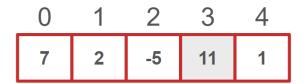
# Сортировка и поиск

**Поиск в программировании** - это ключевая операция, направленная на нахождение элемента в наборе данных или определение его точной позиции. Один из простейших алгоритмов поиска - линейный поиск, который осуществляется путем последовательного перебора элементов в данных. Этот метод особенно полезен, когда структура данных не упорядочена и не имеет каких-либо специальных свойств. Однако, линейный поиск может быть неэффективным на больших объемах данных, особенно в худшем случае, когда искомый элемент отсутствует в коллекции. В таких ситуациях алгоритму приходится проверять каждый элемент, что приводит к увеличению времени выполнения. Это подчеркивает важность выбора подходящего алгоритма поиска в зависимости от характера и размера набора данных.

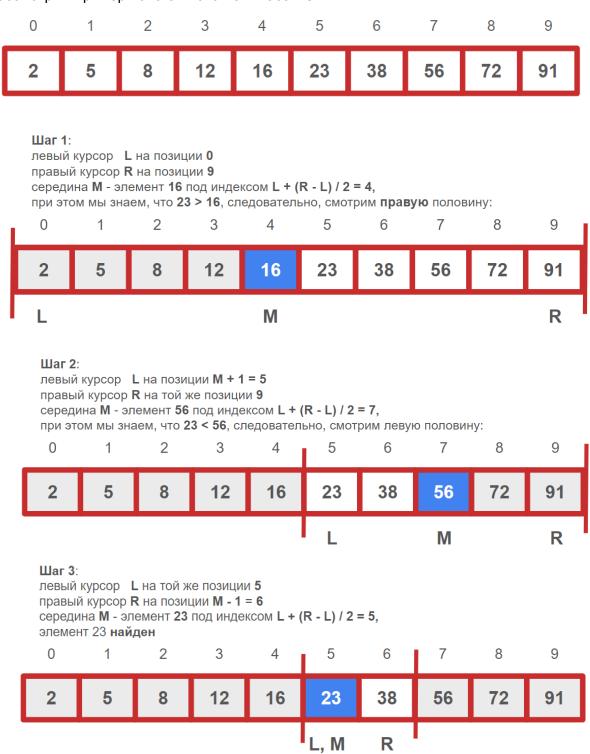


Поиск числа **11** в зависимости от реализации вернет либо **true** (присутствует), либо **3** (позиция элемента)

Поиск числа **777** в зависимости от реализации вернет либо **false** (отсутствует), либо **-1** (невозможный индекс). При этом будут проверены **все** элементы.

**Бинарный поиск** - это эффективный алгоритм поиска, который находит широкое применение в случаях, когда данные уже упорядочены. Принцип его работы схож с методом поиска слова в словаре: вместо того чтобы перебирать все слова, мы делим словарь на половины до тех пор, пока не найдем нужное слово. В контексте бинарного поиска это означает, что при каждом шаге алгоритм сравнивает искомый элемент с элементом в середине диапазона, сокращая область поиска вдвое. Это значительно ускоряет поиск по сравнению с линейным методом, особенно в больших наборах данных. Таким образом, бинарный поиск идеально подходит для быстрого нахождения элементов в больших и упорядоченных наборах данных, предлагая высокую эффективность и скорость выполнения по сравнению с другими алгоритмами поиска.

## Рассмотрим пример поиска числа 23 в массиве:



Таким образом, было выполнено только 3 сравнения.

Линейный и бинарный алгоритмы поиска: <u>анимация</u>

Сортировка - это фундаментальный процесс в области компьютерных наук, который заключается в упорядочивании элементов в наборе данных согласно определенному критерию, например, по возрастанию или убыванию. Один из ключевых аспектов сортировки - это её способность ускорить процессы поиска, так как упорядоченные данные позволяют использовать более эффективные методы поиска, такие как бинарный поиск. Существует множество алгоритмов сортировки, каждый из которых имеет свои особенности и подходит для различных сценариев использования.

Алгоритм сортировки выбором: анимация

Одним из простейших примеров является алгоритм сортировки выбором (selection sort). Этот алгоритм работает, последовательно находя наименьший (или наибольший) элемент из неотсортированной части массива и обменивая его с первым неотсортированным элементом. Несмотря на свою простоту, сортировка выбором не является самым быстрым алгоритмом, особенно на больших наборах данных. Однако, его легкость в понимании и реализации делает его хорошим выбором для введения в концепции сортировки. Сортировка выбором иллюстрирует важный принцип в алгоритмах сортировки - баланс между эффективностью и сложностью, позволяющий выбирать оптимальный метод в зависимости от конкретной задачи и объема данных.

Сортировка выбором (Selection Sort) на наборе данных [8, 6, 2, 5, 1] происходит следующим образом:

## Первый проход:

Находим минимальный элемент во всем массиве. Минимальный элемент - 1. Меняем местами 1 и первый элемент массива (8). Новый массив: [1, 6, 2, 5, 8].

### Второй проход:

Игнорируем уже отсортированный элемент (1) и ищем минимальный элемент в оставшейся части массива. Минимальный элемент - 2. Меняем местами 2 и первый элемент из неотсортированной части (6).

Новый массив: [1, 2, 6, 5, 8].

#### Третий проход:

Продолжаем игнорировать отсортированную часть (1, 2) и находим минимальный элемент в оставшейся части. Здесь это 5.

Меняем местами 5 и первый элемент из неотсортированной части (6).

Новый массив: [1, 2, 5, 6, 8].

## Четвертый проход:

Игнорируем уже отсортированные элементы (1, 2, 5) и выбираем минимальный из оставшихся. Это 6, который уже находится на своем месте.

Поскольку 6 уже находится на своей позиции, никаких обменов не происходит.

Новый массив: [1, 2, 5, 6, 8].

Пятый проход:

Остается только один элемент (8), который уже находится на своем месте. Массив уже полностью отсортирован: [1, 2, 5, 6, 8].

