实验舱
青少年编程
走近科学 走进名校

提高算法班

线段树

Mas

线段树

线段树(Segment Tree)主要用于维护区间信息

线段树可以 $O(\log N)$ 的时间复杂度 实现 单点/区间修改、区间查询(区间求和,区间最值)等操作

相比于树状数组更具通用性

线段树的每个节点都对应一个区间,但不保证所有的区间都是线段树的节点

线段树将长度不为 1 的区间划分成左右两个区间

若节点 p 储存区间 $[L,R]$ 信息,记 $mid=\frac{L+R}{2}$

那么两个子节点分别储存 $[L,mid]$ 和 $[mid+1,R]$ 的信息

左节点对应的区间长度与右节点相同或者比之恰好多 1

线段树—建树

若采用堆式存储建立线段树($2p$ 是 p 的左儿子,$2p+1$是 p的右儿子)

若有 n 个叶子结点,则节点数组范围为$2^{\lceil \log n \rceil+1}$

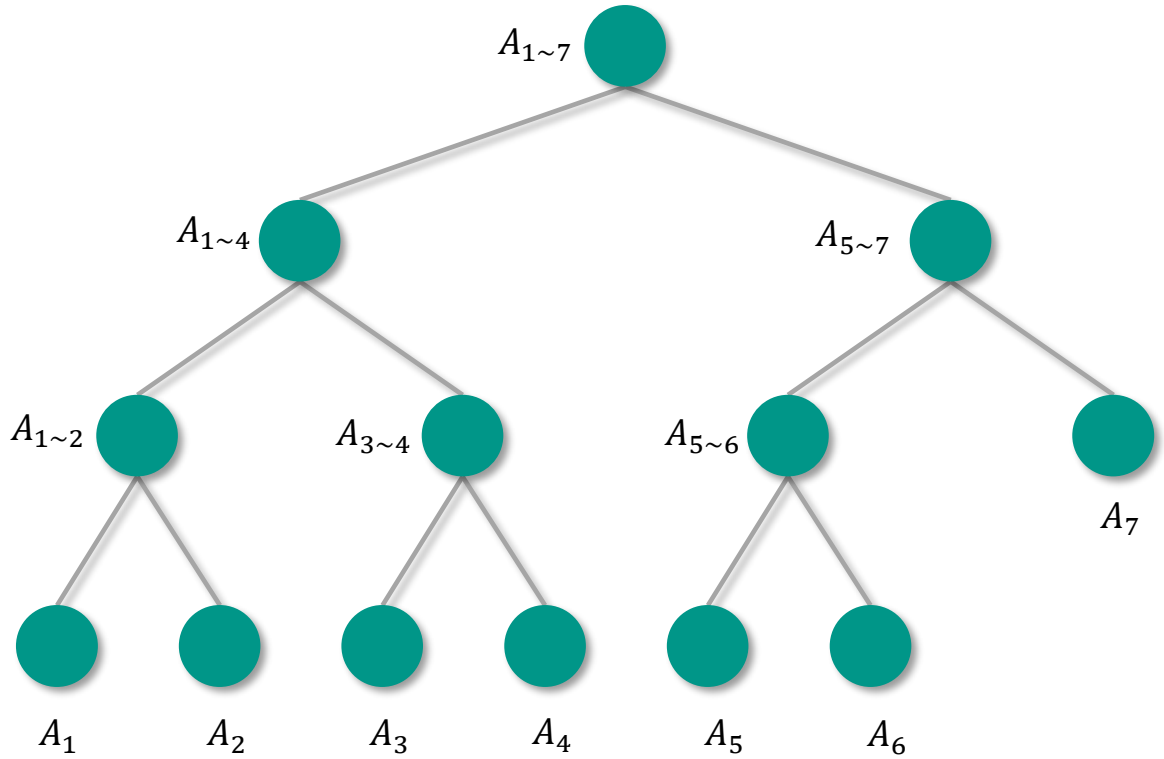
线段树的深度是 $\lceil \log n \rceil$,在堆式储存情况下叶子节点(包括无用的叶子节点)数量为 $2^{\lceil \log n \rceil}$个

又由于其为一棵完全二叉树,则其总节点个数 $2^{\lceil \log n \rceil+1}-1$

$\frac{2^{\lceil \log n \rceil+1}-1}{n}$ 的最大值在$n=2^x+1\ (x \in \mathbb{N}_+)$时取到

此时节点数为 $2^{\lceil \log n \rceil+1}-1=2^{x+1}-1=4n-5$

可以直接把数组长度设为 $\boldsymbol{4n}$



线段树—建树

考虑递归建建立线段树

设当前节点为 p ,若 p 管辖的区间长度为 1

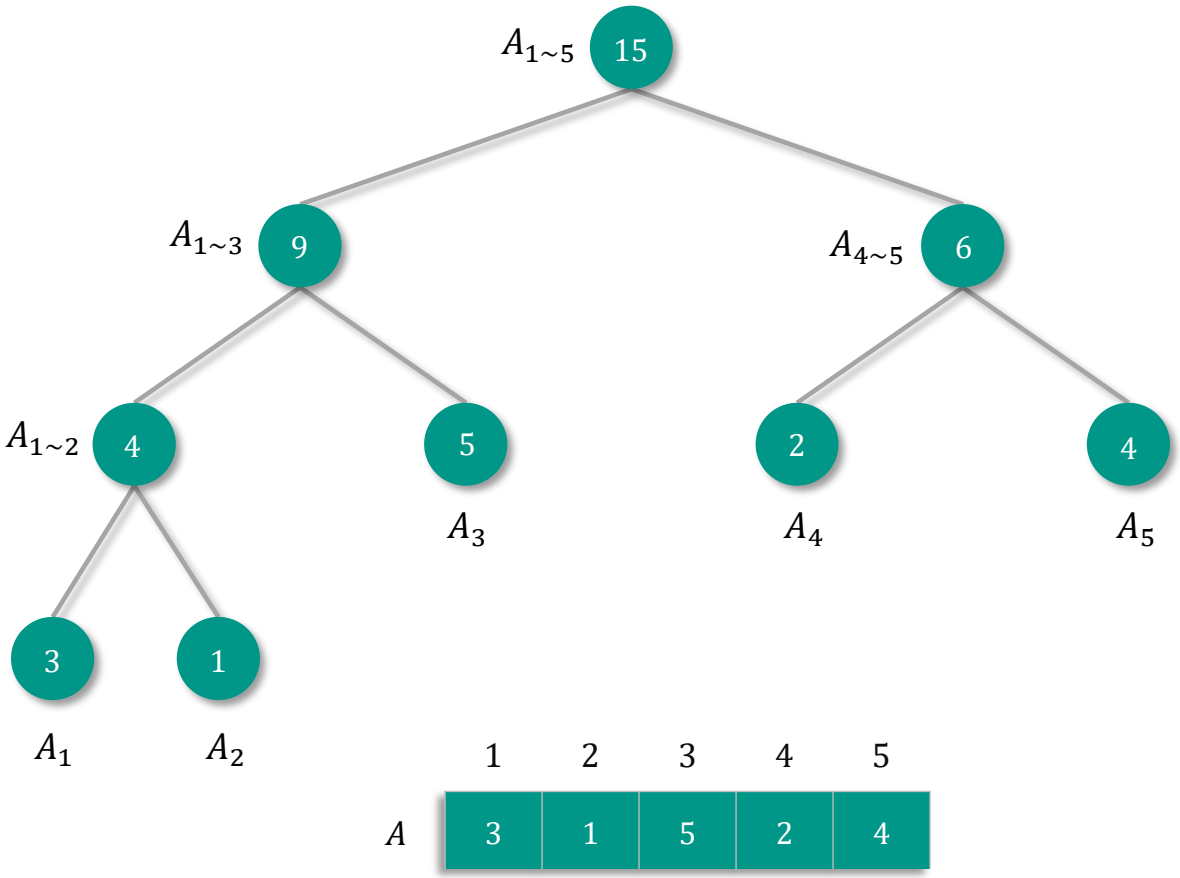
可直接根据原数组上相应位置的值初始化该节点

否则将该区间从中点处分割为两个子区间

分别进入左右子节点递归建树

回溯时合并两个子节点的信息

若区间长度为 n , 建树时间复杂度 $O(n)$



线段树—单点修改

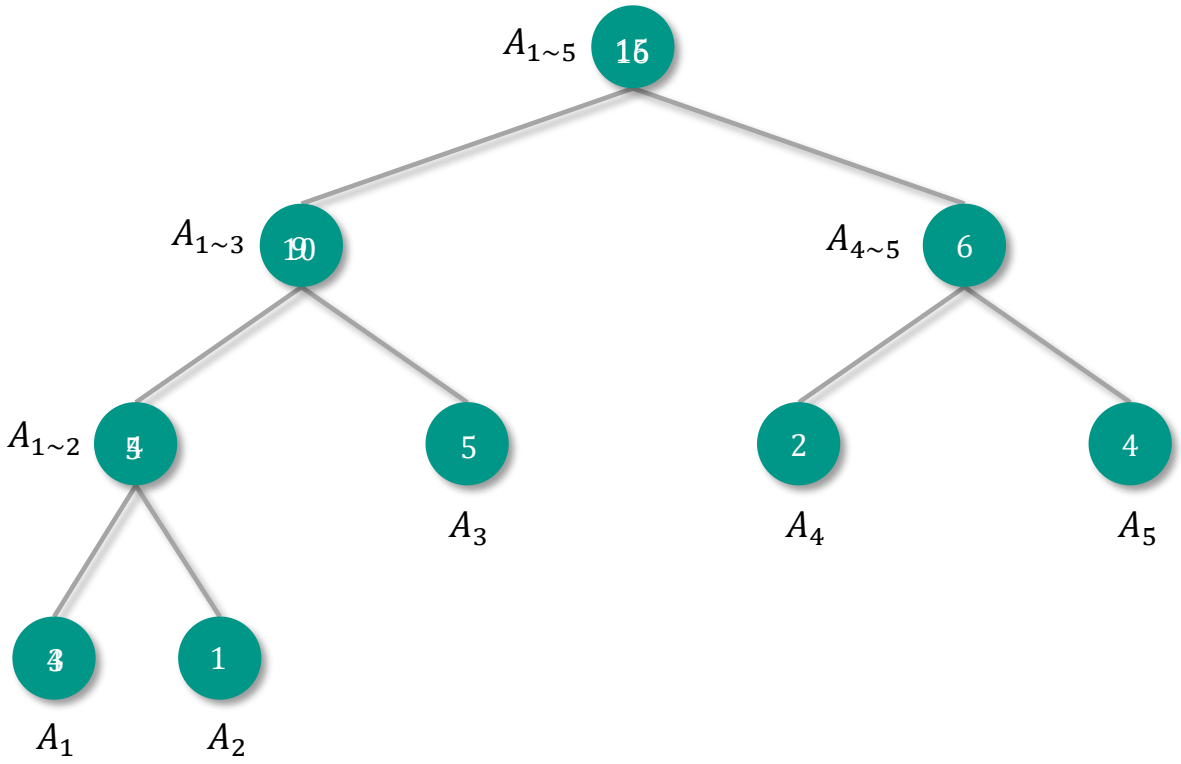
如需修改单点值,参考建树过程不断向下递归

找到该点对应叶子节点后进行修改

回溯时更新父节点的值

如右图更新 $A_1 \mathrel{+}= 1$

每层仅更新一个节点,单点修改时间复杂度 $O(\log n)$



线段树—区间查询

求区间 $[L,R]$ 的总和、求区间最大值/最小值等操作

以右图为例

若要查询区间 [1,5] 的和

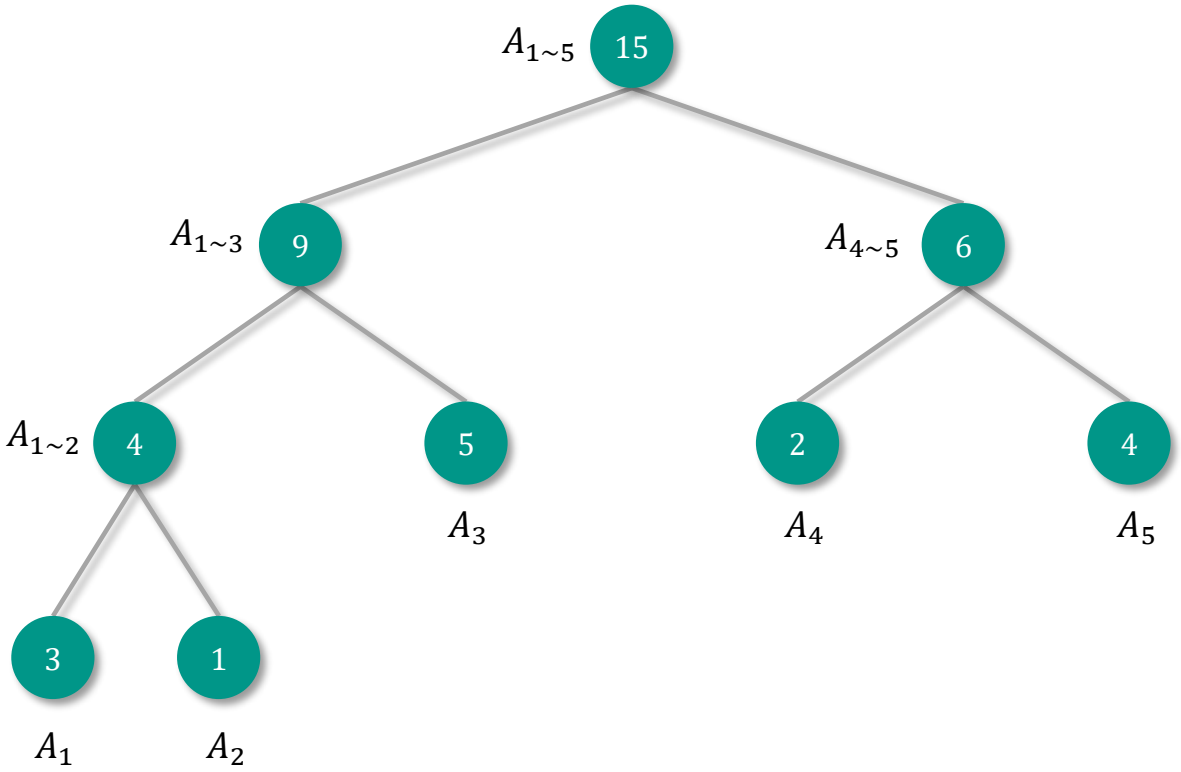
直接获取根节点信息即可

若要查询的区间为[3,5]

此时不能直接获得区间信息

但可将 [3,5] 拆成 [3,3] 和 [4,5] 两个区间

通过合并这些区间来求得答案



线段树—区间查询

若要查询的区间是 $[L,R]$,从根节点出发

记当前节点管辖区间为 $[s,e]$ 区间中点 $mid=\frac{s+e}{2}$

- 若 $L \le s \le e \le R$,即 $[s,e]$ 被 $[L,R]$ 完全包含

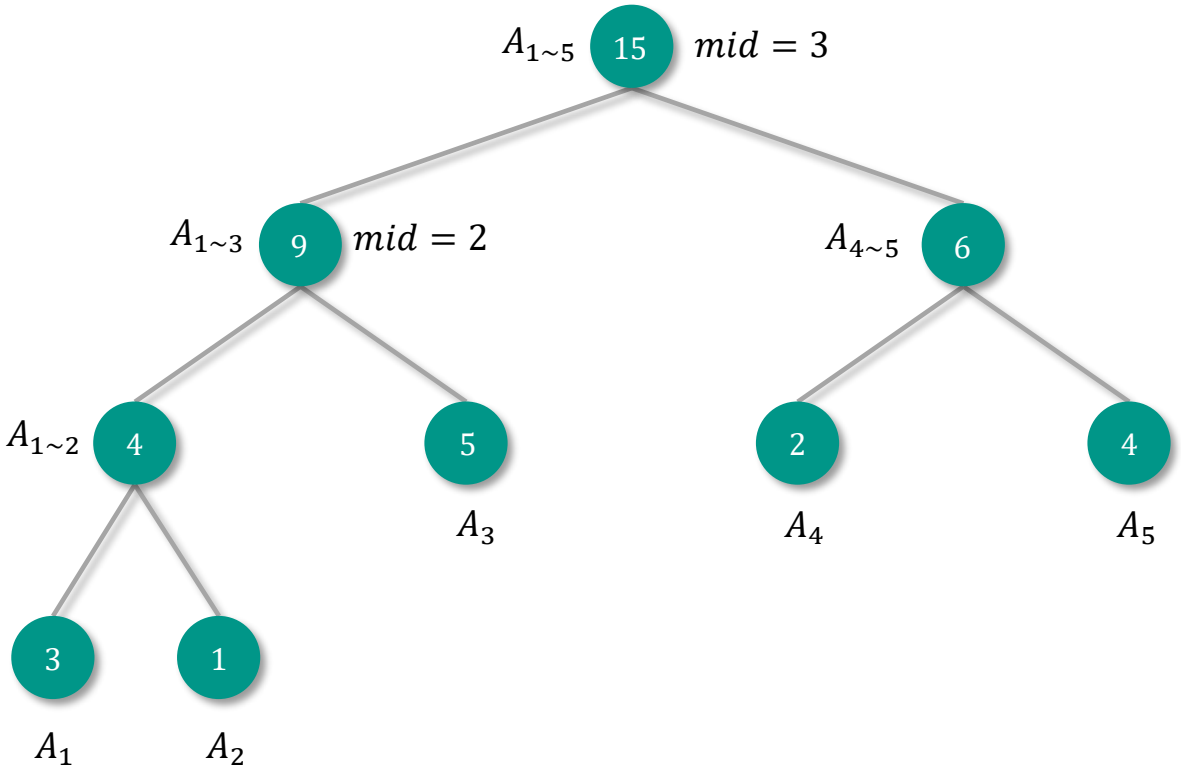
  直接返回当前节点信息

- 若 $L \le mid$

  说明询问区间部分被左孩子管理,查询左子树信息

- 若 $R \ge mid+1$

  说明询问区间部分被右孩子管理,查询右子树信息



合并完全包含的区间即可求出答案

线段树—区间查询

若节点区间完全包含在 $[L,R]$ 内不再向下查询,称这样的节点为完整节点

若节点区间只有一部分在 $[L,R]$ 内则要向下查询,称这样的节点为部分节点

在线段树上某一层中

- 部分节点最多有2个,而且与 $[L,R]$ 交在两端
- 完整节点最多有 2 个

  完整节点的兄弟节点一定不是完整节点,否则它们的父节点也是完整节点

线段树有 $\log n$ 层,每一层至多访问 4 个节点,

更具体的说,每一层最多访问 2 个节点,所以其实只到访了不超过 $2\log n$ 个点

综上区间查询时间复杂度 $O(\log n)$

线段树—区间修改

若要求修改区间 $[L,R]$,将 $[L,R]$ 中的节点都修改一次,时间复杂度无法承受

懒惰标记($Lazy\ Tag$)

到达一个线段树节点时,判断该节点区间 $[s,e]$ 是否完全落在落在区间 $[L,R]$ 里

- 若不完全被包含,正常递归左右孩子(如果它们落在区间里),返回时合并左右孩子信息
- 若完全包含,快速更新当前操作对该区间的影响,同时在该区间打上懒标记,随后退出该节点

懒标记意义

懒标记所在节点已经正确维护了区间信息

但走向它子树节点时需要带上这个标记更新(查询/修改时应拿到懒标记更新后再继续向下递归)

懒标记的原则

懒标记需要满足什么条件才能在线段树上正确运行?

- **懒标记能快速计算出当前节点维护的所有区间信息**
- **懒标记之间能快速合并**

上述“快速”指的是在 $O(1)$ 或 $O(\log n)$ 时间内能稳定计算得到

以区间加区间求和为例

节点上的区间信息为区间和,懒标记为该区间(还未向下传递)的增量

1. 当一个新的懒标记 δ 施加在一个区间上时,可 $O(1)$ 更新区间和,即 $S \mathrel{+}= \delta \times len$
2. 当一个新的懒标记 δ' 施加在原懒标记 δ 时,可 $O(1)$ 更新懒标记,即 $\delta \mathrel{+}= \delta'$

线段树—区间修改

若要给区间 [1,4] 每个数都加上2, 参考区间查询可找到两个极大区间 [1,3] 和 [4,4]

在这两个节点上修改,并给它们打上标记 , 回溯时向上合并区间信息

[1,3] 信息虽然被修改了(管辖 3 个数,权值加上3 × 2)

但两个子节点却还没更新,仍为修改前的信息

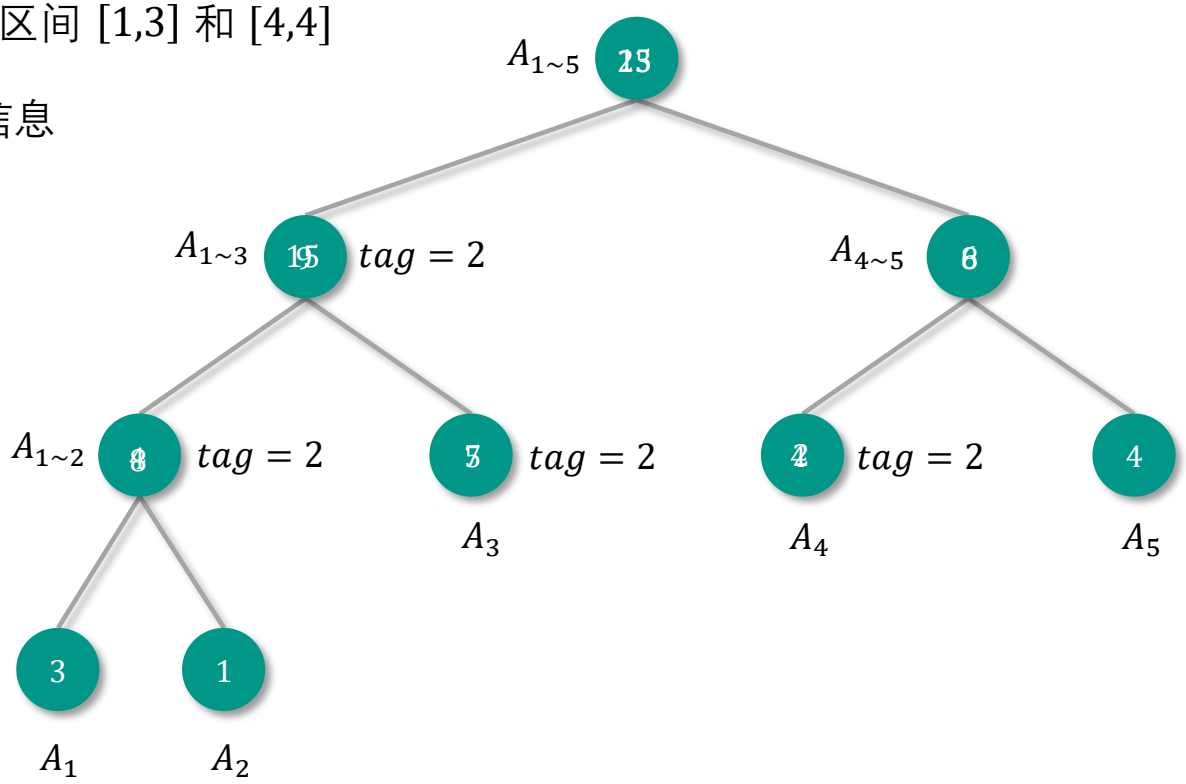
若要查询区间 [1,2] 之和,找到区间[1,3]

发现并非目标区间且还存在标记

此时下放标记,同时将两个子区间的信息更新

并清除该区间上的标记

单次区间修改与区间查询类似,时间复杂度 $O(\log n)$



线段树


```
void pushup(int idx) // 合并区间
{
  tree[idx].val = tree[idx << 1].val + tree[idx << 1 | 1].val;
}
void add(int idx, int val) //树上的点(对应区间)加值
{
  tree[idx].val += (tree[idx].r - tree[idx].l + 1) * val; //加上区间长度 * val
  tree[idx].tag += val; //标记累加
}
void pushdown(int idx)
{
  if (tree[idx].tag) //如果该点有标记值
  {
    add(idx << 1, tree[idx].tag); // 下传给两个子区间
    add(idx << 1 | 1, tree[idx].tag);
    tree[idx].tag = 0;
  }
}

void build(int idx, int l, int r) //建树
{
  tree[idx].l = l, tree[idx].r = r;
  if (l == r)
  {
    tree[idx].val = a[l];
    return;
  }
  int mid = l + r >> 1;
  build(idx << 1, l, mid), build(idx << 1 | 1, mid + 1, r);
  pushup(idx);
}
```

```
long long query(int idx, int l, int r) //区间查询
{
  if (tree[idx].l > r || tree[idx].r < l)
    return 0;
  if (l <= tree[idx].l && tree[idx].r <= r)
    return tree[idx].val;
  pushdown(idx);
  long long res = 0;
  int mid = tree[idx].l + tree[idx].r >> 1;
  if (l <= mid)
    res += query(idx << 1, l, r);
  if (r >= mid + 1)
    res += query(idx << 1 | 1, l, r);
  return res;
}

void update(int idx, int l, int r, int val) // 区间修改
{
  if (l <= tree[idx].l && tree[idx].r <= r)
  {
    add(idx, val);
    return;
  }
  int mid = tree[idx].l + tree[idx].r >> 1;
  pushdown(idx);
  if (l <= mid)
    update(idx << 1, l, r, val);
  if (r >= mid + 1)
    update(idx << 1 | 1, l, r, val);
  pushup(idx);
}
```

#937、树状数组/线段树 3： 区间修改,区间查询

题目描述

给定数列 $a[1], a[2], \ldots, a[n]$,你需要依次进行 q 个操作,操作有两类:

- `1 r x` : 给定 l, r, x ,对于所有 $i \in [l, r]$,将 $a[i]$ 加上 x (换言之,将 $a[l], a[l+1], \ldots, a[r]$ 分别加上 x)
- `2 l r` : 给定 l, r ,求 $\sum_{i=l}^{r} a[i]$ 的值(换言之,求 $a[l] + a[l+1] + \cdots + a[r]$ 的值)

输入格式

第一行包含 2 个正整数 n, q ,表示数列长度和询问个数。保证 $1 \leq n, q \leq 10^6$

第二行 n 个整数 $a[1], a[2], \ldots, a[n]$,表示初始数列。保证 $|a[i]| \leq 10^6$

接下来 q 行,每行一个操作,为以下两种之一:

- `1 l r x` : 对于所有 $i \in [l, r]$,将 $a[i]$ 加上 x
- `2 l r` : 输出 $\sum_{i=l}^{r} a[i]$ 的值

保证 $1 \leq l \leq r \leq n, \ |x| \leq 10^6$

输出格式

对于每个 `2 l r` 操作,输出一行,每行有一个整数,表示所求的结果

数据范围与提示

对于所有数据, $1 \leq n, q \leq 10^6, \ |a[i]| \leq 10^6$, $1 \leq l \leq r \leq n, \ |x| \leq 10^6$

#718、维护序列

题目描述

老师交给小可可一个维护数列的任务,现在小可可希望你来帮他完成

有长为 n 的数列,不妨设为 $a_1, a_2, \cdots, a_n$

有如下三种操作形式:

- 把数列中的一段数全部乘一个值
- 把数列中的一段数全部加一个值

询问数列中的一段数的和,由于答案可能很大,你只需输出这个数模 P 的值

输入格式

第一行两个整数 n 和 P

第二行含有 n 个非负整数,从左到右依次为 $a_1, a_2, \cdots, a_n$

第三行有一个整数 M ,表示操作总数

从第四行开始每行描述一个操作,输入的操作有三种形式:

输出格式

对每个操作 3 ,按照它在输入中出现的顺序
依次输出一行一个整数表示询问结果

数据范围与提示

对于全部测试数据, $1 \leq t \leq g \leq n, 0 \leq c, a_i \leq 10^9, 1 \leq P \leq 10^9$

样例输入

```
7 43
1 2 3 4 5 6 7
5
1 2 5 5
3 2 4
2 3 7 9
3 1 3
3 4 7
```

样例输出

```
2
35
8
```

样例说明

初始时数列为 $\{1, 2, 3, 4, 5, 6, 7\}$

经过第 1 次操作后,数列为 $\{1, 10, 15, 20, 25, 6, 7\}$

对第 2 次操作,和为 $10 + 15 + 20 = 45$,模 43 的结果是 2

经过第 3 次操作后,数列为 $\{1, 10, 24, 29, 34, 15, 16\}$

对第 4 次操作,和为 $1 + 10 + 24 = 35$,模 43 的结果是 35

对第 5 次操作,和为 $29 + 34 + 15 + 16 = 94$,模 43 的结果是 8

操作中存在 乘法 和 加法 两种操作

对于区间修改

需要维护 plus 和 mul 两个 *tag*

mul 初值/清空 值为 1

#718、维护序列

考虑标记下放,记当前节点为 p , 区间长度为 len

若标记下放时先处理 + 再处理 ×

此时 $p.val = (p.val + \text{plus} \times len) \times \text{mul}$

若再出现一次 $+k$ 操作, 为使得操作可由 plus 和 mul 两个 tag 维护

$p.val = \left(p.val + \left(\text{plus} + \frac{k}{\text{mul}}\right) \times len\right) \times \text{mul}$, 需要令 $\text{plus} \leftarrow \text{plus} + \frac{k}{\text{mul}}$ 将出现精度误差

考虑先处理 × 再处理 +

此时 $p.val = p.val \times \text{mul} + \text{plus} \times len$

若再出现一次 $+k$ 操作, $p.val = p.val \times \text{mul} + (\text{plus} + k) \times len$, 仅需令 $\text{plus} \leftarrow \text{plus} + k$

若再出现一次 $\times k$ 操作, $p.val = p.val \times \text{mul} \times k + (\text{plus} \times k) \times len$, 仅需令$\text{plus} \leftarrow \text{plus} \times k$, $\text{mul} \leftarrow \text{mul} \times k$

对于两种操作不同维护标记即可,时间复杂度 $O(m \log n)$

多元组懒标记

懒标记也可以是多元组,彼此之间有计算优先级

以区间求和为例,设懒标记为二元组 $(mul, plus)$,表示当前区间如果向下传递,每个元素 x 会变成 $mul \times x + plus$

1. 区间和的快速更新

   $S_{[l,r]} = x_l + x_2 + \cdots + x_r$,现在 $x' = mul \cdot x + plus$,由乘法分配率 $S' = mul \cdot S + plus$

2. 懒标记的快速合并：假设原懒标记为 $(mul_1, plus_1)$,新懒标记为 $(mul_2, plus_2)$

   对于区间里的一个元素 x,它的最终变化为

   $$x' = mul_2(mul_1 x + plus_1) + plus_2 = mul_1 mul_2 x + mul_2 plus_1 + plus_2$$

   即合并后标记为 $(mul_1 mul_2, mul_2 plus_1 + plus_2)$

依然可通过 $O(1)$ 的时间复杂度完成区间信息的快速更新和懒标记的快速合并

#2652、Can you answer on these queries

题目描述

给定长度为 N 的数列 A ,以及 M 条指令,每条指令可能是以下两种之一:

- `1 x y` ,查询区间 $[x, y]$ 中的最大连续子段和,即

$$max_{x \le l \le r \le y}\{\sum_{i=l}^{r} A_i\}$$

- `2 x y` ,把 A_x 改成 y

对于每个查询指令,输出一个整数表示答案

输入格式

第一行两个整数 N, M

第二行 N 个整数 A_i

接下来 M 行每行 3 个整数 k, x, y ,
$k = 1$ 表示查询(此时如果 $x > y$,请交换 x, y), $k = 2$ 表示修改

输出格式

对于每个查询指令输出一个整数表示答案

每个答案占一行

输入样例

```
5 3
1 2 -3 4 5
1 2 3
2 2 -1
1 3 2
```

输出样例

```
2
-1
```

数据范围

对于全部的数据 $N \le 500000, M \le 100000, -1000 \le A_i \le 1000$

#2652、Can you answer on these queries

线段树节点维护四个信息

lMax	紧靠左端点的最大前缀和	rMax	紧靠右端点的最大前缀和
sum	区间和	maxV	区间最大子段和

记当前节点为 p , 左右孩子分别为 pl 和 pr

考虑向上合并区间

$$p.\text{sum} = pl.\text{sum} + pr.\text{sum}$$

$$p.\text{lMax} = \max(\, pl.\text{lMax}, pl.\text{sum} + pr.\text{lMax})$$

$$p.\text{rMax} = \max(\, pr.\text{rMax}\,, pr.\text{sum} + pl.\text{rMax})$$

$$p.\text{maxV} = \max(\, pl.maxV\,, pr.\text{maxV}\,, pl.\text{rMaxV} + pr.\text{lMaxV}\,)$$

对于每次查询,同样以上述规则合并得到答案

时间复杂度 $O(\, m \log n\,)$

#717、花神游历各国

题目描述

花神喜欢步行游历各国,顺便虐爆各地竞赛。花神有一条游览路线,它是线型的,也就是说,所有游历国家呈一条线的形状排列,花神对每个国家都有一个喜欢程度(当然花神并不一定喜欢所有国家)

每一次旅行中,花神会选择一条旅游路线,它在那一串国家中是连续的一段,这次旅行带来的开心值是这些国家的喜欢度的总和,当然花神对这些国家的喜欢程序并不是恒定的,有时会突然对某些国家产生反感,使他对这些国家的喜欢度 δ 变为 $\sqrt{\delta}$ (可能是花神虐爆了那些国家的 OI ,从而感到乏味)

现在给出花神每次的旅行路线,以及开心度的变化,请求出花神每次旅行的开心值

输入格式

第一行是一个整数 N ,表示有 N 个国家

第二行有 N 个空格隔开的整数,表示每个国家的初始喜欢度 δ_i

第三行是一个整数 M ,表示有 M 条信息要处理

第四行到最后,每行三个整数 x,l,r ,当 $x=1$ 时询问游历国家 l 到 r 的开心值总和,也就是 $\sum_{i=l}^{r}\delta_i$,当 $x=2$ 时国家 l 到 r 中每个国家的喜欢度 δ_i 变为 $\sqrt{\delta_i}$

输出格式

每次 $x=1$ 时,每行一个整数

表示这次旅行的开心度

样例输入

```
4
1 100 5 5
5
1 1 2
2 1 2
1 1 2
2 2 3
1 1 4
```

样例输出

```
101
11
11
```

数据范围与提示

对于全部数据, $1\le n\le 10^5, 1\le m\le 2\times 10^5, 1\le l\le r\le n, 0\le \delta_i\le 10^9$

注:建议使用 `sqrt` 函数,且向下取整

#717、花神游历各国

对于区间的修改,发现不好使用懒标记维护

观察发现 δ_i 至多进行至多执行 5 次 sqrt 操作将会变为 1 , 之后再开根号值大小不变

对于每一个节点维护

sum　　区间和

maxV　　区间最大值

对于每个区间修改操作尝试使用单点修改维护

若当前区间 maxV $\leq$ 1,那么修改将无意义,直接返回即可

时间复杂度 $O(m \log n)$

#2651、亚特兰蒂斯

题目描述

有几个古希腊书籍中包含了对传说中的亚特兰蒂斯岛的描述

其中一些甚至包括岛屿部分地图

但不幸的是,这些地图描述了亚特兰蒂斯的不同区域

您的朋友 $Bill$ 必须知道地图的总面积

你自告奋勇写了一个计算这个总面积的程序

输入格式

第一行包含整数 n ,表示总的地图数量

接下来 n 行,描绘了每张地图

每行包含四个数字 x_1, y_1, x_2, y_2 , (x_1, y_1) 和 (x_2, y_2) 分别是地图的左上角位置和右下角位置

注意,坐标轴 x 轴从上向下延伸, y 轴从左向右延伸

输出格式

总地图面积
即此测试用例中所有矩形的面积并
注意如果一片区域被多个地图包含,则在计算总面积时只计算一次

输入样例

```
2
100 100 200 200
150 150 250 255
```

输出样例

```
18000
```

数据范围

对于全部的数据范围 $1 \leq n \leq 10^5, 0 \leq x_1 < x_2 \leq 10^9, 0 \leq y_1 < y_2 \leq 10^9$

#2651、亚特兰蒂斯

将所有点离散化后处理

将矩形拆分成上边界和下边界两条线段,根据 y 轴排序

假设有一条平行于 x 轴的扫描线

若遇到上边界,将该线段内所有点覆盖次数 $+1$

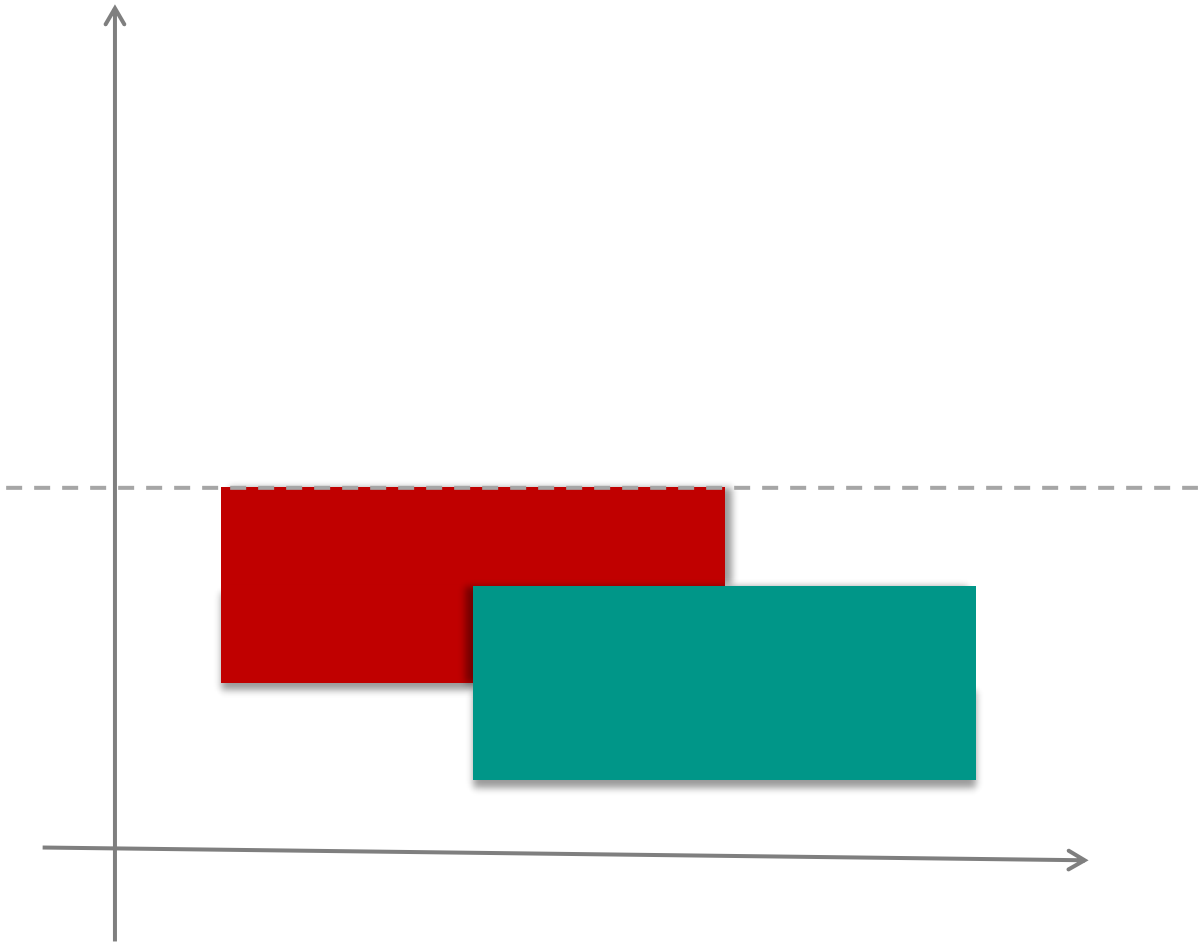
若遇到下边界,将该线段内所有点覆盖次数 -1

当遇到一条边界时

统计被覆盖的长度 len , 与上一条边界的高度差 h

累加 $len \times \ h$即可

时间复杂度 $O(n^2)$



#2651、亚特兰蒂斯

对于区间覆盖的长度, 使用线段树维护各区间内覆盖点的个数

让每个节点维护 $[l, r+1]$ 的信息

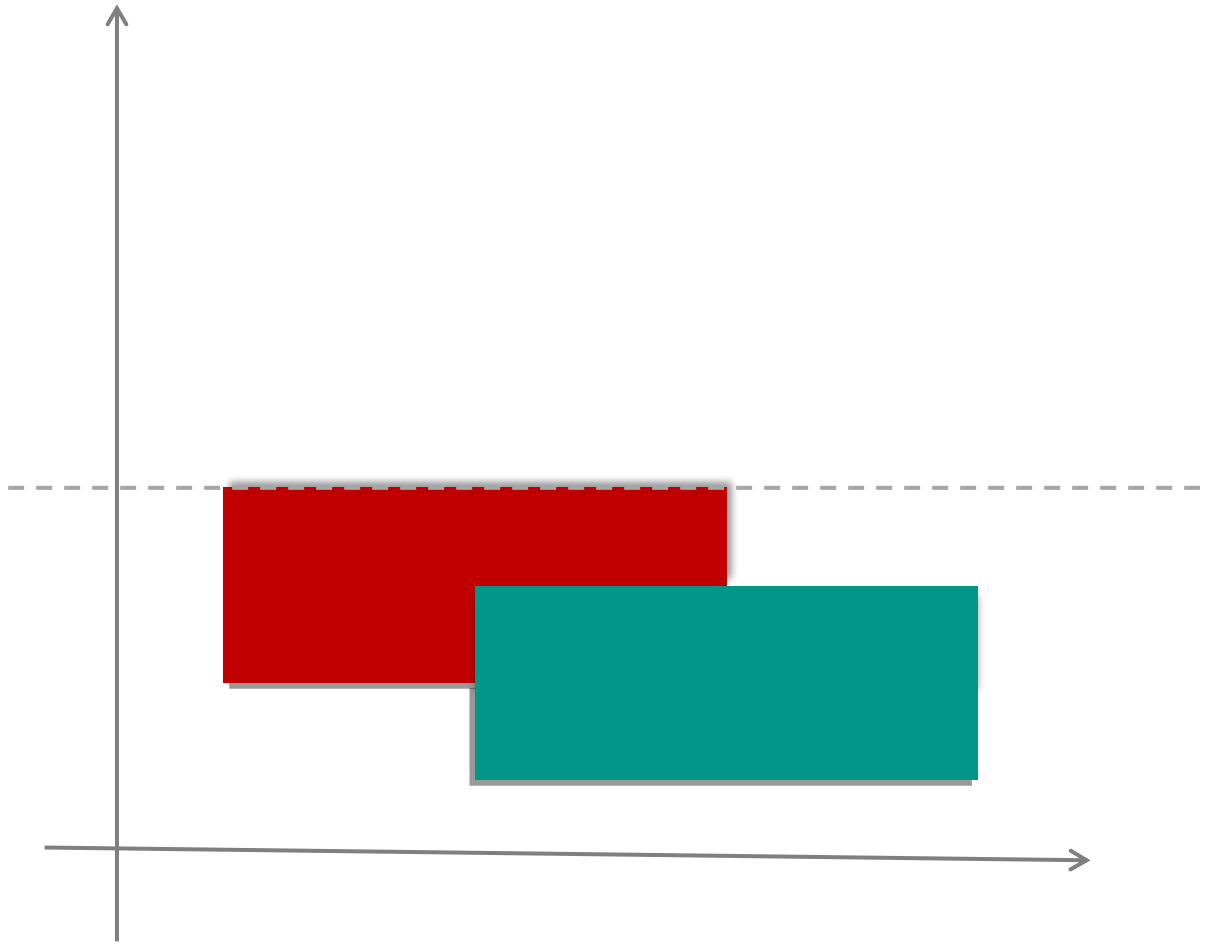
若只维护 $[l, r]$ 叶子节点无意义(即维护格点信息)

当遇到边界时,区间修改维护

向上合并时 , 需要根据覆盖次数累加长度

每次查询根节点维护的值,即为覆盖的点数

时间复杂度$O(n \log n)$



谢谢观看