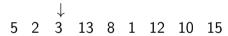
5 2 3 13 8 1 12 10 15

↓
5 2 3 13 8 1 12 10 15

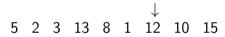
5 2 3 13 8 1 12 10 15





5 2 3 13 8 1 12 10 19

5 2 3 13 8 1 12 10 15



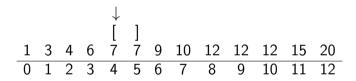


5 2 3 13 8 1 12 10 15

### Алгоритм линейного поиска

```
int Find(int[] array, int query)
{
    for (int i=0;i<array.Length;i++)
        if (array[i]==query)
            return i;
    return -1;
}</pre>
```

1	3	4	6	7	7	9	10	12	12	12	15	20
0	1	2	3	4	5	6	7	8	9	10	11	12



```
int Find(int[] array, int query)
    int left=0;
    int right=array.Length-1;
    while(left<right)
        var middle=(left+right)/2;
        if (query<=array[middle])</pre>
            right=middle;
        else
            left=middle+1;
    }
    if (array[right] == query)
        return right;
    return -1;
```

```
int Find(int[] array, int query)
    int left=0;
    int right=array.Length-1;
    while(left<right)
        var middle=(left+right)/2;
        if (query<=array[middle])</pre>
            right=middle;
        else
            left=middle+1;
    }
       (array[right] == query)
        return right;
    return -1;
```

Утверждение. На каждой итерации значение right-left уменьшается.

```
int Find(int[] array, int query)
    int left=0;
    int right=array.Length-1;
    while(left<right)
        var middle=(left+right)/2;
        if (query<=array[middle])</pre>
            right=middle;
        else
            left=middle+1:
    }
       (array[right] == query)
        return right;
    return -1;
```

Утверждение. На каждой итерации значение right-left уменьшается.

Следовательно, рано или поздно оно станет неположительным, т.е. условие left<right нарушится.

```
int Find(int[] array, int query)
    int left=0;
    int right=array.Length-1;
    while(left<right)
        var middle=(left+right)/2;
        if (query<=array[middle])</pre>
            right=middle;
        else
            left=middle+1:
    }
       (array[right] == query)
        return right;
    return -1;
```

Утверждение. Если в массиве есть искомые значения, то на каждой итерации хотя бы одно из них находится между индексов left и right

```
int Find(int[] array, int query)
    int left=0;
    int right=array.Length-1;
    while(left<right)
        var middle=(left+right)/2;
        if (query<=array[middle])</pre>
            right=middle;
        else
            left=middle+1:
    }
       (array[right] == query)
        return right;
    return -1;
```

Утверждение. Если в массиве есть искомые значения, то на каждой итерации хотя бы одно из них находится между индексов left и right

Следовательно, если по завершению массива значение не обнаружено, то его не было в исходном массиве.

```
int Find(int[] array, int query)
    int left=0;
    int right=array.Length-1;
    while(left<right)
        var middle=(left+right)/2;
        if (query<=array[middle])</pre>
            right=middle;
        else
            left=middle+1;
    }
       (array[right] == query)
        return right;
    return -1;
```

Утверждение. На каждой итерации значение right-left уменьшается не менее чем вдвое.

```
int Find(int[] array, int query)
    int left=0;
    int right=array.Length-1;
    while(left<right)
        var middle=(left+right)/2;
        if (query<=array[middle])</pre>
            right=middle;
        else
            left=middle+1:
    }
       (array[right] == query)
        return right;
    return -1;
```

Утверждение. На каждой итерации значение right-left уменьшается не менее чем вдвое.

Утверждение. Если последовательно уменьшать число N путем целочисленного деления на 2, то в ноль оно обратится не позже, чем через  $\lfloor \log_2 N \rfloor + 1$  шагов

```
int Find(int[] array, int query)
    int left=0;
    int right=array.Length-1;
    while(left<right)
        var middle=(left+right)/2;
        if (query<=array[middle])</pre>
            right=middle;
        else
            left=middle+1:
    }
    if (array[right] == query)
        return right;
    return -1;
```

Утверждение. На каждой итерации значение right-left уменьшается не менее чем вдвое.

Утверждение. Если последовательно уменьшать число N путем целочисленного деления на 2, то в ноль оно обратится не позже, чем через  $\lfloor \log_2 N \rfloor + 1$  шагов

Следовательно, алгоритм совершит порядка  $\log \operatorname{array}.\operatorname{Length}$  шагов, и имеет оценку сложности  $\Theta(\log_n)$ .

$$\frac{l+r}{2} = l + \frac{r-l}{2}$$

Вариант 1.

$$r' = I + \frac{r-1}{2}, \ I' = I$$
  
 $r' - I' = I + \frac{r-1}{2} - I = \frac{r-1}{2} < r-1$ 

Вариант 2.

$$r' = r$$
,  $l' = 1 + l + \frac{r - l}{2}$   
 $r' - l' = r - l - \frac{r - l}{2} - 1 < \frac{r - l}{2} < r - l$ 

$$I = \lceil \log_2 N \rceil + 1$$
  $2^I > N$   $1 = \frac{2^I}{2^I} > \frac{N}{2^I}$ 

### Анализ алгоритма

- 1. Доказательство корректности:
  - 1.1 Алгоритм всегда останавливается?
  - 1.2 Алгоритм всегда возвращает правильный ответ?
- 2. Оценка сложности алгоритма