# Поиск элементов в массиве

Пусть дан массив элементов. Вам периодически приходят запросы вида "Есть ли в массиве элемент $x$". Рассмотрим ряд алгоритмов, позволяющих отвечать на такие запросы. Представленные далее алгоритмы будут возвращать позицию элемента в массиве или $-1$, если элемент не найден.

## 1 Линейный поиск

Самый простой вариант – просто пройтись по всему массиву, проверяя элементы на равенство $x$. Такой алгоритм будет работать за $O(n)$.

```
def linear_search(a, x):
for i in range(len(a)):
    if a[i] == x:
        return i
return -1
```

## 2 Двоичный поиск

Пусть изначально заданный массив был отсортирован или, по крайней мере, условия задачи не запрещают нам этого сделать. Тогда для отсортированного массива можно применить алгоритм поиска быстрее, чем за $O(n)$. Идея двоичного поиска следующая. Две границы $l$ и $r$ задают область поиска на массиве (изначально область поиска – весь массив). Рассмотрим элемент стоящий ровно по середине области, т.е. на месте $m = \lfloor \frac{l+r}{2} \rfloor$. Если $a[m] < x$, то искомый элемент лежит правее $m$, и все что левее $m$ можно исключить из области поиска. Аналогично для $a[m] > x$. Как только границы поиска сомкнутся, т.е. $l + 1 = r$, то алгоритм окончен.

```
def bin_search_left(a, x):
    l, r = -1, len(a)
    while l + 1 < r:
        m = (l + r) // 2
        if a[m] < x:
            l = m
        else:
            r = m
    return r if r < len(a) and a[r] == x else -1
```

Обратите внимание, что в приведенной выше реализации правая граница поиска сдвигается при $a[m] \geq x$. В таком случае $r$ будет указывать на самый первый элемент, равный $x$ (левосторонный двоичный поиск). Если необходимо найти последний элемент, равный $x$ (правосторонний двоичный поиск), то при равенстве $x$ необходимо двигать левую границу.

```
def bin_search_right(a, x):
    l, r = -1, len(a)
    while l + 1 < r:
        m = (l + r) // 2
        if a[m] <= x:
            l = m
        else:
            r = m
    return l if l > -1 and a[l] == x else -1
```

Вне зависимости от реализации, алгоритм двоичного поиска работает за $O(\log n)$.

Обратите внимание, что двоичный поиск может применяться для поиска значения в монотонной функции. Пусть задана монотонная функция $y = f(x)$. Необходимо найти такой $\hat{x}$, что $\hat{y} = f(\hat{x})$ для заданного $\hat{y}$. Работа идет с вещественными числами! В таком случае критерием окончания будет $r - l < \epsilon$, где $\epsilon$ – требуемая точность.

## Список литературы

Кормен, Томас и др. (2013). *Алгоритмы. Построение и анализ. Третье издание*. Издательский дом «Вильямс».