ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

« Алгоритмы сортировки »

Выполнил работу

Игнатенок Филипп

Академическая группа №J3111

Принято

Вершинин Владислав

Санкт-Петербург

2024

**Введение**

Цель работы: изучить алгоритмы сортировки и научиться применять их на практике.

Задачи:

1. Выбрать алгоритмы для реализации
2. Ознакомиться с их устройством
3. Изучить особенности реализации этих алгоритмов на С++
4. Реализовать алгоритмы
5. Провести анализ полученных результатов
6. Подготовить отчёт

**Теоретическая подготовка**

### **Odd-even Sort**

Odd-even Sort — это простой алгоритм сортировки, который является разновидностью пузырьковой сортировки. Он работает путём попарного сравнения соседних элементов массива и их обмена, но с чередованием чётных и нечётных индексов на каждом проходе.

Сложность:

* Лучший случай: O(n).
* Худший случай: O(n^2).
* Средний случай: O(n^2).

Память:

* Основной массив: O(n).
* Переменные: O(1).

### **Pancake Sort**

Pancake Sort — это алгоритм сортировки, в котором элементы массива упорядочиваются с помощью переворотов (reversals) подмассивов. Этот процесс напоминает переворачивание блинов в стопке, чтобы самый большой "блин" оказался на дне, а остальные упорядочились сверху вниз.

Сложность:

* Лучший случай: O(n).
* Худший случай: O(n^2).
* Средний случай: O(n^2).

Память:

* Основной массив: O(n).
* Переменные: O(1).

### **Stooge Sort**

Stooge Sort — это рекурсивный алгоритм сортировки, который упорядочивает массив путем повторного частичного упорядочивания подмассивов. Алгоритм называется так в шутку, так как его подход к сортировке крайне неэффективен, но интересен с точки зрения рекурсивных стратегий.

Сложность:

* Лучший случай: O(n^3).
* Худший случай: O(n^3).
* Средний случай: O(n^3).

Память:

* Основной массив: O(n).
* Переменные: O(1).

**Реализация**

### Используемые алгоритмы

В работе реализованы следующие алгоритмы сортировки:

1. Odd-even Sort — сортировка путем попарного сравнения соседних (четных / нечетных) элементов массива.
2. Pancake Sort — алгоритм сортировки, в котором элементы массива упорядочиваются с помощью переворотов (reversals) подмассивов.
3. Stooge Sort — рекурсивный алгоритм сортировки, который упорядочивает массив путем повторного частичного упорядочивания подмассивов.

Реализация выполнена на языке C++ с использованием следующих библиотек:

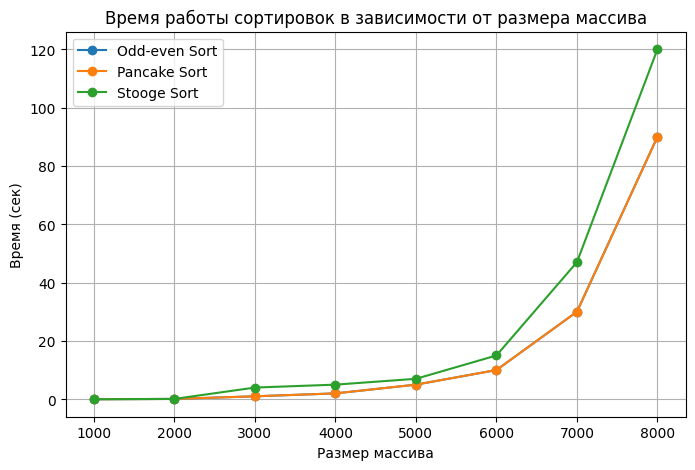
* iostream — ввод/вывод.
* vector — работа с динамическими массивами.
* ctime — замер времени выполнения.
* cstdlib — генерация случайных чисел.
* algorithm — функции стандартной библиотеки.

### Тестирование

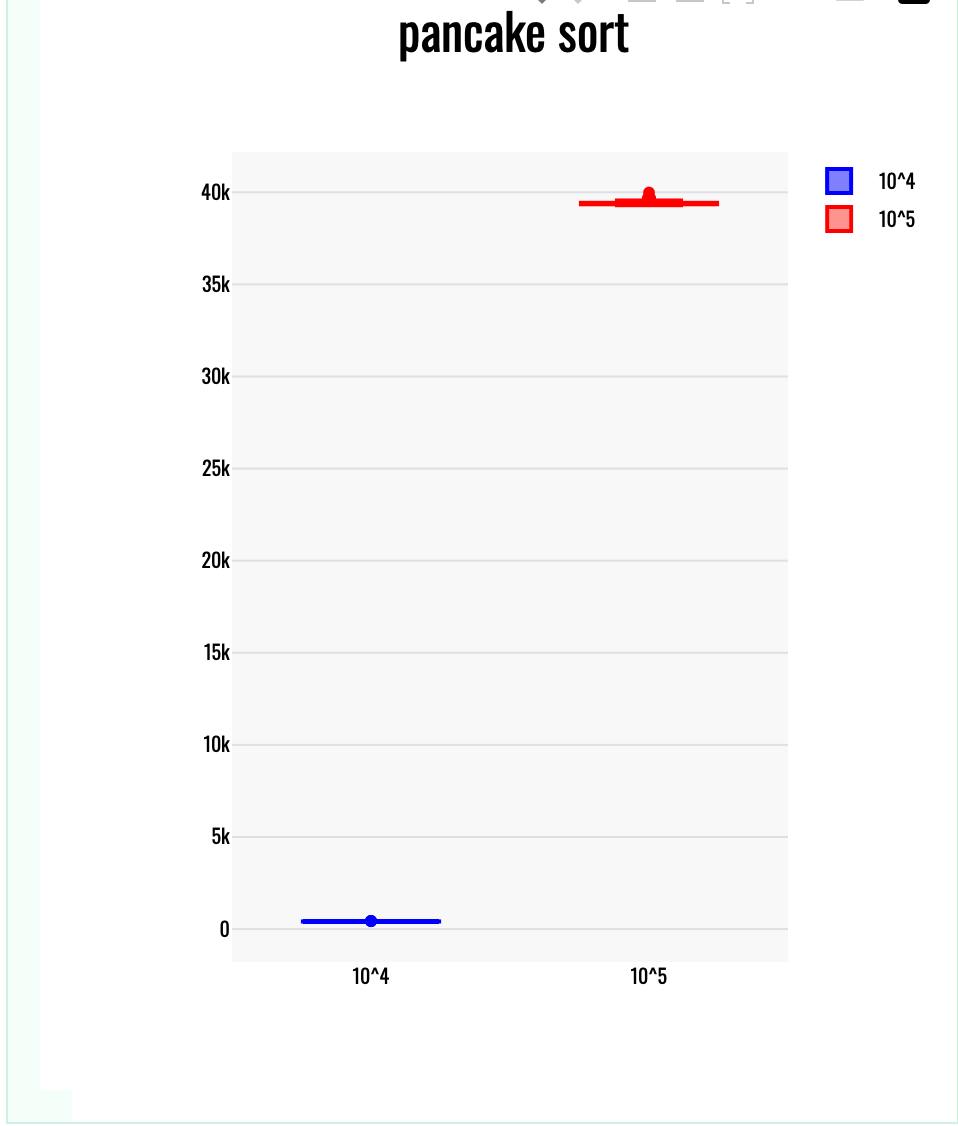
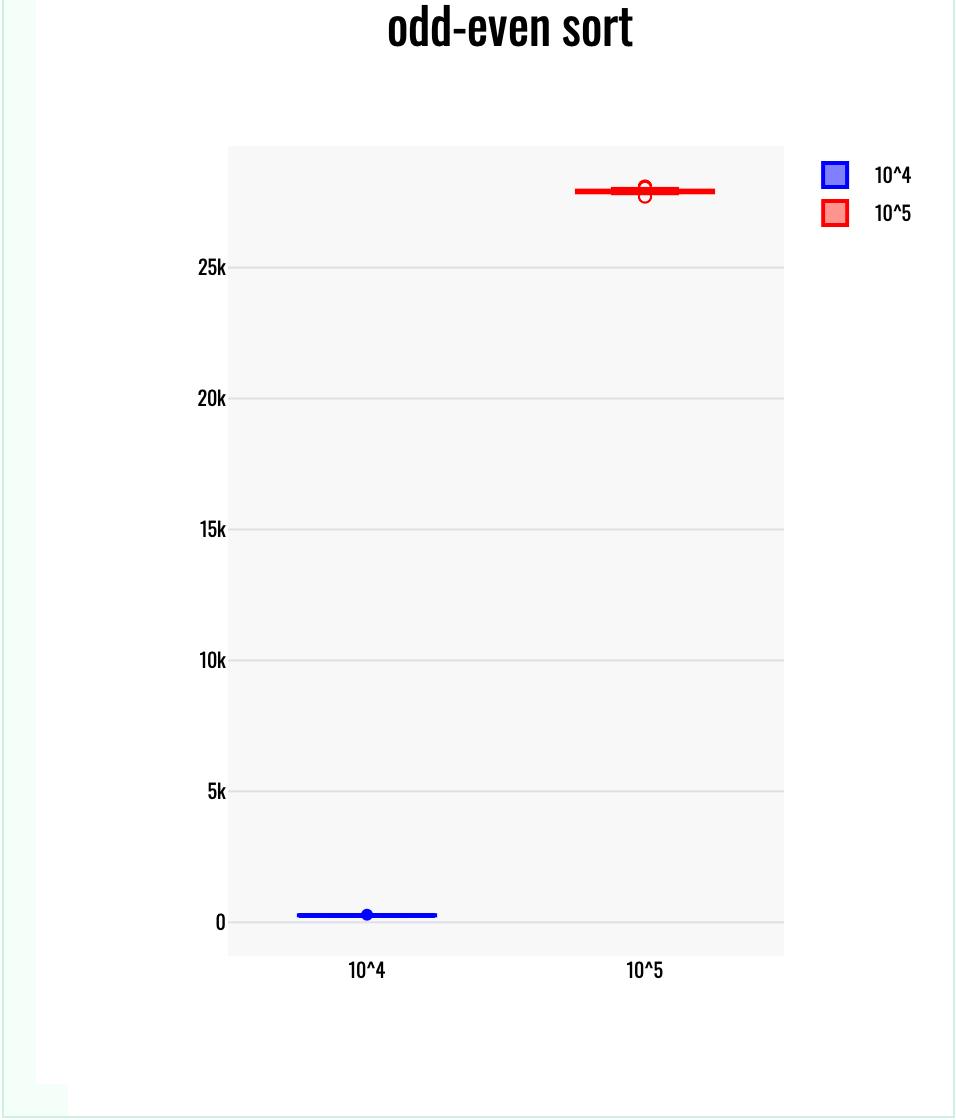
Для каждого алгоритма написаны два набора тестов:

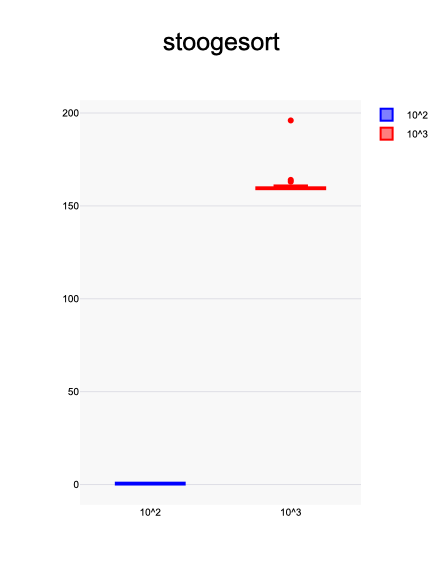
1. Тесты корректности: Проверка, что алгоритм возвращает отсортированный массив.
2. Тесты на производительность: Замер времени выполнения на массивах разной длины.

**Экспериментальная часть**

Затраченное время для каждой сортировки в зависимости от размера массива:  


Распределения затраченного времени для размера 10\*\*4 и 10\*\*5:

  
Распределения затраченного времени для размера 10\*\*2 и 10\*\*3 (Stooge Sort из-за неоптимальности алгоритма):



**Заключение**

В ходе работы были реализованы три алгоритма сортировки: **Odd-Even Sort**, **Pancake Sort** и **Stooge Sort**. Целью было исследование их производительности на массивах различного размера, чтобы выявить особенности их работы и определить их эффективность.

#### Выбросы и их причины:

* TreeSort:

Выбросы наблюдаются на сильно упорядоченных или полностью отсортированных данных, так как дерево становится не сбалансированным, что увеличивает время обработки.

* CountingSort:

Для массивов с большим диапазоном значений время работы увеличивается из-за затрат на инициализацию и обработку массивов частот. Это особенно заметно при наличии редких, но больших значений.

* GnomeSort:

Замедление фиксируется на случайных данных большой длины, так как алгоритм выполняет множество итераций

**Приложения**

#include <iostream>

#include <vector>

#include <algorithm> // Для std::reverse и std::max\_element

#include <cstdlib> // Для rand и srand

#include <ctime> // Для времени

using namespace std;

// Odd-Even Sort

void oddEvenSort(vector<int>& arr) {

bool isSorted = false;

int n = arr.size();

while (!isSorted) {

isSorted = true;

// Чётный проход

for (int i = 0; i < n - 1; i += 2) {

if (arr[i] > arr[i + 1]) {

swap(arr[i], arr[i + 1]);

isSorted = false;

}

}

// Нечётный проход

for (int i = 1; i < n - 1; i += 2) {

if (arr[i] > arr[i + 1]) {

swap(arr[i], arr[i + 1]);

isSorted = false;

}

}

}

}

// Pancake Sort

void flip(vector<int>& arr, int k) {

reverse(arr.begin(), arr.begin() + k);

}

void pancakeSort(vector<int>& arr) {

int n = arr.size();

for (int size = n; size > 1; --size) {

// Найти индекс максимального элемента в текущем подмассиве

int maxIdx = max\_element(arr.begin(), arr.begin() + size) - arr.begin();

// Перевернуть так, чтобы максимальный элемент оказался в начале

if (maxIdx != size - 1) {

if (maxIdx > 0) {

flip(arr, maxIdx + 1);

}

// Перевернуть так, чтобы максимальный элемент оказался на своём месте

flip(arr, size);

}

}

}

// Stooge Sort

void stoogeSort(vector<int>& arr, int l, int r) {

if (arr[l] > arr[r]) {

swap(arr[l], arr[r]);

}

if (r - l > 1) {

int t = (r - l + 1) / 3;

// Рекурсивно сортировать первую 2/3, последние 2/3 и снова первую 2/3

stoogeSort(arr, l, r - t);

stoogeSort(arr, l + t, r);

stoogeSort(arr, l, r - t);

}

}

// Функция для генерации случайного массива

vector<int> generateRandomArray(int size, int minVal = 1, int maxVal = 100) {

vector<int> arr(size);

for (int i = 0; i < size; ++i) {

arr[i] = rand() % (maxVal - minVal + 1) + minVal;

}

return arr;

}

// Функция для вывода массива

void printArray(const vector<int>& arr) {

for (int x : arr) {

cout << x << " ";

}

cout << endl;

}

// Основная программа с тестами

int main() {

srand(time(0)); // Инициализация генератора случайных чисел

// Размеры массивов для тестов

vector<int> sizes = {10, 100};

for (int size : sizes) {

cout << "Тестирование на массиве длинной: " << size << endl;

// Генерация случайного массива

vector<int> originalArray = generateRandomArray(size);

// Odd-Even Sort

vector<int> arr1 = originalArray;

cout << "Изначальный массив: ";

printArray(arr1);

oddEvenSort(arr1);

cout << "Odd-Even Sort: ";

printArray(arr1);

// Pancake Sort

vector<int> arr2 = originalArray;

pancakeSort(arr2);

cout << "Pancake Sort: ";

printArray(arr2);

// Stooge Sort

vector<int> arr3 = originalArray;

stoogeSort(arr3, 0, arr3.size() - 1);

cout << "Stooge Sort: ";

printArray(arr3);

cout << "-------------------------------------" << endl;

if (arr3 == arr2 and arr3 == arr1) {

cout << "Все алгоритмы отработали успешно ";

}

}

return 0;

}