1. 数组A=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]， 一共有25个素数,按照二分搜索算法寻找元素67，并分析算法复杂度。

**输入：** 按非降序排列的 $n$ 个元素的数组 $A[1\cdots n]$和元素 $x$。

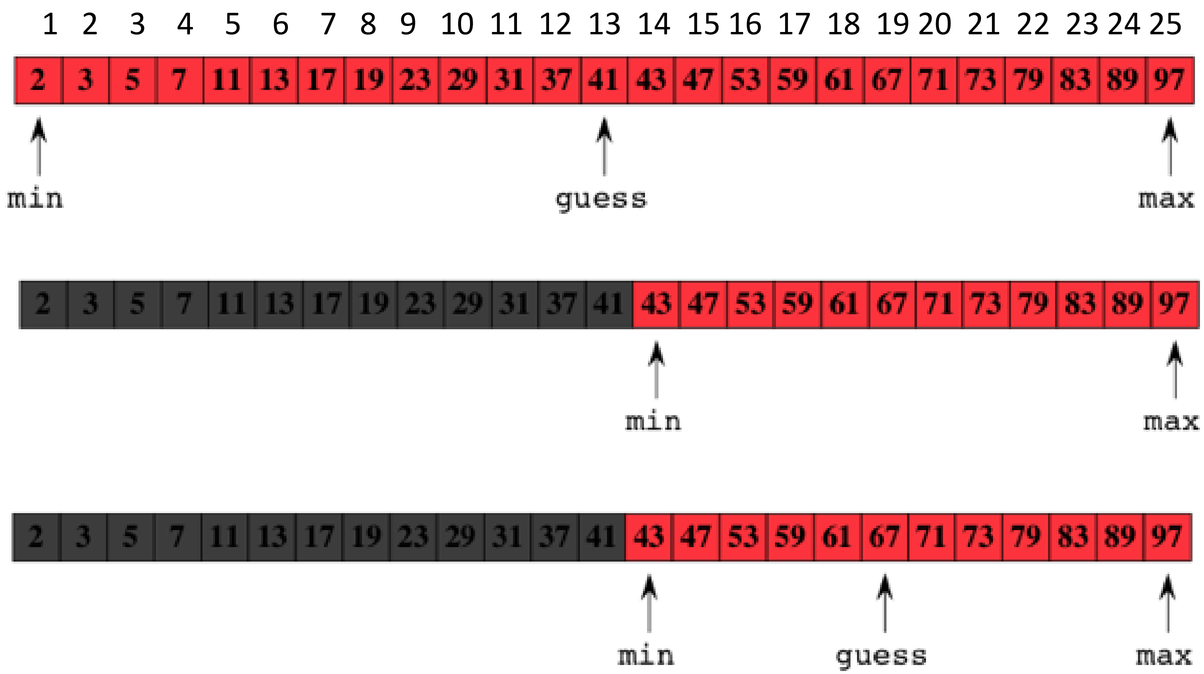
**输出：** 如果 $x = A[j]$，则输出 $j$；否则输出 0。

```
binarysearch(1, n)
```

**过程** *binarysearch*(*low*, *high*)

```
if low > high then return 0
else
    mid ← ⌊(low + high)/2⌋
    if x = A[mid] then return mid
    else if x < A[mid] then return binarysearch(low, mid − 1)
    else return binarysearch(mid + 1, high)
end if
```

1. 数组A=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]， 一共有25个素数, 按照二分搜索算法寻找元素67，并分析算法复杂度。




因此元素67所在的数组位置是19, 一共执行3次比较。

1. 数组A=[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]， 一共有25个素数, 按照二分搜索算法寻找元素67，并分析算法时间复杂性。

当$n$ =0时，即数组为空时，不执行任何比较；

当$n$ =1时，需要执行一次比较；

当$n$ >1时，$C(n)$表示二分搜索算法在最坏情况下执行的比较次数，则$C(n)$的递推式为：

$$C(n) \le \begin{cases} 1, & if\ n = 1 \\ 1 + C\left(\left\lfloor \frac{n}{2} \right\rfloor\right), & if\ n \ge 2 \end{cases}$$

对于整数$k$，满足$2^{k-1} \le n \le 2^k$, 展开上述递推式，

$$\begin{aligned} C(n) &\le 1 + C(\lfloor \mathrm{n}/2 \rfloor) \\ &\le 2 + C(\lfloor \mathrm{n}/4 \rfloor) \\ &= k = \lfloor logn \rfloor + 1 \end{aligned}$$

因此，二分搜索算法的时间复杂性是$O(logn)$.

2. 设有数组A=[44,75,23,43,55,12]，按照划分算法确定划分元素A[low]的新位置。

SPLIT

输入：数组 $A[low \cdots high]$。

输出：(1) 如有必要，输出按上述描述的重新排列的数组 $A$；

(2) 划分元素 $A[low]$的新位置 $w$。

```
i ← low
x ← A[low]
for j ← low + 1 to high
    if A[j] ≤ x then
        i ← i + 1
        if i ≠ j then 互换 A[i]和 A[j]
    end if
end for
互换 A[low]和 A[i]
w ← i
return A 和 w
```

2. 设有数组A=[44, 75, 23, 43, 55, 12]，按照划分算法确定划分元素A[low]的新位置。

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 44 | 75 | 23 | 43 | 55 | 12 |
| i | j | | | | |

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 44 | 75 | 23 | 43 | 55 | 12 |
| i | | j | | | |

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 44 | 23 | 75 | 43 | 55 | 12 |
| | i | j | | | |

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 44 | 23 | 75 | 43 | 55 | 12 |
| | i | | j | | |

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 44 | 23 | 43 | 75 | 55 | 12 |
| | | i | j | | |

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 44 | 23 | 43 | 75 | 55 | 12 |
| | | i | | | j |

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 44 | 23 | 43 | 12 | 55 | 75 |
| | | | i | | j |

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| 12 | 23 | 43 | 44 | 55 | 75 |
| | | | i | | j |

3. 给定数组 A=[44,75,23,43,55,12,64,77,33]，按照快速排序算法进行排序，并分析最坏情况下的时间复杂度。

**QUICKSORT**

**输入**：$n$ 个元素的数组 $A[1\cdots n]$。

**输出**：按非降序排列的数组 $A$ 中的元素。

```
quicksort(A, 1, n)
```

**过程** *quicksort*$(A, low, high)$

```
if low < high then
    SPLIT(A[low···high], w)
    {w 为 A[low]的新位置}
    quicksort(A, low, w − 1)
    quicksort(A, w + 1, high)
end if
```

**SPLIT**

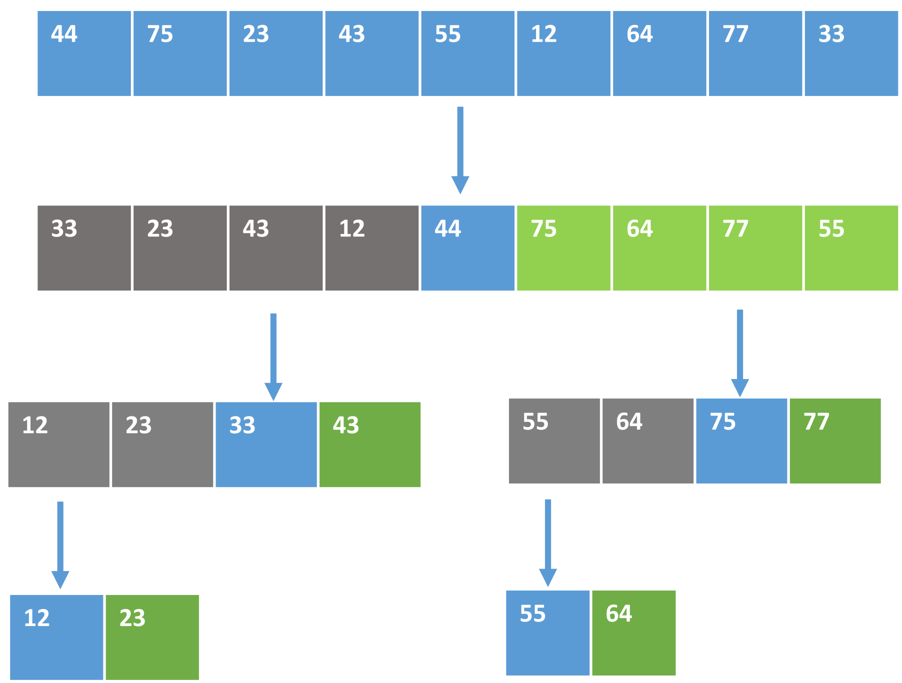
**输入**：数组 $A[low\cdots high]$。

**输出**：(1) 如有必要，输出按上述描述的重新排列的数组 $A$；

(2) 划分元素 $A[low]$的新位置 $w$。

```
i ← low
x ← A[low]
for j ← low + 1 to high
    if A[j] ≤ x then
        i ← i + 1
        if i ≠ j then 互换 A[i]和 A[j]
    end if
end for
互换 A[low]和 A[i]
w ← i
return A 和 w
```

3. 给定数组 A=[44, 75, 23, 43, 55, 12, 64, 77, 33]，按照快速排序算法进行排序，并分析最坏情况下的时间复杂度。



3. 给定数组 A=[44,75,23,43,55,12,64,77,33]，按照快速排序算法进行排序，并分析最坏情况下的时间复杂度。

假设数组A[1…n]是升序排列:

①在quicksort(A,1,n)中，A[1]最小，调用quicksort(A,2,n）

②在quicksort(A,2,n)中，A[2]最小，调用quicksort(A,3,n)

③下面过程调用quicksort(A,4,n)，…，quicksort(A,n,n)

④quicksort(A,1,n)中，spilt算法的比较次数是n-1;

quicksort(A,2,n)中，spilt算法的比较次数是n-2;

…

quicksort(A,n,n)中，spilt算法的比较次数是0;

因此这种情况下，快速排序算法的复杂度是:

$$(n-1)+(n-2)+\cdots+1=\frac{n(n-1)}{2}=\Theta(n^2).$$