МГТУ им. Баумана

Лабораторная работа №3

По курсу: "Анализ алгоритмов"

Алгоритмы сортировки массива

Работу выполнила: Лаврова Анастасия,
ИУ7-55Б

Преподаватели: Волкова Л.Л., Строганов Ю.В.

Москва, 2019

Оглавление

Введение

В данной работе требуется изучить и применить алгоритмы сортировки массива, научиться рассчитывать трудоёмкость алгоритмов, получить в сравнении алгоритмов. Необходимо реализовать описанные ниже алгоритмы:

- сортировка «пузырьком
- сортировка «вставками»
- шейкерная сортировка

и так далее.

В ходе лабораторной работы предстоит:

- изучить работу алгоритмов сортировки
- выполнить полную математическую оценку трудоёмкости для алгоритмов сортировки с указанием лучшего и худшего случаев
- реализовать три алгоритма сортировки на одном из языков программирования
- сравнить работу алгоритмов сортировок и сделать выводы

1 Аналитическая часть

Сортировка массива — одна из самых популярных операций над массивом. Алгоритмы реализуют упорядочивание элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

1.1 Сортировка пузырьком

Алгоритм состоит из повторяющихся проходов по сортируемому массиву. За каждый проход элементы последовательно сравниваются попарно, и если порядок в паре неверный, выполняется обмен элементов. Проходы по массиву повторяются T раз или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает массив отсортирован. При каждом проходе алгоритма по внутреннему циклу, очередной наибольший элемент массива ставится на своё место в конце массива рядом с предыдущим «наибольшим элементом», а наименьший элемент перемещается на одну позицию к началу массива («всплывает» до нужной позиции, как пузырёк в воде — отсюда и название алгоритма).

1.2 Сортировка вставками

На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

1.3 Шейкерная сортировка

Шейкерная сортировка является усовершенствованным методом пузырьковой сортировки. Во-первых, если при движении по части массива перестановки не происходят, то эта часть массива уже отсортирована и, следовательно, её можно исключить из рассмотрения. Во-вторых, при движении от конца массива к началу минимальный элемент «всплывает» на первую позицию, а максимальный элемент сдвигается только на одну позицию вправо. Эти две идеи приводят к следующим модификациям в методе пузырьковой сортировки. Границы рабочей части массива (то есть части массива, где происходит движение) устанавливаются в месте последнего обмена на каждой итерации. Массив просматривается поочередно справа налево и слева направо.

2 | Конструкторская часть

Требования к вводу:
На вход подается массив и его размер
Требования к программе:
Корректная сортировка массива тремя способами

На рис. 2.1 представлена IDEF0 диаграмма сортировки массива:

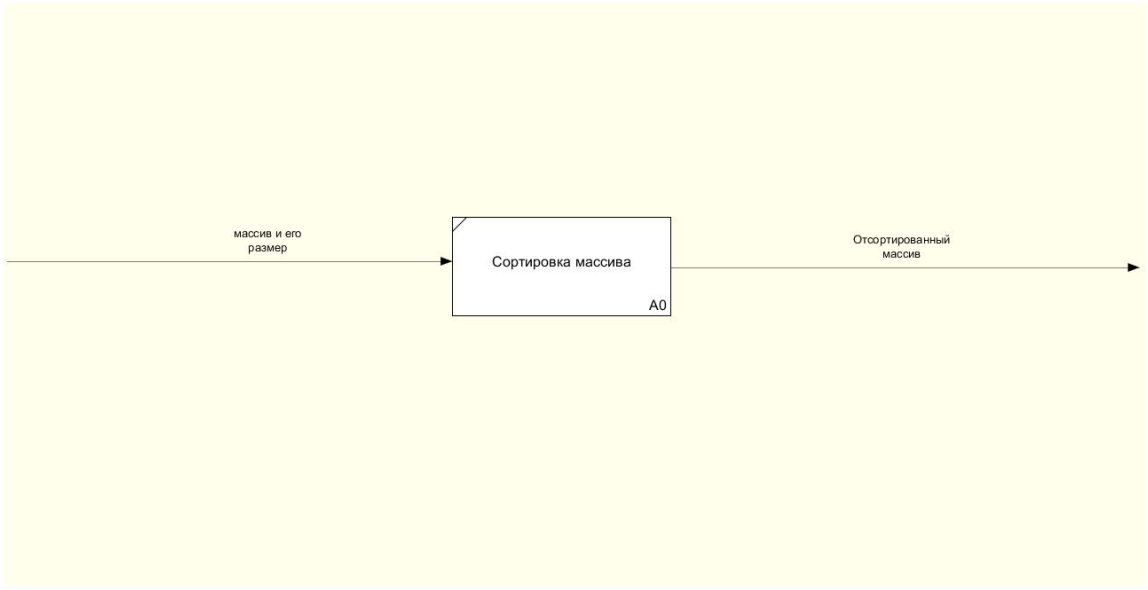

Рис. 2.1: IDEF0-диаграмма, описывающая алгоритм сортировка массива

2.1 Трудоемкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов:

- базовые операции стоимостью 1 — +, -, *, /, =, ==, <=, >=, !=, +=, []
- оценка трудоемкости цикла for от 0 до N с шагом 1 $F_{for} = 2 + N \cdot (2 + F_{body})$, где F_{body} - тело цикла
- стоимость условного перехода применим за 0, стоимость вычисления условия остаётся

Оценим трудоемкость алгоритмов по коду программы

2.1.1 Сортировка пузырьком

Рассмотрим трудоемкость сортировки пузырьком:

Лучший случай: $2 + 2N + 3NN$

Худший случай: $2 + 2N + 6NN$

2.1.2 Сортировка вставками

Рассмотрим трудоемкость сортировки вставками:

Лучший случай: $2 + 9N + 7/2NN$

Худший случай: $2 + 2N + 11/2NN$

2.1.3 Шейкерная сортировка

Рассмотрим трудоемкость шейкерной сортировки:

Лучший случай: $12NN + 7N + 5$

Худший случай: $28NN + 7N + 5$

2.2 Схемы алгоритмов

В данной части будут рассмотрены схемы алгоритмов.
На рис. 2.2 представлена схема алгоритма сортировки "пузырьком".
На рис. 2.3 представлена схема алгоритма сортировки "вставками".
На рис. 2.4 представлена схема алгоритма шейкерной сортировки.

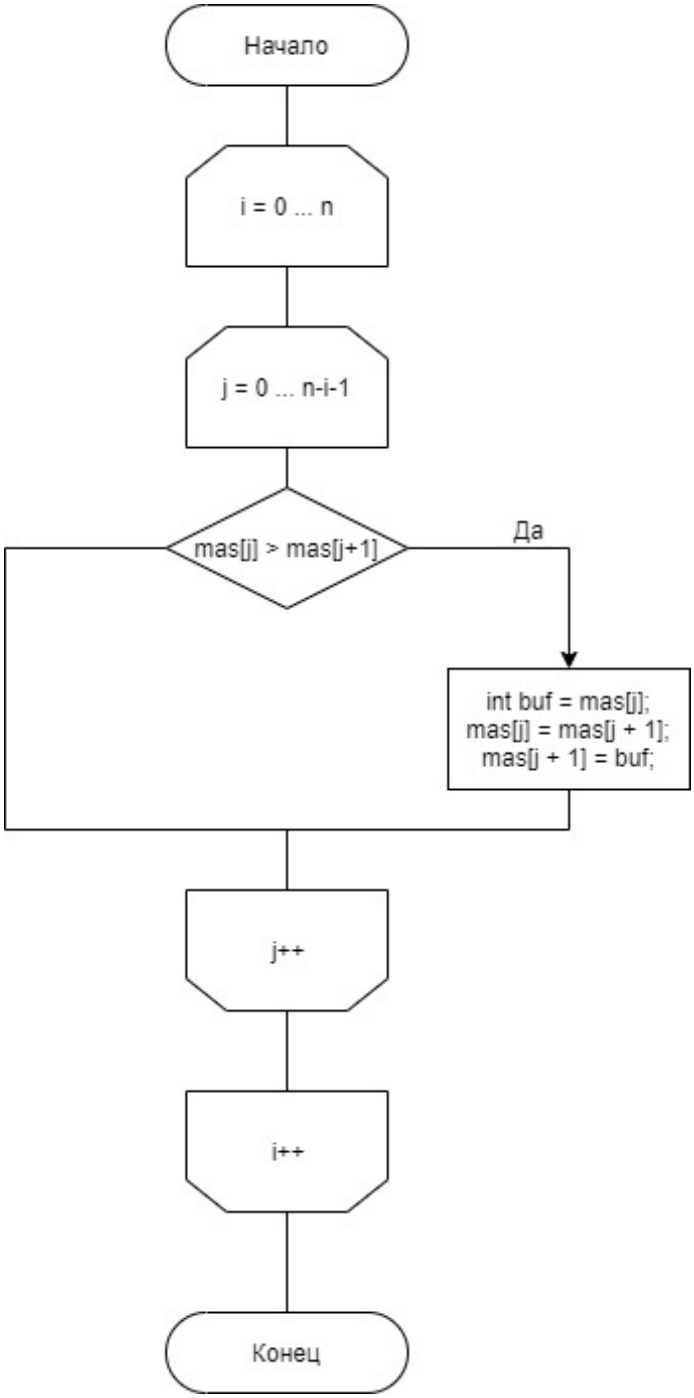

Рис. 2.2: Схема алгоритма сортировки "пузырьком"

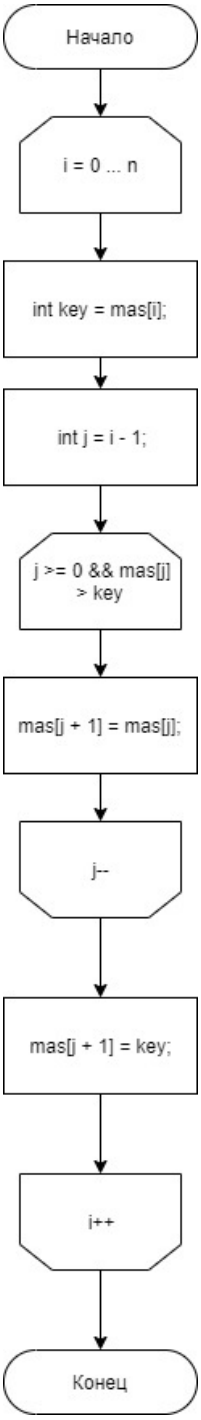

Рис. 2.3: Схема алгоритма сортировки "вставками"

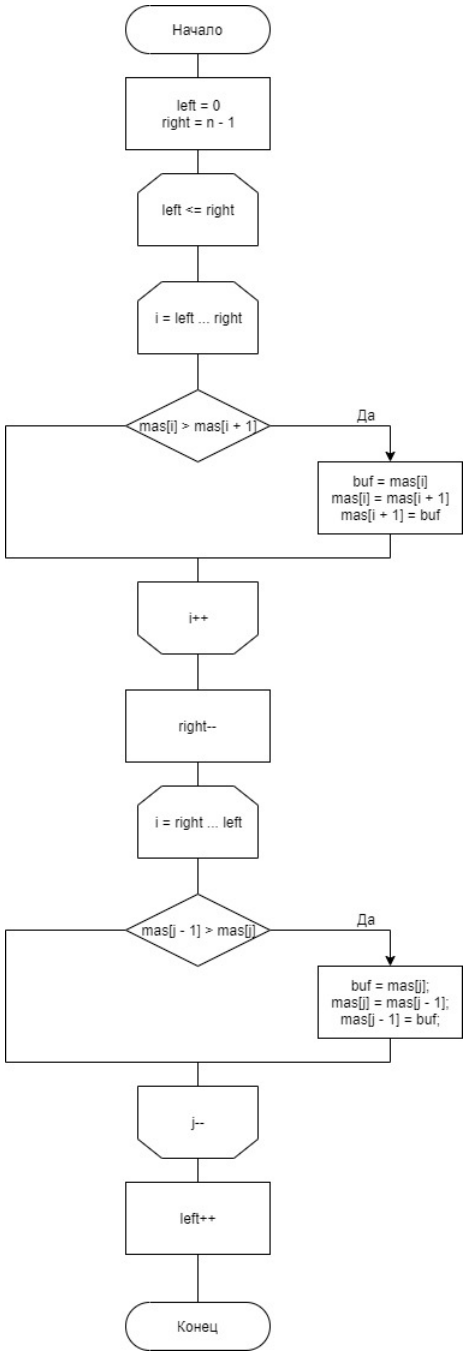

Рис. 2.4: Схема алгоритма шейкерной сортировки

3 Технологическая часть

3.1 Выбор ЯП

Для реализации программ я выбрала язык программирования C++, так имею большой опыт работы с ним. Среда разработки - Visual Studio.

Для замера процессорного времени используется функция, возвращающая количество тиков.

Листинг 3.1: Функция получения тиков

```
unsigned long long getTicks(void)
{
    unsigned long long d;
    __asm__ __volatile__ ("rdtsc" : "=A" (d) );
    return d;
}
```

3.2 Реализация алгоритма

Листинг 3.2: Алгоритм сортировки пузырьком

```
void sortBubble(int* mas, int n)
```

```
{
  for (int i = 0; i < n; i++)
    for (int j = 0; j < n - i - 1; j++)
      if (mas[j] > mas[j + 1])
      {
        int buf = mas[j];
        mas[j] = mas[j + 1];
        mas[j + 1] = buf;
      }
}
```

Листинг 3.3: Алгоритм сортировки вставками

```
void sortInsertion(int* mas, int n)
{
  for (int i = 1; i < n; i++)
  {
    int key = mas[i];
    int j = i - 1;
    for (; j >= 0 && mas[j] > key; j--)
      mas[j + 1] = mas[j];
    mas[j + 1] = key;
  }
}
```

Листинг 3.4: Алгоритм шейкерной сортировки

```
void sortShaker(int *mas, int n)
{
  int left = 0, right = n - 1;
  while (left <= right)
  {
    for (int i = left; i < right; i++)
    {
      if (mas[i] > mas[i + 1])
      {
        int buf = mas[i];
        mas[i] = mas[i + 1];
        mas[i + 1] = buf;
```

```
13        }
14      }
15      right--;
16      for (int j = right; j >= left; j--)
17      {
18        if (mas[j - 1] > mas[j])
19        {
20          int buf = mas[j];
21          mas[j] = mas[j - 1];
22          mas[j - 1] = buf;
23        }
24      }
25      left++;
26    }
27  }
```

3.3 Тестовые данные

Проверка работы алгоритмов на примерах:

- массив, заполненные случайными числами
- отсортированные массивы
- отсортированные в обратном порядке массивы

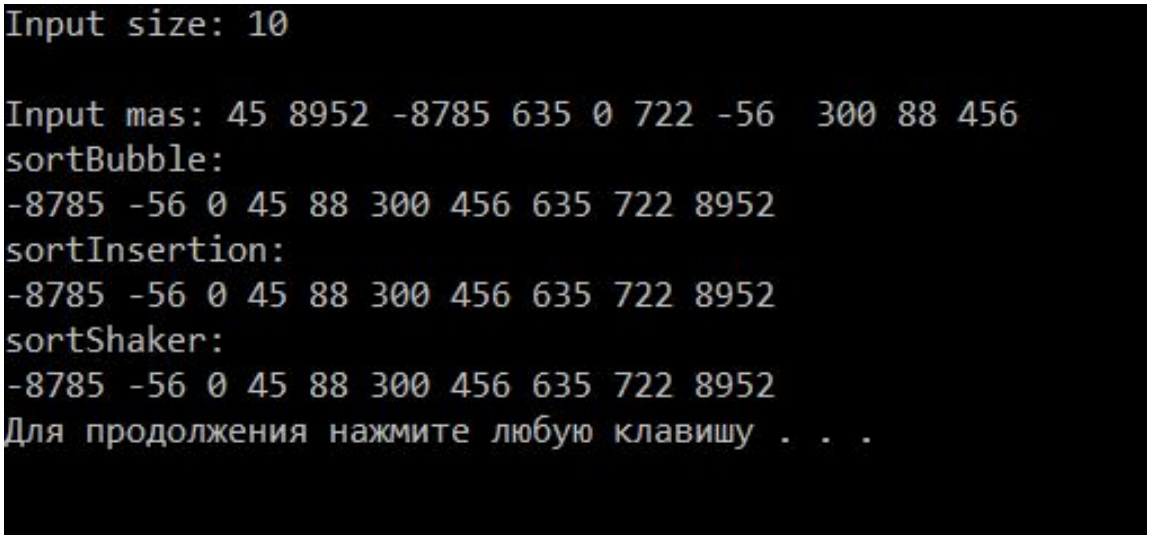

Рис. 3.1: Пример работы алгоритмов с массивом, заполненным случайными числами

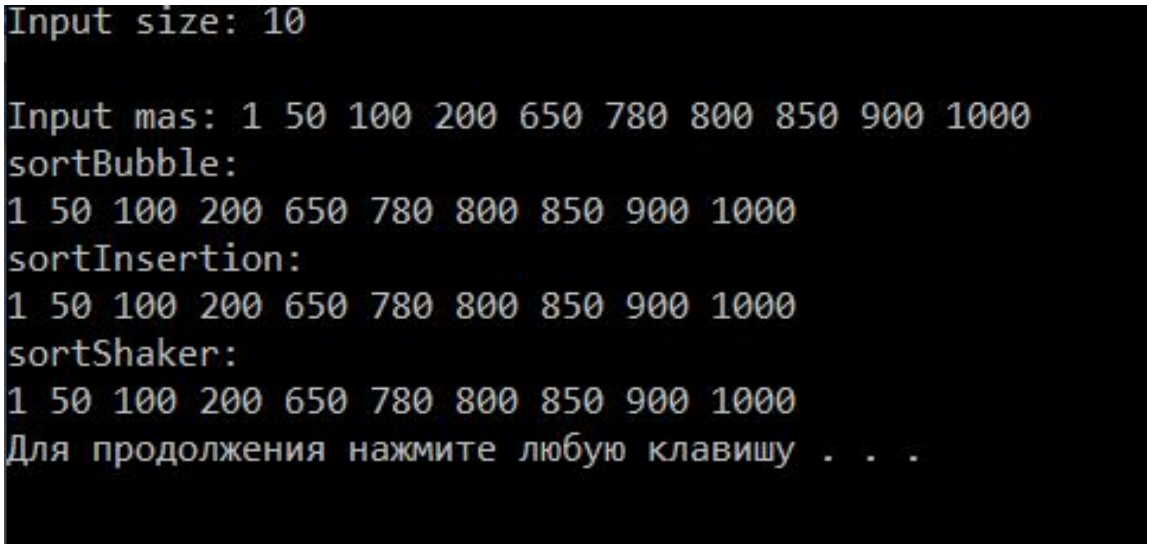

Рис. 3.2: Пример работы алгоритмов с отсортированным массивом

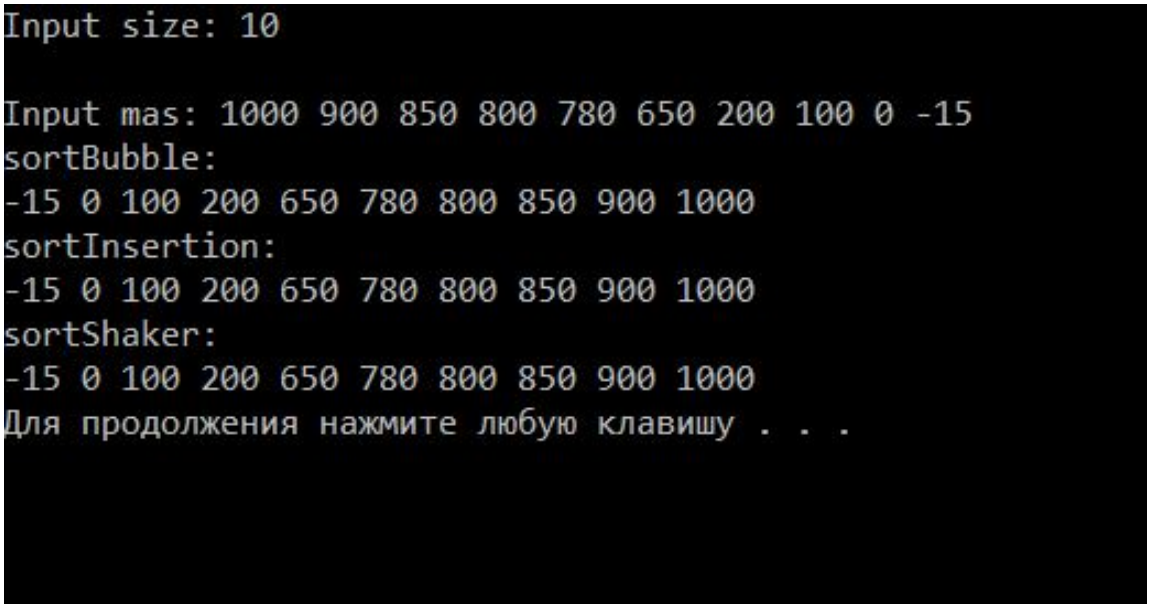

Рис. 3.3: Пример работы алгоритмов с отсортированным в обратном порядке массивом

4 Исследовательская часть

4.1 Сравнительный анализ на основе замеров времени работы алгоритмов

Был проведен замер времени работы каждого из алгоритмов.

Сравнение времени сортировки массивов, заполненных случайными числами:

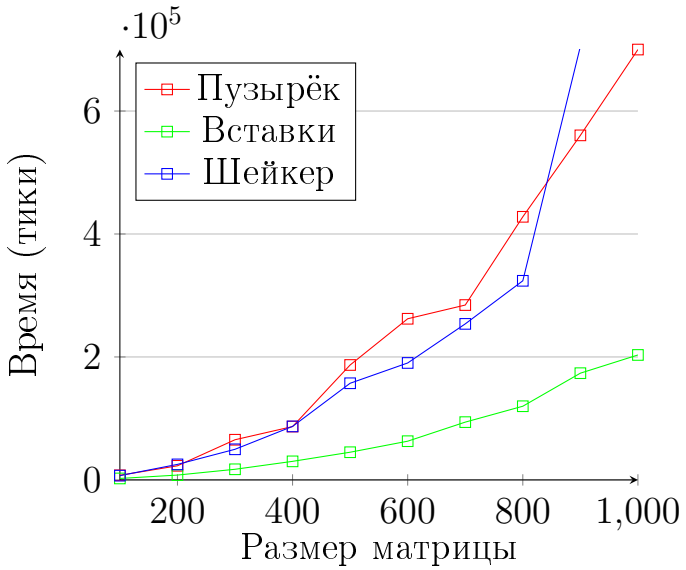

Рисунок 4.4. График времени работы алгоритмов сортировки

массивов, заполненных случайными числами

Сравнение времени сортировки массивов, удовлетворяющие лучшим случаям выполнения алгоритма:

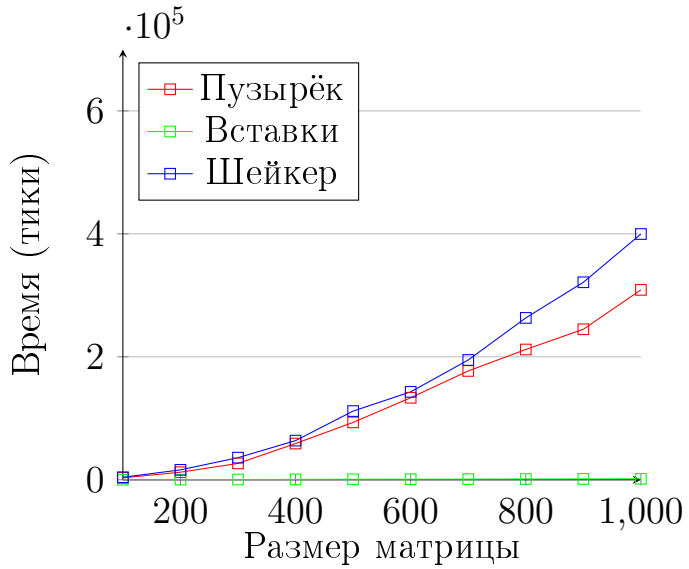

Рисунок 4.5. График времени работы алгоритмов сортировки массивов, удовлетворяющие лучшим случаям выполнения алгоритма

Сравнение времени сортировки массивов, удовлетворяющие худшим случаям выполнения алгоритма:

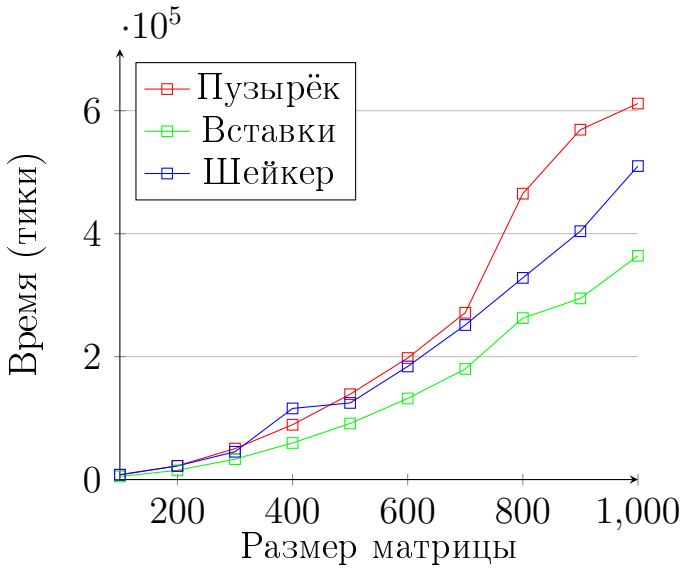

Рисунок 4.6. График времени работы алгоритмов сортировки массивов, удовлетворяющие худшим случаям выполнения алгоритма

4.1.1 Вывод

По результатам тестирования выявлено, что все рассматриваемые алгоритмы реализованы правильно. Самым быстрым алгоритмом, при использовании случайного заполнения, оказался алгоритм сортировки «вставками», а алгоритмы сортировки «пузырьком» и «шейкер» являются разновидностями пузырьковой сортировки, их результаты схожи, однако на массивах большей длины выигрывает классическая сортировка «пузырьком».

Заключение

В ходе работы были изучены алгоритмы сортировки массива: «пузырек», «вставки» и «шейкер». Выполнено сравнение всех рассматриваемых алгоритмов. Изучены зависимости выполнения алгоритмов от длины массива. Также реализован программный код продукта