Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Южно-Уральский государственный университет (НИУ)»

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

ОТЧЕТ

о выполнении практического задания №4

по дисциплине

«Структуры и алгоритмы обработки данных»

Вариант 5

Проверил:

ст. преподаватель кафедры СП

Петрова Л.Н.

Выполнил:

Студент группы КЭз-391

Галиулин Р.Р.

Челябинск
2025

Содержание

1. Описание задачи

Задание: Алгоритмы сортировки данных

Отсортировать одномерный массив вещественных чисел размерности N, применив сортировки бинарным включением и шейкерную.

Входные данные:

- Размер массива n, целое число больше нуля
- Элементы массива, вещественные числа

Все данные вводятся через стандартный поток ввода

Выходные данные:

- Начальные элементы массива, введённой последовательности
- Отсортированный массив элементов, алгоритмом сортировки бинарным включением
- Отсортированный массив элементов, алгоритмом шейкирной сортировки
- Время выполнения каждого из алгоритмов

Данные выводятся через стандартный поток вывода

2. Листинги программ

Язык программирования: C++ 14. Среда разработки: Ubuntu 24.10 (6.11.0-13-generic), gcc 14.2.0, neovim

Листинг 1: Задание: Алгоритмы сортировки данных

```cpp
// Галиулин РР.. КЭз-391
// Структуры и алгоритмы обработки данных
// Практическое занятие №4

// Отсортировать одномерный массив вещественных чисел размерности n,
// применив сортировки бинарным включением и шейкерную.

#include <iostream>
#include <vector>
#include <algorithm>
#include <chrono>

//Печать массива
void PrintArray(const std::vector<double>& arr) {
    for (double num : arr) {
        std::cout << num << " ";
    }
    std::cout << endl;
}

// Функция бинарного поиска для нахождения позиции вставки
// arr - отсортированный массив, в который вставляется элемент
// value - вставляемое значение
// start - начальный индекс диапазона поиска
// end - конечный индекс диапазона поиска
int BinarySearch(const std::vector<double>& arr, double value,
                 int start, int end) {
    while (start <= end) {
        int mid = start + (end - start) / 2; // Находим середину
        if (arr[mid] == value) {
            return mid; // Если угадали сразу
        } else if (arr[mid] < value) { //Определяем в правой или левой части
            start = mid + 1;
        } else {
            end = mid - 1;
        }
    }
    return start; //Индекс, куда нужно вставить значение
}

/// Сортировка бинарным включением
// arr - массив для сортировки
void BinarySort(vector<double>& arr) {
    for (int i = 1; i < arr.size(); ++i) {
        double value = arr[i]; // Текущий элемент для вставки
        int j = BinarySearch(arr, value, 0, i - 1); // Находим куда вставить
        arr.erase(arr.begin() + i); // Удаляем элемент с текущей позиции
        arr.insert(arr.begin() + j, value); // Вставляем элемент в найденную
    позицию
    }
}

// Шейкерная сортировка
// arr - массив для сортировки
```

```
54 void ShakerSort(vector<double>& arr) {
55     int left = 0; // Левая граница
56     int right = arr.size() - 1; //Правая граница
57     while (left < right) {
58         // Идем слева на право - подымаем максимальный эелемент
59         for (int i = left; i < right; ++i) {
60             if (arr[i] > arr[i + 1]) {
61                 swap(arr[i], arr[i + 1]);
62             }
63         }
64         --right; //Подняли и  больше не задействуем
65         for (int i = right; i > left; --i) {
66         // Идем справа на лево - опускаем минимальный элемент
67             if (arr[i] < arr[i - 1]) {
68                 swap(arr[i], arr[i - 1]);
69             }
70         }
71         ++left; // Опустили и больше не задействуем
72     }
73 }
74
75 int main() {
76     int n;
77     std::cout << "Введите размер массива: ";
78     while (!(std::cin >> n) || n <= 0) {
79         std::cout << "Введите корректное положительное число: ";
80         std::cin.clear();
81         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
82     }
83
84     std::vector<double> arr(n);
85     for (int i = 0; i < n; ++i) {
86         std::cout << "Введите элемент " << i + 1 << ": ";
87         while (!(std::cin >> arr[i])) {
88             std::cout << "Введите корректное вещественное число для элемента "
89                       << i + 1 << ": ";
90             std::cin.clear();
91             std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n
    ');
92         }
93     }
94
95     // Исходный массив перед сортировкой
96     std::cout << "Оригинальный массив: ";
97     PrintArray(arr);
98
99     // Сортировка бинарными включениями
100    std::vector<double> arr_binary = arr;
101    auto start = std::chrono::high_resolution_clock::now();
102    BinarySort(arr_binary);
103    auto end = std::chrono::high_resolution_clock::now();
104    std::chrono::duration<double> elapsed = end - start;
105    std::cout << "Массив после сортировки бинарными включениями: ";
106    PrintArray(arr_binary);
107    std::cout << "Время выполнения бинарной сортировки: " << elapsed.count()
108              << " секунд" << endl;
109
110    // Сортировка шейкерная
111    std::vector<double> arr_shaker = arr;
112    start = std::chrono::high_resolution_clock::now();
```

```
113     ShakerSort(arr_shaker);
114     end = std::chrono::high_resolution_clock::now();
115     elapsed = end - start;
116     std::cout << "Массив после шейкерной сортировки: ";
117     PrintArray(arr_shaker);
118     std::cout << "Время выполнения шейкерной сортировки: " << elapsed.count()
119          << " секунд" << std::endl;
120 
121     return 0;
122 }
```

3. Контрольные тесты

3.1 Задание №1: Сортировка

Исходные данные	Результат
10 45 24 1 56 8 9 47 14 20 56	Массив после сортировки бинарными включениями: 1 8 9 14 20 24 45 47 56 56 Время выполнения бинарной сортировки: 3.748e-06 секунд Массив после шейкерной сортировки: 1 8 9 14 20 24 45 47 56 56 Время выполнения шейкерной сортировки: 1.009e-06 секунд
5 8 4 6 2 0	Массив после сортировки бинарными включениями: 0 2 4 6 8 Время выполнения бинарной сортировки: 2.096e-06 секунд Массив после шейкерной сортировки: 0 2 4 6 8 Время выполнения шейкерной сортировки: 5.13e-07 секунд

Таблица 1: Таблица с результатами контрольных тестов Задания №1

4. Контрольные вопросы

4.1 Дайте определение понятию «сортировка»

Это упорядочение элементов множества в некотором порядке, например по возрастанию значений элементов, с целью оптимизации работы с ним, в случае поиска элемента по значению, или анализу (например нахождения медианного значения). Используется в качестве подготовки для анализа данных, поиска и многих других алгоритмов.

4.2 Назовите отличия между внутренней и внешней сортировкой

Внешняя сортировка применяется, когда объём данных, который необходимо отсортировать, превышает объём доступной оперативной памяти. В этом случае для хранения данных используются внешние устройства, такие как жёсткий диск или SSD.

Из-за необходимости чтения и записи данных на внешние носители внешняя сортировка значительно медленнее, чем внутренняя. Однако она позволяет обрабатывать очень большие объёмы данных, недоступные для внутренней сортировки.

Для внешней сортировки требуются специализированные алгоритмы, такие как внешняя сортировка слиянием или многопроходные методы. Эти алгоритмы минимизируют количество операций ввода-вывода, чтобы повысить эффективность процесса.

4.3 Назовите принципы действия сортировки выбором.

Основана на последовательном выборе минимального (или максимального) элемента из неотсортированной части массива и его перемещении в начало (или конец) отсортированной части. Для примера с максимальным элементом:

1. Условно разделяем массив на две части: не отсортированную и пустую отсортированную часть
2. Находим максимальный элемент в неотсортированной части.
3. Перемещаем найденный элемент, меняя его местами с крайним элементом текущей неотсортированной части.
4. Повторяем процесс, постепенно уменьшая размер неотсортированной части.

Алгоритм прост в реализации и не требует дополнительной памяти, так как работает "на месте". Однако его временная сложность $O(n^2)$ делает его неэффективным для работы с большими массивами.

4.4 Назовите принципы действия обменной сортировки

Алгоритм обменной сортировки (он же пузырьковая сортировка) основывается на последовательном сравнении и обмене соседних элементов, если они расположены не в порядке (по убыванию или возрастанию). На первой итерации самый большой (или самый

маленький) элемент "всплывает"и перемещается в конец последовательности. На следующей итерации неотсортированная часть последовательности уменьшается на один элемент. Процесс повторяется, пока размер неотсортированной части не станет равен нулю.

Однако временная сложность алгоритма $O(n^2)$ в худшем и среднем случае делает его неэффективным для работы с большими массивами. Тем не менее, в лучшем случае (при изначально отсортированном массиве) его сложность может быть снижена до $O(n)$.

4.5 Назовите принципы действия шейкерной сортировки

Принцип основан на обменной (пузырьковой) сортировке, но алгоритм проходит последовательность в обоих направлениях, что делает его немного эффективнее:

1. Сначала в прямом порядке, перемещая максимальный элемент в конец последовательности.
2. Затем в обратном порядке, перемещая минимальный элемент в начало.

После каждой итерации область, требующая сортировки, уменьшается на два элемента: максимальный и минимальный элементы занимают свои окончательные позиции. Однако это не решает проблему низкой эффективности, так как временная сложность остаётся $O(n^2)$.

4.6 Назовите принципы действия сортировки вставками

Сортировка вставками упорядочивает массив, последовательно добавляя элементы из неотсортированной части в правильное место в отсортированной части. Последовательность делится на отсортированную (изначально включает только первый элемент) и неотсортированную части. Затем второй и последующие элементы сравниваются с элементами отсортированной части. Все элементы, которые больше (или меньше) текущего, сдвигаются вправо, после чего текущий элемент вставляется на своё место.