0\}$。
3. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 赋为 $x$。
4. 给定整数 $y$，询问 $a_y$。
5. 给定整数 $y$，询问从开始操作前到现在 $a_y$ 的历史最大值。

数据范围：$1 \le n, m \le 5 \times 10^5, 0 \le a_i, x \le 10^9, 1 \le l \le r \le n, 1 \le y \le n$。

**Hint.** 尝试用统一的形式来刻画前三种操作对于单个元素的影响，如函数及函数复合、矩阵及矩阵乘法。

# 例题四（历史信息）

Approach 1.
操作 2 的 $\max$ 和询问 4、询问 5 的 $\max$ 正好是匹配的，所以我们可以用 $(\mathbb{N}, \max, +)$ 上的矩阵统一地刻画一切操作。
这样本题就自然地转化为了前一道题目。

# 例题四（历史信息）

Approach 2.
既然操作 2 需要维护形如 $\max\{0, a_i - x\}$ 的变换，我们不妨将这个函数改造一下以适应所有的操作。
考虑由参数 $a, b$ 确定的变换 $\sigma_{a,b} : x \mapsto \max\{x + a, b\}$，则：

- 变换的复合形如 $\sigma_{c,d} \circ \sigma_{a,b} = \sigma_{a+c,\max\{b+c,d\}}$。
- 操作一可以用 $\sigma_{x,x}$ 来描述。
- 操作二可以用 $\sigma_{-x,0}$ 来描述。
- 操作三可以用 $\sigma_{-\infty,x}$ 来描述。

这样前四种操作/询问都可以被一网打尽。

# 例题四（历史信息）

额外考虑历史最值信息如何维护。

# 例题四（历史信息）

额外考虑历史最值信息如何维护。

我们先假设，已知上一次释放标记后得到的值 $x$，**plus 中间累计的每一次修改的具体信息**，按时序排列为 $\sigma_1, \ldots, \sigma_t$。
那么在这段过程中产生的历史最值就是：

$$\max_{i=1}^{t} \left( (\sigma_i \circ \sigma_{i-1} \circ \cdots \circ \sigma_1)(x) \right)$$

根据 $\max$ 的结合律、交换律，这一团东西依然可以被表述为一个 $\sigma$！

# 例题四（历史信息）

额外考虑历史最值信息如何维护。

我们先假设，已知上一次释放标记后得到的值 $x$，**plus 中间累计的每一次修改的具体信息**，按时序排列为 $\sigma_1, \ldots, \sigma_t$。
那么在这段过程中产生的历史最值就是：

$$\max_{i=1}^{t} \left( (\sigma_i \circ \sigma_{i-1} \circ \cdots \circ \sigma_1)(x) \right)$$

根据 $\max$ 的结合律、交换律，这一团东西依然可以被表述为一个 $\sigma$！

那么，历史最值信息的标记下传，**就是把两队标记拼接在一起（父亲的标记总是发生在儿子的标记之后）。**
手推标记队列合并的算法即可。复杂度仍然是 $O(n \log n)$。

# 历史信息：参考练习

*CF1290E.* Cartesian Tree
*LG P6109.* [Ynoi2009] rprmq1
*LG P9057.* [Ynoi2004] rpfrdtzls
*LG P9990.* [Ynoi Easy Round 2023] TEST90

# 例题五

*LOJ6029.* 「雅礼集训 2017 Day1」市场

给定正整数 $n$ 和一个整数序列 $a_0, \sim, a_{n-1}$，需要解决 $q$ 次询问/操作：

1. 给定整数 $l, r, c$，将所有满足 $l \leq i \leq r$ 的 $a_i$ 加上 $c$。
2. 给定整数 $l, r, d$，将所有满足 $l \leq i \leq r$ 的 $a_i$ 变为 $\left\lfloor \frac{a_i}{d} \right\rfloor$。
3. 给定整数 $l, r$，求 $\min_{l \leq i \leq r} a_i$。
4. 给定整数 $l, r$，求 $\sum_{i=l}^{r} a_i$。

数据范围：$1 \leq n, q \leq 10^5, 0 \leq l \leq r \leq n-1, -10^4 \leq c \leq 10^4, 2 \leq d \leq 10^9$。

# 例题五

*LOJ6029.* 「雅礼集训 2017 Day1」市场

给定正整数 $n$ 和一个整数序列 $a_0, \sim, a_{n-1}$，需要解决 $q$ 次询问/操作：

1. 给定整数 $l, r, c$，将所有满足 $l \le i \le r$ 的 $a_i$ 加上 $c$。
2. 给定整数 $l, r, d$，将所有满足 $l \le i \le r$ 的 $a_i$ 变为 $\left\lfloor \frac{a_i}{d} \right\rfloor$。
3. 给定整数 $l, r$，求 $\min_{l \le i \le r} a_i$。
4. 给定整数 $l, r$，求 $\sum_{i=l}^{r} a_i$。

数据范围：$1 \le n, q \le 10^5, 0 \le l \le r \le n-1, -10^4 \le c \le 10^4, 2 \le d \le 10^9$。

**Hint.** 注意验证自己想法的正确性（尤其是时间复杂度的）。

# 例题五（势能线段树）

操作二没法直接标记解决，所以我们只能先全部递归下去。
但是“取整除”的特殊性提醒我们，“有效”的取整除操作次数是可以被控制的。
“有效”意味着区间中至少有一个数在取整除过程中发生改变。如果不发生改变，这样的数只能是 $0$ 或者 $-1$。
所以我们可以给每个区间，统计出“是否区间中的所有数都是 $0$ 或者 $-1$”，据此判断操作二是否需要进一步递归。

# 例题五（势能线段树）

操作二没法直接标记解决，所以我们只能先全部递归下去。
但是“取整除”的特殊性提醒我们，“有效”的取整除操作次数是可以被控制的。
“有效”意味着区间中至少有一个数在取整除过程中发生改变。如果不发生改变，这样的数只能是 $0$ 或者 $-1$。
所以我们可以给每个区间，统计出“是否区间中的所有数都是 $0$ 或者 $-1$”，据此判断操作二是否需要进一步递归。

然后发现，操作一的存在可以轻而易举地把复杂度弄坏：一个初始为全 $0$ 的序列，每次全局 $+c$ 后反复全局操作二直到序列重新变为全 $0$。在这种情况下，单次操作二会退化到 $O(n)$。
怎么修锅呢？

# 例题五（势能线段树）

自然，我们可以发现，如果一个区间内只有一种数，那么这个区间上的取整除可以打标记解决。
再拓展一下，如果区间内的数在执行一次取整除之后，得到的结果相同，则可以直接打一个区间赋值标记。
想要让它跑很慢也很容易，只需要把序列弄成“犬牙交错”的形态即可，比方说设置为 $2^k$ 和 $2^k - 1$ 交替出现，然后多次执行 $d = 2$ 的操作二。

# 例题五（势能线段树）

自然，我们可以发现，如果一个区间内只有一种数，那么这个区间上的取整除可以打标记解决。
再拓展一下，如果区间内的数在执行一次取整除之后，得到的结果相同，则可以直接打一个区间赋值标记。
想要让它跑很慢也很容易，只需要把序列弄成“犬牙交错”的形态即可，比方说设置为 $2^k$ 和 $2^k-1$ 交替出现，然后多次执行 $d=2$ 的操作二。

不妨先考虑，这种“犬牙交错”的区间怎么快速处理。
其实，可以发现，在上面的反例中，数之间的差没有变，故**取整除其实等效于区间加**。

# 例题五（势能线段树）

自然，我们可以发现，如果一个区间内只有一种数，那么这个区间上的取整除可以打标记解决。
再拓展一下，如果区间内的数在执行一次取整除之后，得到的结果相同，则可以直接打一个区间赋值标记。
想要让它跑很慢也很容易，只需要把序列弄成“犬牙交错”的形态即可，比方说设置为 $2^k$ 和 $2^k-1$ 交替出现，然后多次执行 $d=2$ 的操作二。

不妨先考虑，这种“犬牙交错”的区间怎么快速处理。
其实，可以发现，在上面的反例中，数之间的差没有变，故**取整除其实等效于区间加。**

进一步探讨这种情况出现的条件。
我们考虑除数为 $d$，区间内最大值为 $a=pd+r$，最小值为 $b=qd+s$（其中 $0\le r,s\le d-1$），则等效条件是：

$$p-q=(pd+r)-(qd+s)\Leftrightarrow s-r=(d-1)(p-q)$$

等号成立当且仅当 $p=q\wedge s=r$ 或者 $p=q+1\wedge s=d-1\wedge r=0$；在这种条件下可以直接区间加。

# 例题五（势能线段树）

基于上面的讨论，我们可以得到两种剪枝策略：

1. 如果区间内的数在取整除后的结果相同（通过最大值和最小值去判断），则可以直接打区间赋值标记。
2. 如果区间内的数在取整除后，结果与原数的差值相同，则可以直接打区间加法标记。

下面来分析一下复杂度。特别地，我们指的是操作二带来的“额外”递归的复杂度。

# 例题五（势能线段树）

基于上面的讨论，我们可以得到两种剪枝策略：

1. 如果区间内的数在取整除后的结果相同（通过最大值和最小值去判断），则可以直接打区间赋值标记。
2. 如果区间内的数在取整除后，结果与原数的差值相同，则可以直接打区间加法标记。

下面来分析一下复杂度。特别地，我们指的是操作二带来的“额外”递归的复杂度。

设结点 $x$ 对应区间内的最大值为 $a_x$，最小值为 $b_x$。
一个观察是，在 $a_x - b_x \le 1$ 的时候，我们就可以终止取整除的递归了。
自然，可以将结点势能设置为 $\phi(u) = \log_2 \max\{1, a_x - b_x\}$，总势能设置为 $\Phi = \sum_u \phi(u)$。
由上界 $\lfloor \frac{a}{d} \rfloor - \lfloor \frac{b}{d} \rfloor \le \lceil \frac{a-b}{d} \rceil$，通过分析可以发现，每次额外递归必然导致 $\phi(u)$ 减少至少 $1$，所以额外递归次数可以被总势能及其增量之和控制住。
每次操作一带来的势能变化量，发生在那些“和操作区间有交集但没有被完全覆盖”的结点上。这样的结点数目为 $O(\log n)$，带来的势能变化量可以被 $\log_2 |c|$ 控制。从而，总势能不超过 $O(n \log |a| + q \log n \log |c|)$。

### 初始问题

用线段树维护一个整数序列，需要支持区间取 $\min$ 和区间求和。

# Segment Tree Beats!

**初始问题**

用线段树维护一个整数序列，需要支持区间取 $\min$ 和区间求和。

区间取 $\min$ 和区间求和并不能很好地用标记"兼容"。我们又只能一路递归到叶子修改吗?
但是，一个观察是，如果区间取 $\min$ 仅会影响到至多一种值（很显然，只可能是 $\max$)，则可以快速地计算出区间和的变化量；这种情况就无需继续递归了。
为此，我们只需要维护好区间最大值、区间次大值和区间最大值的个数。

**初始问题**

用线段树维护一个整数序列，需要支持区间取 $\min$ 和区间求和。

区间取 $\min$ 和区间求和并不能很好地用标记“兼容”。我们又只能一路递归到叶子修改吗？
但是，一个观察是，如果区间取 $\min$ 仅会影响到至多一种值（很显然，只可能是 $\max$），则可以快速地计算出区间和的变化量；这种情况就无需继续递归了。
为此，我们只需要维护好区间最大值、区间次大值和区间最大值的个数。

下面来分析复杂度，假定要对区间内的数对于 $x$ 取 $\min$。
每次我们需要“额外”递归的时候，就必然有出现了“次大值小于 $x$”的情况；操作过后，**原先的次大值和最大值会变成同样的值。**
所以，我们可以令线段树结点 $u$ 的势能为：对应区间内部的数的种数。每次额外递归就会导致经过结点的势能减小至少 $1$。总势能为 $O(n \log n)$，从而复杂度可以被控制。

# Segment Tree Beats!

## 略作延伸

用线段树维护一个整数序列，需要支持区间加、区间取 $\min$ 和区间求和。

**略作延伸**

用线段树维护一个整数序列，需要支持区间加、区间取 $\min$ 和区间求和。

可以发现，区间加标记容易直接融入前一个问题的维护方法中。
实现是容易的，但之前的势能分析会被破坏——区间加会导致线段树结点上数的种数发生显著变化。
通过均摊分析，可以证明以上算法的的复杂度为 $O(n\log n + m\log^2 n)$。
c.f. 吉如一，区间最值操作和历史最值问题，2016 年集训队论文集。

**Remark.** 论文同样指出，同时进行两个方向的区间最值操作时，使用类似的技术可以得到复杂度为 $O(n\log^2 n)$。

# 例题六

## *LOJ6565.* 最假女选手

给定 $n$ 和一个整数序列 $a_1, \ldots, a_n$，需要解决 $q$ 次操作/询问：

1. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 加上 $x$。
2. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 变为 $\max\{a_i, x\}$。
3. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 变为 $\min\{a_i, x\}$。
4. 给定整数 $l, r$，求 $\sum_{i=l}^{r} a_i$。
5. 给定整数 $l, r$，求 $\max_{i=l}^{r} a_i$。
6. 给定整数 $l, r$，求 $\min_{i=l}^{r} a_i$。

数据范围：$1 \le n, q \le 5 \times 10^5, |a_i| \le 10^8$，操作一的 $|x| \le 1000$，操作二、三的 $|x| \le 10^8$。

# 例题六

*LOJ6565. 最假女选手*

给定 $n$ 和一个整数序列 $a_1, \dots, a_n$，需要解决 $q$ 次操作/询问：

1. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 加上 $x$。
2. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 变为 $\max\{a_i, x\}$。
3. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 变为 $\min\{a_i, x\}$。
4. 给定整数 $l, r$，求 $\sum_{i=l}^{r} a_i$。
5. 给定整数 $l, r$，求 $\max_{i=l}^{r} a_i$。
6. 给定整数 $l, r$，求 $\min_{i=l}^{r} a_i$。

数据范围：$1 \le n, q \le 5 \times 10^5, |a_i| \le 10^8$，操作一的 $|x| \le 1000$，操作二、三的 $|x| \le 10^8$。

**Hint.** 没有 hint。

# 例题六（势能线段树，Segment Tree Beats!）

我们可以使用相似的方法解决问题。
为了支持区间取 $\min$ 和 $\max$，我们同时维护区间内的：最大值、最大值个数、严格次大值；最小值、最小值个数、严格次小值。更新操作和原始的 Segment Tree Beats 如出一辙，然而需要注意的是，**如果区间内只有两种数，则更新最大值时需要改变次小值，vice versa。**
从而我们得到了一个复杂度为 $O(n\log^2 n)$ 的算法。

# Segment Tree Beats!

上述问题中，我们发现，Segment Tree Beats 的实质就是把区间最值操作用 $\log n$ 的代价转化为了针对区间最值的加减操作。

这个想法可以帮助我们解决更多问题：

# Segment Tree Beats!

上述问题中，我们发现，Segment Tree Beats 的实质就是把区间最值操作用 $\log n$ 的代价转化为了针对区间最值的加减操作。

这个想法可以帮助我们解决更多问题：

## Mzl loves segment tree

给定正整数 $n$ 和整数序列 $a_1, \ldots, a_n$，同时还有一个初始全为 $0$ 的整数序列 $b$，需要解决 $q$ 次操作：

1. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 变为 $\min\{a_i, x\}$。
2. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 变为 $\max\{a_i, x\}$。
3. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 加上 $x$。
4. 给定整数 $l, r$，求 $\sum_{i=l}^{r} b_i$。

每次操作后，对所有 $1 \le i \le n$，如果 $a_i$ 的值因为本次操作发生了变化，则将 $b_i$ 加上一，否则不变。
数据范围：$1 \le n, q \le 3 \times 10^5$。

# Segment Tree Beats!

*LG U180387.* CTSN loves segment tree

给定正整数 $n$ 和两个整数序列 $\{a_i\}_{i=1}^n, \{b_i\}_{i=1}^n$，需要解决 $q$ 次操作：

1. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 变为 $\min\{a_i, x\}$。
2. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $b_i$ 变为 $\min\{b_i, x\}$。
3. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 加上 $x$。
4. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $b_i$ 加上 $x$。
5. 给定整数 $l, r$，求 $\max_{i=l}^r a_i + b_i$。

数据范围：$1 \le n, q \le 3 \times 10^5$。

# Segment Tree Beats! 参考练习

*UOJ515.* 【UR 19】前进四
*UOJ288.* 基础数据结构练习题
*LG P6242.* 【模板】线段树 3（区间最值操作、区间历史最值）

# 例题七

## 赛格蒙特彼茨

给定正整数 $n$ 和整数序列 $a_1, \ldots, a_n$，同时还有一个初始和 $a$ 相同的序列 $b$，需要解决 $q$ 次操作：

1. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 变为 $\max\{a_i, x\}$。
2. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 加上 $x$。
3. 给定整数 $l, r$，求 $\sum_{i=l}^{r} b_i$。

每次操作后，对所有 $1 \le i \le n$，令 $b_i \leftarrow \min\{b_i, a_i\}$。
数据范围：$1 \le n, m \le 10^5$。

# 例题七

## 赛格蒙特彼茨

给定正整数 $n$ 和整数序列 $a_1, \ldots, a_n$，同时还有一个初始和 $a$ 相同的序列 $b$，需要解决 $q$ 次操作：

1. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 变为 $\max\{a_i, x\}$。
2. 给定整数 $l, r, x$，将所有满足 $l \le i \le r$ 的 $a_i$ 加上 $x$。
3. 给定整数 $l, r$，求 $\sum_{i=l}^{r} b_i$。

每次操作后，对所有 $1 \le i \le n$，令 $b_i \leftarrow \min\{b_i, a_i\}$。
数据范围：$1 \le n, m \le 10^5$。

**Hint.** 考察序列 $a - b$。

# 例题七（历史信息 revisit）

前面介绍的线性变换观点就不起作用了——没有办法计算历史最值的区间和。

在求和背景下，我们可以对于 $b$ 作适当拆分——即转而维护 $c = a - b$。这样操作二就变成了对于区间内的 $c$ 执行 $c \leftarrow \max\{c + x, 0\}$，可以使用 $c$ 上的 Segment Tree Beats 技术维护。

接下来，操作一虽然涉及 $\max$，但是我们可以用 $a$ 上的 Segment Tree Beats 技术，将问题转化为针对区间最值的加减操作；这样变化就可以很容易地投射到 $c$ 上去维护了。

# 例题七（历史信息 revisit）

前面介绍的线性变换观点就不起作用了——没有办法计算历史最值的区间和。

在求和背景下，我们可以对于 $b$ 作适当拆分——即转而维护 $c = a - b$。这样操作二就变成了对于区间内的 $c$ 执行 $c \leftarrow \max\{c + x, 0\}$，可以使用 $c$ 上的 Segment Tree Beats 技术维护。

接下来，操作一虽然涉及 $\max$，但是我们可以用 $a$ 上的 Segment Tree Beats 技术，将问题转化为针对区间最值的加减操作；这样变化就可以很容易地投射到 $c$ 上去维护了。

**Remark.** 这样的观点也可以延伸到历史和上。

# 例题八

## *Nowcoder.* 牛半仙的妹子序列

给定一个 $1 \sim n$ 的排列 $p$。
我们称 $p$ 的一个非空子序列 $\varnothing \neq \{i_1, i_2, \dots, i_m\} \subset \{1, 2, \dots, n\}$（其中 $i_1 < i_2 < \dots < i_m$）是“好”的，当且仅当以下条件均满足：

1. 对所有 $1 \le k \le m-1$ 有 $p_{i_k} < p_{i_{k+1}}$。
2. 不存在 $1 \le j \le i_1 - 1$ 使得 $p_j < p_{i_1}$。
3. 不存在 $i_m + 1 \le j \le n$ 使得 $p_{i_m} < p_j$。
4. 对所有 $1 \le k \le m-1$，不存在 $i_k + 1 \le j \le i_{k+1} - 1$ 使得 $p_{i_k} < p_j < p_{i_{k+1}}$。

统计 $p$ 的“好”的非空子序列数目，答案对于 $998244353$ 取模。
数据范围：$1 \le n \le 2 \times 10^5$。

# 例题八（线段树二分）

容易想到写出一个计数 dp（状态记为 $f_i$），具体转移其实就是照着四个条件来搞。我们需要加速转移过程。

# 例题八（线段树二分）

容易想到写出一个计数 dp（状态记为 $f_i$），具体转移其实就是照着四个条件来搞。我们需要加速转移过程。

发现和转移有关的限制其实只有条件一和条件四。假设当前正在计算 $f_i$，正在考虑 $f_j$ 向 $f_i$ 的贡献（$1 \le j < i \le n$），则条件为：

1. $p_j < p_i$。
2. 不存在 $j < k < i$ 使得 $p_j < p_k < p_i$。

# 例题八（线段树二分）

容易想到写出一个计数 dp（状态记为 $f_i$），具体转移其实就是照着四个条件来搞。我们需要加速转移过程。

发现和转移有关的限制其实只有条件一和条件四。假设当前正在计算 $f_i$，正在考虑 $f_j$ 向 $f_i$ 的贡献（$1 \le j < i \le n$），则条件为：

1. $p_j < p_i$。
2. 不存在 $j < k < i$ 使得 $p_j < p_k < p_i$。

条件引导我们使用值域上的数据结构来解决问题。
自然地，我们引入 $p$ 作为置换的逆 $q$，并转而考虑 $f_{q_t}$ 向 $f_i$ 的贡献（$1 \le t < p_i$），条件为：

1. $q_t < i$。
2. 不存在 $t < s < p_i$ 使得 $q_t < q_s < i$。

倘若我们将 $q$ 预先设置为极小值，并在扫描过程中动态更新 $q$，则 $q_t < i$ 自动成立了。于是，第二个条件弱化为了 “$q_t$ 是前缀 $[1, i]$ 上的后缀极大值”。

# 例题八（线段树二分）

考虑可否在线段树区间上，维护出区间内的后缀最值位置上的 $f_{q_i}$ 之和。

假想我们已经拿到了两个子区间各自的后缀最大值（下标）的序列——左区间为 $r_1, \ldots, r_p$，右区间为 $s_1, \ldots, s_q$——我们考虑这个合并的过程是怎么进行的。

# 例题八（线段树二分）

考虑可否在线段树区间上，维护出区间内的后缀最值位置上的 $f_{q_i}$ 之和。

假想我们已经拿到了两个子区间各自的后缀最大值（下标）的序列——左区间为 $r_1, \dots, r_p$，右区间为 $s_1, \dots, s_q$——我们考虑这个合并的过程是怎么进行的。实际上，无非是将 $r$ 的一段后缀弹出，直到尾巴上的最后一个下标满足 $q$ 值 $> q_{s_1}$，然后将两个序列拼在一起。根据单调性，弹出的一段可以通过线段树二分确定。

# 例题八（线段树二分）

考虑可否在线段树区间上，维护出区间内的后缀最值位置上的 $f_{q_i}$ 之和。

假想我们已经拿到了两个子区间各自的后缀最大值（下标）的序列——左区间为 $r_1, \ldots, r_p$，右区间为 $s_1, \ldots, s_q$——我们考虑这个合并的过程是怎么进行的。实际上，无非是将 $r$ 的一段后缀弹出，直到尾巴上的最后一个下标满足 $q$ 值 $> q_{s_1}$，然后将两个序列拼在一起。根据单调性，弹出的一段可以通过线段树二分确定。

具体实现时，我们需要在每个结点上维护好区间最大值、后缀最大值对应的 dp 值之和以及**左子树为了合并弹掉一个尾巴后，其后缀最大值对应的 dp 值之和** (这个值记为 $s$)。
合并时，我们实际上就是要确定 $s$；为此我们拿着右子树的最大值进入左子树二分，寻找最靠右的大于此值的值。与此同时，如果我们在二分过程中递归入结点 $x$ 的右子树 $z$，就说明 $z$ 的最大值可以留存下来，**从而 $z$ 的兄弟左子树的贡献即为 $x$ 储存的 $s$**。将这些贡献求和即可得到所需贡献。复杂度为 $O(n \log^2 n)$。

# 例题八（线段树二分）

考虑可否在线段树区间上，维护出区间内的后缀最值位置上的 $f_{q_i}$ 之和。

假想我们已经拿到了两个子区间各自的后缀最大值（下标）的序列——左区间为 $r_1, \ldots, r_p$，右区间为 $s_1, \ldots, s_q$——我们考虑这个合并的过程是怎么进行的。实际上，无非是将 $r$ 的一段后缀弹出，直到尾巴上的最后一个下标满足 $q$ 值 $> q_{s_1}$，然后将两个序列拼在一起。根据单调性，弹出的一段可以通过线段树二分确定。

具体实现时，我们需要在每个结点上维护好区间最大值、后缀最大值对应的 dp 值之和以及**左子树为了合并弹掉一个尾巴后，其后缀最大值对应的 dp 值之和**(这个值记为 $s$)。
合并时，我们实际上就是要确定 $s$；为此我们拿着右子树的最大值进入左子树二分，寻找最靠右的大于此值的值。与此同时，如果我们在二分过程中递归入结点 $x$ 的右子树 $z$，就说明 $z$ 的最大值可以留存下来，**从而 $z$ 的兄弟左子树的贡献即为 $x$ 储存的 $s$**。将这些贡献求和即可得到所需贡献。复杂度为 $O(n\log^2 n)$。

**Remark.** 本题可能也存在分治、分块等其它更易构建或更快的做法，不过和主题无关。

# 例题九

## 黑白树

给定一棵大小为 $n$ 且以 $1$ 为根的有根树，树上有黑白两种颜色的结点。维护以下五种操作：

1. 改变一个点的颜色，即黑变白、白变黑。
2. 使结点 $x$ 所在的同色点连通块上的每个结点加上 $\delta$。
3. 查询 $x$ 的同色连通块内的点权最大值。
4. 使 $x$ 到 $y$ 的简单路径上的所有结点权值加上 $\delta$。
5. 使 $x$ 子树内所有结点权值加上 $\delta$。

数据范围：$1 \le n, m \le 2 \times 10^5, 0 \le \delta \le 10^5$，且保证答案总 $< 2^{31}$。

# 例题九

## 黑白树

给定一棵大小为 $n$ 且以 $1$ 为根的有根树，树上有黑白两种颜色的结点。
维护以下五种操作：

1. 改变一个点的颜色，即黑变白、白变黑。
2. 使结点 $x$ 所在的同色点连通块上的每个结点加上 $\delta$。
3. 查询 $x$ 的同色连通块内的点权最大值。
4. 使 $x$ 到 $y$ 的简单路径上的所有结点权值加上 $\delta$。
5. 使 $x$ 子树内所有结点权值加上 $\delta$。

数据范围：$1 \le n, m \le 2 \times 10^5, 0 \le \delta \le 10^5$，且保证答案总 $< 2^{31}$。

**Hint.** 从 DFS 序 + 线段树的角度出发。

# 例题九

## 黑白树

给定一棵大小为 $n$ 且以 $1$ 为根的有根树，树上有黑白两种颜色的结点。
维护以下五种操作：

1. 改变一个点的颜色，即黑变白、白变黑。
2. 使结点 $x$ 所在的同色点连通块上的每个结点加上 $\delta$。
3. 查询 $x$ 的同色连通块内的点权最大值。
4. 使 $x$ 到 $y$ 的简单路径上的所有结点权值加上 $\delta$。
5. 使 $x$ 子树内所有结点权值加上 $\delta$。

数据范围：$1 \leq n, m \leq 2 \times 10^5, 0 \leq \delta \leq 10^5$，且保证答案总 $< 2^{31}$。

**Hint.** 从 DFS 序 $+$ 线段树的角度出发。
**Hint.** 将树上连通块想象成：从连通块的最高点挖去不属于连通块的子树。

# 例题九

## 黑白树

给定一棵大小为 $n$ 且以 $1$ 为根的有根树，树上有黑白两种颜色的结点。
维护以下五种操作：

1. 改变一个点的颜色，即黑变白、白变黑。
2. 使结点 $x$ 所在的同色点连通块上的每个结点加上 $\delta$。
3. 查询 $x$ 的同色连通块内的点权最大值。
4. 使 $x$ 到 $y$ 的简单路径上的所有结点权值加上 $\delta$。
5. 使 $x$ 子树内所有结点权值加上 $\delta$。

数据范围：$1 \le n, m \le 2 \times 10^5, 0 \le \delta \le 10^5$，且保证答案总 $< 2^{31}$。

**Hint.** 从 DFS 序 $+$ 线段树的角度出发。
**Hint.** 将树上连通块想象成：从连通块的最高点挖去不属于连通块的子树。
**Hint.** 考虑为线段树结点打上特殊标记，这种标记可以影响其它标记下传和信息更新。

# 例题九（无平凡交区间的维护）

操作四、五的存在使得我们不得不引入重链剖分 + 线段树的结构。所需维护的“最大值”使得我们不能考虑差分之类的技术，只能 bottom-up 合并结果。

# 例题九（无平凡交区间的维护）

操作四、五的存在使得我们不得不引入重链剖分 + 线段树的结构。所需维护的“最大值”使得我们不能考虑差分之类的技术，只能 bottom-up 合并结果。

首先将黑白连通块分开考虑。以黑色连通块为例，我们发现**黑色连通块的形状其实是由连通块内最高的结点 + 该结点子树内的白点所确定的。**
把这种形状拍到 DFS 序上，我们得到的是“一个大区间挖去其中的若干个互不相交的小区间”。
虽然区间加的标记很容易消除，但是我们不可能一板一眼地每次更新都去修正一遍小区间——太慢。

# 例题九（无平凡交区间的维护）

操作四、五的存在使得我们不得不引入重链剖分 + 线段树的结构。所需维护的"最大值"使得我们不能考虑差分之类的技术，只能 bottom-up 合并结果。

首先将黑白连通块分开考虑。以黑色连通块为例，我们发现**黑色连通块的形状其实是由连通块内最高的结点 + 该结点子树内的白点所确定的。**
把这种形状拍到 DFS 序上，我们得到的是"一个大区间挖去其中的若干个互不相交的小区间"。
虽然区间加的标记很容易消除，但是我们不可能一板一眼地每次更新都去修正一遍小区间——太慢。

尝试在线段树结点上直接打永久化标记，来维护"有多少个白点的子树对应的区间恰好完全包含了它"。
(恰好完全包含：自己被包含并且父亲没有被包含)
因为区间加的标记都是从上往下释放的，所以我们可以额外维护一个表示"连通块加"的标记，该标记**只下传给"不被某个白点子树区间恰好包含"的儿子。**
类似地，区间 $\max$ 的信息同样也只考虑"没有被某白点子树对应的区间恰好完全包含"的儿子。

# 例题九（无平凡交区间的维护）

一个具体案例的说明：

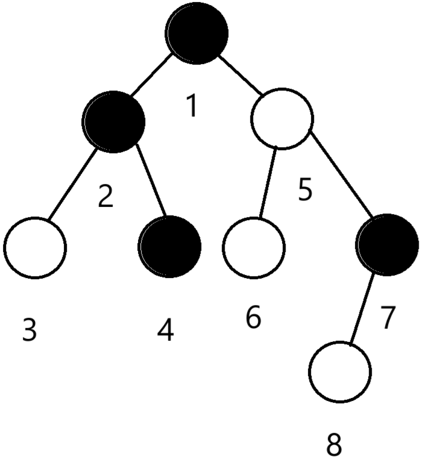


图: 一棵黑白树

# 例题九（无平凡交区间的维护）

思考：

- 如何说明，这种标记方法的正确性，即某个叶子结点会被标记修改到当且仅当它在这个连通块内？
  (还有信息合并，不过是类似的)
- 白色覆盖标记的修改会和“连通块加”标记的下放冲突吗？这一部分标记的维护会和平凡区间加标记的维护冲突吗？

# 例题九（无平凡交区间的维护）

思考：

- 如何说明，这种标记方法的正确性，即某个叶子结点会被标记修改到当且仅当它在这个连通块内？
(还有信息合并，不过是类似的)
- 白色覆盖标记的修改会和“连通块加”标记的下放冲突吗？这一部分标记的维护会和平凡区间加标记的维护冲突吗？

理论上来说，这个技术的威力比较大，但是目前只有这一道例题。有兴趣的话可以用这个 idea 继续编题。

# zkw 线段树

zkw 线段树的优点是：容易编写；无递归，常数小。
但是在 zkw 线段树上维护标记有一定难度。如果标记可持久化则较为容易；如果不可，例如标记之间存在严格时序或者需要维护多种标记时，则可以采用类似于某些平衡树的标记下放方式：先将到根的路径上的所有标记全部下放，然后执行操作。

**Remark.** 理论上，所有自底向上的数据结构都可以这样来释放标记，其复杂度由树高控制。另一个典型例子是 Splay。

# zkw 线段树

zkw 线段树的优点是：容易编写；无递归，常数小。
但是在 zkw 线段树上维护标记有一定难度。如果标记可持久化则较为容易；如果不可，例如标记之间存在严格时序或者需要维护多种标记时，则可以采用类似于某些平衡树的标记下放方式：先将到根的路径上的所有标记全部下放，然后执行操作。

**Remark.** 理论上，所有自底向上的数据结构都可以这样来释放标记，其复杂度由树高控制。另一个典型例子是 Splay。

zkw 线段树上进行线段树二分类似操作也是可行的，具体实现非常类似于 finger search。

# zkw 线段树

zkw 线段树的优点是：容易编写；无递归，常数小。
但是在 zkw 线段树上维护标记有一定难度。如果标记可持久化则较为容易；如果不可，例如标记之间存在严格时序或者需要维护多种标记时，则可以采用类似于某些平衡树的标记下放方式：先将到根的路径上的所有标记全部下放，然后执行操作。

**Remark.** 理论上，所有自底向上的数据结构都可以这样来释放标记，其复杂度由树高控制。另一个典型例子是 Splay。

zkw 线段树上进行线段树二分类似操作也是可行的，具体实现非常类似于 finger search。

然而如果还要将 zkw 线段树进行可持久化、势能均摊等更复杂的操作，那属实有点为难人家了。理论上来说，这些操作都可以通过模仿递归线段树来实现，但是这样一来它的优势就基本丧失了。