Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 1

Выполнил студент группы КС-36 Сары Кристина Ивановна

Ссылка на репозиторий: https://github.com/MUCTR-IKT-CPP/SaryKI\_36\_ALG.git

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: 17.02.2025

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

В рамках лабораторной работы необходимо изучить и реализовать метод сортировки выбором (4 вариант).

В лабораторной работе предлагается изучить способ анализа алгоритма связанный со временем. Рассмотреть для выбранного алгоритма сортировки наилучшие, наихудшее и среднее время и соотнести его с известным для алгоритма показателем эффективности O-большое.

Допускает реализация задания на любом языке программирования, кроме лиспоподобных. Преподаватель может не знать конкретного языка реализации, поэтому вы должны быть способны объяснить алгоритм и нарисовать его без демонстрации непосредственно вашего кода.

Задание:

* Реализовать метод сортировки, соответствующий вашему варианту:

1. Сортировка выбором

* Реализовать проведения тестирования алгоритма сериями расчетов для измерения параметров времени.  
  За один расчет выполняется следующие операции:
  1. Генерируется массив случайных значений
  2. Запоминается время начала расчета алгоритма сортировки
  3. Выполняется алгоритм сортировки
  4. Во время выполнения измерить количество повторных прохождений по массиву.
  5. Во время выполнения измерить количество выполнения операций обмена значений.
  6. Вычисляется время, затраченное на сортировку: текущее время - время начала
  7. Сохраняется время для одной попытки После этого расчет повторяется до окончания серии.
  8. Алгоритм вычисляется 8 сериями по 20 раз за серию.
  9. Алгоритм в каждой серии вычисляется для массива размером M. (1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000)
  10. Массив заполняется значения числами с плавающей запятой в интервале от -1 до 1.
  11. Для серии запоминаются все времена которые были замерены
* По полученным данным времени построить графики зависимости времени от числа элементов в массиве:
  1. Совмешеееый график наихудшего времени выполнения сортировки и сложности алгоритма указанной в нотации O большое.

Для построения графика вычисляется O большое для каждого размера массива. При этом при вычислении функции O(c \* g(N)) подбирается такая константа с, что бы при значении > 1000 график O(N) был выше графика наихудшего случая, но второй график на его фоне не превращался в прямую линию

* 1. Совмещенный график среднего, наилучшего и наихудшего времени исполнения.
  2. График среднего колличества обмена значений.
  3. График повторных обходов массива.
* По результатом расчетов оформляется отчет по предоставленной форме, в отчете:
  1. Приводится описание алгоритма.
  2. Приводится описания выполнения задачи (Описание кода и спецефических элементов реализации)
  3. Приводятся выводы (Графики и их анализ)

# Описание метода/модели.

Сортировка выбором (Selection Sort) - На i-ом шаге алгоритма находим минимальный среди последних n−i+1, и меняем его местами с i-ым элементом в массиве. Алгоритм основам на последовательном выборе минимального элемента и его перемещении в отсортированную часть массива. Алгоритм состоит из следующих шагов:

1. Находим минимальный элемент в неотсортированной части массива.
2. Меняем его местами с первым элементом неотсортированной части.
3. Повторяем процесс для оставшейся части массива, исключая уже отсортированные элементы.

Алгоритм относится к категории алгоритмов сортировки на месте, так как не требует дополнительной памяти для хранения элементов (кроме временной переменной для обмена значений)

**Анализ сложности выполнения сортировки выбором:**

* **Лучший случай (O(n2))**: Даже если массив уже отсортирован, алгоритм все равно проходит по всему массиву дважды.
* **Средний случай (O(n2))**: В среднем случае также выполняются n(n-1)/2 сравнений.
* **Худший случай (O(n2))**: Если массив отсортирован в обратном порядке, количество операций остается таким же.

**Уровень потребляемой памяти**

Сортировка выбором является алгоритмом "на месте" (in-place), то есть не требует дополнительной памяти, кроме временной переменной для обмена элементов. Это делает его эффективным с точки зрения использования памяти (O(1)).

**Преимущества и недостатки алгоритма**

**Преимущества:**

* Простота реализации.
* Отсутствие необходимости в дополнительной памяти.
* Стабильное поведение независимо от изначального порядка элементов.

**Недостатки:**

* Низкая скорость на больших объемах данных.
* Квадратичная сложность делает его менее эффективным по сравнению с более сложными алгоритмами, такими как быстрая сортировка (Quick Sort) или сортировка слиянием (Merge Sort).
* Даже если массив уже отсортирован, выполняется полное количество итераций.

# Выполнение задачи.

Программа реализована на языке C++.

**Организация программы**

Программа включает в себя следующие основные компоненты:

* Функция selectionSort, реализующая алгоритм сортировки выбором и подсчитывающая количество проходов и перестановок.
* Функция runSortingTest, выполняющая тестирование сортировки для различных размеров массива и записывающая результаты в файл.
* Основная функция main, которая организует тестирование на нескольких размерах массивов и записывает результаты в CSV-файл.

**Проведенные тесты**

Программа тестировалась на массивах различных размеров: 1000, 2000, 4000, 8000, 16000, 32000, 64000 и 128000 элементов. Для каждого размера выполнялось по 20 прогонов с генерацией случайных чисел в диапазоне от -1 до 1.

Код:

#include <iostream>

#include <vector>

#include <random>

#include <chrono>

#include <fstream>

#include <algorithm>

using namespace std;

using namespace chrono;

/\*\*

\* Функция сортировки методом выбора.

\* Выполняет сортировку массива и подсчитывает количество проходов и перестановок.

\*

\* @param arr Вектор чисел для сортировки.

\* @param passCount Счетчик количества проходов по массиву.

\* @param swapCount Счетчик количества перестановок элементов.

\*

\* @return Нет, результат сортировки происходит по месту.

\*/

void selectionSort(vector<double>& arr, long long& passCount, long long& swapCount) {

int n = arr.size(); // Размер массива

passCount = 0; // Инициализация счетчика количества проходов

swapCount = 0; // Инициализация счетчика количества перестановок

// Внешний цикл, проходящий по всем элементам массива

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

int minIndex = i; // Индекс минимального элемента в текущем проходе

// Внутренний цикл для поиска минимального элемента в оставшейся части массива

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

++passCount; // Увеличиваем счетчик проходов

if (arr[j] < arr[minIndex]) {

minIndex = j; // Обновляем индекс минимального элемента

}

}

// Если минимальный элемент не на текущей позиции, выполняем обмен

if (minIndex != i) {

swap(arr[i], arr[minIndex]); // Меняем местами элементы

++swapCount; // Увеличиваем счетчик перестановок

}

}

}

/\*\*

\* Функция для тестирования сортировки методом выбора для нескольких массивов.

\* Выполняет несколько прогонов для массива каждого размера и записывает результаты в файл.

\*

\* @param arraySize Размер массива для сортировки.

\* @param numRuns Количество прогонов для каждого размера массива.

\* @param outFile Поток вывода, в который записываются результаты.

\*/

void runSortingTest(int arraySize, int numRuns, ofstream& outFile) {

mt19937 engine(time(0)); // Генератор случайных чисел, инициализируем его временем

uniform\_real\_distribution<double> gen(-1.0, 1.0); // Диапазон случайных чисел от -1 до 1

vector<double> times; // Вектор для хранения времени выполнения для каждого прогона

double minTime = numeric\_limits<double>::max(); // Инициализация минимального времени

double maxTime = 0; // Инициализация максимального времени

double totalTime = 0; // Общее время всех прогонов

long long totalPassCount = 0; // Общее количество проходов

long long totalSwapCount = 0; // Общее количество перестановок

// Многократный прогон сортировки для получения статистики

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

vector<double> arr(arraySize); // Создаем массив заданного размера

// Заполняем массив случайными числами

for (auto& el : arr) {

el = gen(engine);

}

long long passCount = 0, swapCount = 0; // Локальные счетчики для текущего прогона

auto start = high\_resolution\_clock::now(); // Засекаем время начала сортировки

selectionSort(arr, passCount, swapCount); // Выполняем сортировку

auto end = high\_resolution\_clock::now(); // Засекаем время окончания сортировки

// Вычисляем время выполнения текущего прогона

duration<double, milli> elapsed = end - start;

times.push\_back(elapsed.count()); // Добавляем время в список

totalTime += elapsed.count(); // Обновляем общее время

minTime = min(minTime, elapsed.count()); // Обновляем минимальное время

maxTime = max(maxTime, elapsed.count()); // Обновляем максимальное время

totalPassCount += passCount; // Добавляем количество проходов

totalSwapCount += swapCount; // Добавляем количество перестановок

// Выводим результаты для текущего прогона

cout << "Run " << i + 1 << " - Time: " << elapsed.count() << " ms, Passes: " << passCount << ", Swaps: " << swapCount << endl;

outFile << arraySize << "," << elapsed.count() << "," << passCount << "," << swapCount << endl; // Записываем результаты в файл

}

// Рассчитываем средние значения

double avgTime = totalTime / numRuns;

double avgPasses = totalPassCount / numRuns;

double avgSwaps = totalSwapCount / numRuns;

// Записываем статистику в файл

outFile << arraySize << ",AVG," << avgTime << "," << avgPasses << "," << avgSwaps << endl;

outFile << arraySize << ",MIN," << minTime << ",0,0" << endl;

outFile << arraySize << ",MAX," << maxTime << ",0,0" << endl;

// Сообщаем, что завершили тестирование для текущего размера массива

cout << "\nCompleted series for array size " << arraySize << "\n";

}

int main() {

// Массив размеров для тестирования

vector<int> sizes = {1000, 2000, 4000, 8000, 16000, 32000, 64000, 128000};

int numRuns = 20; // Количество прогонов для каждого размера массива

// Открываем файл для записи результатов

ofstream outFile("sorting\_results.csv");

outFile << "ArraySize,Type,Time(ms),Passes,Swaps" << endl; // Записываем заголовки в файл

// Запускаем тесты сортировки для каждого размера массива

for (int size : sizes) {

runSortingTest(size, numRuns, outFile); // Запуск теста для данного размера массива

cout << "--------------------------------------\n"; // Разделитель для каждого теста

}

// Закрываем файл после завершения всех тестов

outFile.close();

return 0; // Завершаем программу

}

**Полученные результаты:**

Run 1 - Time: 1.7603 ms, Passes: 499500, Swaps: 992

Run 2 - Time: 1.7361 ms, Passes: 499500, Swaps: 992

Run 3 - Time: 1.7168 ms, Passes: 499500, Swaps: 990

Run 4 - Time: 1.7266 ms, Passes: 499500, Swaps: 990

Run 5 - Time: 1.7589 ms, Passes: 499500, Swaps: 993

Run 6 - Time: 1.7997 ms, Passes: 499500, Swaps: 992

Run 7 - Time: 1.7201 ms, Passes: 499500, Swaps: 991

Run 8 - Time: 1.7066 ms, Passes: 499500, Swaps: 991

Run 9 - Time: 1.7092 ms, Passes: 499500, Swaps: 993

Run 10 - Time: 1.8533 ms, Passes: 499500, Swaps: 998

Run 11 - Time: 1.753 ms, Passes: 499500, Swaps: 992

Run 12 - Time: 1.7176 ms, Passes: 499500, Swaps: 996

Run 13 - Time: 1.7934 ms, Passes: 499500, Swaps: 993

Run 14 - Time: 1.9288 ms, Passes: 499500, Swaps: 991

Run 15 - Time: 1.8427 ms, Passes: 499500, Swaps: 994

Run 16 - Time: 1.8444 ms, Passes: 499500, Swaps: 996

Run 17 - Time: 1.718 ms, Passes: 499500, Swaps: 988

Run 18 - Time: 2.8464 ms, Passes: 499500, Swaps: 993

Run 19 - Time: 3.2211 ms, Passes: 499500, Swaps: 990

Run 20 - Time: 1.9677 ms, Passes: 499500, Swaps: 991

Completed series for array size 1000

--------------------------------------

Run 1 - Time: 8.6999 ms, Passes: 1999000, Swaps: 1993

Run 2 - Time: 7.6565 ms, Passes: 1999000, Swaps: 1991

Run 3 - Time: 7.5818 ms, Passes: 1999000, Swaps: 1993

Run 4 - Time: 8.2646 ms, Passes: 1999000, Swaps: 1992

Run 5 - Time: 9.5158 ms, Passes: 1999000, Swaps: 1993

Run 6 - Time: 9.6864 ms, Passes: 1999000, Swaps: 1993

Run 7 - Time: 9.1517 ms, Passes: 1999000, Swaps: 1984

Run 8 - Time: 8.8655 ms, Passes: 1999000, Swaps: 1988

Run 9 - Time: 8.835 ms, Passes: 1999000, Swaps: 1993

Run 10 - Time: 8.4272 ms, Passes: 1999000, Swaps: 1989

Run 11 - Time: 10.1294 ms, Passes: 1999000, Swaps: 1994

Run 12 - Time: 10.1691 ms, Passes: 1999000, Swaps: 1992

Run 13 - Time: 8.4003 ms, Passes: 1999000, Swaps: 1990

Run 14 - Time: 10.6472 ms, Passes: 1999000, Swaps: 1988

Run 15 - Time: 9.5055 ms, Passes: 1999000, Swaps: 1988

Run 16 - Time: 9.664 ms, Passes: 1999000, Swaps: 1988

Run 17 - Time: 9.3811 ms, Passes: 1999000, Swaps: 1993

Run 18 - Time: 10.3088 ms, Passes: 1999000, Swaps: 1988

Run 19 - Time: 8.2185 ms, Passes: 1999000, Swaps: 1995

Run 20 - Time: 7.7389 ms, Passes: 1999000, Swaps: 1992

Completed series for array size 2000

--------------------------------------

Run 1 - Time: 37.1461 ms, Passes: 7998000, Swaps: 3989

Run 2 - Time: 30.362 ms, Passes: 7998000, Swaps: 3989

Run 3 - Time: 34.3397 ms, Passes: 7998000, Swaps: 3996

Run 4 - Time: 29.8732 ms, Passes: 7998000, Swaps: 3994

Run 5 - Time: 41.4825 ms, Passes: 7998000, Swaps: 3991

Run 6 - Time: 33.0688 ms, Passes: 7998000, Swaps: 3992

Run 7 - Time: 32.7323 ms, Passes: 7998000, Swaps: 3996

Run 8 - Time: 33.933 ms, Passes: 7998000, Swaps: 3992

Run 9 - Time: 33.949 ms, Passes: 7998000, Swaps: 3988

Run 10 - Time: 34.0478 ms, Passes: 7998000, Swaps: 3986

Run 11 - Time: 32.3614 ms, Passes: 7998000, Swaps: 3988

Run 12 - Time: 32.9792 ms, Passes: 7998000, Swaps: 3992

Run 13 - Time: 35.4914 ms, Passes: 7998000, Swaps: 3990

Run 14 - Time: 33.2857 ms, Passes: 7998000, Swaps: 3993

Run 15 - Time: 33.1269 ms, Passes: 7998000, Swaps: 3990

Run 16 - Time: 34.6348 ms, Passes: 7998000, Swaps: 3995

Run 17 - Time: 33.9432 ms, Passes: 7998000, Swaps: 3991

Run 18 - Time: 33.9821 ms, Passes: 7998000, Swaps: 3992

Run 19 - Time: 31.4991 ms, Passes: 7998000, Swaps: 3989

Run 20 - Time: 31.2549 ms, Passes: 7998000, Swaps: 3993

Completed series for array size 4000

--------------------------------------

Run 1 - Time: 106.153 ms, Passes: 31996000, Swaps: 7988

Run 2 - Time: 152.272 ms, Passes: 31996000, Swaps: 7993

Run 3 - Time: 121.024 ms, Passes: 31996000, Swaps: 7989

Run 4 - Time: 125.642 ms, Passes: 31996000, Swaps: 7993

Run 5 - Time: 135.779 ms, Passes: 31996000, Swaps: 7992

Run 6 - Time: 151.903 ms, Passes: 31996000, Swaps: 7995

Run 7 - Time: 123.653 ms, Passes: 31996000, Swaps: 7988

Run 8 - Time: 126.091 ms, Passes: 31996000, Swaps: 7991

Run 9 - Time: 135.331 ms, Passes: 31996000, Swaps: 7994

Run 10 - Time: 118.578 ms, Passes: 31996000, Swaps: 7987

Run 11 - Time: 122.529 ms, Passes: 31996000, Swaps: 7995

Run 12 - Time: 117.459 ms, Passes: 31996000, Swaps: 7989

Run 13 - Time: 126.386 ms, Passes: 31996000, Swaps: 7993

Run 14 - Time: 117.772 ms, Passes: 31996000, Swaps: 7994

Run 15 - Time: 114.327 ms, Passes: 31996000, Swaps: 7989

Run 16 - Time: 108.223 ms, Passes: 31996000, Swaps: 7995

Run 17 - Time: 108.367 ms, Passes: 31996000, Swaps: 7993

Run 18 - Time: 108.627 ms, Passes: 31996000, Swaps: 7987

Run 19 - Time: 120.309 ms, Passes: 31996000, Swaps: 7994

Run 20 - Time: 140.605 ms, Passes: 31996000, Swaps: 7992

Completed series for array size 8000

--------------------------------------

Run 1 - Time: 427.111 ms, Passes: 127992000, Swaps: 15991

Run 2 - Time: 423.245 ms, Passes: 127992000, Swaps: 15985

Run 3 - Time: 422.084 ms, Passes: 127992000, Swaps: 15987

Run 4 - Time: 422.037 ms, Passes: 127992000, Swaps: 15993

Run 5 - Time: 422.167 ms, Passes: 127992000, Swaps: 15989

Run 6 - Time: 421.786 ms, Passes: 127992000, Swaps: 15992

Run 7 - Time: 421.266 ms, Passes: 127992000, Swaps: 15990

Run 8 - Time: 420.849 ms, Passes: 127992000, Swaps: 15987

Run 9 - Time: 453.803 ms, Passes: 127992000, Swaps: 15994

Run 10 - Time: 421.565 ms, Passes: 127992000, Swaps: 15991

Run 11 - Time: 426.637 ms, Passes: 127992000, Swaps: 15992

Run 12 - Time: 430.832 ms, Passes: 127992000, Swaps: 15989

Run 13 - Time: 425.827 ms, Passes: 127992000, Swaps: 15988

Run 14 - Time: 423.202 ms, Passes: 127992000, Swaps: 15990

Run 15 - Time: 422.162 ms, Passes: 127992000, Swaps: 15989

Run 16 - Time: 427.784 ms, Passes: 127992000, Swaps: 15983

Run 17 - Time: 433.625 ms, Passes: 127992000, Swaps: 15992

Run 18 - Time: 424.773 ms, Passes: 127992000, Swaps: 15991

Run 19 - Time: 427.671 ms, Passes: 127992000, Swaps: 15994

Run 20 - Time: 424.968 ms, Passes: 127992000, Swaps: 15992

Completed series for array size 16000

--------------------------------------

Run 1 - Time: 1713.97 ms, Passes: 511984000, Swaps: 31986

Run 2 - Time: 1713.16 ms, Passes: 511984000, Swaps: 31993

Run 3 - Time: 1737.86 ms, Passes: 511984000, Swaps: 31985

Run 4 - Time: 1737.83 ms, Passes: 511984000, Swaps: 31991

Run 5 - Time: 1684.72 ms, Passes: 511984000, Swaps: 31988

Run 6 - Time: 1687.62 ms, Passes: 511984000, Swaps: 31985

Run 7 - Time: 1694.05 ms, Passes: 511984000, Swaps: 31989

Run 8 - Time: 1710.06 ms, Passes: 511984000, Swaps: 31992

Run 9 - Time: 1723.78 ms, Passes: 511984000, Swaps: 31988

Run 10 - Time: 1704.25 ms, Passes: 511984000, Swaps: 31992

Run 11 - Time: 1701.56 ms, Passes: 511984000, Swaps: 31991

Run 12 - Time: 1710.02 ms, Passes: 511984000, Swaps: 31990

Run 13 - Time: 1756.52 ms, Passes: 511984000, Swaps: 31986

Run 14 - Time: 1701.69 ms, Passes: 511984000, Swaps: 31986

Run 15 - Time: 1710.93 ms, Passes: 511984000, Swaps: 31990

Run 16 - Time: 1700.34 ms, Passes: 511984000, Swaps: 31988

Run 17 - Time: 1716.61 ms, Passes: 511984000, Swaps: 31990

Run 18 - Time: 1729.22 ms, Passes: 511984000, Swaps: 31988

Run 19 - Time: 1704.34 ms, Passes: 511984000, Swaps: 31990

Run 20 - Time: 1699.53 ms, Passes: 511984000, Swaps: 31991

Completed series for array size 32000

--------------------------------------

Run 1 - Time: 6965.03 ms, Passes: 2047968000, Swaps: 63993

Run 2 - Time: 6876.48 ms, Passes: 2047968000, Swaps: 63988

Run 3 - Time: 6892.69 ms, Passes: 2047968000, Swaps: 63986

Run 4 - Time: 6915.02 ms, Passes: 2047968000, Swaps: 63988

Run 5 - Time: 6912.79 ms, Passes: 2047968000, Swaps: 63986

Run 6 - Time: 7163.87 ms, Passes: 2047968000, Swaps: 63984

Run 7 - Time: 6979.92 ms, Passes: 2047968000, Swaps: 63990

Run 8 - Time: 6914.45 ms, Passes: 2047968000, Swaps: 63992

Run 9 - Time: 7117.87 ms, Passes: 2047968000, Swaps: 63992

Run 10 - Time: 6903.07 ms, Passes: 2047968000, Swaps: 63991

Run 11 - Time: 6927.59 ms, Passes: 2047968000, Swaps: 63989

Run 12 - Time: 6964.32 ms, Passes: 2047968000, Swaps: 63985

Run 13 - Time: 7095.57 ms, Passes: 2047968000, Swaps: 63982

Run 14 - Time: 7065.56 ms, Passes: 2047968000, Swaps: 63988

Run 15 - Time: 6830.11 ms, Passes: 2047968000, Swaps: 63989

Run 16 - Time: 6814.39 ms, Passes: 2047968000, Swaps: 63984

Run 17 - Time: 6938.35 ms, Passes: 2047968000, Swaps: 63993

Run 18 - Time: 7008.69 ms, Passes: 2047968000, Swaps: 63988

Run 19 - Time: 6992.25 ms, Passes: 2047968000, Swaps: 63985

Run 20 - Time: 6916.44 ms, Passes: 2047968000, Swaps: 63991

Completed series for array size 64000

--------------------------------------

Run 1 - Time: 27727.1 ms, Passes: 8191936000, Swaps: 127990

Run 2 - Time: 27633.6 ms, Passes: 8191936000, Swaps: 127990

Run 3 - Time: 27237.3 ms, Passes: 8191936000, Swaps: 127984

Run 4 - Time: 27172.8 ms, Passes: 8191936000, Swaps: 127990

Run 5 - Time: 27167.7 ms, Passes: 8191936000, Swaps: 127988

Run 6 - Time: 27131.8 ms, Passes: 8191936000, Swaps: 127990

Run 7 - Time: 27320.8 ms, Passes: 8191936000, Swaps: 127984

Run 8 - Time: 27228.1 ms, Passes: 8191936000, Swaps: 127991

Run 9 - Time: 27287.8 ms, Passes: 8191936000, Swaps: 127991

Run 10 - Time: 27405.3 ms, Passes: 8191936000, Swaps: 127988

Run 11 - Time: 27942.2 ms, Passes: 8191936000, Swaps: 127988

Run 12 - Time: 27707.8 ms, Passes: 8191936000, Swaps: 127991

Run 13 - Time: 27733.3 ms, Passes: 8191936000, Swaps: 127983

Run 14 - Time: 27554.9 ms, Passes: 8191936000, Swaps: 127983

Run 15 - Time: 27632.5 ms, Passes: 8191936000, Swaps: 127992

Run 16 - Time: 27424.5 ms, Passes: 8191936000, Swaps: 127986

Run 17 - Time: 27634.9 ms, Passes: 8191936000, Swaps: 127986

Run 18 - Time: 27605.9 ms, Passes: 8191936000, Swaps: 127983

Run 19 - Time: 27558.2 ms, Passes: 8191936000, Swaps: 127987

Run 20 - Time: 27767.9 ms, Passes: 8191936000, Swaps: 127988

Completed series for array size 128000

--------------------------------------

Process finished with exit code 0

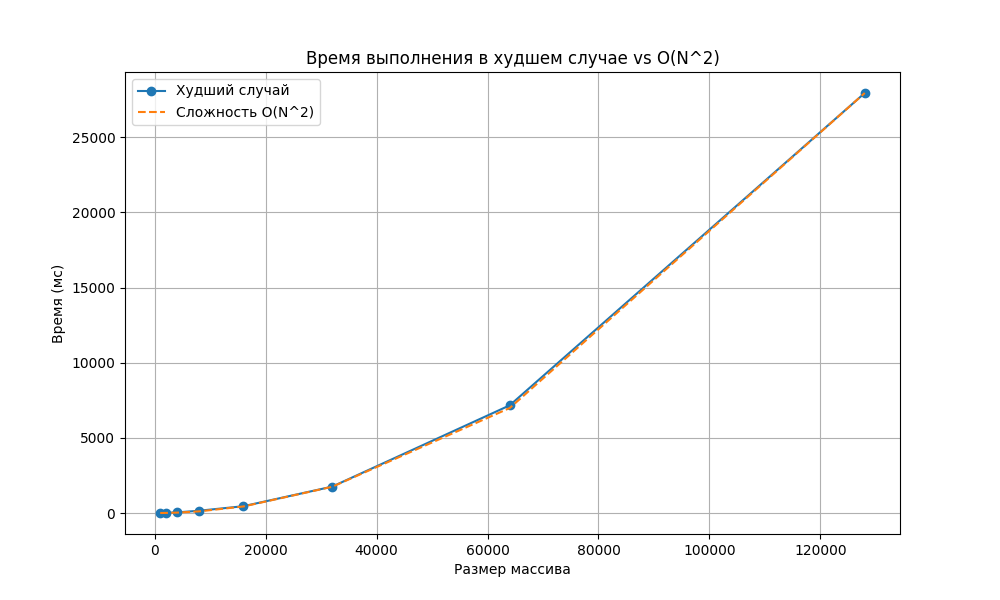
Графики для анализа полученных результатов построены при помощи библиотеки matplotlib языка Python

Рисунок 1 — Совмещённый график наихудшего времени выполнения сортировки и сложности алгоритма указанной в нотации O большое.

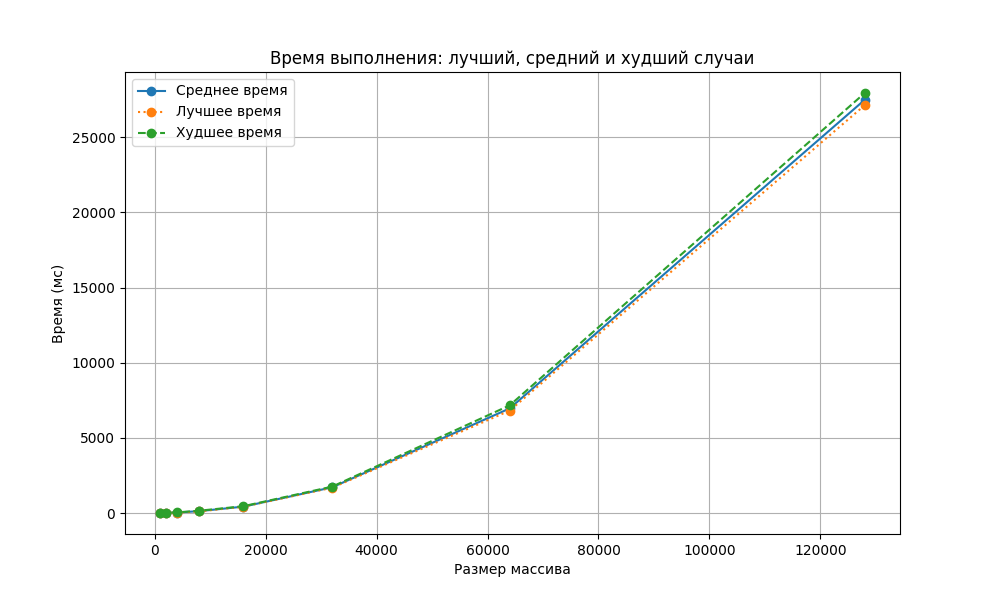


Рисунок 2 — Совмещённый график среднего, наилучшего и наихудшего времени исполнения

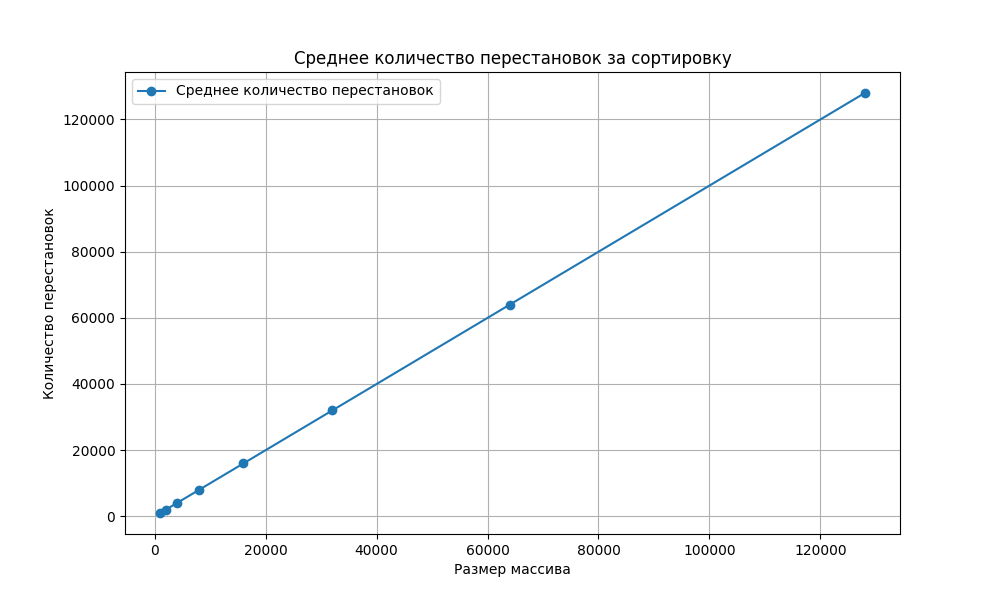


Рисунок 3 — График среднего количества обмена значений

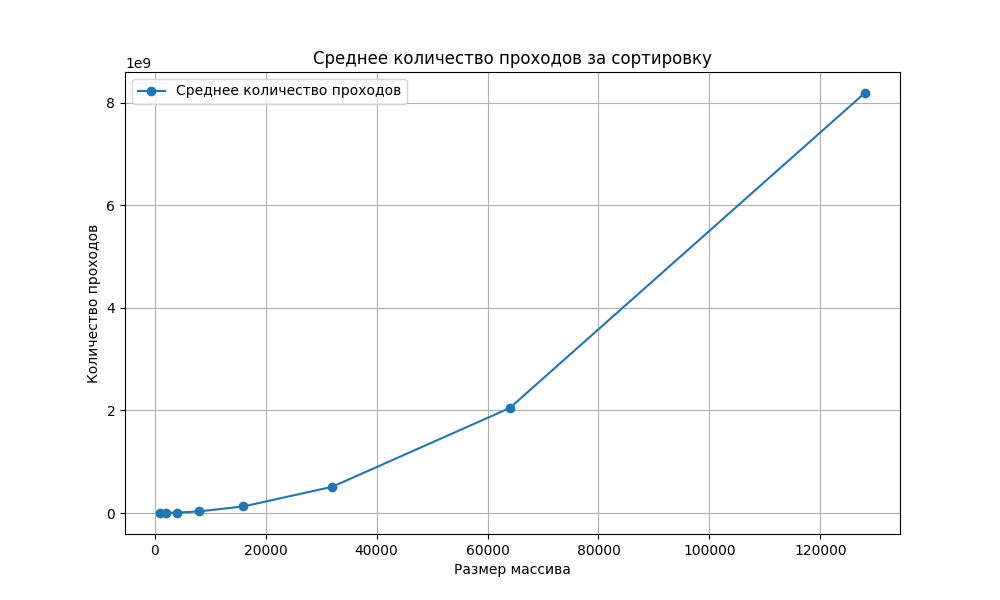


Рисунок 4 — График повторных обходов массива

# Заключение.

В ходе выполнения лабораторной работы были сделаны следующие выводы:

Сортировка выбором, несмотря на простоту, оказывается неэффективной для работы с большими объемами данных.

Результаты тестирования подтвердили квадратичную зависимость времени выполнения от размера массива, что делает алгоритм непрактичным для реальных задач с большими данными.

Однако он может быть полезен в ситуациях, когда размер массива мал или когда требуется реализовать алгоритм с минимальным потреблением памяти.

Количество проходов всегда равно n−1n-1n−1, а количество перестановок зависит от порядка исходных данных, но не превышает n−1n-1n−1.

При тестировании на случайных массивах с элементами от -1 до 1 наблюдалось значительное увеличение времени сортировки при переходе к массивам размером 32 000 элементов и выше.