分块算法经典例题讲解

例题 1：区间乘法、区间加法与单点查询

题目描述

给出一个长度为 n 的数列 $a_1, a_2, \ldots, a_n$，以及 n 个操作。

操作类型

- **操作 0**（区间加法）：$\mathrm{opt} = 0, l, r, c$

  将区间 $[l, r]$ 中的所有数字都加上 c

- **操作 1**（区间乘法）：$\mathrm{opt} = 1, l, r, c$

  将区间 $[l, r]$ 中的所有数字都乘以 c

- **操作 2**（单点查询）：$\mathrm{opt} = 2, r$

  查询 a_r 的当前值

输入格式

- 第一行：整数 n

- 第二行：n 个整数 $a_1, a_2, \ldots, a_n$

- 接下来 n 行：每行一个操作

解法：分块 + 懒标记

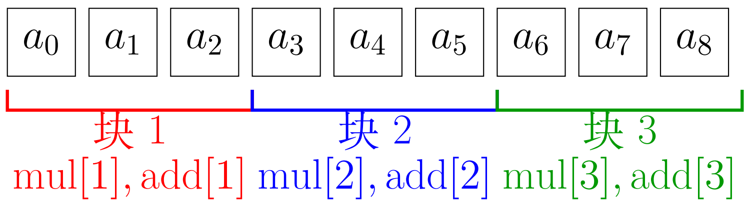
核心思想

将长度为 n 的数列分成 $\sqrt{n}$ 个块，每块大小约为 $\sqrt{n}$。

对每个块维护两个懒标记：

- $\mathrm{mul}[i]$：第 i 块的乘法标记（初始为 1）
- $\mathrm{add}[i]$：第 i 块的加法标记（初始为 0）

可视化示意



操作实现

1. **区间加法/乘法**：
   - 对于完整覆盖的块：只更新懒标记
   - 对于部分覆盖的块：暴力修改每个元素
2. **单点查询**：

   查询 a_r 时，返回 $a_r \times \mathrm{mul}[\mathrm{block}(r)] + \mathrm{add}[\mathrm{block}(r)]$

标记下传规则

当需要访问块内具体元素时:

$$a_i \leftarrow a_i \times \mathrm{mul}[\mathrm{block}(i)] + \mathrm{add}[\mathrm{block}(i)]$$

然后重置标记: $\mathrm{mul}[\mathrm{block}(i)] = 1, \mathrm{add}[\mathrm{block}(i)] = 0$

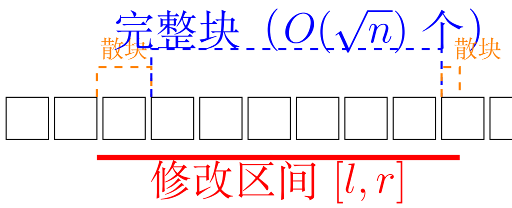
复杂度分析

空间复杂度

- 原数组：$O(n)$
- 块标记：$O(\sqrt{n})$ 个块，每块 2 个标记
- 总空间：$O(n)$

时间复杂度

区间修改操作（加法或乘法）：



- 散块（两端不完整的块）：暴力修改，$O(\sqrt{n})$
- 完整块：只修改标记，最多 $O(\sqrt{n})$ 个块，每个 $O(1)$
- **单次操作**：$O(\sqrt{n})$

单点查询操作：

直接通过下标计算所属块号，应用标记，$O(1)$

总复杂度：

n 个操作，每次 $O(\sqrt{n})$ 或 $O(1)$，总时间复杂度：$\boxed{O(n\sqrt{n})}$

例题 2：区间查询与区间赋值

题目描述

给出一个长度为 n 的数列 $a_1, a_2, \ldots, a_n$，以及 n 个操作。

操作内容

每个操作由三个参数 l, r, c 组成，执行以下两步：

1. **查询**：统计区间 $[l, r]$ 中有多少个元素等于 c
2. **修改**：将区间 $[l, r]$ 中的所有元素都赋值为 c

输入格式

- 第一行：整数 n
- 第二行：n 个整数 $a_1, a_2, \ldots, a_n$
- 接下来 n 行：每行三个整数 l, r, c

示例

初始数组：$[1, 2, 2, 3, 3]$

操作 $l = 2, r = 4, c = 2$：

- 查询结果：区间 $[2, 4]$ 是 $[2, 2, 3]$，有 2 个元素等于 2

- 修改后：$[1, 2, 2, 2, 3]$

解法：分块 + 区间赋值标记

核心思想

将数组分成 $\sqrt{n}$ 个块，每块维护：

- tag[i]：整块赋值标记（-1 表示无标记）
- 每个元素的实际值

可视化操作流程

初始: | 1 | 2 | 2 | 3 | 3 | 1 | 4 | 4 | 5 |

操作：$l = 2, r = 7, c = 2$（查询并赋值）

操作后: | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 5 |

散块 整块标记 散块

操作实现

区间查询与赋值：

1. **散块**（两端不完整块）：
   - 先下传该块的标记（如果有）
   - 遍历散块元素，统计等于 c 的个数
   - 将散块元素逐个赋值为 c
2. **整块**（完全覆盖的块）：

- 如果该块有标记 $\mathrm{tag}[i]$：
  - 若 $\mathrm{tag}[i] = c$，贡献块大小个 c
  - 否则贡献 0 个 c
- 否则遍历块内所有元素统计
- 给整块打上标记 $\mathrm{tag}[i] = c$

复杂度分析

块大小选择

设块大小为 B，则有 $\frac{n}{B}$ 个块。

单次操作时间复杂度

操作部分	**时间复杂度**
散块处理	$O(B)$
整块标记	$O(\frac{n}{B})$
整块查询（最坏）	$O(B \cdot \frac{n}{B}) = O(n)$
单次总复杂度	$\boxed{O(n)}$

优化分析

关键观察: 整块查询只在块没有标记时遍历，一旦打上标记后，之后的查询都是 $O(1)$。

均摊分析:

- 每个元素最多被遍历常数次（打标记前）
- 打标记后的查询都是 $O(1)$
- 散块操作：$O(B) = O(\sqrt{n})$
- 整块标记：$O(\frac{n}{B}) = O(\sqrt{n})$

选择 $B = \sqrt{n}$，单次操作均摊：$O(\sqrt{n})$

总时间复杂度（均摊）：$\boxed{O(n\sqrt{n})}$

例题 3：区间开方与区间求和

题目描述

给出一个长度为 n 的数列 $a_1, a_2, \ldots, a_n$，以及 n 个操作。

操作类型

- **操作 0**（区间开方）：$\mathrm{opt} = 0, l, r$

  对区间 $[l, r]$ 中的每个元素 a_i 进行开方：

  $$a_i \leftarrow \lfloor \sqrt{a_i} \rfloor$$

- **操作 1**（区间求和）：$\mathrm{opt} = 1, l, r$

  查询区间 $[l, r]$ 中所有元素的和：

  $$\sum_{i=l}^{r} a_i$$

输入格式

- 第一行：整数 n
- 第二行：n 个整数 $a_1, a_2, \ldots, a_n$
- 接下来 n 行：每行一个操作

示例

初始：$[16, 9, 4, 1]$

操作 0，$l = 1, r = 3$：$[4, 3, 2, 1]$

操作 1，$l = 1, r = 4$：输出 $4 + 3 + 2 + 1 = 10$

解法：分块 + 区间和维护

核心思想

关键性质：开方操作使数字快速减小

$$10^9 \longrightarrow 31622 \longrightarrow 177 \longrightarrow 13 \longrightarrow 3 \longrightarrow 1$$

约 $\log\log n$ 次后变为 1

将数组分成 $\sqrt{n}$ 个块，每块维护：

- $\text{sum}[i]$：第 i 块的元素和
- $\max[i]$：第 i 块的最大值

可视化数据结构

16	9	4	16	25	36

块 1　　块 2

$\text{sum}[1] = 29, \max[1] = 16$　　$\text{sum}[2] = 61, \max[2] = 36$

操作实现

区间开方：

1. 对于整块：如果 $\max[i] \leq 1$，跳过（已收敛）
2. 否则遍历块内元素，逐个开方，更新 $\text{sum}[i]$ 和 $\max[i]$
3. 散块：直接暴力修改

区间求和：

1. 整块：直接累加 $\mathrm{sum}[i]$，$O(1)$

2. 散块：遍历累加，$O(\sqrt{n})$

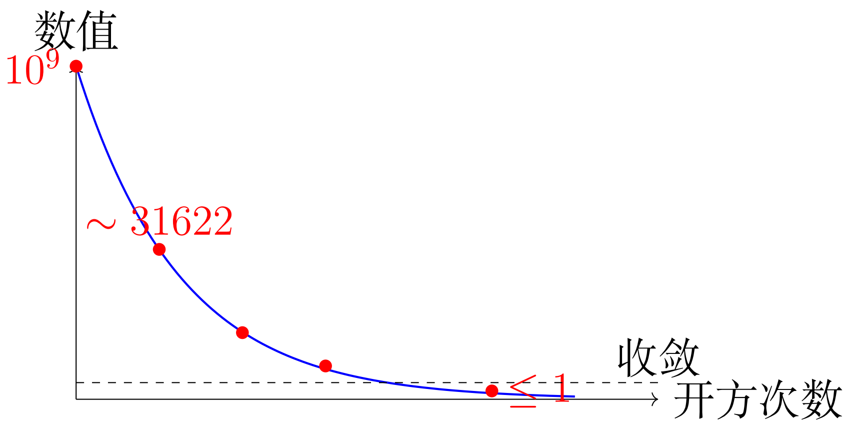
复杂度分析

空间复杂度

- 原数组：$O(n)$
- 块信息：$O(\sqrt{n})$ 个块，每块存储和与最大值
- 总空间：$O(n)$

时间复杂度 - 关键分析

开方操作的收敛性：



每个元素最多被开方 $O(\log\log V)$ 次，其中 V 是元素的最大值。

操作复杂度

区间求和：

- 整块：$O(\sqrt{n})$ 个块，每块 $O(1)$
- 散块：$O(\sqrt{n})$

- 单次：$O(\sqrt{n})$

区间开方（均摊）：

- 每个元素被真正修改的次数：$O(\log\log V)$
- 单次操作：最坏 $O(n)$，但均摊 $O(\sqrt{n}\log\log V)$

总时间复杂度：$\boxed{O(n\sqrt{n}\log\log V)}$

例题 4：区间生长与区间计数

题目描述

有 n 株花，每株花有一个初始高度（≤ 1000 的自然数）。

有两个角色执行 q 个操作：

- **Lily White**：使花儿生长
- **Yuka**：统计满足条件的花

操作类型

- **操作 M**（生长）：`M` $l\ r\ h$

  使区间 $[l, r]$ 内所有花的高度增加 h

- **操作 A**（询问）：`A` $l\ r\ k$

  查询区间 $[l, r]$ 内有多少花的高度不低于 k（即统计满足 $a_i \geq k$ 的花的数量）

输入格式

- 第一行：n（花的数量）和 q（操作数量）
- 第二行：n 个整数表示初始高度
- 接下来 q 行：每行一个操作

示例

初始：$[5,3,8,2]$

`M 1 3 2`：$[7,5,10,2]$

`A 1 4 6`：输出 2（$a_1 = 7 \geq 6$，$a_3 = 10 \geq 6$）

解法：分块 + 懒标记 + 块内排序

核心思想

将数组分成 $\sqrt{n}$ 个块，每块维护：

- add$[i]$：第 i 块的加法懒标记
- sorted$[i]$：第 i 块元素的有序副本

数据结构可视化

原数组: $\boxed{5}\ \boxed{3}\ \boxed{8}\ \boxed{2}\ \boxed{7}\ \boxed{1}$

块1：add$[1] = 0$，有序：$[3, 5, 8]$

块2：add$[2] = 0$，有序：$[1, 2, 7]$

操作实现

区间加法（M 操作）:

1. **整块**：直接增加 add$[i]$ 标记，$O(1)$
2. **散块**:
   - 下传标记到块内所有元素
   - 逐个修改散块元素
   - 重新排序该块的有序副本，$O(\sqrt{n} \log \sqrt{n})$

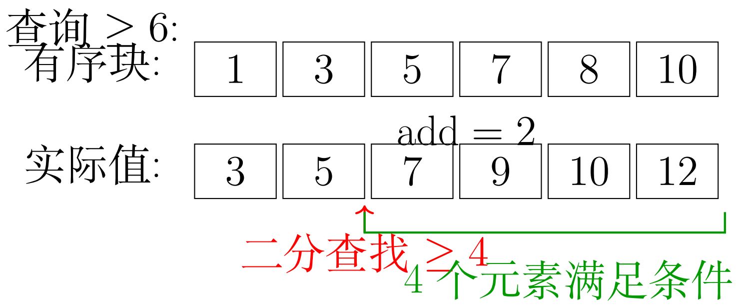
区间计数（A 操作）:

1. **整块**:

   - 在有序副本中二分查找第一个 $\geq k - \text{add}[i]$ 的位置
   - 该位置右侧的元素都满足条件，$O(\log\sqrt{n})$

2. **散块**：遍历元素，逐个判断，$O(\sqrt{n})$

查询可视化



复杂度分析

空间复杂度

- 原数组：$O(n)$
- 有序副本：每块 $O(\sqrt{n})$，共 $O(\sqrt{n})$ 个块，总计 $O(n)$
- 懒标记：$O(\sqrt{n})$
- 总空间：$O(n)$

时间复杂度

M 操作（区间加法）：

部分	**复杂度**
整块标记	$O(\sqrt{n})$
散块修改 + 重排	$O(\sqrt{n}\log\sqrt{n})$
单次	$\boxed{O(\sqrt{n}\log\sqrt{n})}$

A 操作（区间计数）：

部分	**复杂度**
整块二分	$O(\sqrt{n}\log\sqrt{n})$
散块遍历	$O(\sqrt{n})$
单次	$\boxed{O(\sqrt{n}\log\sqrt{n})}$

总复杂度

q 个操作，每次 $O(\sqrt{n}\log\sqrt{n})$

总时间复杂度：$\boxed{O(q\sqrt{n}\log n)}$