区间问题

西南大学附属中学 High School Affiliated to Southwest University

题目描述

给出一个长为 n 的数列，以及 m 个操作，操作涉及区间加法，单点查值，区间求和，$n,m<=10^5$，后续的n,m数据规模也如此。

算法	特点	整体时间复杂度
树状数组	实现容易 扩展性弱	O((n+m)logn)
线段树	实现麻烦 扩展性强	O((n+m)logn) 常数比树状数组大

对于区间问题，除了线段树和树状数组
还可以使用“优雅的暴力”——分块
特别是当求众数等区间不满足可加性时，运用分块更容易实现。

信息学暑期

根号算法-数列分块

分块

分块是维护序列的数据结构，是一种“准暴力”手段，它比树形数据结构好写，也便于调试，但代码不一定短

分块的基本思想是通过适当的划分，预处理适当大小的信息并保存下来，用空间换取时间，使得修改以及查询的时间复杂度都能被接受，达到“时空平衡”。

支持的操作：

1.单点修改/查询；

2.区间修改/查询；

分块步骤：

1.预处理；

2.修改；

3.查询；

预处理

首先定义块操作的基本要素,便于后续维护和处理:

(1)**块的大小**: 用block表示。

(2)**块的数量**: 用t表示。

(3)**块的左右边界**: 用数组st[],ed[]表示, st[i], ed[i]就是第i块的起点与终点。

```
st[i]=(i-1)*block+1
ed[i]=i*block
```

(4)**每个元素所属的块**: belong[i]表示第i个元素属于哪个块。

通常我们把block的大小设定为$\sqrt{n}$取整, 大多数情况下这样的算法效率最高
但有时候也要根据题目灵活调整块的大小

预处理

证明

$$O(\text{siz} \times \text{处理角块的复杂度} + \text{num} \times \text{处理整块的复杂度})$$

由于$O(\text{siz} \times \text{num}) = O(N)$，我们可以使用均值不等式（即基本不等式）调整两个参数的值。

我们设$u =$ 处理角块的复杂度，$v =$ 处理整块的复杂度，则有

$$O = u \times \text{siz} + v \times \text{num}$$

$$O = u \times \text{siz} + v \times \frac{N}{\text{siz}}$$

$$O \geq \sqrt{u \times \text{siz} \times v \times \frac{N}{\text{siz}}} = \sqrt{u \times v \times N}$$

当且仅当$\text{siz} = \sqrt{\frac{v \times N}{u}}$时等号成立，复杂度最低

分块的时间复杂度为O((n+m)$\sqrt{n}$)

具体题目还应当将其他操作的时间复杂度计算进来

分块-根号算法

预处理

```
int block = sqrt(n);              //块的大小：每块有block个元素。
int t = n/block;                  //块的数量：共分为t块
if(n % block) t++;                //sqrt(n)的结果不是整数，最后加一小块
for(int i=1; i<=t; i++){          //遍历块
    st[i] = (i-1)*block+1;
    ed[i] = i*block;
}
ed[t] = n;                        //sqrt(n)的结果不是整数，最后一块较小
for(int i=1; i<=n; i++)           //遍历所有元素的位置
 belong[i]=(i-1)/block + 1;
```

操作-区间修改

以[L,R]每个数加上一个d为例。

首先需要记录原本每块的总和

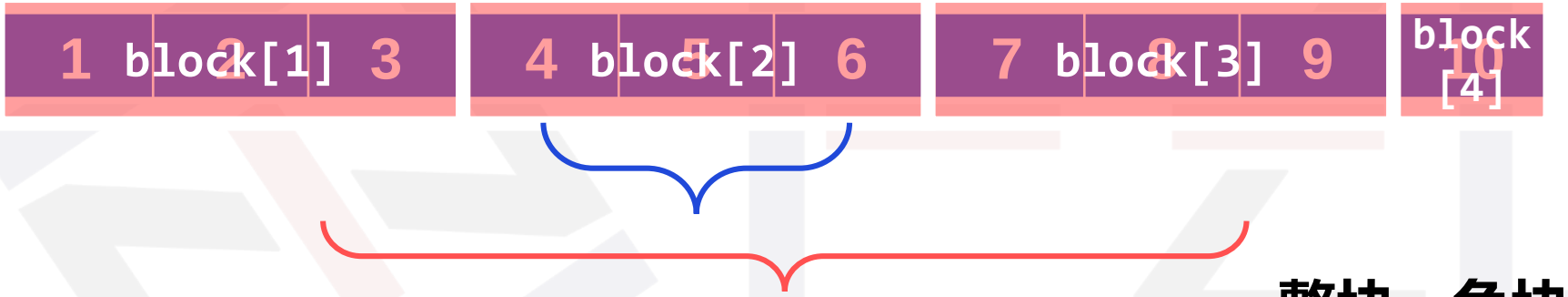
引入数组sum

sum[i]辅助记录第i块的和

```
for(int i=1; i<=t; i++) //遍历所有的块
    for(int j=st[i]; j<=ed[i];j++) //遍历块i内的所有元素
        sum[i] += a[j];
```

操作-区间修改

以[L,R]每个数加上一个d为例。



[L,R]分为两种情况：
1、完全被整块包含
2、不能完全被整块包含，但零散块(角块)长度和不超过2t

整块、角块的信息如何维护？

再引入数组add
add[i]标记第i块加上了多少
类似于线段树的lazy

大段维护，局部朴素
“整块打包维护，碎片逐个枚举”

优雅的暴力

区间修改代码


```
void change(int L,int R,int d){
    int p = belong[L], q = belong[R];
    if(p==q){                          //情况1
       for(int i=L;i<=R;i++)  a[i]+=d;
       sum[p]+=d*(R-L+1);
    }
    else{                              //情况2
       for(int i=p+1;i<=q-1;i++)    add[i]+=d; //整块，有m=(q-1)-(p+1)+1个。计算m次
       for(int i=L;i<=ed[p];i++)    a[i]+=d;   //整块前面的碎片。计算n/t次
       sum[p]+=d*(ed[p]-L+1);
       for( int i=st[q];i<=R;i++)   a[i]+=d;   //整块后面的碎片。计算n/t次
       sum[q]+=d*(R-st[q]+1);
    }
 }
```

整块直接标记不会超过$\sqrt{n}$块，两边暴力修改$\sqrt{n}$次

时间复杂度O($\sqrt{n}$)

操作-单点修改

先将所在块的add标记下传，操作$\sqrt{n}$次，在暴力修改对应位置
时间复杂度O($\sqrt{n}$)

操作-单点询问

输出a[i]加上对应块的add标记的值即可
时间复杂度O(1)

操作-区间询问

以[L,R]的和为例。

[L,R]分为两种情况：
1、完全被整块包含
2、不能完全被整块包含

区间询问和区间修改是类似的

区间询问代码


```
long long query(int L,int R) {
    int p=belong[L],q=belong[R];
    long long ans=0;
    if(p==q){                          //情况1
       for(int i=L;i<=R;i++)           ans += a[i];
       ans+=add[p]*(R-L+1);
    }
    else{  //情况2
       for(int i=p+1;i<=q-1;i++)  ans+=sum[i]+add[i]*(ed[i]-st[i]+1);//整块
       for(int i=L;i<=ed[p];i++)  ans += a[i];   //整块前面的碎片
       ans += add[p]*(ed[p]-L+1);
       for(int i=st[q];i<=R;i++)  ans += a[i];   //整块后面的碎片
       ans += add[q]*(R-st[q]+1);
    }
    return ans;
 }
```

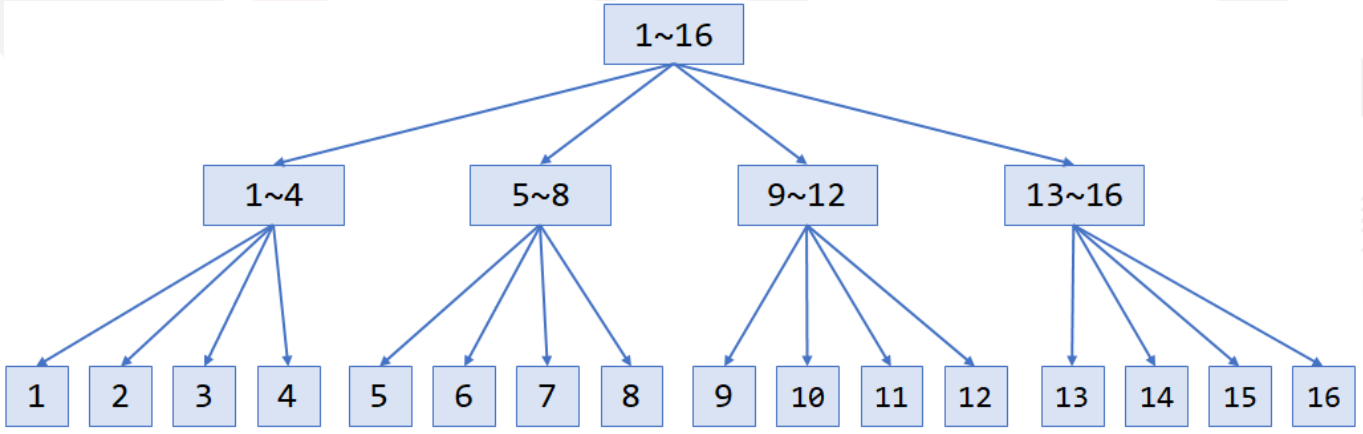
时间复杂度O($\sqrt{n}$)

小结

分块算法实质上是一种是通过分成多块后在每块上打标记以实现快速区间修改，区间查询的一种算法。

虽然分块时间复杂度没有树形结构算法优秀，但综合编程复杂度、可扩展性等，在考场上也是常见的解题方法。

如果像线段树一样组织的话，分块是一棵3层树



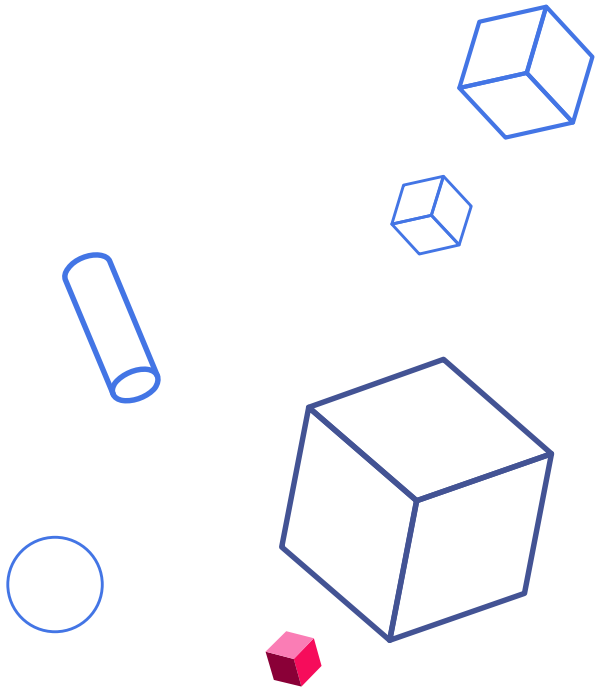
第一层：序列

第二层：块

第三层：单个值

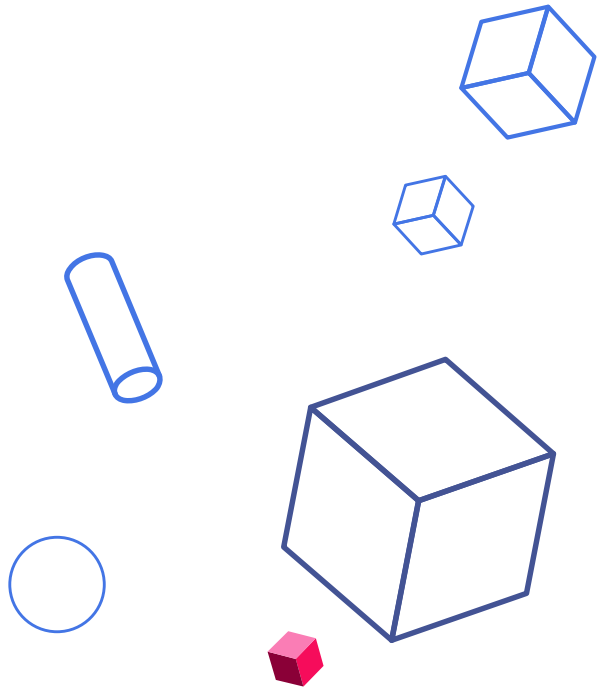
根据刚才的讲解
40min：完成数列分块1、4

代码展示



FZOI
西大附中信息学竞赛
High School Affiliated to Southwest University

西南大学附属中学
High School Affiliated to Southwest University



分块九题部分讲解

根据题目的相似或相关性
并不一定按照顺序讲解

数列分块2

题目描述

给出一个长为n的数列，以及n个操作，操作涉及区间加法，询问区间内小于某个值的元素个数。

对于区间查询：

对于不完整的块直接枚举统计即可；
对于完整块，要在每个整块内寻找小于一个值的元素数，这要求块内元素是有序的，才能使用二分法对块内查询，需要预处理时每块做一遍排序，复杂度O(nlogn)，每次查询在$\sqrt{n}$个块内二分，以及暴力$2\sqrt{n}$个元素，总时间复杂度O(nlogn+n $\sqrt{n}$ log $\sqrt{n}$)

数列分块2

题目描述

给出一个长为n的数列，以及n个操作，操作涉及区间加法，询问区间内小于某个值的元素个数。

对于区间加法：

对于不完整的块，修改后会使该块内数字乱序。因此对于每次区间修改的不完整块需要重新排序

对于完整的块直接维护加法标记即可。

由于每个块都需要排序，有重复元素也可以尝试块内维护multiset等数据结构，如果有插入和删除操作，multiset会更快些

数列分块3

题目描述

给出一个长为 n的数列，以及 n个操作，操作涉及区间加法，询问区间内小于某个值 x 的前驱（比其小的最大元素）。

修改数列分块2的二分

由于是要找的是 x 的前驱，因此暴力找左右不完整块的比 x 小最大值，再二分找整块比 x 小的最大值，取其中最大即可

时间复杂度不变

题目描述

给出一个长为n的数列，以及n个操作，操作涉及区间乘法，区间加法，单点询问。

很显然，如果只有区间乘法，和分块入门 1 的做法没有本质区别，但要思考如何同时维护两种标记。在《「AHOI2009」维护序列》线段树做法中，也涉及过此类维护。

两种标记具有优先级关系乘法标记的优先级高于加法。
若当前的一个块乘以m1后加上a1，这时进行一个乘m2的操作，则原来的标记变成m1*m2，a1*m2
若当前的一个块乘以m1后加上a1，这时进行一个加a2的操作，则原来的标记变成m1，a1+a2

数列分块5

题目描述

给出一个长为 n的数列，以及 n个操作，操作涉及区间开方，区间求和。

由于涉及开方，整块无法开方，需要对每一个元素进行操作后，才能知道开方后的总和。

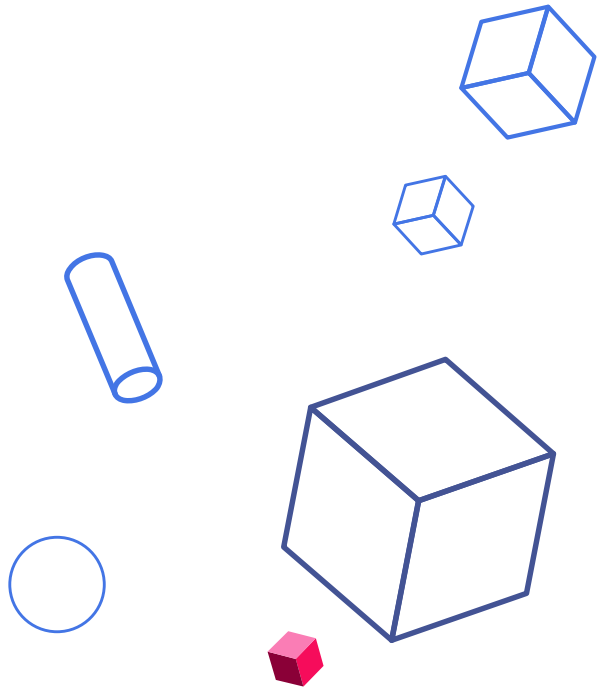
问题：难以对一个块进行维护，怎么办？

一个数经过几次开方之后，它的值就会变成 0 或者 1。

如果每次区间开方只不涉及完整的块，意味着不超过2 $\sqrt{n}$个元素，直接暴力即可。如果涉及了一些完整的块，这些块经过几次操作以后就会都变成 0 / 1，于是我们采取一种分块优化的暴力做法，只要每个整块暴力开方后，开数组记录一下元素是否都变成了 0 / 1，区间修改时跳过那些全为 0 / 1 的块即可。

由于数列中的元素最大为 $2^{31}-1$，因此最多开方 5 次后就为 1，时间复杂度在允许范围之内

拓展学习部分



数列分块8

题目描述

给出一个长为 n的数列，以及 n个操作，操作涉及区间询问等于一个数 c 的元素，并将这个区间的所有元素改为 c。

刚开始是有不同的权值，但随着询问次数增多，可能就几种权值区间了

使用分块5的方法，建立一个标志数组，在维护每个整块的时候，判断该块是否只有一种权值，这样在区间操作的时候，对于同权值的一个块能在 O(1) 即可统计答案，若不是只有一种权值，那么就暴力统计答案，然后修改标记，而对于不完整块，同样使用暴力维护。

这样一来，最差的情况每次都会消耗 O(n) 的时间，但实际上，可以这样进行分析：假设初始序列都是同一个值，那么查询是 $O(\sqrt{n})$，如果此时进行一个区间操作，其最多破坏首尾两个块的标记，因此只能使后面的询问至多多两个块的暴力时间，因此均摊后每次操作的复杂度仍为 $O(\sqrt{n})$，简单来说，要想让一个操作花费 O(n) 的时间，就要先花费 $\sqrt{n}$ 个操作对数列进行修改

数列分块6

题目描述

给出一个长为 n的数列，以及 n个操作，操作涉及单点插入，单点询问，数据随机生成。

数据是随机生成的，可以在每块内维护一个 vector，每次插入时找到位置所在的块，再暴力插入，将块内其他元素直接向后移动一位，这样来实现插入操作，时间复杂度不会太高。

但对于数据不是随机的情况，如果在一个块内有大量的单点插入，会使得这个块的大小超过 $\sqrt{n}$，这样对块内暴力的复杂度就没有保证了，此时需要引入一个新的操作：重构，即重新分块。

重构，是每 $\sqrt{n}$次插入后，重新将数列进行分块，由于重构所需的时间复杂度为 O(n)，重构次数为 $\sqrt{n}$，因此重构的时间复杂度没有问题，且保证了每个块的大小相对均衡。

数列分块9

题目描述

给出一个长为 n的数列，以及 n个操作，操作涉及询问区间的最小众数。

众数不满足可加性，这时候线段树无法直接求解，分块的优势就出来了

设序列a分为T段，则每块的长度L=N/T
对于每个询问[x,y]，设x处于第p块，y处于第q块
区间[x,y]包含了三部分：

1.开头不足一块的区间[x,ed[p]]
2.第p+1~q-1块构成的区间
3.结尾不足一块的区间[st[q],y]

数列分块9(在线求众数)

题目描述

给出一个长为 n的数列，以及 n个操作，操作涉及询问区间的最小众数。

1.开头不足一块的区间[x,ed[p]]
2.第p+1~q-1块构成的区间
3.结尾不足一块的区间[st[q],y]

对于每一个数a[i]都用vector存储一下他出现过的位置，每次用二分查询就是可以得到数字个数。由于a[i]的范围过大，且不涉及修改，所以需要离散化一下。

对于1、3种情况，由于是角块，我们可以直接枚举出现的每个数，复杂度为$O(2\sqrt{n})$;

对于第2种情况，直接暴力枚举显然不现实。

预处理出f[i][j]表示第i块到第j块的众数,开个桶遍历一遍。

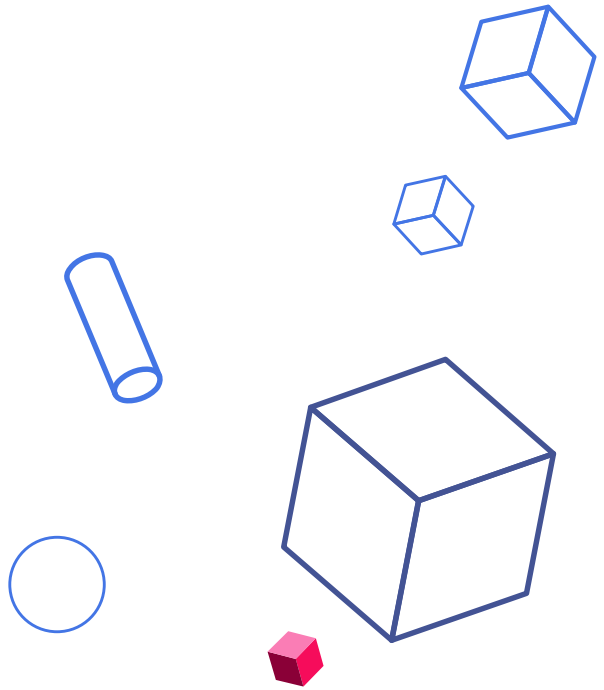
Thanks

For Your Watching

FZOI
西大附中信息学竞赛
High School Affilited to Southwest University

FZOI
西大附中信息学竞赛
High School Affiliated to Southwest University

西南大学附属中学
High School Affiliated to Southwest University



休息一下