МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

АНАЛИЗ СЛОЖНОСТИ СОРТИРОВОК, НЕ ИСПОЛЬЗУЮЩИХ СРАВНЕНИЕ ЭЛЕМЕНТОВ

ОТЧЁТ

студента 2 курса 251 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Толстова Роберта Сергеевича

Проверено:

доцент, к. ф.-м. н. _____ М. И. Сафрончик

Саратов 2025

СОДЕРЖАНИЕ

1 Сортировка подсчётом

1.1 Реализация на C++

```
void countingSort(vector<int>& arr) {
    if (arr.empty()) return;

    int maxVal = *max_element(arr.begin(), arr.end());
    int minVal = *min_element(arr.begin(), arr.end());
    size_t range = maxVal - minVal + 1;

    vector<int> count(range, 0);

    for (int num : arr) count[num - minVal]++;

    size_t arrIndex = 0;
    for (size_t i = 0; i < range; ++i) {
        while (count[i] > 0) {
            arr[arrIndex++] = i + minVal;
            count[i]--;
        }
    }
}
```

1.2 Анализ сложности

Проверка на пустой массив:

O(1) — проверка на пустой массив занимает постоянное время.

Поиск минимального и максимального элемента:

O(n) — проходим по всему массиву для нахождения минимального и максимального элементов (n — количество элементов в массиве).

Создание вспомогательного массива для подсчёта:

O(k), где k — это разница между максимальным и минимальным элементами. В худшем случае $k \sim n$

Подсчет вхождений каждого элемента:

O(n) — снова проходим по всему массиву и увеличиваем соответствующие счетчики.

Перезапись исходного массива в отсортированном порядке:

O(n + k) — проходим по счетчикам по счетчику, который имеет размер k, и восстанавливаем элементы исходного массива. В коде это вложенные циклы `for` и `while`

Общая временная сложность:

O(n + k), n — количество элементов в исходном массиве, k — разница между максимальным и минимальным элементами.

2 Поразрядная сортировка

2.1 Реализация на C++

```
void countingSortForRadix(vector<int>& arr, int exp) {
    size_t n = arr.size();
    vector<int> output(n);
    vector<int> count(10, 0);

    for (size_t i = 0; i < n; i++) count[(arr[i] / exp) % 10]++;

    for (int i = 1; i < 10; i++) count[i] += count[i - 1];

    for (int i = n - 1; i >= 0; i--) {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }

    for (size_t i = 0; i < n; i++) arr[i] = output[i];
}

void radixSort(vector<int>& arr) {
    if (arr.empty()) return;

    int maxVal = *max_element(arr.begin(), arr.end());

    for (int exp = 1; maxVal / exp > 0; exp *= 10)
countingSortForRadix(arr, exp);
}
```

2.2 Анализ сложности

Проверка на пустой массив:

O(1) – проверка на пустой массив занимает постоянное время.

Поиск максимального элемента

O(n) – это ситуация в худшем случае

Проходы по разрядам

O(n*d), где d – число разрядов в максимальном числе, а n – число элементов в массиве

Общая временная сложность

O(n*d) + O(n) + O(1) = O(n*d)