蛟龙五班

二分查找、二分快速幂

Mas

二分查找

二分查找($binary\ search$),也称折半搜索($half - interval\ search$)、对数搜索($logarithmic\ search$)

是用来在一个**有序**数组中查找某一元素的算法

以在一个升序数组中查找一个数为例

每次考察数组当前部分的中间元素

- 如果中间元素刚好是要找的,就结束搜索过程
- 如果中间元素小于所查找的值,那么左侧的只会更小,不会有所查找的元素,只需到右侧查找
- 如果中间元素大于所查找的值同理,只需到左侧查找

二分查找

在有序数组中查找8

1	3	8	20	90	101	102

中间点$a[4]$为20

1	3	8	20	90	101	102

$20 > 8$,舍弃$a[4 \sim 7]$

新区间中间点$a[2]$为3

1	3	8	20	90	101	102

$3 < 8$,舍弃$a[1 \sim 2]$

新区间中间点$a[3]$为8

查找成功

在有序数组中查找91

1	3	8	20	90	101	102

中间点$a[4]$为20

1	3	8	20	90	101	102

$20 < 91$,舍弃$a[1 \sim 4]$

新区间中间点$a[6]$为101

1	3	8	20	90	101	102

$101 > 91$,舍弃$a[6 \sim 7]$

新区间中间点$a[5]$为90

1	3	8	20	90	101	102

$90 < 91$,舍弃$a[6 \sim 6]$

区间为空

查找失败

当查找失败时,l和r所停留的下标与被查找数有什么关系?

二分查找

```
int binarySearch(int l, int r, int x)
{
  while (l <= r) //区间不为空继续查找
  {
    int mid = (l + r) / 2;
    if (a[mid] == x)
      return mid; //已经找到
    if (a[mid] > x)
      r = mid - 1; //舍弃右半部分
    else
      l = mid + 1; //舍弃左半部分
  }
  return -1; //未找到
}
```

能否写成$l < r$

能否写成$r = mid$?

能否写成$l = mid$?

二分查找的最优时间复杂度为 $O(1)$

二分查找的平均时间复杂度和最坏时间复杂度均为$O(\log n)$

因为在二分搜索过程中,算法每次都把查询的区间减半,所以对于一个长度为 n的数组,至多会进行 $\log n$ 次查找

#147、双倍查找

题目描述

给定一个长度为 n 的数列 $A_1 \sim A_n$ ，问这个数列中有多少个数，它的两倍的数也在这个数列中？
即问有多少个 A_i， $2 \times A_i$ 也在数列中。

输入格式

第一行输入一个正整数 n 。
第二行输入 n 个非负整数 $A_1 \sim A_n$ 。

输出格式

输出一行，包含一个非负整数。

输入样例

```
6
8 2 4 0 5 4
```

输出样例

```
4
```

数据规模与约定

对于前 30% 的数据有 $n \le 100, 0 \le A_i \le 1000$ ；
对于前 60% 的数据有 $n \le 100, 0 \le A_i \le 100000000$ ；
对于所有数据有 $n \le 100000, 0 \le A_i \le 100000000$ 。

一层循环枚举A_i,另一层循环枚举$2 \times A_i$

时间复杂度$O(n^2)$

若使用 数组记录 每个数出现的次数

时间复杂度$O(n)$,空间复杂度$O(\max\{A_i\})$

将数组排序

一层循环枚举A_i,对于$2 \times A_i$直接使用二分查找

时间复杂度$O(n\log n)$

#1663、Mas的数组查找

题目描述

在一个有序的数组 A 中查找 x ,你可以使用二分查找快速的找到 x 的位置。
现在 Mas 拿到了一个奇怪的数组 S ,数组 S 的是由一个有序数组按未知的旋转轴旋转得到的
如

```
1 2 3 4 5 6        ==>        4 5 6 1 2 3
```

现在请你帮 Mas 在找到 x 在 S 的位置。

输入格式

第一行两个个正整数 n m
第二行 n 个整数 S_i ,保证 S_i 仅出现一次
接下来 m 行，每行表示一个询问 X

输出格式

输出 m 行
对于每个询问，输入 x 在 S 中的位置,如果不存在输出 -1

输入样例

```
6 1
4 5 6 1 2 3
2
```

输出样例

```
5
```

数据规模

对于 10% 的数据 $1 \le n \le 100$
对于 40% 的数据 $1 \le n \le 2 \times 10^5$
对于 100% 的数据 $1 \le n, m \le 6 \times 10^5, -10^6 \le S_i \le 10^6$

#1663、Mas的数组查找

思路1

不难发现以旋转点为分界,是两个有序的数组

分界点x满足$a[x] > a[x-1]$

将原数组下标记录并将两个数组合并成新的有序数组(时间复杂度$O(n)$)

再对新数组进行二分查找,输出原数组中的下标

总时间复杂度$O(Q\log n)$

思路2

找到分界点 x 后,直接把数组划成两部分,每次对每一部分分别使用二分查找

注意其中一部分会查找不到数据,就不会产生 = 的情况,注意二分边界

总时间复杂度仍旧是 $O(Q\log n)$

#1663、Mas的数组查找

20	90	101	102	1	3	8

101	102	1	3	8	20	90

思路3

对于每一次二分查找x,如果$a[mid] = x$ 说明找到结束查找

否则分情况讨论旋转点情况

若旋转点在mid右侧(含),有$a[l] \le a[mid]$

若此时 $a[l] \le x$且$a[mid] > x$,说明x位于$1 \sim mid - 1$ 之间

否则说明x位于 $mid + 1 \sim r$ 之间

否则旋转点在mid左侧

若此时 $x \le a[r]$ 且$a[mid] < x$,说明 x 位于$mid + 1 \sim r$之间

否则说明 x 位于$1 \sim mid - 1$之间

时间复杂度$O(Qlogn)$

```
int binarySearch(int l, int r, int x)
{
  while (l <= r)
  {
    int mid = (l + r) >> 1;
    if (x == a[mid])      return mid;
    if (a[l] <= a[mid])
    {
      if (a[l] <= x && a[mid] > x)
        r = mid - 1;
      else
        l = mid + 1;
    }
    else
    {
      if (a[r] >= x && a[mid] < x)
        l = mid + 1;
      else
        r = mid - 1;
    }
  }
  return -1;
}
```

#2049、二分查找1

题目描述

现在长度为 N（$1 \le N \le 10^5$）的有序递增数组。数组中的任意一个数记为 $A_i(-10^9 \le A_i \le 10^9)$。

给定一个数据 num，请在数值中找到这个数据的位置。如果数组中有该数据，输出数据对应的数组第一次出现的下标，如果数组中没有该数据，输出 `-1`。

注意数据 num 在数组中可能出现多次。

输入

第一行，两个整数，用空格隔开，分别为 N 和 M。

第二行到 $N+1$ 行，包含 N 个整数的数组，数据之间用空格隔开。

接下来 M 行,每行一个 num .

输出

M 行，对于每一个 num 需要输出 num 对应的数组第一次出现的下标，如果数组中没有该数据，输出 `-1`。

样例输入

```
22 1
-5 0 1 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 7 8 9
4
```

样例输出

```
4
```

#2049、二分查找1

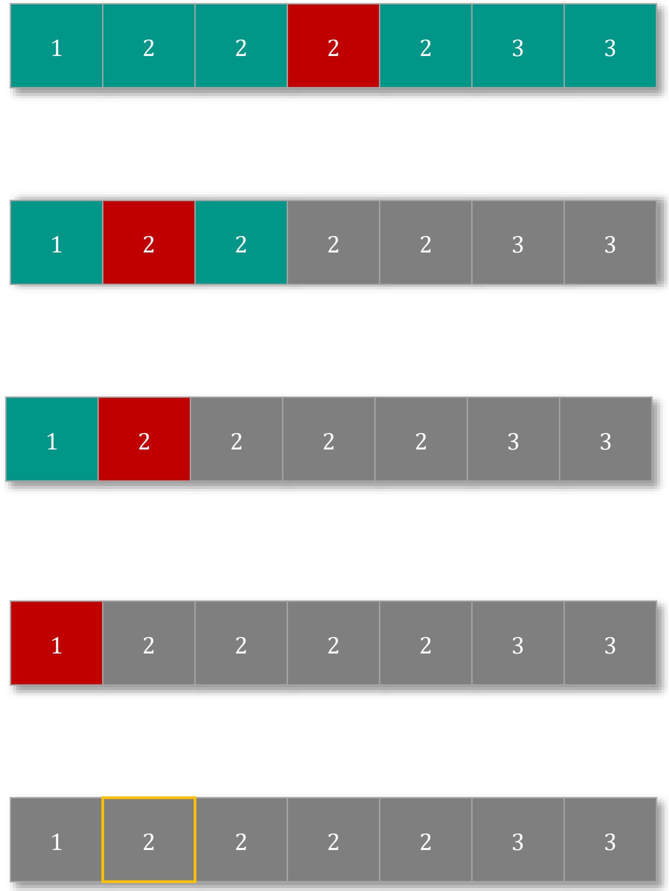
对于每一次二分查找x

如果 $a[mid] \geq x$说明答案不在mid右侧,令 $r = mid - 1$

否则说明答案不在mid左侧,令 $l = mid + 1$

最终区间为空,不难发现l为第一个不小于x的位置

判断$a[l]$是否为x即可



#2049、二分查找2

题目描述

现在长度为 N（$1 \le N \le 10^6$）的有序递增数组。数组中的任意一个数记为 $A_i(-10^9 \le A_i \le 10^9)$。

给定一个数据 num，请在数值中找到这个数据的位置。如果数组中有该数据，输出数据对应的数组最后一次出现的下标。如果数组中没有该数据，输出 `-1`。

注意数据 num 在数组中可能出现多次。

输入

第一行，两个整数，用空格隔开，分别为 N 和 num。
第二行到 $N+1$ 行，包含 N 个整数的数组，数据之间用空格隔开。

输出

一个整数，num 在数组中的位置。

样例输入

```
22 4
-5 0 1 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 7 8 9
```

样例输出

```
17
```

对于每一次二分查找x

如果 $a[mid] \ge x$说明答案不在mid右侧

令 $r = mid - 1$

否则说明答案不在mid左侧

令 $l = mid + 1$

最终区间为空

不难发现r为最后一个不小于x的位置

判断$a[r]$是否为x即可

#2054 查找出现的次数

结合前两题,计算差值即可

题目描述

现在长度为 N（$1 \le N \le 10^7$）的有序递增数组。数组中的任意一个数记为 $A_i(-10^9 \le A_i \le 10^9)$。

给定一个数据 num，请在统计 num 在数组中的出现次数。

注意数据 num 在数组中可能出现多次。

输入

第一行，两个整数，用空格隔开，分别为 N 和 num 。
第二行到 $N+1$ 行，包含 N 个整数的数组，数据之间用空格隔开。

输出

一个整数， num 在数组中出现的次数。

样例输入

```
22 4
-5 0 1 2 4 4 4 4 4 4 4 4 4 4 4 4 4 4 5 7 8 9
```

样例输出

```
14
```

STL中的二分查找相关函数

方法	功能
binary_search(first,last,val)	在区间$[first, last)$查找val返回$true/false$
lower_bound(first,last,val)	在区间$[first, last)$查找第一个不小于val返回元素地址/迭代器
upper_bound(first,last,val)	在区间$[first, last)$查找第一个大于val返回元素地址/迭代器
equal_range(first,last,val)	在区间$[first, last)$查找val出现的左右边界,返回一个pair

每个函数最后还可以加一个 cmp 参数,表示比较函数。
假设数组 $a[0]$~$a[n-1]$里存放了n个有序的 $pair$,我们可以用如下的方法查找:

```
bool cmp(int a, int b){
   return a.first < b.first || a.first == b.first && a.second < b.second;
}
int pos = lower_bound(a, a+n, val, cmp) – a;
```

快速幂

a 的 n 次方表示将 n 个 a 乘在一起,循环迭代实现,时间复杂度$O(n)$

$$a^{b+c} = a^b \cdot a^c$$

$$a^{2b} = a^b \cdot a^b = \left(a^b\right)^2$$

对于求解a^n,考虑求出$a^{\lfloor \frac{n}{2} \rfloor}$

- n为偶数答案为$a^{\lfloor \frac{n}{2} \rfloor} \cdot a^{\lfloor \frac{n}{2} \rfloor}$
- n为奇数答案为$a^{\lfloor \frac{n}{2} \rfloor} \cdot a^{\lfloor \frac{n}{2} \rfloor} \cdot a$

不断向下递归求解直到指数变为0,时间复杂度$O(\log n)$

```
long long qpow(long long a, long long b)
{
    if (b == 0)
        return 1;
    long long res = qpow(a, b / 2);
    if (b & 1)
        return res * res * a;
    return res * res;
}
```

也可将将取幂的任务按照指数的**二进制表示**分割成更小的任务,将n表示为2进制

$$3^{13} = 3^{(1101)_2} = 3^8 \cdot 3^4 \cdot 3^1$$

因为n有 $\lfloor \log_2 n \rfloor + 1$ 个二进制位,当已知 $a^1, a^2, a^4, a^8, \ldots, a^{\lfloor \log_2 n \rfloor}$后,只需计算$\log_2 n$次乘法就可以计算出$a^n$

快速幂

只需要快速计算上述3的2^k次幂的序列,不难发现序列中(除第一个)任意一个元素就是其前一个元素的平方

$$3^1 = 3$$

$$3^2 = (3^1)^2 = 9$$

$$3^4 = (3^2)^2 = 81$$

$$3^8 = (3^4)^2 = 6561$$

只需要将对应二进制位为1的整系数幂乘起来即可

$$3^{13} = 6561 \cdot 81 \cdot 3 = 1594323$$

将上述过程说得形式化一些$n = n_t 2^t + n_{t-1} 2^{t-1} + \cdots + n_1 2^1 + n_0 2^0$其中$n_i \in \{0,1\}$

$$a^n = a^{n_t 2^t + n_{t-1} 2^{t-1} + \cdots + n_1 2^1 + n_0 2^0} = a^{n_t 2^t} \times a^{n_{t-1} 2^{t-1}} \times \cdots a^{n_1 2^1} \times a^{n_0 2^0}$$

时间复杂度$O(\log n)$

```
long long qpow(long long a, long long b)
{
    long long res = 1, t = a;
    while (b)
    {
        if (b & 1)
            res = (res * t) % MOD;
        t = (t * t) % MOD;
        b >>= 1;
    }
    return res;
}
```

#779、A 的 B 次方

题目描述

给出三个整数 a, b, m ,求 $a^b \bmod m$ 的值

输入格式

一行三个整数 a, b, m

输出格式

一个整数,表示 $a^b \bmod m$ 的值

样例输入

```
2 100 1007
```

样例输出

```
169
```

数据范围与提示

对于全部数据, $1 \le a, b, m \le 10^9$

#2698、二进制串

题目描述

Mas 对二进制 01 序列很感兴趣,尤其是长度为 n 的二进制串

当 $n=3$,有 8 个长度为 3 的二进制串, `000、001、010、011、100、101、110、111`

输入格式

输入一个非负整数 n .

输出格式

输出长度为 n 的二进制串数量,答案可能很大输出对 99999991239959 取余的结果

输出样例1

```
3
```

输入样例1

```
8
```

数据规模

对于 60% 的数据 $1 \leq n < 31$
对于 70% 的数据 $1 \leq n < 1000$
对于 100% 的数据 $1 \leq n \leq 10^{12}$

答案显然为2^n

若直接快速幂求解,由于模数超过10^{12}

在一次乘法操作中,可能直接导致数据溢出

考虑将乘法操作改为加法操作,参考快速幂实现(龟速乘)

费马小定理

若 p 为素数$(a,p)=1$,则可以得到

$$a^{p-1} \equiv 1(\operatorname{mod} p)$$

简要证明

$p-1$个整数,$a,2a,3a,\ldots(p-1),a$ 中没有一个是 p 的倍数,而且没有任意两个模 p同余

所以这 $p-1$ 个数对模 p 的同余是 $1,2,3\ldots,(p-1)$ 的排列

可得

$$a \times 2a \times 3a \times \cdots \times (p-1)a \equiv 1 \times 2 \times 3 \times \cdots \times (p-1)(\operatorname{mod} p)$$

可化简为

$$a^{p-1} \times (p-1)! \equiv (p-1)!\,(\operatorname{mod} p)$$

即 $a^{p-1} \equiv 1(\operatorname{mod} p)$

给定两个整数 a 和 p,假设存在一个 x 使得 $ax \equiv 1(\operatorname{mod} p)$

那么称 x 为 a 关于 p 的乘法逆元记作 a^{-1},因为 $ax \equiv 1\ (mod\ p)$,根据费马小定理 $ax \equiv a^{p-1}\ (mod\ p)$

所以

$$x \equiv a^{p-2}\ (mod\ p)$$

#2742、单个数的逆元

题目描述

给定 n 组 a_i, p_i ，其中 p_i 是质数，求 a_i 模 p_i 的乘法逆元，若逆元不存在则输出 `impossible` 。

> *请求出在 $0 \sim p-1$ 之间的逆元。*

乘法逆元的定义

> *若整数 b, m 互质,并且对于任意的整数 a ,如果满足 $b \mid a$ ，则存在一个整数 x ,使得 $\frac{a}{b} \equiv ax \pmod m$ ，则称 x 为 b 的模 m 乘法逆元，记为 $b^{-1} \pmod m$ 。*
>
> *b 存在乘法逆元的充要条件是 b 与模数 m 互质。当模数 m 为质数时，b^{m-2} 即为 b 的乘法逆元。*

输入格式

第一行包含整数 n 。

接下来 n 行，每行包含一个数组 a_i, p_i ，数据保证 p_i 是质数。

输出格式

输出共 n 行，每组数据输出一个结果，每个结果占一行。

若 a_i 模 p_i 的乘法逆元存在，则输出一个整数，表示逆元，否则输出 `impossible` 。

实验舱

青少年编程

走近科学 走进名校

谢谢观看