

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра вычислительных технологий

ЛАБОРАТОРНАЯ РАБОТА №1
Дисциплина: «Оценка сложности алгоритмов»

Работу выполнил: _____ А. А. Костров

Направление подготовки: 02.03.02 Фундаментальная информатика и
информационные технологии

Преподаватель: _____ А. А. Яхонтов

Краснодар
2026

Тема. Работа с векторами и массивами.

Цель. Найти функцию сложности алгоритмов: скалярное умножение векторов, перемножение квадратных матриц и сортировки массивов.

Реализуем алгоритм скалярного умножения векторов:

```
def inner_prod(u, v, n):  
    prod = 0  
    for i in range(n):  
        prod = prod + u[i] * v[i]  
    return prod
```

Временная сложность:

$T(n) = 3n + C = O(n)$ — функция сложности линейная

Ёмкостная сложность:

$S(n) = C = O(1)$ — функция сложности константная

Реализуем алгоритм перемножения квадратных матриц:

```
def matrix_prod(a, b, n):  
    c = [[0 for i in range(n)] for j in range(n)]  
    for i in range(n):  
        for j in range(n):  
            for k in range(n):  
                c[i][j] = c[i][j] + a[i][k] * b[k][j]  
    return c
```

Временная сложность:

$T(n) = 3n^3 + n^2 + C = O(n^3)$ — функция сложности кубическая

Ёмкостная сложность:

$S(n) = n^2 + C = O(n^2)$ — функция сложности квадратичная

Реализуем алгоритм сортировки вставкой:

```
def insertion_sort(arr, n):  
    for i in range(1, n):  
        key = arr[i]  
        j = i - 1  
        while j ≥ 0 and arr[j] > key:  
            arr[j + 1] = arr[j]  
            j -= 1  
        arr[j + 1] = key  
    return arr
```

Временная сложность для лучшего случая (когда на вход поступает полностью отсортированный массив):

$T(n) = 5n + C = O(n)$ — функция сложности линейная

Временная сложность для среднего случая (когда на вход поступает наполовину отсортированный массив):

$T(n) = \frac{n^2}{2} + 5n + C = O(n^2)$ — функция сложности квадратичная

Временная сложность для худшего случая (когда на вход поступает полностью неотсортированный массив):

$T(n) = n^2 + 5n + C = O(n^2)$ — функция сложности квадратичная

Ёмкостная сложность:

$S(n) = C = O(1)$ — функция сложности константная

Реализуем алгоритм сортировки выбором:

```
def selection_sort(arr, n):  
    for i in range(n):  
        min_idx = i  
        for j in range(i + 1, n):  
            if arr[j] < arr[min_idx]:  
                min_idx = j  
        arr[i], arr[min_idx] = arr[min_idx], arr[i]  
    return arr
```

Временная сложность для лучшего случая (когда на вход поступает полностью отсортированный массив):

$$T(n) = \frac{n^2}{2} + 2n + C = O(n^2) \text{ — функция сложности квадратичная}$$

Временная сложность для среднего случая (когда на вход поступает наполовину отсортированный массив):

$$T(n) = \frac{3n^2}{4} + 2n + C = O(n^2) \text{ — функция сложности квадратичная}$$

Временная сложность для худшего случая (когда на вход поступает полностью неотсортированный массив):

$$T(n) = n^2 + 2n + C = O(n^2) \text{ — функция сложности квадратичная}$$

Ёмкостная сложность:

$$S(n) = C = O(1) \text{ — функция сложности константная}$$

Реализуем алгоритм сортировки пузырьком:

```
def bubble_sort(arr, n):  
    for i in range(n):  
        for j in range(0, n - i - 1):  
            if arr[j] > arr[j + 1]:  
                arr[j], arr[j + 1] = arr[j + 1], arr[j]  
    return arr
```

Временная сложность для лучшего случая (когда на вход поступает полностью отсортированный массив):

$$T(n) = \frac{n^2}{2} + C = O(n^2) \text{ — функция сложности квадратичная}$$

Временная сложность для среднего случая (когда на вход поступает наполовину отсортированный массив):

$$T(n) = \frac{3n^2}{4} + C = O(n^2) \text{ — функция сложности квадратичная}$$

Временная сложность для худшего случая (когда на вход поступает полностью неотсортированный массив):

$$T(n) = n^2 + C = O(n^2) \text{ — функция сложности квадратичная}$$

Ёмкостная сложность:

$$S(n) = C = O(1) \text{ — функция сложности константная}$$