Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

Отчет по лабораторной работе №3
По курсу: «Анализ алгоритмов»

По теме: «Алгоритмы сортировки массива»

Студент:
Коротков Андрей Владимирович
Группа: ИУ7-55Б

Преподаватели:
Волкова Лилия Леонидовна
Строганов Юрий Владимирович

Москва, 2019 г.

Содержание

Введение

В данной работе требуется изучить и применить алгоритмы сортировки массива, научиться рассчитывать трудоёмкость алгоритмов, получить в сравнении алгоритмов.

Необходимо реализовать описанные ниже алгоритмы:

1. сортировка «пузырьком»;
2. сортировка «расческой»;
3. сортировка «вставками».

1. Аналитическая часть

В данном разделе будет рассмотрено описание алгоритмов сортировки массива.

1.1 Описание алгоритмов

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.

Эти алгоритмы активно применяются во всех областях, которым необходимо хранить последовательность чисел, такие как:

- математика;
- физика;
- экономика;
- и так далее.

1.1.1 Алгоритм сортировки «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 $N - 1$ 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).

1.1.2 Алгоритм сортировки «расчёской»

Основная идея «расчёски» в том, чтобы первоначально брать достаточно большое расстояние между сравниваемыми элементами и по мере упорядочивания массива сужать это расстояние вплоть до минимального. Таким образом, мы как бы причёсываем массив, постепенно разглаживая на всё более аккуратные пряди. Первоначальный разрыв между сравниваемыми элементами лучше брать с учётом специальной величины, называемой фактором уменьшения, оптимальное значение которой равно примерно 1,247. Сначала расстояние между элементами равно размеру массива, разделённого на фактор уменьшения (результат округляется до ближайшего целого). Затем, пройдя массив с этим шагом, необходимо поделить шаг на фактор уменьшения и пройти по списку вновь. Так продолжается до тех пор, пока разность индексов не достигнет единицы. Сортировка завершается обычным пузырьком.

Оптимальное значение фактора уменьшения $1{,}247... = \frac{1}{1 - e^{-\phi}}$, где e — основание натурального логарифма, а ϕ — золотое сечение.

1.1.3 Алгоритм сортировки «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.

1.1.4 Вывод

Были рассмотрены поверхностно алгоритмы сортировки «пузырьком», «расческой», «вставками», которые по разному обходят массив, в следствие чего сортируют по-разному.

2. Конструкторская часть

В данном разделе будут рассмотрены схемы алгоритмов:

- сортировка «пузырьком»;
- сортировка «расческой»;
- сортировка «вставками».

2.1 Разработка алгоритмов

Ниже на рисунках (1) - (3) приведены схемы алгоритмов умножения матриц:

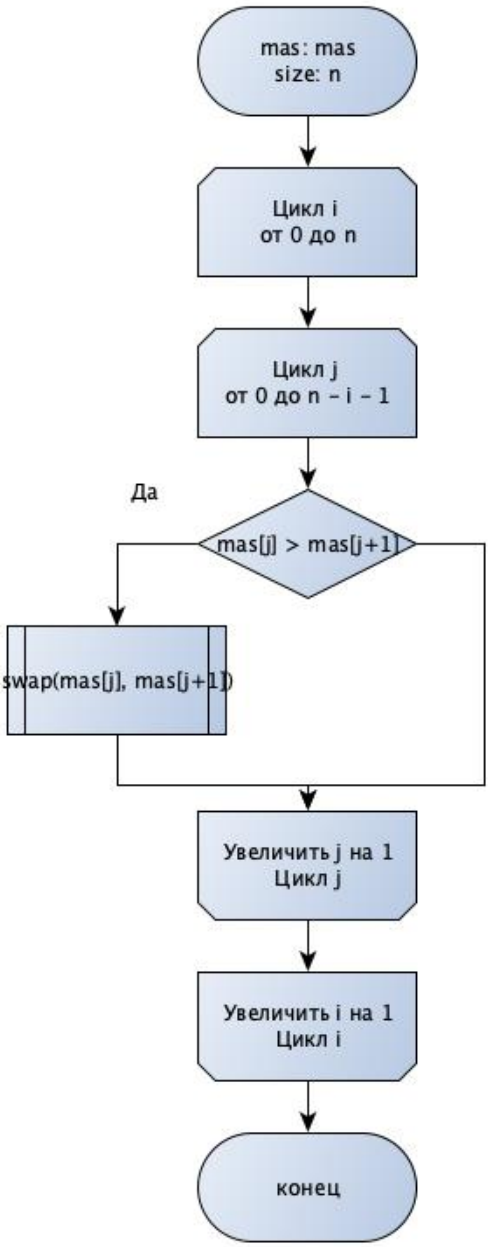

Рисунок 1. Схема алгоритма сортировки массива «пузырьком».

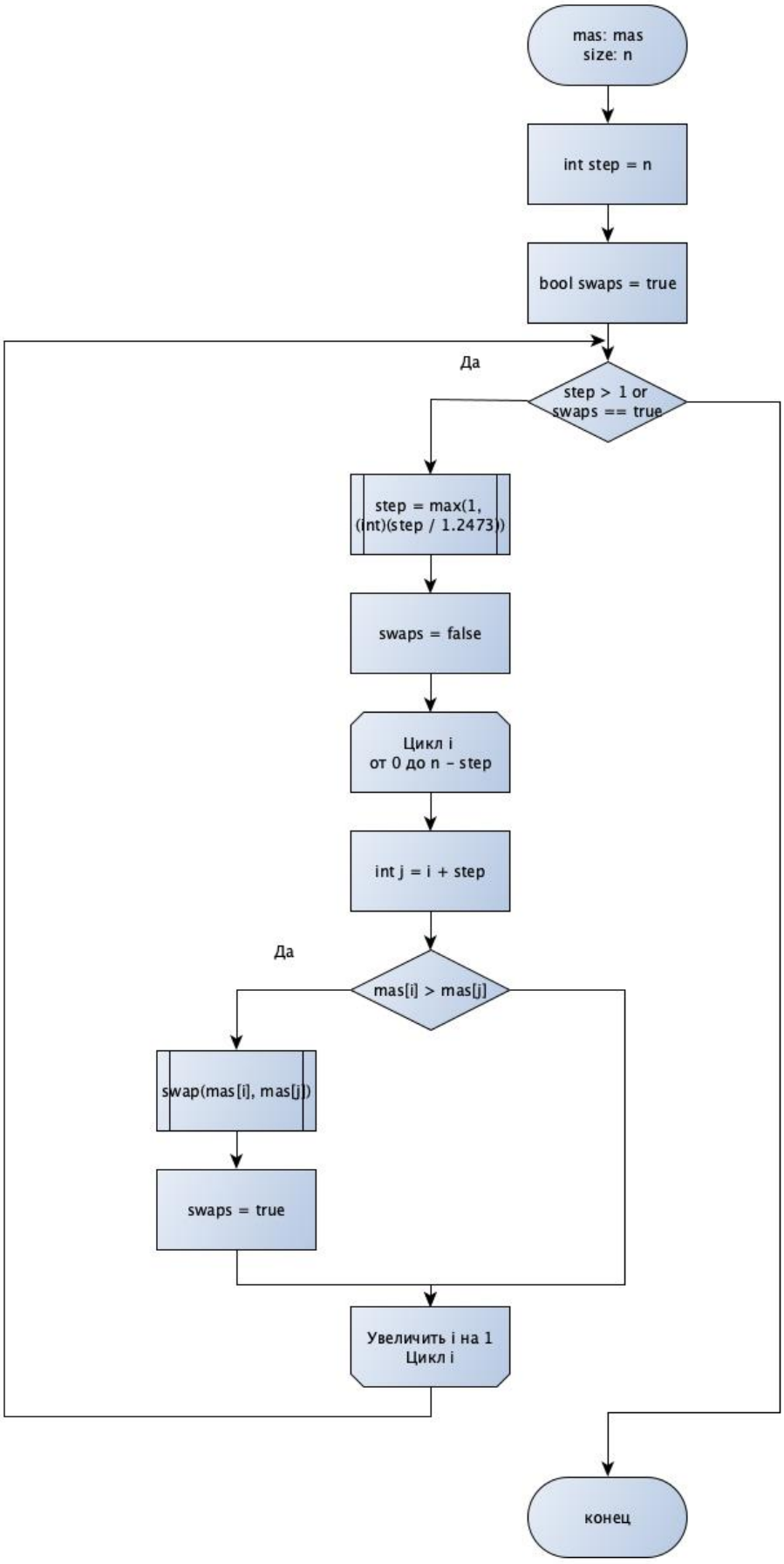

Рисунок 2. Схема алгоритма сортировки массива «расчёской».

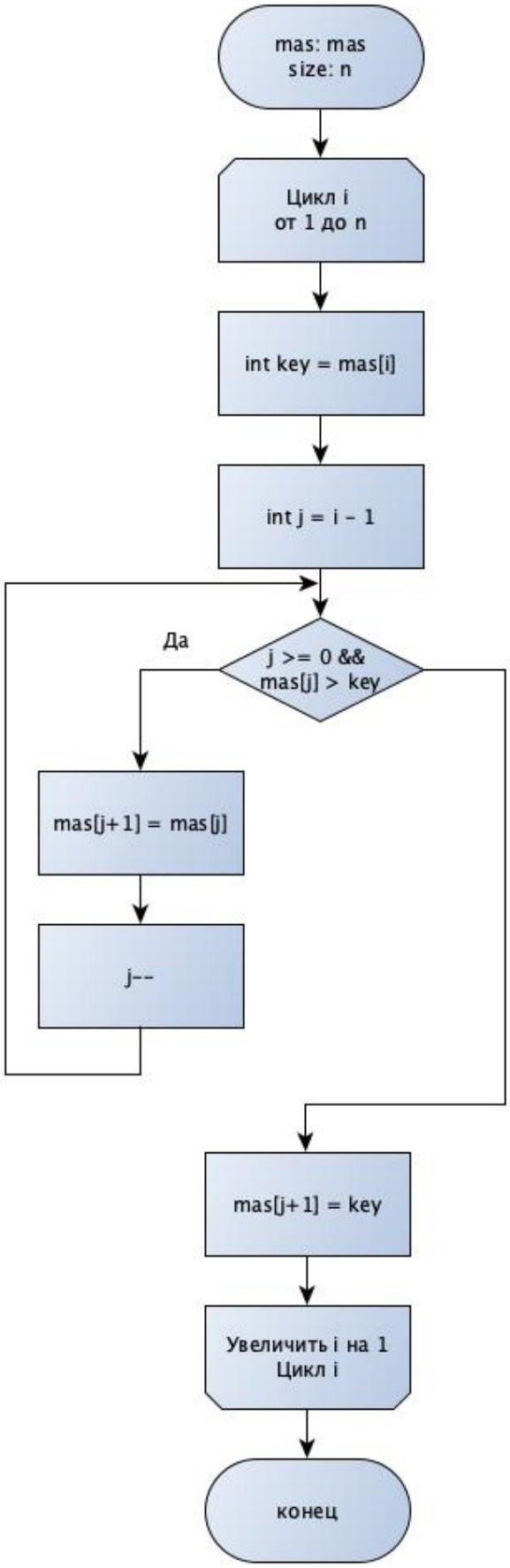

Рисунок 3. Схема алгоритма сортировки массива «вставками».

2.2 Трудоемкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов:

1) базовые операции стоимостью 1 — +, -, *, /, =, ==, <=, >=, !=, +=, [];
2) оценка трудоемкости цикла:

Листинг 1. Общий пример цикла в Си.

```
for (int i = 0; i <= N; i++)
{
      //тело цикла
}
```

Fц = 2 + N*(2 + Fтела)

3) стоимость условного перехода возьмем за 0, стоимость вычисления условия остаётся.

2.2.1 Трудоемкость алгоритмов сортировки массива

Ниже представлены на таблицах (1) - (3) представлены оценки трудоемкости алгоритмов.

Таблица 1. Оценка трудоемкости алгоритма сортировки «пузырьком».

Трудоемкость	Оценка трудоемкости
Fbubble	2 + N*(2 + N/2*(2 + Fтела))
Fтела	$\begin{bmatrix} 4 \\ 4+6 \end{bmatrix}$
Fbubble	$2 + 2*N + 3*N*N + N*N* \begin{bmatrix} 0 \\ 3 \end{bmatrix}$

Таблица 2. Оценка трудоемкости алгоритма сортировки «расчёской».

Трудоемкость	Оценка трудоемкости
Fcomb	2 + 2 + N(2 + 4 + N/2(2 + 3 + Fтела))
Fтела	$\begin{bmatrix} 5 \\ 5+6 \end{bmatrix}$
Fcomb	$4 + 6*N + 5/2*N*N + N*N/2 \begin{bmatrix} 0 \\ 6 \end{bmatrix}$

Таблица 3. Оценка трудоемкости алгоритма сортировки «расчёской».

Трудоемкость	Оценка трудоемкости
Finsrt	$2 + N(2 + 7 + N/2(2 + 5 + \begin{bmatrix}0\\4\end{bmatrix}))$
Finsrt	$2 + 9*N + 7/2*N*N + 1/2*N*N*\begin{bmatrix}0\\4\end{bmatrix}$

2.3 Вывод

Так как оценка дается по самому быстрому растущему слагаемому, в нашем случае это квадрат длины массива, в алгоритме сортировки «пузырьком» коэффициент равен 3 в лучшем случае и 6 в худшем случае, «расческой» — 5/2 в лучшем случае и 11/2 в худшем случае, «вставками» — 7/2 в лучшем случае и 11/2 в худшем случае.

3. Технологическая часть

В данном разделе будут рассмотрены требования к программному обеспечению, средства реализации и представлен листинг кода.

3.1 Требования к программному обеспечению

Входные данные: mas - массив чисел, n - длина массива.

Выходные данные: отсортированный массив.

Рисунок 4. IDEF0-диаграмма, описывающая алгоритм сортировки массива.

3.2 Средства реализации

В данной работе используется язык программирования «C». Проект выполнен в среде Xcode. Для отключения оптимизации компилятора указан флаг «-o0».

Для замера процессорного времени используется функция, возвращающая количество тиков.

Листинг 2. Функция замера времени.

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

3.3 Листинг кода

В данном пункте представлен листинг кода, а именно:

- сортировка «пузырьком»;
- сортировка «расческой»;
- сортировка «вставками».

Ниже приведен листинг трех алгоритмов сортировки массива:

Листинг 3. 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])
                swap(&mas[j], &mas[j+1]);
}
```

Листинг 4. Алгоритм сортировки «расчёской».

```
void sortComb(int* mas, int n)
{
    int step = n;
    bool swaps = true;
    while (step > 1 || swaps)
    {
        step = max(1, (int)(step / 1.2473));
        swaps = false;
        for (int i = 0;  i < n - step; i++)
        {
            int j = i + step;
            if (mas[i] > mas[j])
            {
                swap(&mas[i], &mas[j]);
                swaps = true;
            }
        }
    }
}
```

Листинг 5. 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;
    }
}
```

3.4 Вывод

В данном разделе были представлены реализации алгоритмов сортировки «пузырьком», «вставками» и «расчёской».

4. Экспериментальная часть

В данном разделе будут рассмотрены примеры работы программы.
Также будет проведен эксперимент и сравнительный анализ данных.

4.1 Примеры работы

Проверки работы алгоритмов:

- массивы, заполненные случайными числами;
- отсортированные массивы, со случайными числами;
- отсортированные массивы в обратном порядке, со случайными числами.

В листингах (6) - (10) представлены результаты работы программы на разных входных данных:

Листинг 6. Сортировка массива, заполненного случайными числами.

```
Mas:
49302 67257 74751 68256 83618 81989 26443 33320 36007 53003
--------------------------------
Bubble sort:
26443 33320 36007 49302 53003 67257 68256 74751 81989 83618
Comb sort:
26443 33320 36007 49302 53003 67257 68256 74751 81989 83618
Insert sort:
26443 33320 36007 49302 53003 67257 68256 74751 81989 83618
```

Листинг 7. Сортировка отсортированного массива, заполненного случайными числами.

```
Mas:
3066 9579 16886 20002 45006 50313 52164 53978 66235 82982
--------------------------------
Bubble sort:
3066 9579 16886 20002 45006 50313 52164 53978 66235 82982
Comb sort:
3066 9579 16886 20002 45006 50313 52164 53978 66235 82982
Insert sort:
3066 9579 16886 20002 45006 50313 52164 53978 66235 82982
```

.

Листинг 8. Сортировка отсортированного массива в обратном порядке, заполненного случайными числами.

```
Mas:
94705 80090 71578 69550 67925 66025 43768 41858 26963 2168
--------------------------------
Bubble sort:
2168 26963 41858 43768 66025 67925 69550 71578 80090 94705
Comb sort:
2168 26963 41858 43768 66025 67925 69550 71578 80090 94705
Insert sort:
2168 26963 41858 43768 66025 67925 69550 71578 80090 94705
```

Листинг 9. Сортировка массива, заполненного нулями.

```
Mas:
0 0 0 0 0
--------------------------------
Bubble sort:
0 0 0 0 0
Comb sort:
0 0 0 0 0
Insert sort:
0 0 0 0 0
```

Листинг 10. Сортировка пустого массива.

```
Mas:
Empty
--------------------------------
Bubble sort:
Empty
Comb sort:
Empty
Insert sort:
Empty
```

4.2 Постановка эксперимента

В рамках данного проекта были проведены эксперименты, описанные ниже.

1. Сравнение времени выполнения трех алгоритмов в лучшем случае. Создаются массивы длиной от 100 до 2000 элементов с шагом 100 элементов. Для каждого алгоритма создается такой массив, который удовлетворяет лучшему случаю выполнения алгоритма.
2. Сравнение времени выполнения трех алгоритмов в худшем случае. Создаются массивы длиной от 100 до 2000 элементов с шагом 100 элементов. Для каждого алгоритма создается такой массив, который удовлетворяет худшему случаю выполнения алгоритма.
3. Сравнение времени выполнения трех алгоритмов в произвольном случае. Создаются массивы длиной от 100 до 2000 элементов с шагом 100 элементов. Для каждого алгоритма создается случайный массив.

4.3 Сравнительный анализ на материале экспериментальных данных

Ниже на рисунках (5) - (7) представлены графики работы алгоритмов по сортировке массивов разных размерностей:

Сравнение времени сортировки массивов, удовлетворяющие лучшим случаям выполнения алгоритма:

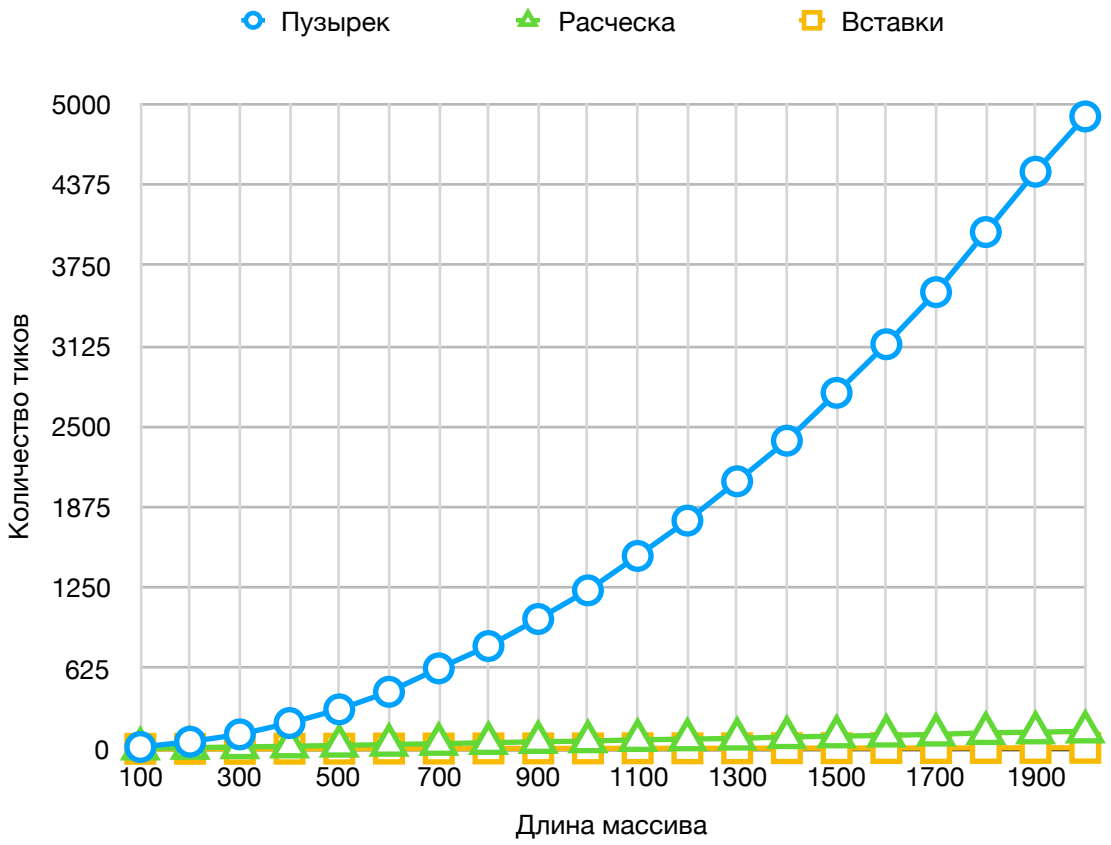

Рисунок 5. График работы алгоритмов по сортировки массива (ось абсцисс — время работы алгоритмов (тики), ось ординат — длина массива).

Сравнение времени сортировки массивов, удовлетворяющие худшим случаям выполнения алгоритма:

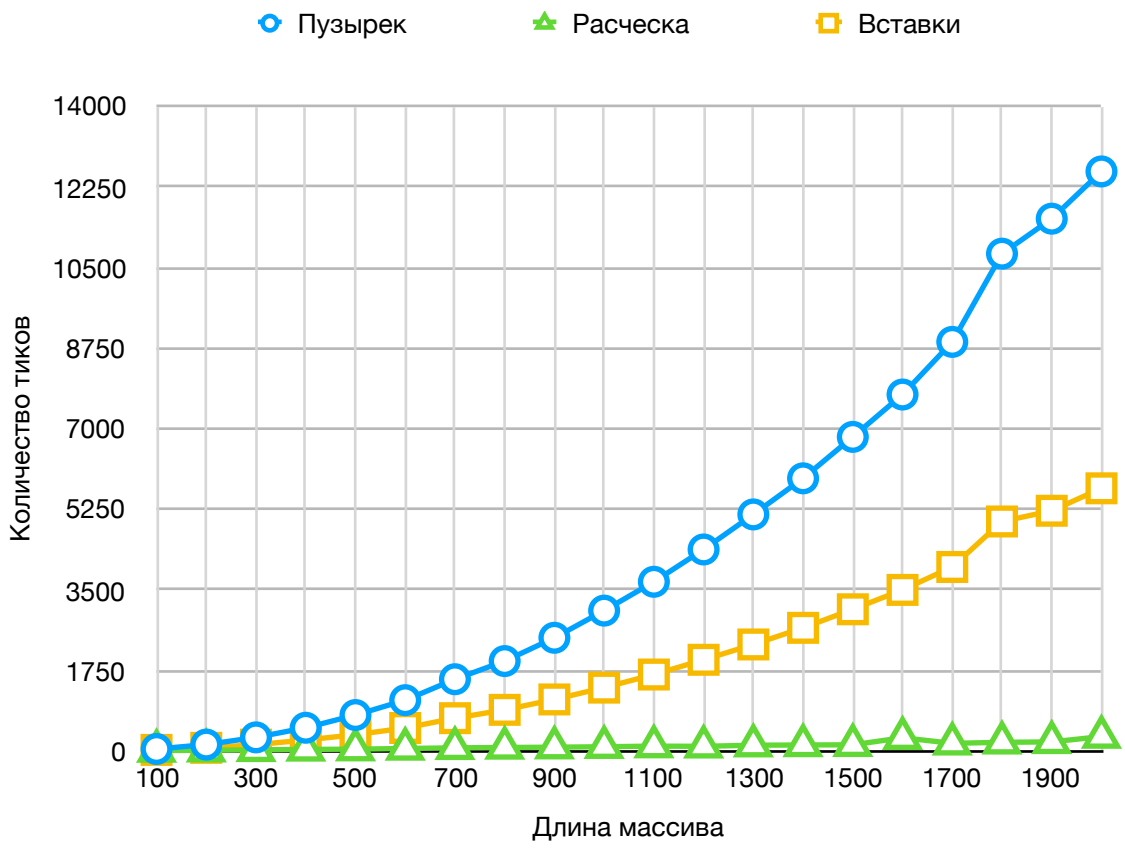

Рисунок 6. График работы алгоритмов по сортировки массива (ось абсцисс — время работы алгоритмов (тики), ось ординат — длина массива).

Сравнение времени сортировки массивов, заполненных случайными числами:

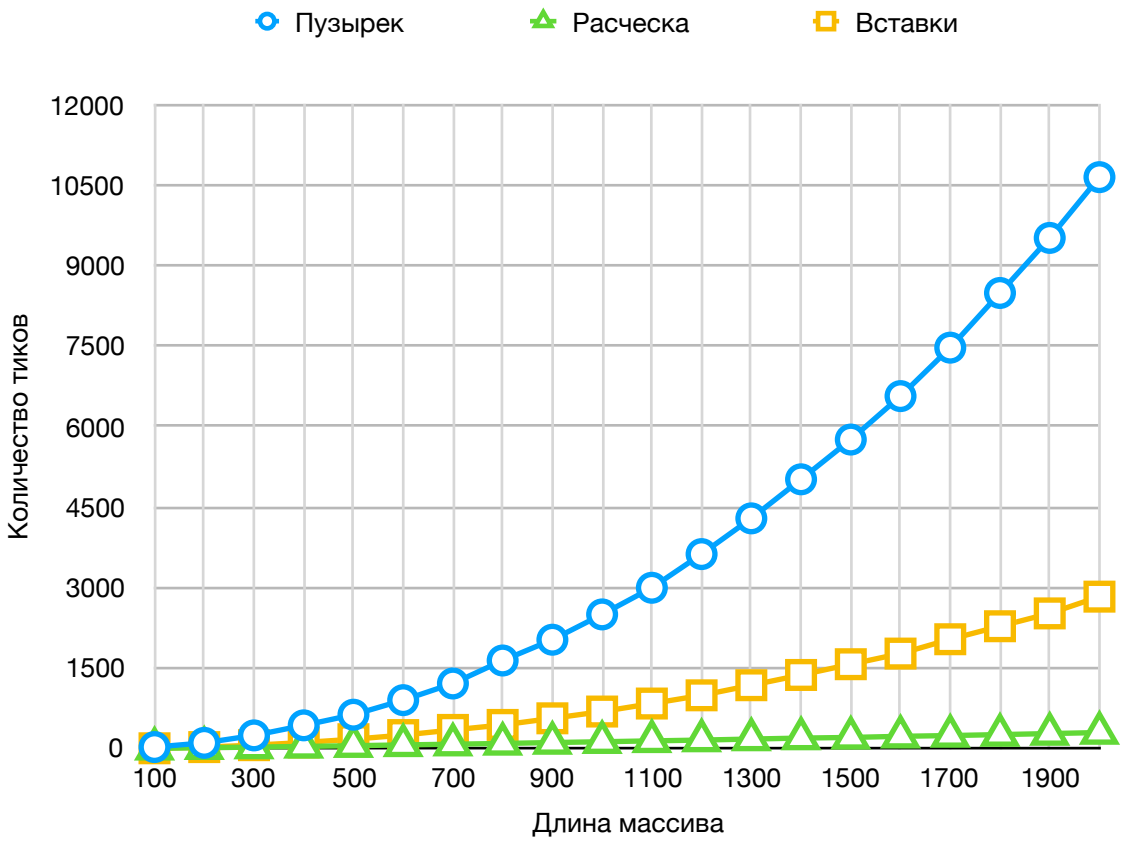

Рисунок 7. График работы алгоритмов по сортировки массива (ось абсцисс — время работы алгоритмов (тики), ось ординат — длина массива).

4.4 Вывод

По результатам тестирования выявлено, что все рассматриваемые алгоритмы реализованы правильно. Самым быстрым алгоритмом, при использовании случайного заполнения, оказался алгоритм сортировки «расческой», а самым медленным — алгоритм сортировки «пузырьком».

Заключение

В ходе работы были изучены алгоритмы сортировки массива: «пузырек», «расческа» и «вставки». Выполнено сравнение всех рассматриваемых алгоритмов. В ходе исследования было установлено, что алгоритм сортировки «расческой» является самым быстрым алгоритмом по трудоемкости и времени выполнения среди рассмотренных алгоритмов сортировки массива. Изучены зависимости выполнения алгоритмов от длины массива. Также реализован программный код продукта.

Список использованных источников

1. Кнут Д. Э. Искусство программирования [Электронный ресурс] // books.google.ru [сайт]. [2010]. URL: https://books.google.ru/booksid=92rWnktlbgC&printsec: [сайт]. [2014].
2. Рассел Джесси. Сортировка вставками. - М.:Техносфера, 2012.
3. Лорин Г. Сортировка и системы сортировки. - М.:Наука, 2010.
4. Вирт Н. Алгоритмы и структуры данных. - М.: Вершина, 2013.