

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。

1. *binarysearch*(1, n)

过程 *binarysearch*(*low*, *high*)

1. **if** *low* > *high* **then return** 0

2. **else**

3. $mid \leftarrow \lfloor (low + high)/2 \rfloor$

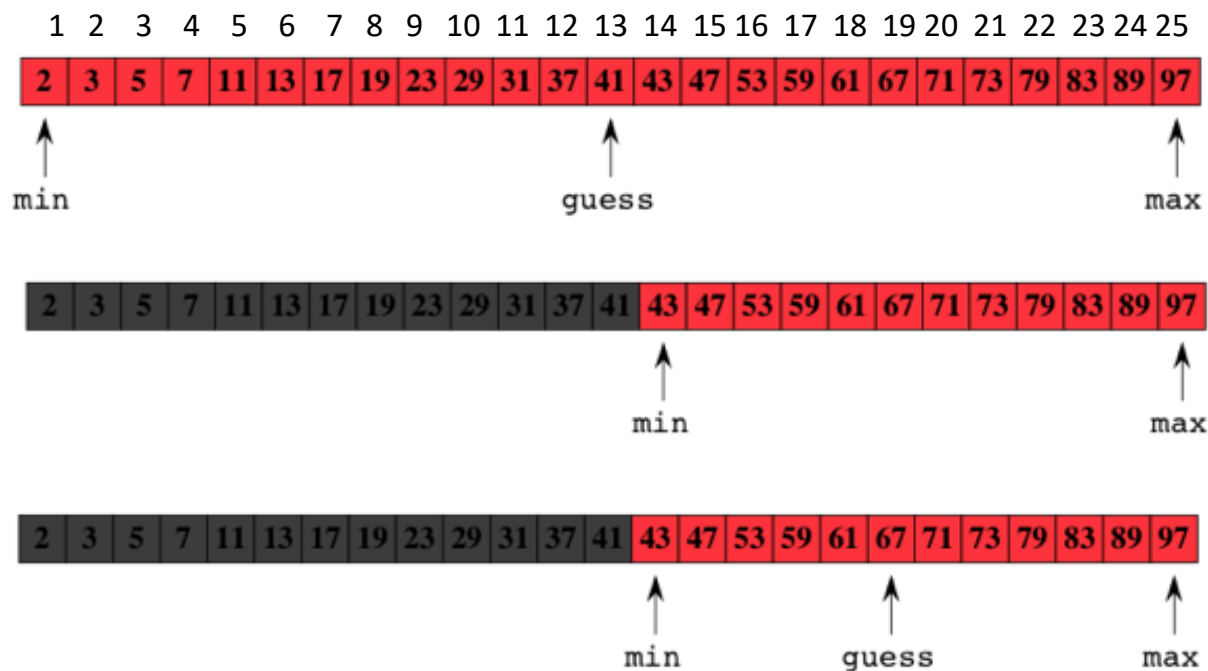
4. **if** $x = A[mid]$ **then return** *mid*

5. **else if** $x < A[mid]$ **then return** *binarysearch*(*low*, *mid* - 1)

6. **else return** *binarysearch*(*mid* + 1, *high*)

7. **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:

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

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

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

因此, 二分搜索算法的时间复杂性是 $O(\log n)$.

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

SPLIT

输入：数组 $A[\text{low} \cdots \text{high}]$ 。

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 $A[\text{low}]$ 的新位置 w 。

```
1.  $i \leftarrow \text{low}$ 
2.  $x \leftarrow A[\text{low}]$ 
3. for  $j \leftarrow \text{low} + 1$  to  $\text{high}$ 
4.   if  $A[j] \leq x$  then
5.      $i \leftarrow i + 1$ 
6.     if  $i \neq j$  then 互换  $A[i]$  和  $A[j]$ 
7.   end if
8. end for
9. 互换  $A[\text{low}]$  和  $A[i]$ 
10.  $w \leftarrow i$ 
11. 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。

1. $quicksort(A, 1, n)$

过程 $quicksort(A, low, high)$

1. **if** $low < high$ **then**
2. $SPLIT(A[low \cdots high], w)$
 $\{w$ 为 $A[low]$ 的新位置 $\}$
3. $quicksort(A, low, w - 1)$
4. $quicksort(A, w + 1, high)$
5. **end if**

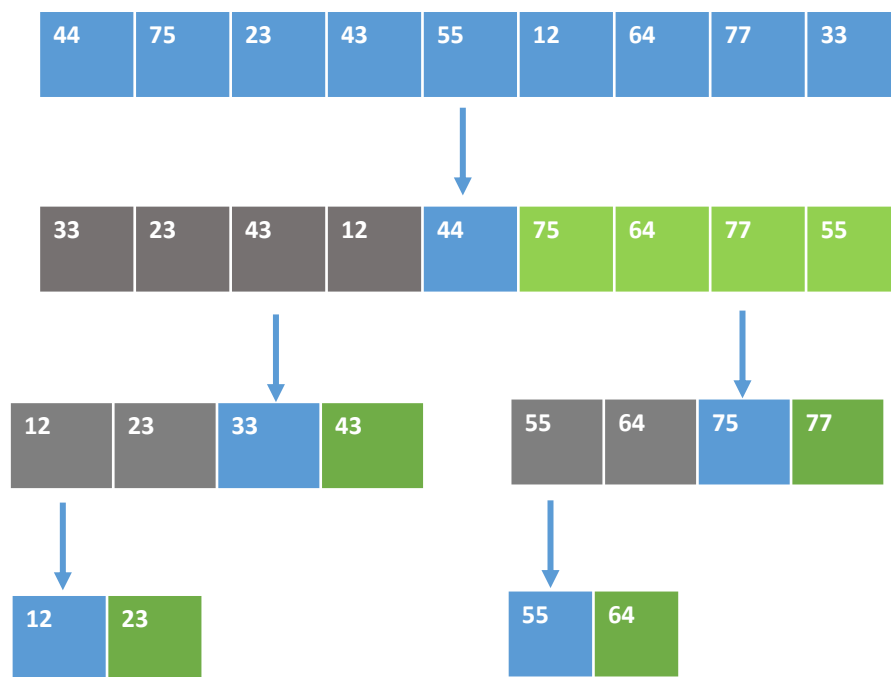
SPLIT

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 。

1. $i \leftarrow low$
2. $x \leftarrow A[low]$
3. **for** $j \leftarrow low + 1$ **to** $high$
4. **if** $A[j] \leq x$ **then**
5. $i \leftarrow i + 1$
6. **if** $i \neq j$ **then** 互换 $A[i]$ 和 $A[j]$
7. **end if**
8. **end for**
9. 互换 $A[low]$ 和 $A[i]$
10. $w \leftarrow i$
11. **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 $A[1 \cdots n]$ 是升序排列：

①在 $\text{quicksort}(A, 1, n)$ 中， $A[1]$ 最小，调用 $\text{quicksort}(A, 2, n)$

②在 $\text{quicksort}(A, 2, n)$ 中， $A[2]$ 最小，调用 $\text{quicksort}(A, 3, n)$

③下面过程调用 $\text{quicksort}(A, 4, n)$ ， \cdots ， $\text{quicksort}(A, n, n)$

④ $\text{quicksort}(A, 1, n)$ 中，spilt 算法的比较次数是 $n-1$ ；

$\text{quicksort}(A, 2, n)$ 中，spilt 算法的比较次数是 $n-2$ ；

\cdots

$\text{quicksort}(A, n, n)$ 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：

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