**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра САПР**

**ОТЧЁТ**

**по лабораторной работе №3**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Сортировка**

Студент гр. 0322 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Руссу В.А.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Пестерев Д.О.

Санкт-Петербург

2022

# Задача

Реализовать следующие алгоритмы сортировки:

1. Сортировка вставками (Insertion sort)
2. Сортировка выбором (Selection sort)
3. Пузырьковая сортировка (Bubble sort)
4. Сортировка слиянием (Merge sort)
5. Сортировка Шелла (Shell sort)
6. Быстрая сортировка (Quick sort)

# Описание реализуемого класса и методов

void show(int\* arr, int length) – вывод массива.

void InsertionSort(int\* copy\_array, int length) – сортировка вставками

void swap (int\* xp, int\* yp) – меняет значения двух принимаемых величин местами

void SelectionSort(int\* copy\_array, int length) – сортировка выбором

void BubbleSort(int\* copy\_array, int length) – сортировка пузырьком

void merge(int\* copy\_array, int l, int m, int length) и void MergeSort(int\* copy\_array, int l, int length) – сортировка слиянием

void ShellSort(int\* copy\_array, int length) – сортировка Шелла

int partition(int\* copy\_array, int low, int high) – вспомогательная функция для быстрой сортировки

void QuickSort(int\* copy\_array, int low, int high) – быстрая сортировка

Лучший случай (массив отсортирован)

Худший случай (массив отсортирован в обратном порядке)

Средний случай (значения стоят в хаотичном порядке)

Пространственная сложность (сколько дополнительной памяти требует алгоритм)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название\Асимптотическая сложность | Лучший случай | Средний случай | Худший случай | Пространственная сложность |
| Вставками | O(n) | O(n2) | O(n2) | O(1) |
| Выбором | O(n2) | O(n2) | O(n2) | O(1) |
| Пузырьком | O(n) | O(n2) | O(n2) | O(1) |
| Слиянием | O(nlogn) | O(nlogn) | O(nlogn) | O(n) |
| Шелла | O(nlog^2(n)) | O(n2) | O(n2) | O(1) |
| Быстрая | O(nlogn) | O(nlogn) | O(n2) | O(nlogn) |

Построим для каждого из методов график зависимости времени выполнения от размера входных данных.

Сортировка вставками (Рисунок 1):

100 элементов в массиве – 9 микросекунд

1000 элементов в массиве – 751 микросекунд

5000 элементов в массиве – 19347 микросекунд

10000 элементов в массиве –74732 микросекунд

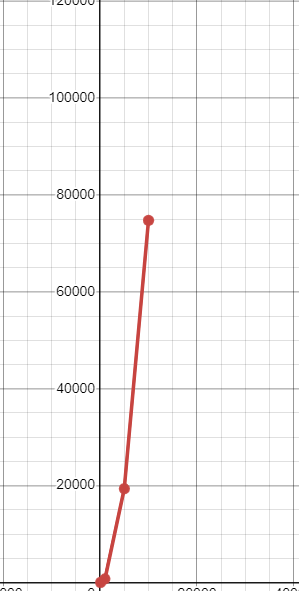


Рисунок 1

Сортировка выбором (Рисунок 2):

100 элементов в массиве – 18 микросекунд

1000 элементов в массиве – 1368 микросекунд

5000 элементов в массиве – 32942 микросекунд

10000 элементов в массиве – 128753 микросекунд

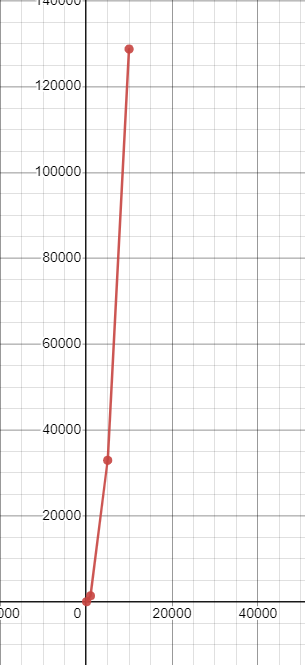


Рисунок 2

Сортировка пузырьком (Рисунок 3):

100 элементов в массиве – 27 микросекунд

1000 элементов в массиве – 2317 микросекунд

5000 элементов в массиве – 63916 микросекунд

10000 элементов в массиве – 253114 микросекунд



Рисунок 3

Сортировка слиянием (Рисунок 4):

100 элементов в массиве – 107 микросекунд

1000 элементов в массиве – 734 микросекунд

5000 элементов в массиве – 3759 микросекунд

10000 элементов в массиве – 8517 микросекунд

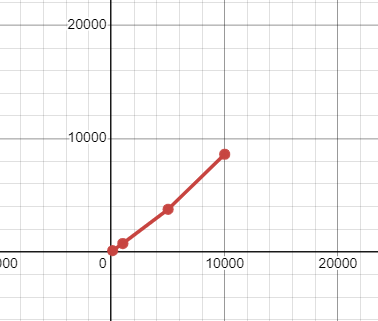


Рисунок 4

Сортировка Шелла (Рисунок 5):

100 элементов в массиве – 8 микросекунд

1000 элементов в массиве – 117 микросекунд

5000 элементов в массиве – 765 микросекунд

10000 элементов в массиве – 1662 микросекунд

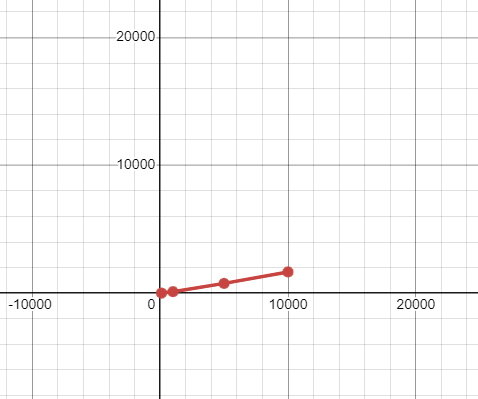


Рисунок 5

Быстрая сортировка (Рисунок 6):

100 элементов в массиве – 8 микросекунд

1000 элементов в массиве – 121 микросекунд

5000 элементов в массиве – 659 микросекунд

10000 элементов в массиве – 1423 микросекунд

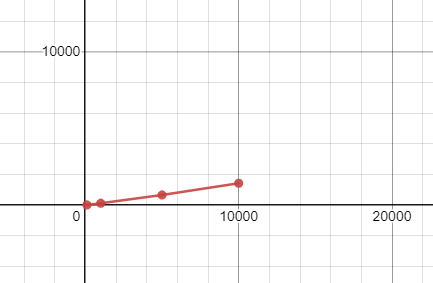


Рисунок 6

Найдем временную асимптотическую сложность для лучшего (числа идут по возрастанию), худшего (числа расположены хаотично) и среднего случая (числа расположены в порядке убывания).

Для примера будем брать массив из 300 элементов. Время в микросекундах.

|  |  |  |  |
| --- | --- | --- | --- |
| Сортировка\Сложность | Лучший случай | Средний случай | Худший случай |
| Вставками | 1 | 68 | 139 |
| Выбором | 131 | 138 | 145 |
| Пузырьком | 133 | 206 | 195 |
| Слиянием | 220 | 221 | 206 |
| Шелла | 8 | 28 | 12 |
| Быстрая | 514 | 28 | 332 |

Лучше всего, в данном случае (было взято 300 элементов в массиве), показала себя сортировка Шелла. Её мы и будем сравнивать со встроенной сортировкой sort в C++.

Эксперимент проведём для 100 и 10000 элементов.

Сортировка Шелла:

100 эл. – 9 микросекунд

10000 эл. – 1504 микросекунд

Встроенная сортировка sort:

100 эл. – 29 микросекунд

10000 эл. – 3751 микросекунд

Ссылка:

https://github.com/RVA-t/aisd\_lab\_3