提高算法班

树状数组

Mas

树状数组

树状数组/二叉索引树(Binary Index Tree) , 又以其发明者命名 Fenwick Tree

树状数组是一种简洁优美的数据结构

普通树状数组维护的信息及运算要满足 **结合律** 且 **可差分**

如加法、乘法、异或等

模意义下的乘法若要可差分，需保证每个数都存在逆元

结合律

$(x \circ y) \circ z = x \circ (y \circ z)$, 其中 $\circ$ 是一个二元运算符

可差分

具有逆运算的运算即已知 $x \circ y$ 和 x 可以求出 y

树状数组

最简单的树状数组支持两种操作，时间复杂度均为 $O(\log n)$

- 单点修改: 更改数组中一个元素的值
- 区间查询: 查询一个区间内所有元素的和

常见的运算是求和，由于加法满足相减性，即可以利用前缀查询做到区间查询

树状数组和线段树具有相似的功能

树状数组 支持的操作 线段树 一定支持，线段树 支持的操作 树状数组 不一定支持

但 树状数组 的代码要比 线段树 短，思维更清晰,速度也更快

在解决一些单点修改的问题时,树状数组 是不二之选

树状数组

考虑区间求和,预处理前缀和可做到 $O(1)$ 询问

若考虑单点修改可做到 $O(1)$ 修改,但需重新计算前缀和，时间复杂度$O(n)$

考虑一个节点维护若干个小区间

- 单点修改时

  只更新包含该元素的区间

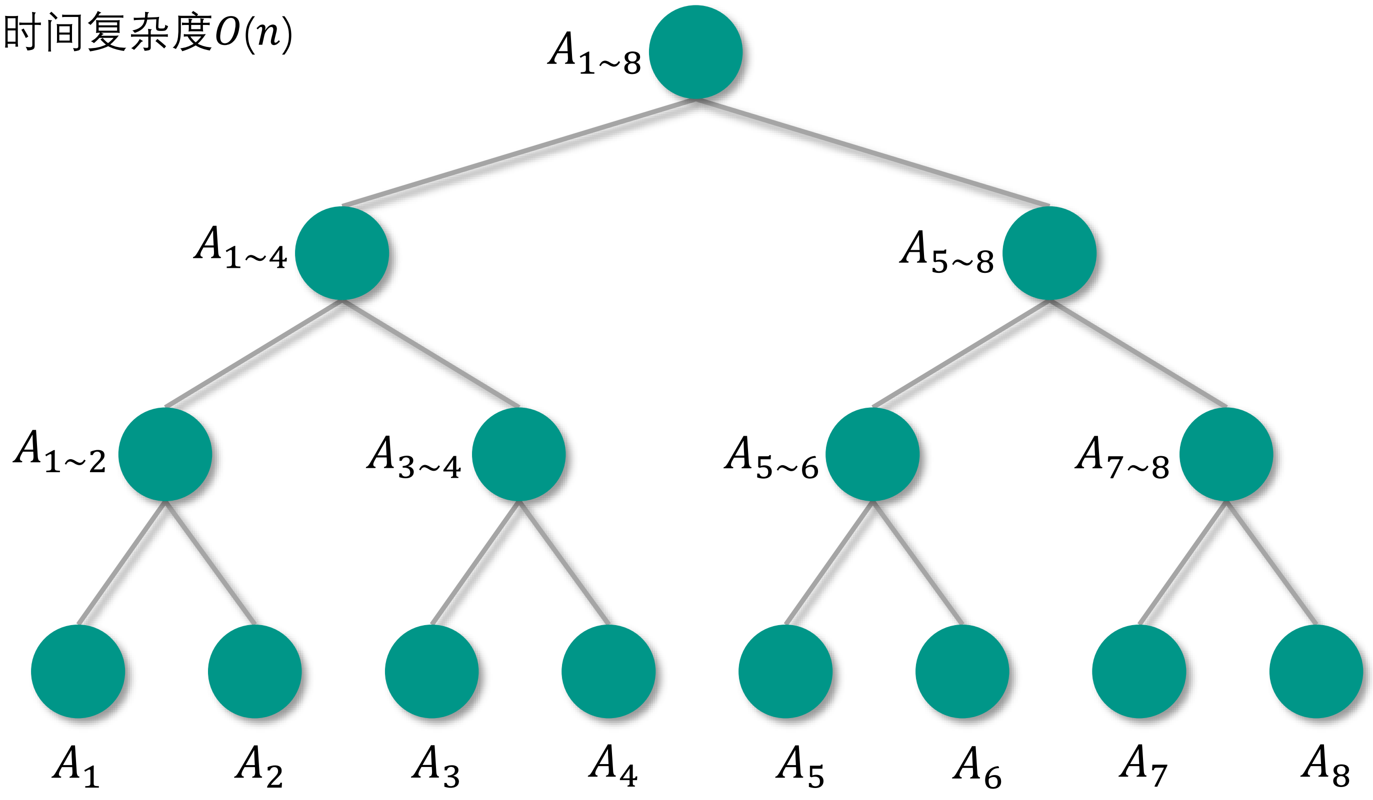
- 求前缀和时

  将区间进行拆分,再对用到的区间求和

不难想到如右图所示的数组划分方式

每个节点需存储/管理其左右孩子对应区间信息

对于叶子节点仅管理单个元素信息



树状数组

若仅考虑**前缀求和**

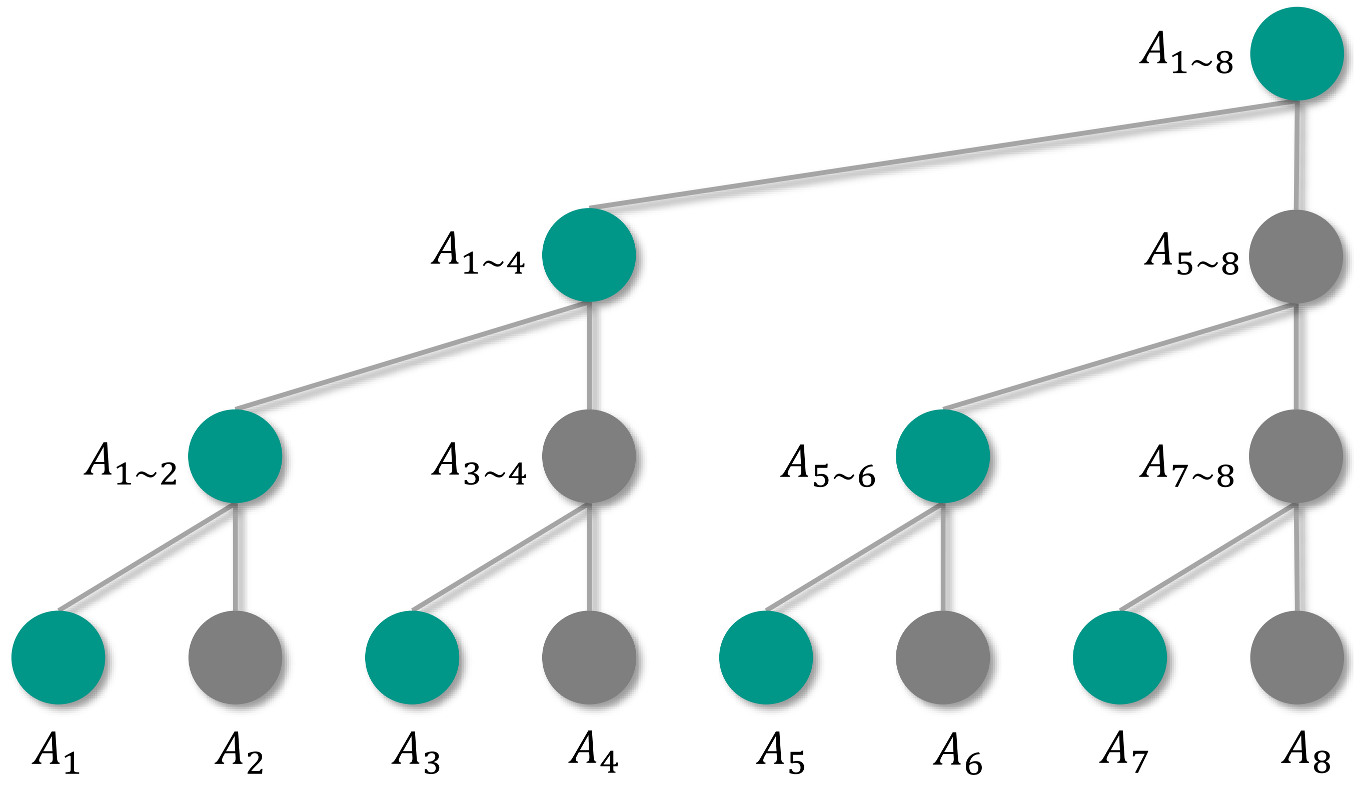
发现部分节点在实际计算中并不需要

如管理元素 A_2 的节点可被管理 $A_{1\sim2}$ 的节点完全替代

管理元素 $A_{3\sim4}$ 的节点可被管理 $A_{1\sim4}$ 的节点完全替代

管理元素 $A_{5\sim8}$ 的节点可被管理 $A_{1\sim8}$ 的节点完全替代

发现每一层的偶数节点存在没有意义



树状数组

将无意义节点移除,发现仅剩下 n 个节点,若规定最底层为第 0 层,发现第 i 层管理的元素个数为 2^i

将这些节点编号后可放入一个数组 C 内

在数组 C 中编号为节点管理最后的元素在原数组 A 中的编号

发现在节点在数组 C 中编号,其二进制位

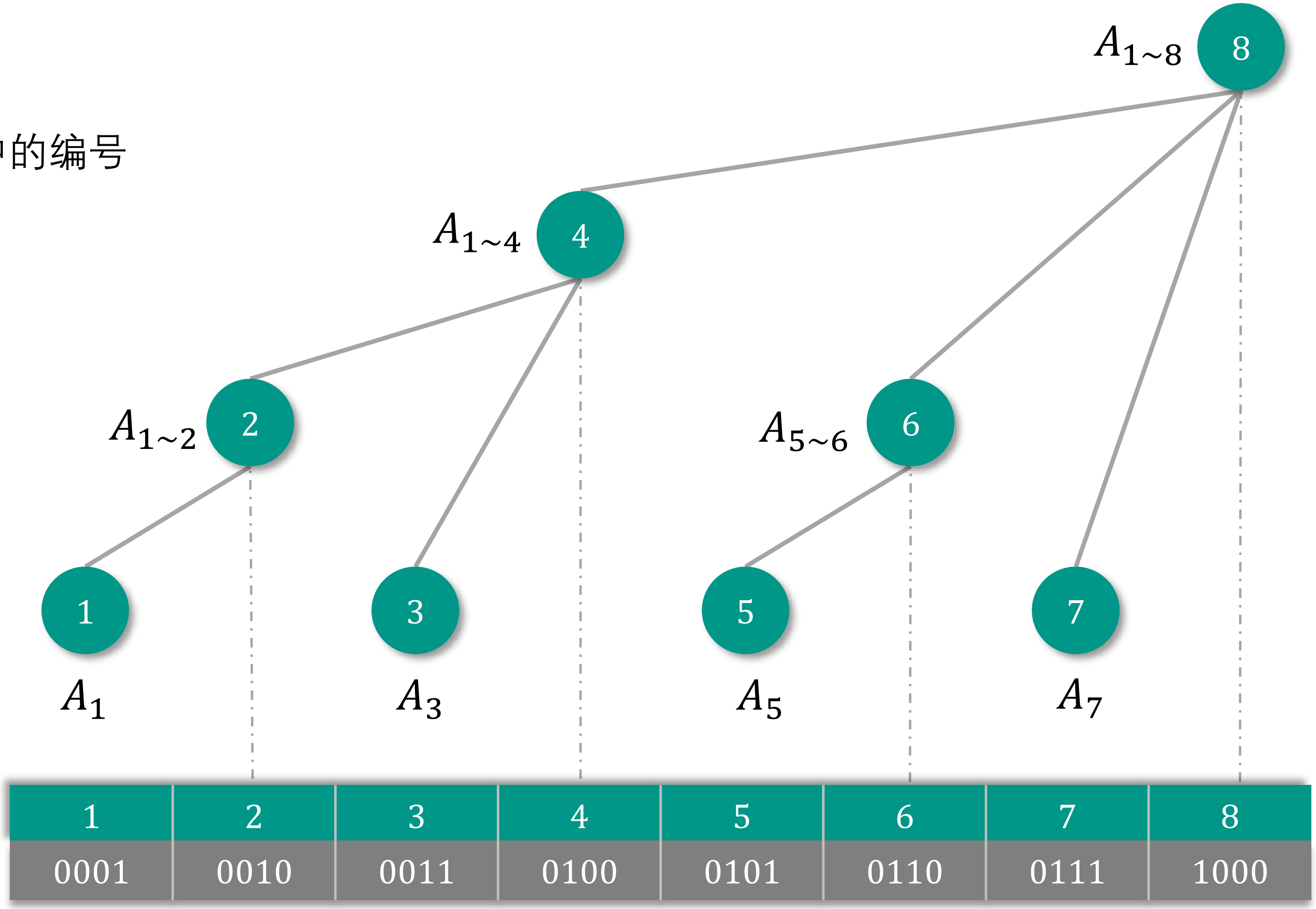
最低且为 1 的位权代表节点管理元素个数

如右图中

C_7 管理 A_7

C_6 管理 $A_{5\sim6}$

C_4 管理 $A_{1\sim4}$



树状数组

考虑在精简后的数组中进行求和

如求 $\sum_{i=1}^{7} A_i$

尝试找出与 $\sum_{i=1}^{7} A_i$ 相关的区间，其显然与 C_7 有关

观察 7 二进制形式

$$(7)_{10} = (0111)_2$$

可分别查询 $(6,7]$ ，$(4,6]$ ，$(0,4]$ 对应节点信息再相加

上述区间端点二进制 $(0110_2, 0111_2]$ ，$(0100_2, 0110_2]$ ，$(0000_2, 0100_2]$

不断地减去区间右端点二进制位最低且为 1 的位权

即可得到拆分区间

$$(0110\,,0111]$$

$$(0100\,,0110]$$

$$(0000\,,0100]$$

树状数组

定义整数 x 的二进制位中最低且为 1 的位权为 $\text{lowbit}(x)$

如 $\text{lowbit}(7)=1$, $\text{lowbit}(12)=4$

那么 C_x 维护区间 $(x-\text{lowbit}(x), x]$ 的信息

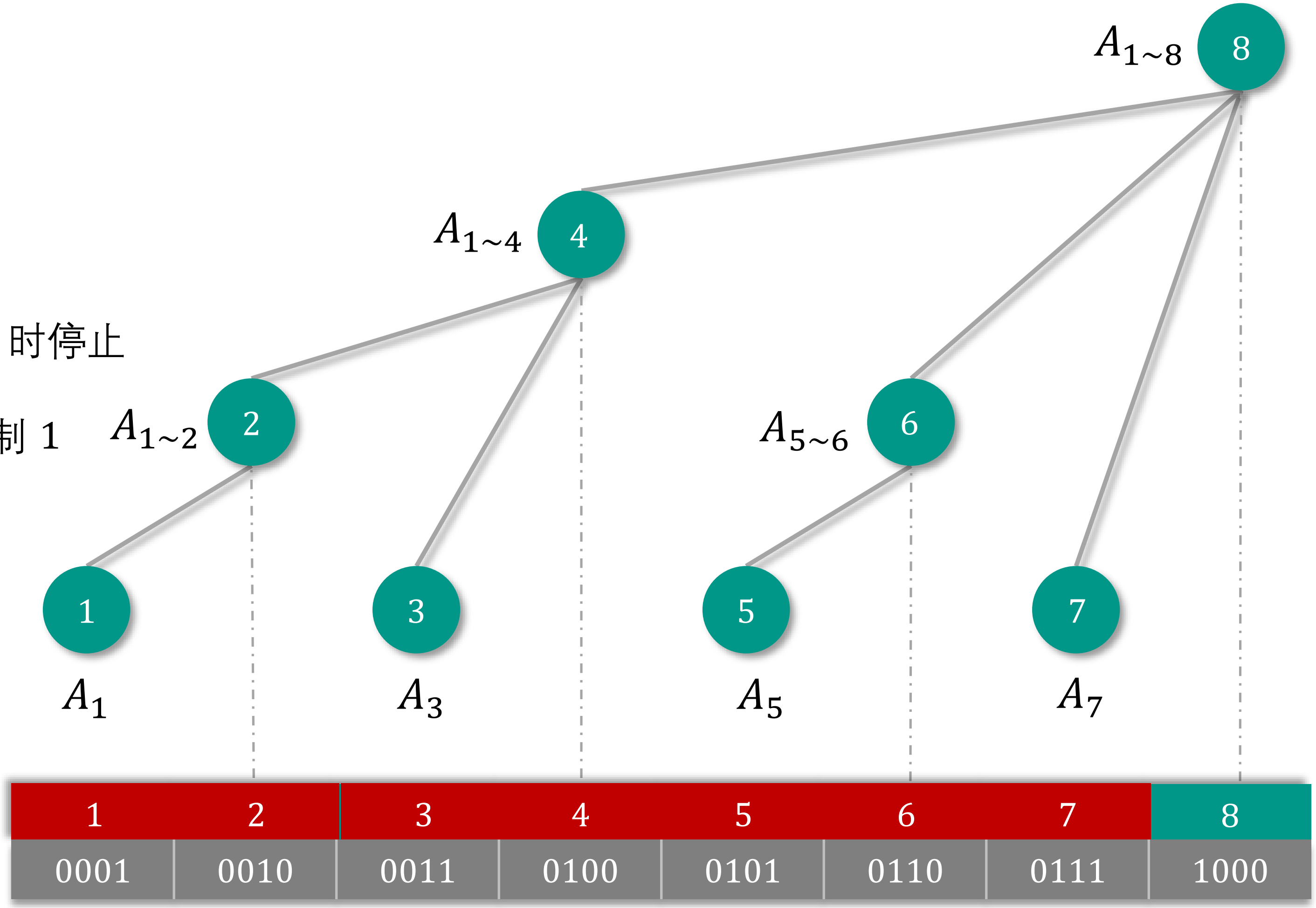
对于求 $\sum_{i=1}^{x} A_i$

累加 C_x 并不断令 $x \leftarrow x-\text{lowbit}(x)$, 直到 x 变为 0 时停止

若有 n 个元素每次减去 lowbit 可看作消去一位二进制 1

所以考察的区间数不超过 $\log n$ 个

时间复杂度 $O(\log n)$



lowbit

考虑求出 $\text{lowbit}(x)$

$$\text{lowbit}(x) = x \,\&\, -x$$

计算机里有符号数一般是以 **补码** 的形式存储

在补码表示, x 的相反数 $-x = \sim x + 1$

$-x$ 相当于 x **按位取反再加 1**，会把结尾处原来 1000 ... 的形式,变成0111 ... ,再变成1000 ...

而前面每一位都与原来相反，再将其和 x 按位与，得到的结果便是 $\text{lowbit}(x)$

原	01001000	原	01100000
反	10110111	反	10011111
补	10111000	补	10100000
	00001000		00100000

树状数组

实验舱 青少年编程 LAB 走近科学 走进名校

记 $\mathrm{l}(x) = x - \mathrm{lowbit}(x) + 1$ 即 $\mathrm{l}(x)$ 为 $C[x]$ 管辖左端点

对于任意正整数 x , 总能将 x 表示为 $s \times 2^{k+1} + 2^k$, 其中 $s \in N$ 且 $2^k = \mathrm{lowbit}(x)$

下列描述中使用 $C[x]$ 表示 $C[x]$ 所辖区间

- **对于 $x \le y$ 要么有 $C[x]$ 和 $C[y]$ 不相交要么有 $C[x]$ 包含于 $C[y]$**

  假设 $C[x]$ 与 $C[y]$ 相交 , 即 $[\,\mathrm{l}(x), x\,]$ 与 $[\,\mathrm{l}(y), y\,]$ 相交 有 $\mathrm{l}(y) \le x \le y$

  设 $y = s \times 2^{k+1} + 2^k$ 那么 $l(y) = s \times 2^{k+1} + 1$

  设 $x = s \times 2^{k+1} + b$ 其中 $1 \le b \le 2^k$

  显然 $\mathrm{lowbit}(x) = \mathrm{lowbit}(b)$,又由于 $b - \mathrm{lowbit}(b) \ge 0$ 所以

$$\mathrm{l}(x) = x - \mathrm{lowbit}(x) + 1 = s \times 2^{k+1} + b - \mathrm{lowbit}(b) + 1$$

$$\ge s \times 2^{k+1} + 1 = \mathrm{l}(y)$$

  显然 $\mathrm{l}(y) \le \mathrm{l}(x) \le x \le y$, 命题得证

树状数组

- **在 $C[x]$ 真包含于 $C[x+\mathrm{lowbit}(x)]$**

  设 $y = x + \mathrm{lowbit}(x), x = s \times 2^{k+1} + 2^k$，则 $y = (s+1) \times 2^{k+1}$， $l(x) = s \times 2^{k+1} + 1$

  不难发现 $\mathrm{lowbit}(y) \geq 2^{k+1}$

  所以

  $$\mathrm{l}(y) = (s+1) \times 2^{k+1} - \mathrm{lowbit}(y) + 1 \leq s \times 2^{k+1} + 1 = \mathrm{l}(x)$$

  即 $\mathrm{l}(y) \leq \mathrm{l}(x) \leq x < y$，命题得证

- **对于任意 $x < y < x + \mathrm{lowbit}(x)$，有 $C[x]$ 和 $C[y]$ 不相交**

  设 $x = s \times 2^{k+1} + 2^k$，则 $y = x + b = s \times 2^{k+1} + 2^k + b$，其中 $1 \leq b < 2^k$

  不难发现 $\mathrm{lowbit}(y) = \mathrm{lowbit}(b)$，又因为 $b - \mathrm{lowbit}(b) \geq 0$ 所以

  $$\mathrm{l}(y) = y - \mathrm{lowbit}(y) + 1 = x + b - \mathrm{lowbit}(b) + 1 > x$$

  即 $\mathrm{l}(x) \leq x < \mathrm{l}(y) \leq y$，命题得证

树状数组

- **若 $\boldsymbol{u = s \times 2^{k+1} + 2^k}$, 则其儿子数量为 $\boldsymbol{k = \log_2 \mathrm{lowbit}(u)}$ 编号分别为 $\boldsymbol{u - 2^t (0 \le t < k)}$**

  x 减去 2^t 其二进制第 t 位反转更低位不变

  考虑 u 的儿子 v 有 $v + \mathrm{lowbit}(v) = u$, 即 $v = u - 2^t$ 且 $\mathrm{lowbit}(v) = 2^t$

  设 $u = s \times 2^{k+1} + 2^k$

  考虑 $0 \le t < k$

  u 的第 t 位及其后均为 0 , 所以 $v = u - 2^t$ 的第 t 位变为 1 其后仍为 0 满足 $\mathrm{lowbit}(v) = 2^t$

  考虑 $t = k$

  $v = u - 2^k$, v 的第 k 位变为 0 不满足 $\mathrm{lowbit}(v) = 2^t$

  考虑 $t = k$

  $v = u - 2^k$, v 的第 k 位是 1,所以 $\mathrm{lowbit}(v) = 2^k$ 不满足 $\mathrm{lowbit}(v) = 2^t$

树状数组

考虑更新单点的值，更新单点需将所有受影响的区间更新

观察发现更新是不断向上爬升完成的

如更新 A_2 的值，那么 C_2 、 C_4 、 C_8 会受到影响

即区间 $(0000_2, 0010_2]$，$(0010_2, 0100_2]$，$(0100_2, 1000_2]$

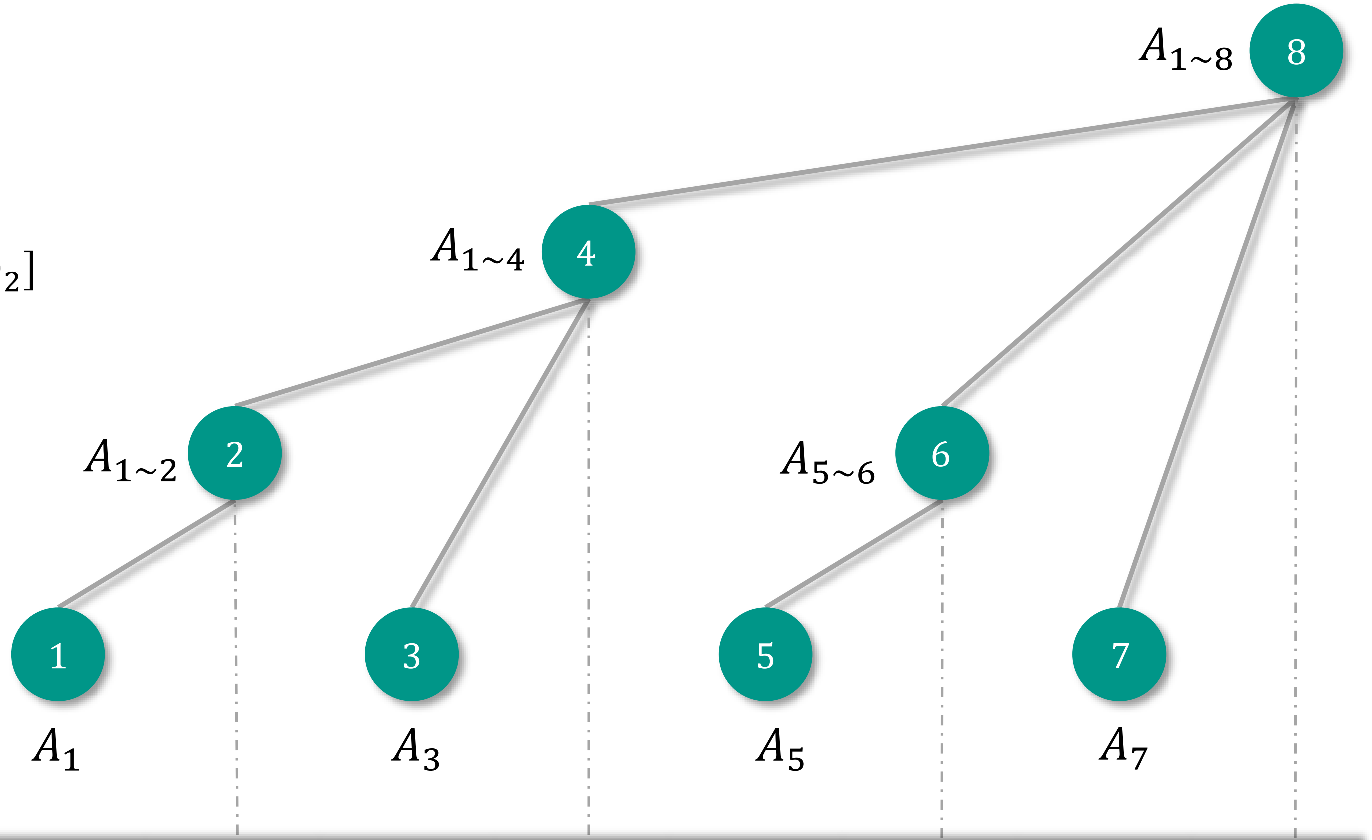
若要更新 A_x 考虑正确修改包含 A_x 的 C 数组信息

不断令 $x \leftarrow x + \text{lowbit}(x)$ 就能跳到 x 的上一级节点

不断修改，直到再无上一级节点时停止

树状数组若有 n 个元素，上跳区间数不超过 $\log n$ 个

时间复杂度 $O(\log n)$



1	2	3	4	5	6	7	8
0001	0010	0011	0100	0101	0110	0111	1000

单点更新、区间求和

根据上述实现进行单点更新

对于区间 $[L,R]$ 求和

求出 sum_R 和 sum_{L-1} 计算 $Sum_R - Sum_{L-1}$ 即可

$O(n)$ 建立树状数组

1. 每一个节点的值是由所有与自己直接相连的儿子的值求和得到的

   每次确定完儿子的值后,用自己的值更新自己的直接父亲即可

2. 由于 C_x 维护区间 $(x-\text{lowbit}(x)\,,x]$ 的信息

   预处理一个前缀和数组 sum , $C_x = sum_x - sum_{x-\text{lowbit}_\text{x}}$

```
int lowbit(int x)
{
    return x & (-x);
}
void update(int pos, int val)
{
    for (int i = pos; i <= n; i += lowbit(i))
        c[i] += val;
}
int query(int pos)
{
    int res = 0;
    for (int i = pos; i >= 1; i -= lowbit(i))
        res += c[i];
    return res;
}
```

#936、树状数组 1 ：单点修改,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 i x` : 给定 i, x ,将 a_i 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 $1 \le n, q \le 10^6$ 。

第二行 n 个整数 $a_1, a_2, \ldots, a_n$,表示初始数列。保证 $|a_i| \le 10^6$ 。

接下来 q 行,每行一个操作,为以下两种之一:

- `1 i x` : 给定 i, x ,将 a_i 加上 x
- `2 l r` : 给定 l, r ,求 $\sum_{i=l}^{r} a[i]$ 的值

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

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

样例输入

```
3 2
1 2 3
1 2 0
2 1 3
```

样例输出

```
6
```

数据范围与提示

对于所有数据, $1 \le n, q \le 10^6,\ \ |a_i| \le 10^6$, $1 \le l \le r \le n,\ \ |x| \le 10^6$

#900、树状数组 2： 区间修改,单点查询

实验舱 青少年编程 LAB 走近科学 走进名校

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 l 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 i` : 给定 i ,求 $a[i]$ 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 i` : 给定 i ,求 $a[i]$ 的值。

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

输出格式

对于每个 `2 i` 操作,输出一行,每行有一个整数,表示所求的结果

样例输入

```
3 2
1 2 3
1 1 3 0
2 2
```

样例输出

```
2
```

数据范围与提示

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

区间修改、单点查询

实验舱 青少年编程 走近科学 走进名校

通过差分可将问题转为单点修改、区间查询

原数组为 A ,记差分数组为 d

根据差分数组定义

$$A_x = \sum_{i=1}^{x} d_i$$

对于区间 $[L,R]$ 加上定值 v 只需要令

$$d_L \mathrel{+}= v$$

$$d_{R+1} \mathrel{-}= v$$

单点查询直接查询 $\sum_{i=1}^{x} d_i$ 即可

单次修改/查询 $O(\log n)$

#937、树状数组 3： 区间修改,区间查询

题目描述

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

> `l 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* $a[l],a[l+1],\dots,a[r]$ *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* $a[l]+a[l+1]+\dots+a[r]$ *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 $1\le n,q\le 10^6$

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

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

样例输入

```
5 10
2 6 6 1 1
2 1 4
1 2 5 10
2 1 3
2 2 3
1 2 2 8
1 2 3 7
1 4 4 10
2 1 2
1 4 5 6
2 3 4
```

样例输出

```
15
34
32
33
50
```

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, $1\le n,q\le 10^6,\ \ |a[i]|\le 10^6$, $1\le l\le r\le n,\ \ |x|\le 10^6$

区间修改、区间求和

根据差分数组定义

$$\sum_{i=1}^{x} A_i = \sum_{i=1}^{x}\sum_{j=1}^{i} d_j = \sum_{i=1}^{x}(x-i+1)\times d_i$$

$$= (x+1)\times\sum_{i=1}^{x} d_i - \sum_{i=1}^{x}(d_i \times i)$$

维护两个数组 $C1_i = d_i$, $C2_i = d_i \times i$

对于区间 $[L, R]$ 加上定值 v

令 $C1_L \mathrel{+}= v$, $C1_{R+1} \mathrel{-}= v$

同时令 $C2_L \mathrel{+}= v \times L, C2_{R+1} \mathrel{-}= v \times (R+1)$

对于求和累加 $(x+1)\times C1_x - C2_x$ 即为答案

单次修改/查询 $O(\log n)$

二维树状数组

- 单点修改、区域查询

  增加一个维度即可

- 区间修改、单点查询

  二维差分数组维护即可

- 区域修改、区域查询

$$\sum_{i=1}^{x}\sum_{j=1}^{y}A_{ij}=\sum_{i=1}^{x}\sum_{j=1}^{y}\sum_{k=1}^{i}\sum_{h=1}^{j}d_{kh}=$$

$$(x+1)\times(y+1)\times\sum_{i=1}^{x}\sum_{j=1}^{y}(d_{ij})-(y+1)\times\sum_{i=1}^{x}\sum_{j=1}^{y}(d_{ij}\times i)-(x+1)\times\sum_{i=1}^{x}\sum_{j=1}^{y}(d_{ij}\times j)+\sum_{i=1}^{x}\sum_{j=1}^{y}(d_{ij}\times i\times j)$$

参考一维差分,分别使用四个数组维护

#322、逆序对数

题目描述

给你一个 n 个数的数组，逆序对定义为:

存在两个整数 $i < j$ 使得 $a_i > a_j$

你需要求出逆序对的个数

输入

第一行 n ，代表数组长度
第二行 n 个数，代表 $a_i (1 \le i \le n)$

输出

输出逆序对数

输入样例

```
4
4 3 2 1
```

输出样例

```
6
```

数据规模

对于 10% 的数据，$n \le 100, 1 \le a_i \le 100$
对于 40% 的数据，$n \le 10000, 1 \le a_i \le 10^9$
对于 100% 的数据，$n \le 100000, 1 \le a_i \le 10^9$

#322、逆序对数

可以将数组各值离散化后处理, 维护各值出现的次数

- 逆序遍历数组,对于当前元素 A_i

  若知道 $j > i$ 且 $A_j < A_i$ 值的数量 , 即为 A_i 能构成的逆序对数量

  同时将 A_i 所在位置点的个数加一

  这是一个单点修改、区间求和的问题

  可以考虑树状数组优化,时间复杂度 $O(n\log n)$

- 顺序遍历数组,对于当前元素 A_i

  设 x 为 A_i 插入之前有小于A_i数的数量,那么 $i - x$ 就是 在 A_i 插入之前的大于A_i 数的个数,累加即可

  同样可以考虑数组数组优化,时间复杂度$O(n\log n)$

#1038、最长上升子序列

题目描述

设有整数序列 $b_1, b_2, b_3, \ldots, b_m$

若存在

$$i_1 < i_2 < i_3 < \ldots < i_n$$

且

$$b_{i_1} < b_{i_2} < \cdots < b_{i_n}$$

则称 $b_1, b_2, b_3, \ldots, b_m$ 中有长度为 m 的不下降序列

$$b_{i_1}, b_{i_2}, \ldots, b_{i_n}$$

求序列 $b_1, b_2, b_3, \ldots, b_m$ 中所有长度(n)最大不下降子序列

输入格式

第一行一个整数 M（$M \leq 10000$）
接下来输入 M 个用空格隔开的整数 (≤ 20000) 序列;

输出格式

一行一个整数，表示最大长度 n

可以将数组各值离散化后处理

记 $dp[i]$ 表示 以 $A[i]$ 结尾的最长上升子序列长度

将 dp 数组根据值 $A[i]$ 大小映射到树状数组上

逆序遍历数组,对于当前元素 $A[i]$

查询 $1 \sim A[i]-1$ 范围内查询最大值，最大值 $+1$ 即为答案

不可差分信息

区间查询

以查询 $\max\limits_{L \le i \le R} a_i$ 为例

从 R 沿着 lowbit 一直向前跳,但不可跳到 L 的左边

若跳到了 $C[x]$ 需判断 $x - \text{lowbit}(x)$

- 若 $x - \text{lowbit}(x) < L$

  说明越界，将 $a[x]$ 合并到总信息跳至 $C[x - 1]$

- 若 $x - \text{lowbit}(x) \ge L$

  说明没越界, 正常合并 $c[x]$,然后跳到 $C[x - \text{lowbit}(x)]$ 即可

考虑 R 和 L 不同的最高位，必有 R 在该位为 1,而 L 在该位为 0

若 R 在该位后仍然有 1 ,必有 $R - \text{lowbit}(R) \ge L$ ，下一步即为将 R 的最低位 1 填为 0

若 R 的这一位 1 就是 R 的最低位 1，无论是 $R \leftarrow R - \text{lowbit}(R)$ 还是 $R \leftarrow R - 1$，R 的这一位 1 一定会变为 0

```
int queryMax(int l, int r)
{
    int res = 0;
    while (r >= l)
    {
        res = max(res, a[r--]);
        for (; r - lowbit(r) >= l; r -= lowbit(r))
            res = max(res, C[r]);
    }
    return res;
}
```

循环条件不可写为 $R - \text{lowbit}(R) + 1 \ge L$

不可差分信息

因此 R 经过至多 $\log n$ 次变换后 R 和 L 不同的最高位必可下降一位

时间复杂度 $O(\log^2 n)$

单点更新

以 $a[x] \leftarrow p$ 后续查询最大值为例，记 y 为 x 在树状数组中的上级编号

$C[y] = \max(C[y], p)$？

模拟 $A = [1,2,3,4,5,0,0,0]$ 将 5 修改成 4

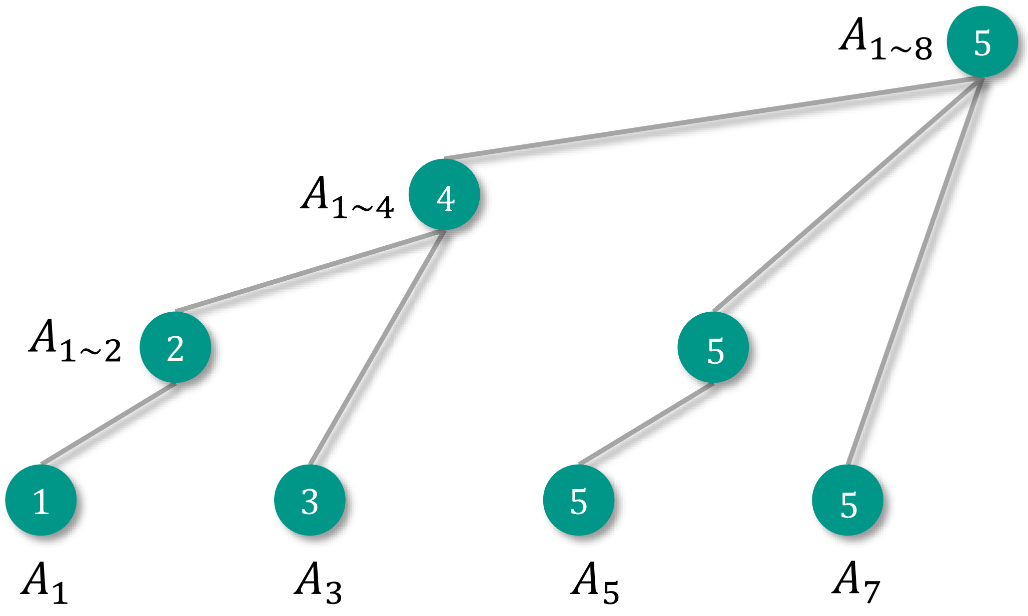
$C[y] = p$？

模拟 $A = [1,2,3,4,5,0,0,0]$ 将 3 修改成 4？

移除时对区间信息而言相当于做逆运算

修改可视为将原数从区间移除再加入新数

由于不可差分信息不存在逆运算，所以无法直接修改达成目的



不可差分信息

考虑对于每个 $C[y]$ 重构区间信息

事实上 $C[y]$ 的子节点信息必然正确（先更新子节点再更新父节点）

由于子节点组成了（$y-\text{lowbit}(y)$，$y\ -\ 1$］

那么再合并一个 $a[y]$ 即可合并得出 （$y-\text{lowbit}(y)$，y ］

至多用 $\log n$ 个区间重构合并出续修改的 $C[y]$

即用 $C[y-1]$， $C[y-2]$， $C[y-4]$， $\cdots$ ， $C[y-\text{lowbit}(y)]$ 与 $a[y]$ 得出正确的 （$y-\text{lowbit}(y)$，y ］

由于至多上跳 $\log n$ 个节点，每个节点进行 $\log n$ 次合并

单点修改时间复杂度为 $O(\log^2 n)$

```
void update(int x, int v)
{
    a[x] = v;
    for (int i = x; i <= n; i += lowbit(i))
    {
        C[i] = a[i];
        for (int j = 1; j < lowbit(i); j <<= 1)
            C[i] = max(C[i], C[i - j]);
    }
}
```

线段树仅需 $O(\log n)$ 即可实现单调修改、区间查询

#2639、子段的平均数

题目描述

给你一个长度为 n 整数数组 A 以及一个整数 k

数组 A 有 $\frac{n \times (n+1)}{2}$ 个连续区间 $A_l, A_{l+1}, \cdots, A_{r-1}, A_r$

请你统计有多少个连续子区间的算术平均值大于等于 k

输入格式

第一行输入两个正整数 n, k
接下来每行一个正整数 A_i

输出格式

请你输出有多少个连续子区间的算术平均值大于等于 k

数据规模

对于 15% 的数据 $1 \leq N \leq 1000$
对于全部的数据 $1 \leq N \leq 2 \times 10^5, 1 \leq k, a_i \leq 10^9$

输入样例1

```
3 6
7
5
7
```

输出样例1

```
5
```

样例解释1

一共 5 个区间满足条件 $\{7\}$，$\{7,5\}$，$\{7,5,7\}$，$\{5,7\}$，$\{7\}$

#2639、子段的平均数

求出前缀和数组 sum

要求二元组 (l,r) 满足 $sum_r - sum_{l-1} \geq (r-l+1) \times K$ 的数量

不妨令 $A_i \leftarrow A_i - K$

再求出前缀和 sum

问题转化为

要求二元组 (l,r) 满足 $sum_r \geq sum_{l-1}$ 的数量

将 sum 离散化后使用权值树状数组维护

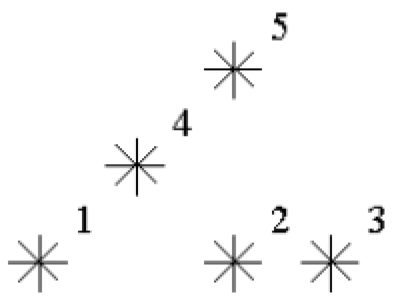
时间复杂度 $O(N \log N)$

#707、数星星 Stars

题目描述

天空中有一些星星,这些星星都在不同的位置,每个星星有个坐标

如果一个星星的左下方（包含正左和正下）有 k 颗星星,就说这颗星星是 k 级的



例如,上图中星星 5 是 3 级的（$1, 2, 4$ 在它左下）,星星 $2, 4$ 是 1 级的

例图中有 1 个 0 级, 2 个 1 级, 1 个 2 级, 1 个 3 级的星星

给定星星的位置,输出各级星星的数目

一句话题意 给定 n 个点,定义每个点的等级是在该点左下方（含正左、正下）的点的数目,试统计每个等级有多少个点

数据范围与提示

对于全部数据, $1 \leq N \leq 1.5 \times 10^4, 0 \leq x, y \leq 3.2 \times 10^4$

输入格式

第一行一个整数 N ,表示星星的数目

接下来 N 行给出每颗星星的坐标,坐标用两个整数 x, y 表示

不会有星星重叠

星星按 y 坐标增序给出, y 坐标相同的按 x 坐标增序给出

输出格式

N 行,每行一个整数,分别是 0 级, 1 级, 2 级,……, $N-1$ 级的星星的数目

#707、数星星 Stars

由于 y 轴已经有序，仅需考虑 x 轴

对于每个点 (x_i, y_i)，仅需知道 $j < i$ 且 $x_j \le x_i$ 的星星数量(二维偏序)

树状数组维护即可

注意坐标可能为 0，将下标 统一 +1 处理

时间复杂度 $O(N \log \max(x))$

实验舱
青少年编程
走近科学 走进名校

谢谢观看