МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

АНАЛИЗ СЛОЖНОСТИ ДЛЯ СОРТИРОВОК, НЕ ИСПОЛЬЗУЮЩИХ СРАВНЕНИЕ ЭЛЕМЕНТОВ

ОТЧЕТ

студентки 2 курса 211 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета компьютерных наук и информационных технологий
Никитенко Яны Валерьевны

Саратов 2025

1 Сортировка подсчетом

1.1 Текст программы

```
    // Функция сортировки
vector<int> countSort(vector<int>& inputArray)
{
    int N = inputArray.size();
    int M = 0; // Нахождение максимального элемента массива inputArray[].

    for (int i = 0; i < N; i++)
        M = max(M, inputArray[i]);



   vector<int> countArray(M + 1, 0); // Массив запоненный нулями для хранения
    всех вхождений входного массива


    //
   for (int i = 0; i < N; i++)     // Элементы из ввода сопоставленны с индексами
        countArray[inputArray[i]]++;
    //


    //
    for (int i = 1; i <= M; i++)            // Выичсиление суммы префикса
    по каждому индексу
        countArray[i] += countArray[i - 1];
    //

    // Создания массива вывода
    vector<int> outputArray(N);

    for (int i = N - 1; i >= 0; i--) {
        outputArray[countArray[inputArray[i]] - 1]
            = inputArray[i];

        countArray[inputArray[i]]--;
```

```
    }
    return outputArray;
    //
}
//
```

1.2 Анализ сложности

Инициализация массива output происходит за время O(n), потому что размер массива равен n. Инициализация массива count происходит за время O(1), потому что размер массива фиксирован и равен 10. Первый цикл, который заполняет массив count, работает за O(n), где n – количество элементов.

Второй цикл, который преобразует массив count в префиксную сумму, работает за O(1) (поскольку размер массива фиксирован, равен 10). Третий цикл, который заполняет массив output, работает за O(n). Копирование временного массива output в исходный массив arr происходит за O(n) Таким образом, общая временная сложность составляет: O(n) + O(1) + O(n) + O(n+k) + O(n) = O(n+k)

2 Поразрядная сортировка (LSD)

2.1 Текст программы

```
// Функция для получения максимального элемента в массиве
int getMax(const vector<int>& arr) {
    return *max_element(arr.begin(), arr.end());
}
//

// Функция для сортировки массива по разряду
void countingSort(vector<int>& arr, int exp) {

    int n = arr.size();
    vector<int> output(n); // Временный массив
    для хранения отсортированных элементов
    int count[10] = { 0 }; // Массив для подсчета вхождений (0-9)

    // Подсчет вхождений для текущего разряда
    for (int i = 0; i < n; i++) {
        count[(arr[i] / exp) % 10]++;
    }
    //

    // Изменение count[i], чтобы count[i]
    содержал позицию этого разряда в output
    for (int i = 1; i < 10; i++) {
        count[i] += count[i - 1];
    }
    //

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

```
    //

    // Копирование отсортированных элементов обратно в массив
    for (int i = 0; i < n; i++) {
        arr[i] = output[i];
    }
    //
}
//

// Основная функция поразрядной сортировки
void radixSort(vector<int>& arr) {
    int maxVal = getMax(arr); // Нахождение максимального элемента
    for (int exp = 1; maxVal / exp > 0; exp *= 10) {
        countingSort(arr, exp);
    }
}
//
```

2.2 Анализ сложности

1.Инициализация переменных: - Инициализация переменной n равна O(1), так как она зависит только от размера вектора vec. - Инициализация переменной max итерируется по всем элементам вектора vec, следовательно, это займет O(n) времени. 2.Внешний цикл сортировки по разрядам: - Внешний цикл выполняется для каждого разряда числа, который оценивается как O(k), где k - количество разрядов в максимальном числе

3.Внутренние циклы: - Внутри внешнего цикла выполняются четыре внутренних цикла, каждый из которых имеет сложность O(n). Таким образом, общая сложность внутренних циклов составляет O(4n), что можно сократить до O(n). Итак, общая временная сложность алгоритма равна сумме сложностей всех его частей: O(1) + O(n) + O(k) * O(n) = O(kn)

Если количество разрядов большое, то сложность будет квадратичная. Если количество разрядов мало, то сложность будет линейная.