Решение задач с использованием линейных массивов: **реализация простых алгоритмов сортировки**.

Для решения многих задач необходимо упорядочить данные по определенному признаку. Этот процесс называют *сортировкой*.

Сортировка простым выбором.

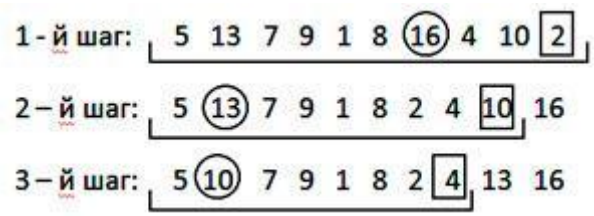
Пусть исходный массив состоит из 10 элементов: 5 13 7 9 1 8 16 4 10 2

После сортировки массив должен выглядеть так: 1 2 4 5 7 8 9 10 13 16

Максимальный элемент заключен в

Элемент, с которым происходит обмен

Скобкой помечена рассматриваемая часть массива, т.е. неупорядоченная.



и т. д.

```
def selection_sort(arr):
    # Проходим по всем элементам массива
    for i in range(len(arr)):
        # Находим минимальный элемент в оставшейся части массива
        min_index = i
        for j in range(i + 1, len(arr)):
            if arr[j] < arr[min_index]:
                min_index = j
        # Меняем местами найденный минимальный элемент с первым элементом
        arr[i], arr[min_index] = arr[min_index], arr[i]
    return arr

# Пример использования
numbers = [64, 25, 12, 22, 11]
sorted_numbers = selection_sort(numbers)
print("Отсортированный массив:", sorted_numbers)
```

Сортировка простым обменом.

Отсортируем по возрастанию массив: 5 4 8 2 9

Длина текущей (неупорядоченной) части массива - (N - k + 1), где k -номер просмотра, i - номер первого элемента проверяемой пары, номер последней пары - (N - k).

1 - й просмотр. Рассматривается весь массив.

i = 1 5 4 8 2 9

> меняем

i = 2 4 5 8 2 9

< не меняем

i = 3 4 5 8 2 9

> меняем

$i = 4$ 4 5 2 8 9

< не меняем

9 находится на своем месте.

2 - й просмотр. Рассматривается часть массива с 1-го до предпоследнего элемента.

$i = 1$ 4 5 2 8 9

< не меняем

$i = 2$ 4 5 2 8 9

> меняем

$i = 3$ 4 2 5 8 9

< не меняем

8 на своем месте.

3 - й просмотр. Рассматривается часть массива содержащая три первых элемента.

$i = 1$ 4 2 5 8 9

> меняем

$i = 2$ 2 4 5 8 9

< не меняем

5 на своем месте.

4 - й просмотр. Рассматриваем последнюю пару элементов.

$i = 1$ 2 4 5 8 9

< не меняем

4 на своем месте.

Наименьший элемент - 2 оказывается на первом месте. Количество просмотров элементов массива равно N - 1. Этот метод также называют методом "пузырька".

```python
def bubble_sort(arr):
    n = len(arr)
    # Проходим по всем элементам массива
    for i in range(n):
        # Последние i элементов уже отсортированы
        for j in range(0, n-i-1):
            # Меняем местами, если элемент найден больше, чем следующий
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]

# Пример использования
arr = [64, 34, 25, 12, 22, 11, 90]
bubble_sort(arr)
print("Отсортированный массив:", arr)
```

Сортировка простыми вставками.

Пусть требуется отсортировать массив из 10 элементов по возрастанию:

1-й шаг 13 6 8 11 3 1 5 9 15 7

Рассматриваем часть массива из одного элемента 13. Вставляем второй элемент массива 6 так, что упорядоченность сохранялась. Т. к. 6 < 13, записываем 6 на первое место. отсортированная часть массива содержит два элемента (6 и 13).

2 - й шаг 6 13 8 11 3 1 5 9 15 7

Возьмем третий элемент массива 8 и подберем для него место в упорядоченной части массива 8 > 6 и 8 < 13, следовательно, его место - второе.

3 - й шаг 6 8 13 11 3 1 5 9 15 7

Следующий элемент - 11. Он записывается в упорядоченную часть массива на третье место, т. к. 11 > 8, но 11 < 13

4 - й шаг 6 8 11 13 3 1 5 9 15 7

Далее, действуя аналогичным образом, определяем, что 3 необходимо записать на первое место.

5 - шаг 3 6 8 11 13 1 5 9 15 7

По той же причине 1 записываем на первое место.

6 - шаг 1 3 6 8 11 13 5 9 15 7

Т. к. 5 > 3, но 5 < 6, то место 5 в упорядоченной части - третье.

7 - шаг 1 3 5 6 8 11 13 9 15 7

Место числа 9 - шестое.

8 - й шаг 1 3 5 6 8 9 11 13 15 7

Определяем место для предпоследнего элемента 15. Оказывается, что этот элемент массива уже находится на своем месте.

9 - шаг 1 3 5 6 8 9 11 13 15 7

Осталось подобрать подходящее место для последнего элемента 7.

1 3 5 6 7 8 9 11 13 15

Массив отсортирован полностью.

```python
def insertion_sort(arr):
    # Проходим по всем элементам массива, начиная со второго
    for i in range(1, len(arr)):
        key = arr[i]  # Текущий элемент для вставки
        j = i - 1

        # Сравниваем текущий элемент с отсортированной частью массива
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]  # Сдвигаем элемент вправо
            j -= 1

        arr[j + 1] = key  # Вставляем текущий элемент на его место

# Пример использования
arr = [12, 11, 13, 5, 6]
insertion_sort(arr)
print("Отсортированный массив:", arr)
```