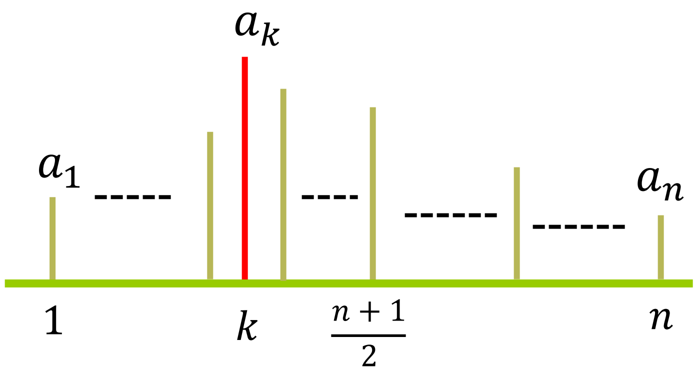
Binary search: Implementation

非单调序列上的二分搜索I： 山峰数组a$[1..n]$

$$0 < a[1] < \cdots < a[k-1] < a[k] > a[k+1] > \cdots > a[n] > 0$$

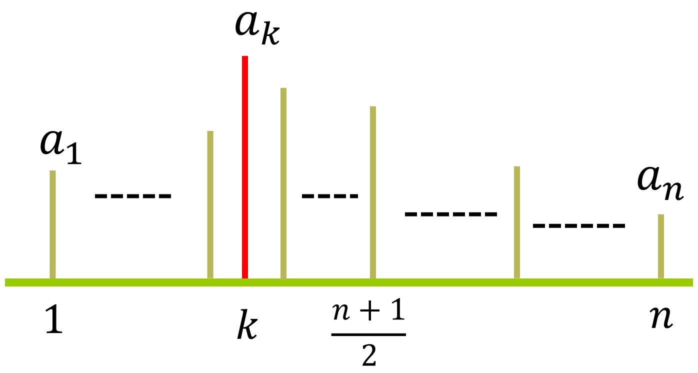


Q：查找最大值的位置k $(1 < k < n)$

Binary search: Implementation

山峰数组a[1..n]:

$$0 < a[1] < \cdots < a[k-1] < a[k] > a[k+1] > \cdots > a[n] > 0$$



- 查找最大值的位置k $(1 < k < n)$

（1）顺序查找，时间复杂度O(k)

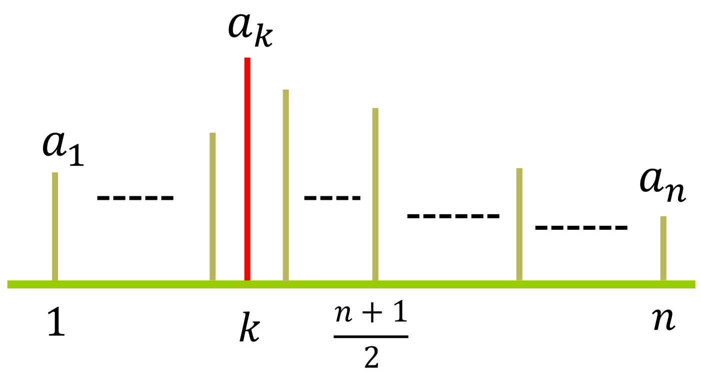
（2）二分法

问题点：
1. $a[k]$两边的数值大小关系不确定，如何二分？
2. 如何通过O(1)次比较实现搜索范围减半？

Binary search: Implementation

山峰数组a[1..n]:

$$0 < a[1] < \cdots < a[k-1] < a[k] > a[k+1] > \cdots > a[n] > 0$$



- 查找最大值的位置k ($1 < k < n$)

设$a[0] = a[n+1] = 0$

（2）二分法

问题点：
1. $a[k]$两边的数值大小关系不确定
2. 如何通过O(1)次比较实现搜索范围减半？

i. $\forall i \in [1, k)$: $a[i-1] < a[i] < a[i+1]$

ii. 最大值$a[k]$： $a[k-1] < a[k] > a[k+1]$

iii. $\forall j \in (k, n]$: $a[j-1] > a[j] > a[j+1]$

[二分原则] 如果连续三值升序（降序）排列，则最大值一定在右边（左边）

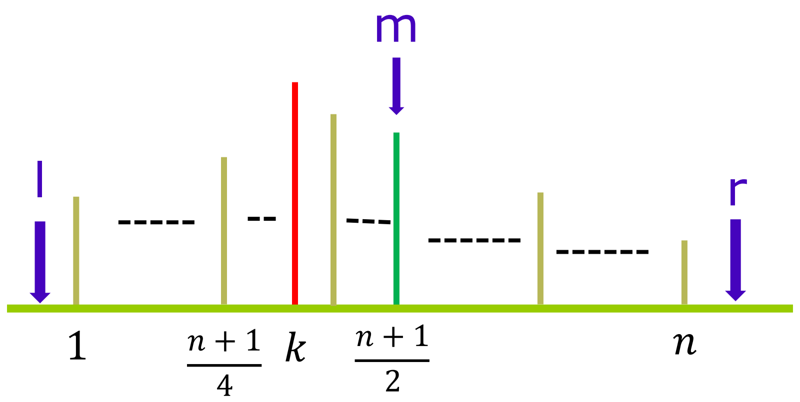
Binary search: Implementation

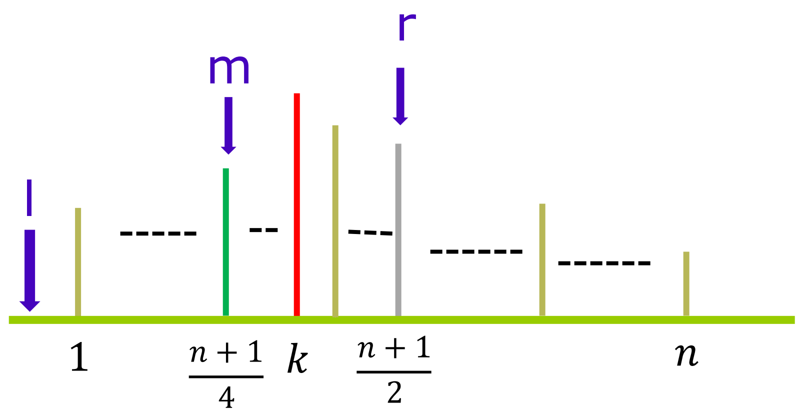
山峰数组a[1..n]:

$$0 < a[1] < \cdots < a[k-1] < a[k] > a[k+1] > \cdots > a[n] > 0$$

```
int binary(E A[], int n)  {  //山峰数组A
    A[0] = A[n+1] = 0;   //简化
    int l = 0;              //始终在最大值左边
    int r = n+1;            //始终在最大值右边

    while(l+1 < r)  {
        int m = (l+r) / 2;     //中间位置

        if (A[m]>A[m-1] && A[m]>A[m+1])
            return m;            //m是最大值位置

        if(A[m] < A[m+1])     //最大值在右边
            l = m;             //查找(m, r)
        else                    //最大值在左边
            r = m;           //查找(l, m)
    }
}
```

- 查找最大值的位置k $(1 < k < n)$





Binary search: Implementation

非单调序列上的二分搜索II：找极大值

(注：极大值 $\neq$ 最大值)

$a_1, a_2, \ldots, a_n$是长度为n的正数序列且值互异。假设$a_0 = a_{n+1} = 0$。a_k $(k \in [1, n])$是该序列的极大值当且仅当 $a_k > \max(a_{k+1}, a_{k-1})$。 找出序列中任何1个极大值的位置，时间复杂度$O(\log(n))$。

例： 0，8, 3, 15, 9, 7, 14, 12，0

极大值：8, 15, 14

命题1：如果$\exists k \in [1, n]$：$a_k < a_{k+1}$，则存在$k < j \leq n$且a_j是极大值。

（证明） 如果$a_{k+1} > a_{k+2}$， a_{k+1}就是极大值$(j = k+1)$;
否则$a_{k+1} < a_{k+2}$，判断a_{k+2}是否极大值，如果不是继续找a_{k+2}后面的极大值。
显然，该过程最多持续到a_n，即$a_k < a_{k+1} < \cdots < a_n > 0$，则$a_n$就是要找的极大值。

Binary search: Implementation

非单调序列上的二分搜索（找极大值）

$a_1, a_2, \ldots, a_n$是长度为n的正数序列且值互异。假设$a_0 = a_{n+1} = 0$。
a_k $(k \in [1, n])$是该序列的极大值当且仅当 $a_k > \max(a_{k+1}, a_{k-1})$。
找出序列中任意1个极大值的位置，时间复杂度$O(\log(n))$。

命题1：如果$\exists k \in [1, n)$：$a_{k+1} > a_k$，则存在$k < j \leq n$且a_j是极大值。

命题2：如果$\exists k \in (1, n]$：$a_{k-1} > a_k$，则存在$1 \leq j < k$且a_j是极大值。

[二分原则] 如果一个数的右（左）邻比自己大，
只需在右（左）边找极大值。

Binary search: Implementation

找极大值:

- $a_1, a_2, \ldots, a_n$是长度为n的正数序列且值互异。假设$a_0 = a_{n+1} = 0$。
- a_k $(k \in [1, n])$是该序列的极大值当且仅当 $a_k > \max(a_{k+1}, a_{k-1})$。
- 找出序列中任意1个极大值的位置，时间复杂度$O(\log(n))$。

```
int findMaximum (int A[], int n)  {
                          //正数数组A[1..n]
    A[0] = A[n+1] = 0;   //简化
    int l = 0;           //位置右边有极大值
    int r = n+1;         //位置左边有极大值

    while(l+1 < r)  {
        int m = (l+r) / 2;    //中间位置

        if (A[m]>A[m-1] && A[m]>A[m+1])
            return m;            //m是极大值位置

        if(A[m] < A[m+1])   //右边有极大值
            l = m;          //在(m, r)中找
        else                //左边有极大值
            r = m;          //在(l, m)中找

    }
}
```

Binary search: Implementation

非单调序列上的二分搜索（找极大值）

$a_1, a_2, \ldots, a_n$是长度为n的正数序列且值互异。假设$a_0 = a_{n+1} = 0$。
$a_k\ (k \in [1, n])$是该序列的极大值当且仅当 $a_k > \max(a_{k+1}, a_{k-1})$。
找出序列中任意1个极大值的位置，时间复杂度$O(\log(n))$。

l				m				r
0	8	3	15	9	7	14	12	0

↓

l		m		r				
0	8	3	15	9	7	14	12	0

↓

		l	m	r				
0	8	3	15	9	7	14	12	0

maximum!

Binary search: Implementation

（面试题） 两个升序（非降序）序列$\mathrm{a}[1..\mathrm{n}], \mathrm{b}[1..\mathrm{m}]$ $(\mathrm{n}, \mathrm{m} \geq 1)$， 查找合并后的中位数，即排在$\left\lfloor \frac{m+n}{2} \right\rfloor$位的元素.

输入：

a	1	2	4	5	6	12
b	3	7	8	9	10	11

输出： 6

（1）归并 时间复杂度 $\Theta(\frac{n+m}{2})$

（2）双二分 时间复杂度 $O(\log(n)\log(m))$

（*）区间二分 时间复杂度 $O(\log(n+\mathrm{m}))$

Binary search: Implementation

（面试题） 两个升序（非降序）序列$a[1..n], b[1..m]\ (n, m \geq 1)$， 查找合并后的中位数，即排在$\left\lfloor \frac{m+n}{2} \right\rfloor$位的元素.

（1）归并 时间复杂度 $\Theta(\frac{n+m}{2})$

- 归并两个排好序的子序列，第k次比较返回的值就是合并后的第k小值
- 因为是找中位数，不需要排序以及使用辅助空间

i)

1	2	4	5	6	12
3	7	8	9	10	11

返回1

ii)

1	2	4	5	6	12
3	7	8	9	10	11

返回2

iii)

1	2	4	5	6	12
3	7	8	9	10	11

返回3

iv)

1	2	4	5	6	12
3	7	8	9	10	11

返回4

v)

1	2	4	5	6	12
3	7	8	9	10	11

返回5

iv)

1	2	4	5	6	12
3	7	8	9	10	11

返回6

Binary search: Implementation

（面试题） 两个升序（非降序）序列$a[1..n], b[1..m]$ $(n, m \geq 1)$， 查找合并后的中位数，即排在$\left\lfloor \frac{m+n}{2} \right\rfloor$位的元素.

（2）双二分 时间复杂度 $O(\log(n)\log(m))$

- 对数组b的元素$b[i]$ $(1 \leq i \leq m)$，定义 $\#^a(i) = |\{a[j] \mid 1 \leq j \leq n \wedge a[j] \leq b[i]\}|$，即数组a中小于等于$b[i]$的元素数量。$\#^a(i) = 0$ 表示数组a中所有元素比$b[i]$大，$\#^a(i) = n$ 表示所有元素小于等于$b[i]$，除此之外$a[\#^a(i)] \leq b[i] < a[\#^a(i) + 1]$

- 定义$T(i) = i + \#^a(i)$， $T(i)$表示$b[i]$在合并后的序列中的（绝对）位次

  - $b[1] \leq \cdots \leq b[m] \Rightarrow \#^a(1) \leq \cdots \leq \#^a(m) \Rightarrow T(1) < \cdots < T(m)$

- 双二分查找 $\min\limits_{1 \leq c \leq m} \left(\left|\frac{m+n}{2} - T(c)\right|\right)$ ： 二分b[1..m] + 二分搜索a[1..n]

- 找出中位数
$$\begin{cases} 1)\ T[c] = \frac{m+n}{2} \Rightarrow b[c]\text{是中位数} \\ 2)\ T[c] > \frac{m+n}{2} \Rightarrow a[\frac{m+n}{2} - c + 1]\text{是中位数} \\ 3)\ T[c] < \frac{m+n}{2} \Rightarrow a[\frac{m+n}{2} - c]\text{是中位数} \end{cases}$$

Binary search: Implementation

（2）双二分　　时间复杂度 $O(\log(n)\log(m))$

i)

a	1	2	4	5	6	12
b	3	7	8	9	10	11
	l		m			r

$\underbrace{l = 1, r = 6, m = 3}_{\text{二分b}} \mid \underline{\#^a(3) = 5} \Rightarrow T(3) = 8 \quad (> 6)$

二分搜索a

ii)

a	1	2	4	5	6	12
b	3	7	8	9	10	11
	l	m	r			

$l = 1, r = 3, m = 2 \mid \#^a(2) = 5 \Rightarrow T(3) = 7 \quad (> 6)$

iii)

a	1	2	4	5	6	12
b	3	7	8	9	10	11
	l m	r				

$l = 1, r = 2, m = 1 \mid \#^a(1) = 2 \Rightarrow T(1) = 3 \quad (< 6)$

iv)

a	1	2	4	5	6	12
b	3	7	8	9	10	11
		l r				

$\underline{l = r = m = 2} \mid \#^a(2) = 5 \Rightarrow T(2) = 7 \ (> 6)$

双二分结束

$\boxed{\text{中位数：} a[6-2+1] = a[5] = 6}$

Binary search: Implementation

不似“二分”，恰是二分

快速求幂 $a^n \ (a, n > 0)$

- 直接迭代：$a^n = a^{n-1} * a$　　　时间复杂度（相乘次数）$\Theta(n)$

- 二分递归：$\boxed{a^n = a^{n\%2} * a^{\frac{n}{2}} * a^{\frac{n}{2}}}$

时间 $\Theta(\log(n))$

Binary search: Implementation

快速求幂 $a^n \ (a, n > 0)$

- 线性递归：$a^n = a^{n-1} * a$　　时间复杂度（相乘次数）$\Theta(n)$

- 二分递归：

$$a^n = a^{n\%2} * a^{\frac{n}{2}} * a^{\frac{n}{2}}$$

时间 $\Theta(\log(n))$

```
int power(int a, int n)  {
    if(n == 0) return 1;

    int p = power(a, n/2);
                //二分，求a^(n/2)
    return p*p*(n&1? a : 1);
}
```

Binary search: Implementation

不似“二分”，恰是二分

班長の食卓

- 卓越班n个人在实验室，到了中午，大家决定一起从乡村基点同样的套餐。
- 按照惯例，大家一人一票投票表决，选获票数最多的套餐一起点。
- 今天，童心未泯的班长突然想让全班一起吃**儿童套餐**，怎么办呢？
- 开始时每个人已经定好各自想吃的套餐，因此要让儿童套餐获票最多，必须说服足够多的人改变原先的选择，这可真是费时费神的事情！
- 因此，班长让你设计一个拉票方案，使儿童套餐的获票最多（不允许票数并列），同时需要说服的人数最少。

Binary search: Implementation

不似“二分”，恰是二分

班長の食卓