分块和树状数组

上海交通大学 方泓杰

分块

PART 1

约定和前置技能

$a+b \geq 2\sqrt{ab}$;

$O(n\sqrt{n})$的算法在$n = 10^5$时是可以承受的（除非常数太大）；

若题目中没有特殊限制，则时间限制（TL）：1s；空间限制（ML）：256Mb；

以下内容中，K意义为一个块内有多少数。

简单题1

给你一个长度为n的序列，要求支持m个操作，操作有两种：

1. 修改单点的值；

2. 询问一个区间内数的和。

$1 \leq n, m \leq 50000$

别用树状数组/线段树！

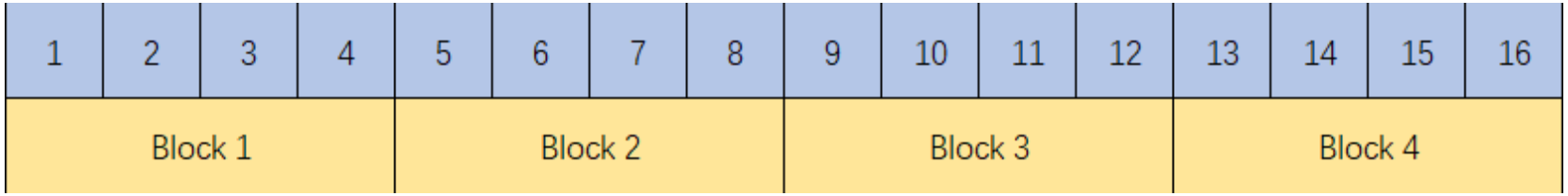
简单题1

我会暴力！

将序列划分成若干块，每块维护块内的和。

1. 考虑修改操作，单次修改，我们暴力修改原序列，并且更新块内和。

2. 考虑查询操作，例如，若$n = 16, K = 4$时：



简单题1

如果我要查询[7,13]的区间和，其涉及到三个块：块2，块3，块4。

其中，我们需要知道整个块3内的数的和，以及块2和块4一部分数的和。

于是，我们可以把涉及到的块分为两类：①完整的块；②不完整的块。

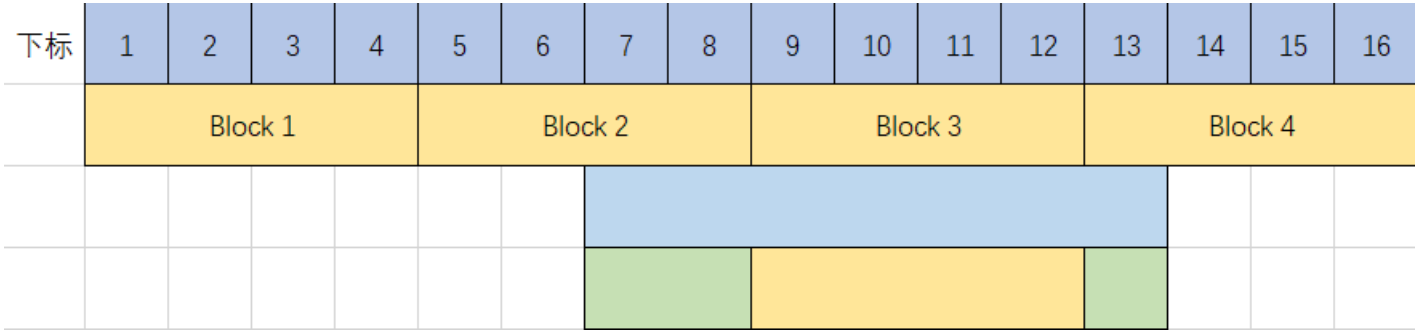
对于完整的块，我们直接调用已经维护的块内和即可。

对于不完整的块，我们暴力求出在查询区间内的数的和即可。

分析以下复杂度：

完整块：每块$O(1)$，最多$\frac{n}{K}$块。

不完整块：每块$O(K)$，最多2块。



简单题1

那么单次询问总共复杂度：$O\left(\frac{n}{K}+K\right)$，由基本不等式知道：

当$K=\sqrt{n}$时，$\frac{n}{K}+K$取最小值，此时复杂度$O(\sqrt{n})$；

所以总复杂度$O(m\sqrt{n})$，可以通过本题。

简单题2

给你一个长度为n的序列，要求支持m个操作，操作有两种：

1. 将一个区间内的数都加上x；

2. 询问单点的数的和。

$1 \leq n, m \leq 50000$

别用树状数组/线段树！

LOJ 6277 简单题2

我会暴力!

经过上题我们对分块有了初步的认识，接下来你想怎么做分块题？

LOJ 6277 简单题2

我会暴力!

经过上题我们对分块有了初步的认识，接下来你想怎么做分块题？

我们一般提出三个问题：

1. 维护什么数据？
2. 对于每种操作，不完整的块怎么处理？
3. 对于每种操作，完整的块怎么处理？

LOJ 6277 简单题2

1. 维护什么数据？

2. 对于每种操作，不完整的块怎么处理？

3. 对于每种操作，完整的块怎么处理？

LOJ 6277 简单题2

1. 维护什么数据？

每个点的值，以及一整块整体被加了多少。

2. 对于每种操作，不完整的块怎么处理？

对于修改，不完整的块暴力加x，复杂度$O(K)$；

对于查询，答案就是每个点的值加上这一整块整体被加了多少。

3. 对于每种操作，完整的块怎么处理？

对于修改，完整的块我们记录以下这一块整体被加了多少。

LOJ 6277 简单题2

时间复杂度分析和上题相同，询问$O(1)$，修改$O\left(\frac{n}{K}+K\right)$。

当$K=\sqrt{n}$时复杂度为$O(\sqrt{n})$，故总复杂度$O(m\sqrt{n})$。

实现方法？

LOJ 6277 简单题2

时间复杂度分析和上题相同，询问$O(1)$，修改$O\left(\frac{n}{K}+K\right)$。

当$K=\sqrt{n}$时复杂度为$O(\sqrt{n})$，故总复杂度$O(m\sqrt{n})$。

实现方法？

两种实现方法：

1. 块大小K固定（对于所有n都相同），取一个在$\sqrt{n_{\max}}$附近的值；
2. 块大小K不固定，对于每个n取$\sqrt{n}$。

个人习惯于第一种实现。

LOJ 6277 简单题2

如何调试？

令$K=1$，你的分块就是一个显然正确的大暴力了，复杂度$O(mn)$！

将$K=1$的程序和$K=\sqrt{n}$的程序对拍即可。

所以，写分块实际上也顺带写了暴力！

```
const int BLOCK_SIZE = 200, MAX_BLOCK = 305;
int tag[MAX_BLOCK];
int bel[M];

# define bg(x) ((x-1) * BLOCK_SIZE + 1)
# define ed(x) (x * BLOCK_SIZE)
```

LOJ 6277 简单题2

```
for (int i=1; i<=n; ++i) bel[i] = (i-1) / BLOCK_SIZE + 1;
```

```
for (int T=1; T<=n; ++T) {
    scanf("%d%d%d%d", &op, &l, &r, &c);
    if(op == 0) {
        int L = bel[l], R = bel[r];
        if(L == R) {
            for (int i=l; i<=r; ++i) a[i] += c;
        } else {
            for (int i=ed(L); i>=l; --i) a[i] += c;
            for (int i=bg(R); i<=r; ++i) a[i] += c;
            for (int i=L+1; i<=R-1; ++i) tag[i] += c;
        }
    } else printf("%d\n", a[r] + tag[bel[r]]);
}
```

简单题3

给你一个长度为n的序列，要求支持m个操作，操作有两种：

1. 将一个区间内的数都加上x；

2. 询问一个区间内的数的和。

$1 \leq n, m \leq 50000$

别用树状数组/线段树！

LOJ 6280 简单题3

1. 维护什么数据？

2. 对于每种操作，不完整的块怎么处理？

3. 对于每种操作，完整的块怎么处理？

LOJ 6280 简单题3

1. 维护什么数据？

每个点的值，以及一整块整体被加了多少，整块的和。

2. 对于每种操作，不完整的块怎么处理？

对于修改，不完整的块暴力加x，同时更新整块的和，复杂度$O(K)$；

对于查询，不完整的块我们暴力统计，复杂度$O(K)$。

3. 对于每种操作，完整的块怎么处理？

对于修改，完整的块我们记录以下这一块整体被加了多少，复杂度$O\left(\frac{n}{K}\right)$。

对于查询，完整的块我们可以通过调用整块的和来得到答案，复杂度$O\left(\frac{n}{K}\right)$。

因此，总时间复杂度为$O(m\sqrt{n})(K=\sqrt{n})$

简单题4

给你一个长度为n的序列，要求支持m个操作，操作有两种：

1. 将一个区间内的数都加上x；

2. 询问一个区间内有多少个数小于x。

$1 \leq n, m \leq 50000$

LOJ 6278 简单题4

1. 维护什么数据？

2. 对于每种操作，不完整的块怎么处理？

3. 对于每种操作，完整的块怎么处理？

LOJ 6278 简单题4

1. 维护什么数据？

每个数的值，每个块内整体加了多少，每个块内经过排序后的数。

2. 对于每种操作，不完整的块怎么处理？

修改：暴力修改在区间内的若干数，然后重新将块内排序，复杂度$O(K\log K)$。

询问：暴力统计在区间内的数有多少小于x，复杂度$O(K)$。

3. 对于每种操作，完整的块怎么处理？

修改：我们记录以下这一块整体被加了多少，复杂度$O\left(\frac{n}{K}\right)$。

查询：在每一块内的排序后的数中二分即可得到这一块的答案，复杂度$O\left(\frac{n}{K}\log K\right)$。

LOJ 6278 简单题4

单次平均时间复杂度$O\left(K\log K+\frac{n}{K}\right)$，当$K=\sqrt{\frac{n}{\log n}}$时最小，为$O\left(\sqrt{n\log n}\right)$；

LOJ 6278 简单题4

单次平均时间复杂度$O\left(K\log K+\frac{n}{K}\right)$，当$K=\sqrt{\frac{n}{\log n}}$时最小，为$O\left(\sqrt{n\log n}\right)$；

调参：当n较大的时候，虽然理论上当$K=\sqrt{\frac{n}{\log n}}$时最优，但是实际操作中，由于各部分常数不同，经常需要自己通过调试来找到一个相对最优的K。

本题中，经过对于大数据的尝试，我们发现当$K=56$时最优。

LOJ 6278 简单题4

单次平均时间复杂度$O\left(K\log K+\frac{n}{K}\right)$，当$K=\sqrt{\frac{n}{\log n}}$时最小，为$O\left(\sqrt{n\log n}\right)$；

调参：当n较大的时候，虽然理论上当$K=\sqrt{\frac{n}{\log n}}$时最优，但是实际操作中，由于各部分常数不同，经常需要自己通过调试来找到一个相对最优的K。

本题中，经过对于大数据的尝试，我们发现当$K=56$时最优。

对于块内数排序和重构我们可以用vector实现。

```
inline void build(int b) {
    v[b].clear();
    for (int i=bg(b), ito=ed(b); i<=ito; ++i) v[b].push_back(a[i]);
    sort(v[b].begin(), v[b].end());
}
```

LOJ 6278 简单题4

一些代码实现。

右边的为修改操作，下面的为询问操作。

```
int L = bl[l], R = bl[r];
if(L == R) {
    for (int i=l; i<=r; ++i) a[i] += c;
    build(L);
} else {
    for (int i=ed(L); i>=l; --i) a[i] += c;
    build(L);
    for (int i=bg(R); i<=r; ++i) a[i] += c;
    build(R);
    for (int i=L+1; i<R; ++i) tg[i] += c;
}
```

```
int L = bl[l], R = bl[r], sum = 0; c = c * c;
if(L == R) {
    for (int i=l; i<=r; ++i) if(a[i] + tg[L] < c) ++sum;
} else {
    for (int i=ed(L); i>=l; --i) if(a[i] + tg[L] < c) ++sum;
    for (int i=bg(R); i<=r; ++i) if(a[i] + tg[R] < c) ++sum;
    for (int i=L+1; i<R; ++i)
        sum += lower_bound(v[i].begin(), v[i].end(), c - tg[i]) - v[i].begin();
}
printf("%d\n", sum);
```

简单题5

给你一个长度为n的序列，要求支持m个操作，操作有两种：

1. 将一个区间内的数都加上x；

2. 询问一个区间内比x小的最大的数。

$1 \leq n, m \leq 50000$

LOJ 6279 简单题5

1. 维护什么数据？

2. 对于每种操作，不完整的块怎么处理？

3. 对于每种操作，完整的块怎么处理？

LOJ 6279 简单题5

上面就留给大家自己填空了！

事实上，本题的做法和“简单题4”几乎一模一样。

更为一般的， 如果比较懒且开了O2的话，我们可以用STL中的set实现！

简单题6

给你一个长度为n的非负整数序列，要求支持m个操作，操作有两种：

1. 将一个区间内的每个数x都变为$[\sqrt{x}]$（开方后下取整）；

2. 询问一个区间内的数的和。

$1 \leq n, m \leq 50000, 1 \leq a_i \leq 10^9$

LOJ 6281 简单题6

1. 维护什么数据？

2. 对于每种操作，不完整的块怎么处理？

3. 对于每种操作，完整的块怎么处理？

LOJ 6281 简单题6

似乎无从下手？那我们就需要观察题目的一些具体性质了！

可以发现，一个正整数被开方下取整最多5次过后就变成了0/1！

那么我们每个块仍然维护区间和，多记录一个标记表示该块内的数是否全是0/1。

这样每个块最多被整体操作5次，每个数最多被操作10次（块操作+暴力操作）

复杂度就可以接受了。

```
int L = bl[l], R = bl[r];
if(L == R) {
  if(! o[L]) {
    for (int i=l; i<=r; ++i) a[i] = sqrt(a[i]);
    chk(L);
  }
} else {
  if(! o[L]) {
    for (int i=ed(L); i>=l; --i) a[i] = sqrt(a[i]);
    chk(L);
  }
  if(! o[R]) {
    for (int i=bg(R); i<=r; ++i) a[i] = sqrt(a[i]);
    chk(R);
  }
  for (int i=L+1; i<R; ++i) {
    if(o[i]) continue;
    for (int j=bg(i), jto=ed(i); j<=jto; ++j) a[j] = sqrt(a[j]);
    chk(i);
  }
}
```

简单题7

给你一个长度为n的序列，要求支持m个操作，操作有两种：

1. 单点插入一个数；

2. 询问单点的数。

$1 \leq n, m \leq 50000$

LOJ 6282 简单题7

1. 维护什么数据？

2. 对于每种操作，不完整的块怎么处理？

3. 对于每种操作，完整的块怎么处理？

LOJ 6282 简单题7

1. 维护什么数据？

一个vector维护块内的所有数。

2. 对于每种操作，不完整的块怎么处理？

查询：暴力查询；

插入：暴力插入。

3. 对于每种操作，完整的块怎么处理？

没有完整块的操作，不用处理！

感觉有一丝丝不对？复杂度？

LOJ 6282 简单题7

如果把所有数都插入到一个块中，那么……

LOJ 6282 简单题7

如果把所有数都插入到一个块中，那么……

所以我们需要引入新的操作：块重构！

LOJ 6282 简单题7

有两种块重构的方法：

1. 当一个块大小达到$2K$时，将其分成两个块；
2. 每$\sqrt{n}$个操作后，对整个序列重新进行分块操作。

上述两种方法复杂度相同，均为$O(m\sqrt{n})$，第二种较好实现。下面给出第一种的部分代码

```
inline void rebuild(int x) {
  int y = nxt[x]; ++B;
  nxt[x] = B, nxt[B] = y;
  for (int i=a[x].size()/2, ito=a[x].size(); i<ito; ++i) a[B].push_back(a[x][i]);
  for (int i=a[x].size()/2, ito=a[x].size(); i<ito; ++i) a[x].pop_back();
}
```

```
if(op == 0) {
  --l;
  for (int i=fir; i; i=nxt[i]) {
    if(l <= a[i].size()) {
      a[i].insert(a[i].begin() + l, r);
      if(a[i].size() >= (BLOCK_SIZE * 2)) rebuild(i);
      break;
    }
    l -= a[i].size();
  }
} else {
  l = r;
  for (int i=fir; i; i=nxt[i]) {
    if(l <= a[i].size()) {
      printf("%d\n", a[i][l-1]);
      break;
    }
    l -= a[i].size();
  }
}
```

简单题3·改

给你一个长度为n的序列，要求支持m个操作，操作有三种：

1. 将一个区间内的数都加上x；

2. 将一个区间内的数都乘以x；

2. 询问一个区间内的数的和。

$1 \leq n, m \leq 50000$

别用树状数组/线段树！

LOJ 6283 简单题3·改

整体思路和“简单题3”没有区别。

考虑两个标记共存的问题。

乘一个数y的时候，标记如何变化？

加一个数x的时候，标记如何变化？

乘法和加法优先级谁高？

考虑清楚这些问题，这道题就没有问题了。

LOJ 6283 简单题3·改

假设你有一个乘法标记q，加法标记p，块内存的某个数为x，

那么实际上这个数时$qx+p$。

乘一个数c的时候，相当于$c(qx+p)=(cq)x+(cp)$，因此p,q均要乘c；

加一个数d的时候，相当于$qx+p+d=qx+(p+d)$，只要p加d即可！

简单题8

给你一个长度为n的序列，要求支持m个操作，操作只有一种：

询问一个区间内等于c的元素个数，并将这个区间所有元素都改为c。

$1 \leq n, m \leq 50000$

LOJ 6284 简单题8

容易发现，进行过一些操作过后，整个区间一定是一段一段连续的数。

然后我们考虑分块。块内维护权值是否相同。

对于完整块，如果权值全相同则$O(1)$查询&修改；否则暴力查询&修改即可。

对于不完整的块，我们直接暴力修改即可。

这个复杂度是对的吗？

LOJ 6284 简单题8

暴力出奇迹！

这样看似最差情况每次都会耗费$O(n)$的时间，但其实可以这样分析：

假设初始序列都是同一个值，那么查询是$O(\sqrt{n})$。

如果这时进行一个区间操作，它最多破坏首尾2个块的标记。

要想让一个操作耗费$O(n)$的时间，要先花费$O(\sqrt{n})$个操作对数列进行修改。

经过这么一波分析，复杂度就正确啦！为$O(m\sqrt{n})$。

简单题9

给定一个长度为n的序列，求有多少个区间$[l,r]$满足：

区间内每个数都出现奇数次。

$1 \leq n \leq 2 \times 10^5, 1 \leq a_i \leq 10^6$

TL：3s

简单题9

给每一个数x赋一个随机权值H_x，那么转化为：

求有多少个区间，使得**区间内的所有数权值异或和等于区间内出现过的数权值异或和**。

简单题9

给每一个数x赋一个随机权值H_x，那么转化为：

求有多少个区间，使得区间内的所有数权值异或和等于区间内出现过的数权值异或和。

记pre_i表示i这个数上一次出现的位置。

那么问题转化为：

枚举区间右端点r，每次两个操作：

1. 给区间$[1, pre_{a_r}]$异或一个数；
2. 询问$[1, r]$中为0的数的个数。

简单题9

区间异或我们可以打标记，区间询问即为统计值为标记的数的个数！

我们可以用STL的map来实现！

遇到不完整的块的时候，直接暴力处理过后把块的信息重新统计即可！

这样时间复杂度$O\left(n\sqrt{n\log n}\right)$（由于map有$O(\log n)$的复杂度）

但是过不去……

简单题9

实际上，我们把map改成hash表就能通过啦！

这样复杂度是$O(n\sqrt{n})$。

参考代码：https://paste.ubuntu.com/p/W4XtXvTgV8/

蒲公英

给出一个长度为n的序列，m次询问，求区间的众数（有多个输出编号小的那个）。

强制在线。

$1 \leq n \leq 40000, 1 \leq a_i \leq 50000$

TL：2s

BZOJ 2724 蒲公英

分块进阶套路。

考虑分块后，令$ans_{i,j}$表示第i块到第j块的答案，这部分是可以在$O(n\sqrt{n})$内预处理的。

（枚举左端点是第几块，每次一块一块加进来统计答案）

然后对于每一个值开一个vector，内部存的是这个值所有的出现位置的下标。

可以证明，答案只会在整块的答案 和 至多两个不完整块中的 所有数中选取。

于是对于这些数，我们在他们对应的vector中二分可以找出他们在询问区间内出现次数。

从而得到答案。

BZOJ 2724 蒲公英

可能称为答案的数共有$O(K)$个，二分的复杂度为$O(\log K)$，总共复杂度$O(K \log K)$。

根据复杂度分析，可以知道最优复杂度是$O\left(n\sqrt{n \log n}\right)$。

记录下第i的块到第j个块的答案也是一种常见分块套路。

作诗

有n个数，m组询问，每次问$[l,r]$中有多少个数出现正偶数次。

强制在线。

数据范围$1 \leq n, m, a_i \leq 10^5$

TL：2.5s

BZOJ 2821 作诗

套用前一题的方法，令$f_{i,j}$表示第i块到第j块的答案。

令$g_{i,j}$表示j这个数在$[1,i]$块中出现了几次。

那么这两个数组都可以预处理出来。

BZOJ 2821 作诗

套用前一题的方法，令$f_{i,j}$表示第i块到第j块的答案。

令$g_{i,j}$表示j这个数在$[1,i]$块中出现了几次。

那么这两个数组都可以预处理出来。

碰到整块的直接调用答案。

碰到不完整块的可以通过g数组调用出在整块中，这个数出现了几次。

然后把两边不完整块出现次数加上去统计即可。

时间复杂度$O(K)$。注意不要写成多一个log！

Anton and Permutation

给出一个初始满足$a_i = i$的n个数的排列。有q次操作，每次把位置x和位置y的数交换。

也就是$swap(a_x, a_y)$。问每次操作后序列的逆序对个数。

$1 \leq n \leq 2 \times 10^5, 1 \leq q \leq 50000$

CodeForces 785E Anton and Permutation

当交换(a_x, a_y)时，讨论区间(x, y)，有：

$ans = ans +$ 区间内比a_y小的数的个数 $-$ 区间内比a_x小的数的个数 $+$ 区间内比a_x大的数的个数 $-$ 区间内比a_y大的数的个数 $- [a_x > a_y] + [a_x < a_y]$

转化为求一个区间内比给定值小/大的元素个数。

对于每个块维护一个vector，参照之前的方法即可！

修改的话只有单点修改，所以直接暴力重新构造块！

时间复杂度$O\left(n\sqrt{n \log n}\right)$

（标解应该是BIT套平衡树）

Serega and Fun

给出一个长度为n的序列$\{a_n\}$，q次操作，操作有两种：

1. 将区间$[l,r]$循环移位（即$a_l \to a_{l+1}, a_{l+1} \to a_{l+2}, \dots, a_r \to a_l$）

2. 询问区间$[l,r]$内等于k的数有多少个。

强制在线。

$1 \le n, q \le 10^5, 1 \le a_i \le n$

CodeForces 455D Serega and Fun

这个循环移位听起来好像是splay的操作？

不用！我们可以分块！

每一个整块内我们维护一个双端队列。

考虑一个循环移位操作，整块的显然只要双端队列一边进一边出即可。

不完整块我们暴力移位。

怎么统计答案？

CodeForces 455D Serega and Fun

一开始我们令$f(i,j)$表示i这个数在前j块出现的次数，那么可以预处理出$f(i,j)$。

因为每次整块只更改两个数，所以也是可以在处理整块的时候维护$f(i,j)$的！

有了$f(i,j)$，参考“作诗”一题的方法，我们知道了中间整块部分的答案。

对于两边不完整块的答案，我们暴力统计即可！

时间复杂度$O(q\sqrt{n})$。

比splay好写多了。

GukiZ and GukiZiana

给一个正整数序列$\{a_n\}$，进行q次操作，有两种操作：

1. 区间$[l,r]$增加x；

2. 找到数组内值等于y的两个数的下标最大距离。

$1 \le n \le 5 \times 10^5, 1 \le q \le 50000$

CodeForces 551E GukiZ and GukiZiana

每块维护排序后的序列。查询二分查找即可。

时间复杂度$O\left(q\sqrt{n\log n}\right)$。

Ant colony

给定一个长度为n的序列，q个询问。

每次给定区间$[l,r]$，将$[l,r]$中所有数两两比较。

如果$a,b \in [l,r]$，$a|b$则a得一分，$b|a$则b得一分。

问有多少个数没有得到满分。

$1 \leq n,q \leq 10^5, 1 \leq a_i \leq 10^9$

CodeForces 474F Ant colony

答案为区间长度减去这个区间内值为gcd的数的个数。

先预处理出每一块的gcd，然后询问时查询每个块gcd后和不完整块合并即可得到总的gcd。

那么接下来就是查询区间内有多少个数等于某个确定的数了。

可以将所有数离散，然后令$f(i,j)$表示离散后为i的数在$1\sim j$块出现了几次。

和“作诗”一题套路相同，可以在$O(q\sqrt{n})$的时间内解决问题。

DZY Loves Colors

给定两个长度为n的序列$\{a_i\},\{b_i\}$，初始$a_i = i, b_i = 0$。

m次操作，操作有两种：

1. 将$a_l, a_{l+1}, \ldots, a_r$变为$x$，同时$b_i = b_i + |x - a_i| (l \le i \le r)$

2. 统计b的区间和。

$1 \le n, m \le 10^5, 1 \le x \le 10^8$

CodeForces 444C DZY Loves Colors

根据“简单题8”的分析，我们发现这题和“简单题8”有着相同的性质。

因此我们维护一个标记表示区间内的数是否全相同即可。

如果全相同很容易算出a,b数组需要更新多少，打标记即可。

否则暴力修改。

复杂度分析和“简单题8”相同，为$O(m\sqrt{n})$。

Holes

一条直线上有n个洞。

每当一个球进入一个洞时，会被弹到后面距它l_i的洞中或直接被弹飞，要求支持两种操作：

1. 修改某个洞的弹力l_i；

2.询问小球放入某个洞后，被弹几次之后会飞出，飞出之前进入的最后一个洞是什么。

共有m次操作。

$1 \leq n, m \leq 10^5$

CodeForces 13E Holes

分块。对于每一块维护其中每个洞弹出这个块需要几次，并且会被弹到哪一个洞。

询问的时候以块为单位暴力找即可。

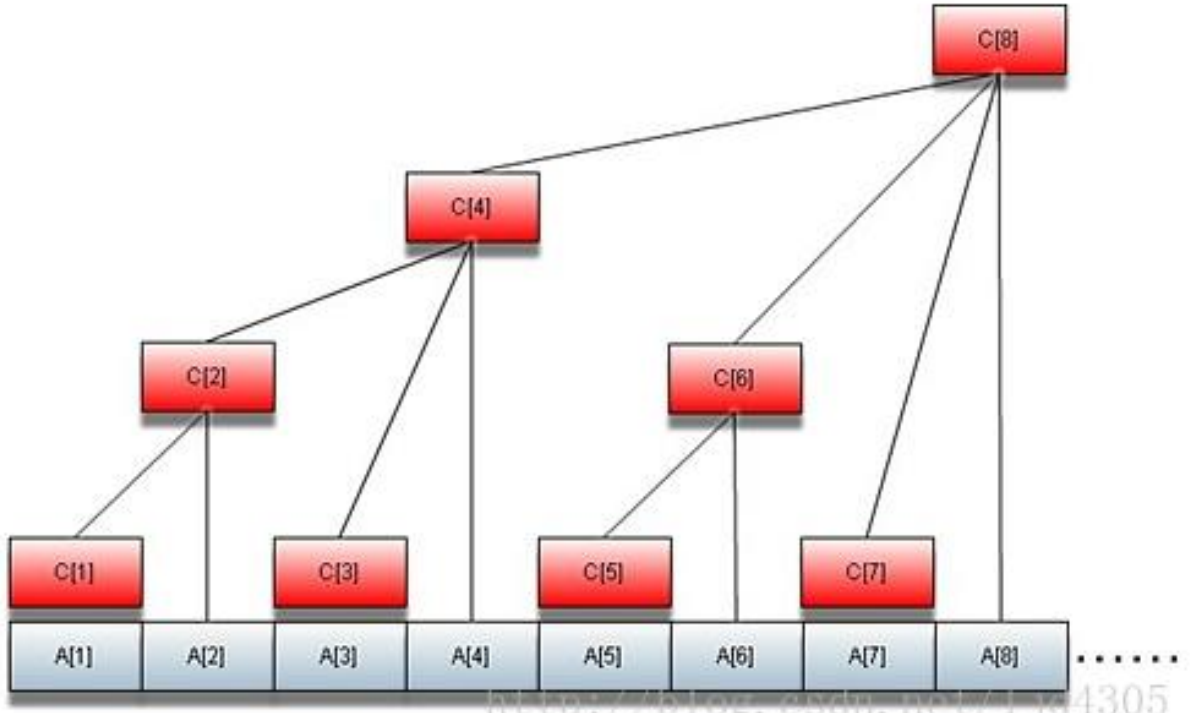
修改的时候，暴力修改这个块内所有洞的数据。

时间复杂度$O(m\sqrt{n})$。

树状数组

PART 2

树状数组（BIT）



树状数组

$C[1]=A[1];C[2]=A[1]+A[2];$

$C[3]=A[3];C[4]=A[1]+A[2]+A[3]+A[4];$

$C[5]=A[5];C[6]=A[5]+A[6];$

$C[7]=A[7];C[8]=A[1]+A[2]+A[3]+A[4]+A[5]+A[6]+A[7]+A[8];$

e.g. 关于$A[3]$出现了几次？$C[3],C[4],C[8]$。

关于$A[1]$出现了几次？$C[1],C[2],C[4],C[8]$

什么规律？我们写成二进制：关于$A[3]:0011\rightarrow 0100\rightarrow 1000$

关于$A[1]:0001\rightarrow 0010\rightarrow 0100\rightarrow 1000.$

$a=a+lowbit(a)$；lowbit(x)表示x最后一个1单独提出来的数。

树状数组

据此归纳我们可以得到树状数组的单点修改的代码：

我们要对x单点修改，那么就从x开始，每次加lowbit(x)，经过路上的这些节点都需要修改。

可以发现，每次加lowbit(x)，时间复杂度是$O(\log n)$级别的。

考虑我们要查询一个区间：

我们把区间转化为前缀和相减，那么就相当于查询[1,x]的和。

观察：$[1,5] = C[4] + C[5], [1,3] = C[2] + C[3], [1,7] = C[4] + C[6] + C[7]$

转化为二进制：$[1,5]: 0101 \to 0100,\ [1,3]: 0011 \to 0010,\ [1,7]: 0111 \to 0110 \to 0100$

发现是每次减去lowbit(x)，于是一样操作即可，复杂度$O(\log n)$.

区修区查树状数组

以目前我们的树状数组知识，可以实现：

1. 单点修改；

2. 区间求和。

如果我们需要实现：

1. 区间修改；

2. 求区间$[l,r]$的和。

怎么利用树状数组实现呢？

区修区查树状数组

设b_i表示$[i,n]$整体加了多少。

那么$[x,y]$区间加d就相当于b_x加d，b_{y+1}减d。

查询$[x,y]$区间呢？我们同样转化成前缀和，即查询$[1,x]$。

$$sum = \sum_{i=1}^{x}\sum_{j=1}^{i} b_j = \sum_{i=1}^{x} b_i(x-i+1) = \sum_{i=1}^{x} b_i(x+1) - \sum_{i=1}^{x} b_i \times i$$

我们如果维护出了$b_i \times i$，就能求出答案了！

那我们就多维护一下$b_i \times i$就行啦！

用两棵BIT实现即可。

二维树状数组

在一个二维空间，需要维护：

1. 单点加；

2. 矩形求和。

矩形求和$[a,b] \to [c,d]$的矩形可以前缀和化为四个以[1,1]为左上点矩形的加减。

实现方法和普通BIT完全相同，只是换成两重循环即可。

时间复杂度$O(\log^2 n)$。

区修区查二维树状数组

在一个二维空间，需要维护：

1. 矩形加；

2. 矩形求和。

同样的方法，我们记录$b_{i,j}$为$[i,j] \to [n,n]$的整体增量。

矩形加可以化成四个矩形的加减。

考虑矩形求和。

区修区查二维树状数组

同样转化成左上角为$(1,1)$的矩形求和，设右下角为(x,y)，那么：

$$sum=\sum_{i=1}^{x}\sum_{j=1}^{y}\sum_{u=1}^{i}\sum_{v=1}^{j}b_{i,j}=\sum_{i=1}^{x}\sum_{j=1}^{y}b_{i,j}(x+1-i)(y+1-j)$$

$$sum=\sum_{i=1}^{x}\sum_{j=1}^{y}b_{i,j}(x+1)(y+1)-\left(b_{i,j}\times i\right)(y+1)-\left(b_{i,j}\times j\right)(x+1)+b_{i,j}\times i\times j$$

我们只要维护$b_{i,j}\times i, b_{i,j}\times j, b_{i,j}\times i\times j$即可。

时间复杂度为$O(\log^2 n)$。

阿里巴巴

宝藏在了一块可分成 $n \times n$ 网格的空地下，网格按$(1,1),(1,2)\ldots(1,n),(2,1)\ldots(2,n)\ldots(n,n)$的方式编号。

宝藏物品总共有 50 种。大盗们特地开发了一套程序来管理这些宝藏。初始时地里没有物品，他们总共会操作m次，每一次都会选择这个大网格图的一个子矩形，操作分两种：

对于埋物品操作，会给出一个物品序列，如“5 件 A、7 件 C”，程序将会自动在这个子矩形的每一个方格内新添上述物品序列给出的物品（比如在(2, 2)里放 5 个 A、7 个 C，再在(2, 3)里放 5 个 A、7 个 C……）；

对于每个查询操作，需要打印一行 50 个整数，依次分别表示给定子矩形内每种物品数量的奇偶性。

现在，阿里巴巴的传人想要破解这套系统，首先就需要模拟以上程序的执行，你能帮助他吗？

$1 \le n \le 10^3, 1 \le m \le 10^5$

阿里巴巴

把50种宝物二进制存储，那么每次埋宝藏相当于区间异或，用区修区查二维BIT即可。

异或的话其实只要把所有的+/-改成异或即可。

时间复杂度$O(m\log^2 n)$。

BIT vs 线段树

优势：

1. 好写，写的快；
2. 容易查错，不容易出错；
3. 常数小。

劣势：

支持的操作不如线段树多。

习题

由于树状数组大部分的题目和线段树都有所重合，所以剩下一些例题我们明天统一讲。

今天的习题主要是分块的内容。

https://vjudge.net/contest/283316

密码：hailiangC0214

完成后的同学可以做一下 bzoj 4241 历史研究

二维树状数组可以自行练习，题目：luogu4514 上帝造题的七分钟

前面的题号中，LOJ 6277-6285分别对应HLUOJ676-684，大家可以在HLUOJ中提交！