Министерство науки и высшего образования 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»

Отчёт

по лабораторной работе № 3

Название: Алгоритмы сортировки

Дисциплина: Анализ алгоритмов

Студент ИУ7-55Б _____ Д.О. Склифасовский

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподователь _____ Л.Л. Волкова

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

Содержание

Введение

Цель работы: изучение алгоритмов сортировки массивов. В данной лабораторной работе рассматриваются 3 алгоритма:

1) сортировка пузырьком;

2) сортировка выбором;

3) сортировка вставками.

Также требуется изучить рассчет сложности алгоритмов. В ходе лабораторной работы необходимо:

1) изучить алгоритмы сортировки;

2) дать теоритическую оценку сортировок пузырьком, шейкером и вставками;

3) реализовать три алгоритма сортировки на одном из языков программирования;

4) сравнить алгоритмы сортировки.

1 Аналитический раздел

В данном разделе представлено описание алгоритмов сортировки массивов.[1]

1.1 Сортировка пузырьком

Сортировка пузырьком — один из самых известных алгоритмов сортировки. Здесь нужно последовательно сравнивать значения соседних элементов и менять числа местами, если предыдущее оказывается больше последующего. Таким образом элементы с большими значениями оказываются в конце списка, а с меньшими остаются в начале.

Этот алгоритм считается учебным и почти не применяется на практике из-за низкой эффективности: он медленно работает на тестах, в которых маленькие элементы (их называют «черепахами») стоят в конце массива. Однако на нём основаны многие другие методы, например, шейкерная сортировка и сортировка расчёской.

1.2 Сортировка выбором

В сортировке выбором мы целенаправленно ищем максимальный элемент (или минимальный), которым дополняем отсортированную часть массива.

1.3 Сортировка вставками

При сортировке вставками массив постепенно перебирается слева направо. При этом каждый последующий элемент размещается так, чтобы он оказался между ближайшими элементами с минимальным и максимальным значением.

1.4 Вывод

Было представлено описание алгоритмов сортировки массивов. В основном все алгоритмы сортировок основаны на алгоритме сортировки пузырьком.

2 Конструкторский раздел

В данном разделе представлены съемы разработанных алгоритмов. Также оценивается трудоемкость алгоритмов.

2.1 Разработка алгоритмов

На рисунке 1 изображена схема алгоритма сортировки пузырьком.

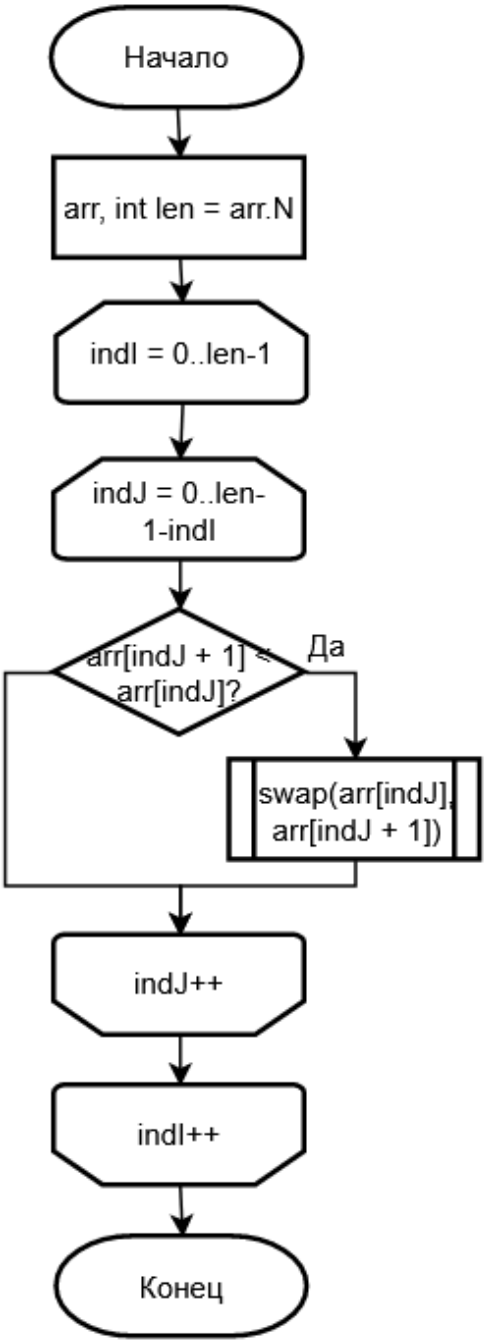

Рисунок 1 – Схема алгоритма сортировки пузырьком

На рисунке 2 изображена схема алгоритма сортировки выбором.

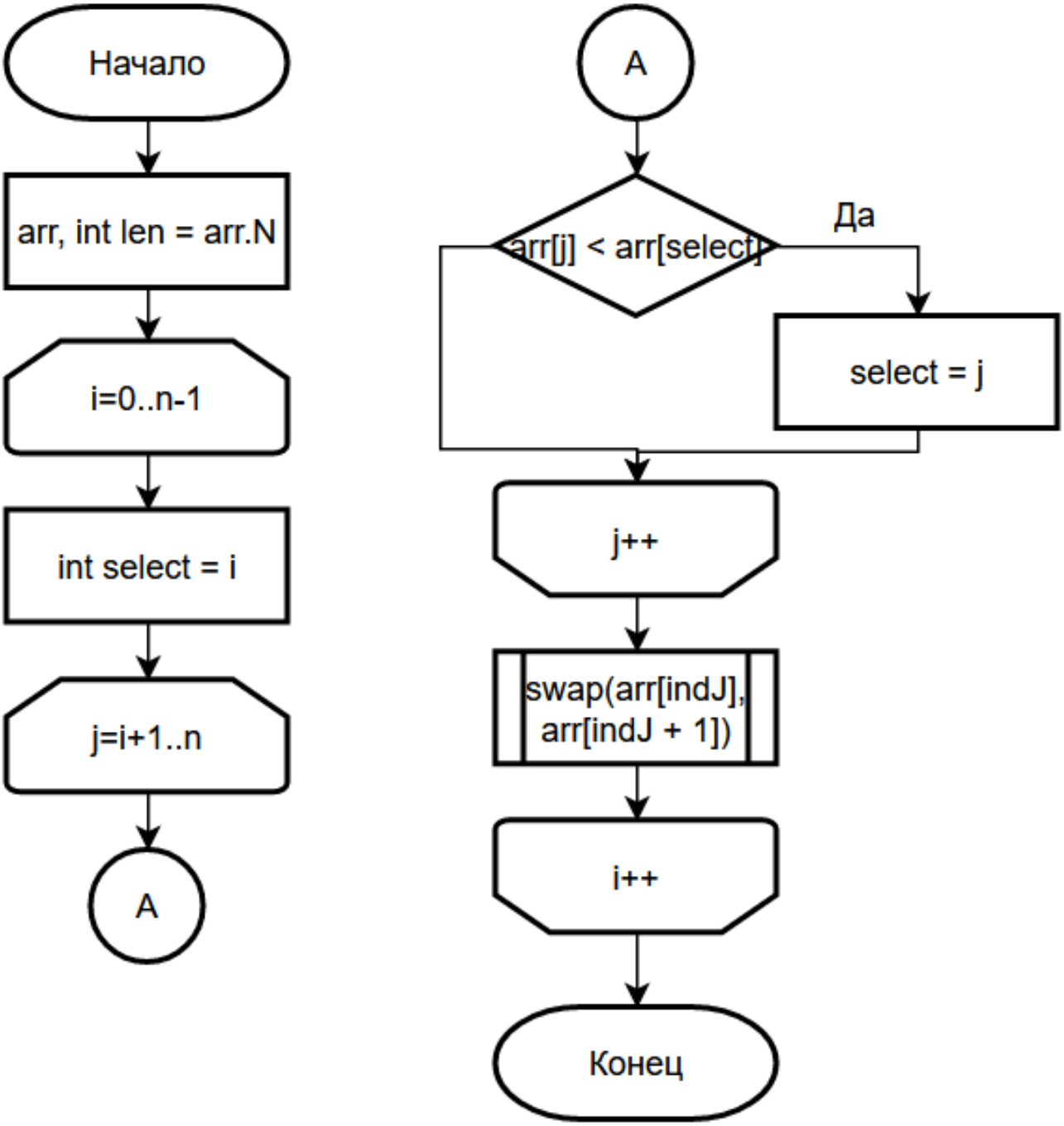

Рисунок 2 – Схема алгоритма сортировки выбором

На рисунке 3 изображена схема алгоритма сортировки вставками.

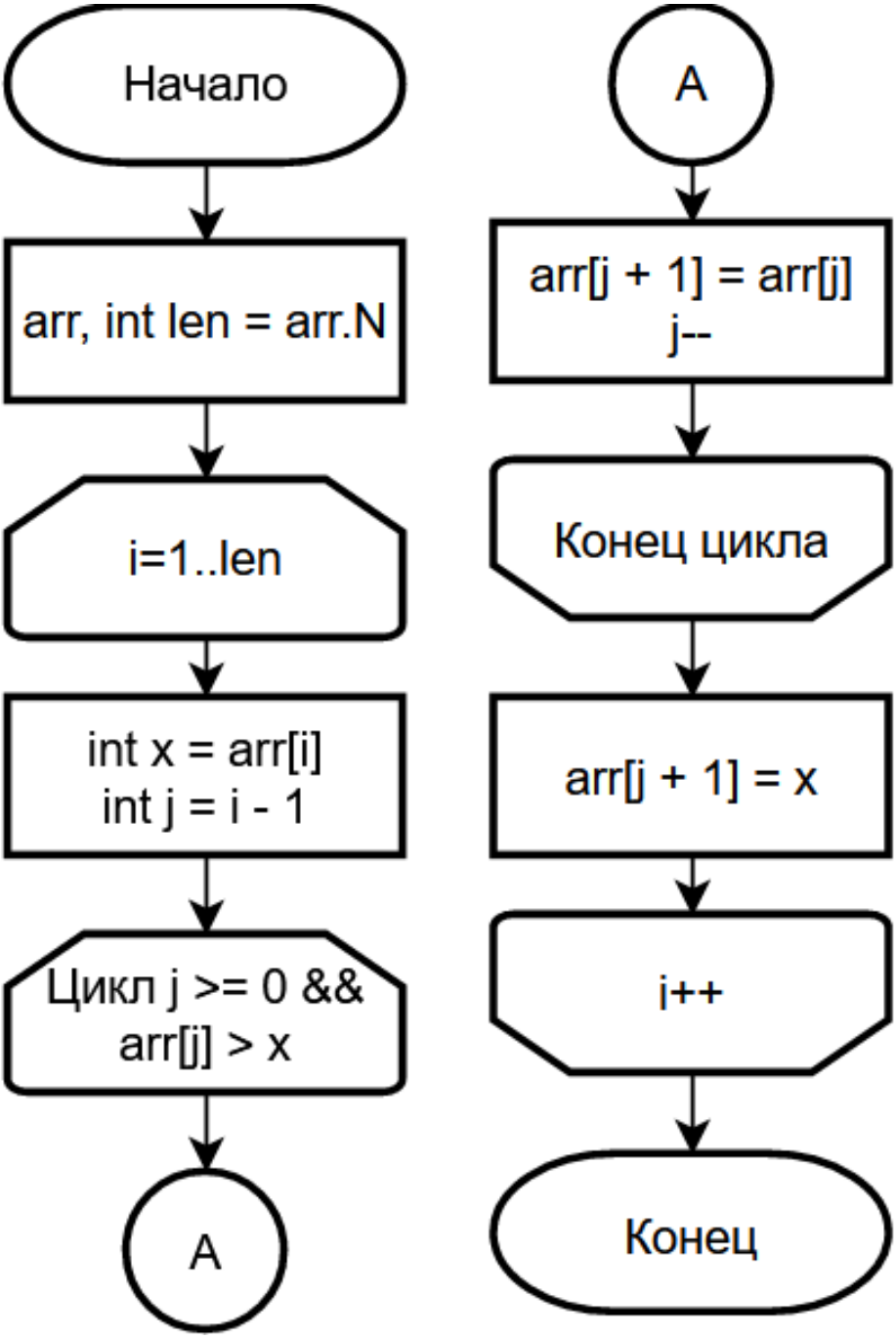

Рисунок 3 – Схема алгоритма сортировки вставками

2.2 Модель трудоемкости

Модель трудоемкости для оценки алгоритмов:

1) стоимость базовых операций единица:

$=, +, *, \simeq, <, >, \geq, \leq, ==, !=, [], +=, -=, *=, /=, ++, --;$

2) стоимость цикла:

$f_{for} = f_{init} + f_{comp} + M(f_{body} + f_{increment} + f_{comp})$

Пример: $for(i = 0, i < M; i + +)/ * body * /$

Результат: $2 + M(2 + f_{body})$;

3) стоимость условного оператора

Пусть goto (переход к одной из ветвей) стоит 0, тогда

$$f_f = \begin{cases} min(f_A, f_B), & \text{лучший случай} \\ max(f_A, f_B), & \text{худший случай} \end{cases}$$

4) операция обращения к ячейки матрицы [i, j] имеет трудоёмкость равную двум.

2.3 Оценка трудоемкости алгоритмов сортировки

Оценим трудоемкость алгоритмов.

Трудоемкость функции Swap

$$f_{swap} = 2 + 3 + 3 = 8$$

Сортировка пузырьком

Внутренний цикл будет выполняться: $n - 0 - 1, n - 1 - 1, n - 2 - 1, ..., n - (n - 2) - 1$ раз. Эта последовательность является арифметической прогрессией и ее можно записать как:

$$s_{bubble} = \frac{(n-1)n}{2}$$

Лучший случай (массив отсортирован): $f_{bubble} = 2+(3+4+(n-0-1)(4+4))+(4+3+(n-1-1)(4+4))+...+(4+3+(n-(n-2)-1)(4+4)) = 2+7(n-1)+8s_{bubble} = 2+7(n-1)+8\frac{(n-1)n}{2} = 4n^2+3n-5 \approx 4n^2$

Худший случай (массив отсортирован в порядке, обратном нужному, т.е. каждый раз будет выполняться тело условного оператора): $f_{bubble} = 3+(3+4+(n-0-1)(4+4+f_{swap}))+(3+4+(n-1-1)(4+4+f_{swap}))+...+(3+4+(n-(n-2)-1)(4+4+f_{swap})) = 3+7(n-1)+15s_{bubble} = 3+7(n-1)+15\frac{(n-1)n}{2} \approx \frac{15}{2}n^2$

Сортировка выбором

Внутренний цикл будет выполняться: $n-1, n-2, n-3, ..., n-(n-1)$ раз. Эта последовательность является арифметической прогрессией и ее можно записать как:

$$s_{selection} = \frac{(n-1)n}{2}$$

Лучший случай (массив отсортирован): $f_{selection} = 3+(3+1+3+(n-1)(2+3)+2+f_{swap})+(3+1+3+(n-2)(2+3)+2+f_{swap})+...+(3+1+3+(n-(n-1))(2+3)+2+f_{swap}) = 3+12(n-1)+12(n-1)+5*s_{selection} = \frac{5}{2}n^2+9n-9 \approx \frac{5}{2}n^2$

Худший случай (массив отсортирован в порядке, обратном нужному, т.е. каждый раз будет выполняться тело условного оператора): $f_{selection} = 3+(3+1+3+(n-1)(2+3+1)+2+f_{swap})+(3+1+3+(n-2)(2+3+1)+2+f_{swap})+...+(3+1+3+(n-(n-1))(2+3+1)+2+f_{swap}) = 3+12(n-1)+6s_{selection} = 3n^2+8n-8 \approx 3n^2$

Сортировка вставками

Внутренний цикл будет выполняться: $1, 2, 3, ..., n-1$ раз. Эта последовательность является арифметической прогрессией и ее можно записать как:

$$s_{insert} = \frac{(n-1)n}{2}$$

Лучший случай (массив отсортирован): $f_{insertion} = 2 + (n-1)(2 + 2 + 2 + 4 + 3) = 13n - 11 \approx 13n$

Худший случай (массив отсортирован в порядке, обратном нужному, т.е. каждый раз будет выполняться тело условного оператора): $f_{insertion} = 2 + (2 + 2 + 4 + 1(5 + 4) + 3) + (2 + 2 + 4 + 2(5 + 4) + 3) + ... + (2 + 2 + 4 + (n-1)(5 + 4) + 3) = 2 + 11(n-1) + 9 * s_{insert} = \frac{9}{2}n^2 + \frac{13}{2}n - 9 \approx \frac{9}{2}n^2$

2.4 Вывод

В данном разделе были рассмотрены схемы алгоритмов сортировки массива, введена модель оценки трудоемкости алгоритма и были рассчитаны трудоемкости алгоритмов.

3 Технологический раздел

В данном разделе даны общие требования к программе, средства реализации и реализация алгоритмов.

3.1 Общие требования к программе

Требования к вводу:

1) вводится размер массива;

2) вводятся или автоматически генерируется массив.

Требования к программе:

1) при вводе неправильных размеров массива программа не должна завершаться аварийно;

2) должна выполняться корректная сортировка массива.

3.2 Средства реализации

В качестве языка программирования был выбран C#[2], так как я знаком с данным языком программирования, имею представление о способах тестирования программы. Средой разработки Visual Studio.[3] Для замеров процессорного времени используется функция Stopwatch.[4][5]

3.3 Сведения о модулях программы

Программа состоит из:

1) Program.cs - главный файл программы, в котором располагается точка входа в программу;

2) Array.cs - файл класса Array;

3) Sort.cs - файл класса Sort. В нем находятся алгоритмы сортировки массивов.

3.4 Листинг кода программы

В листинге 1 реализован класс Array. Он используется для работы с массивами.

Листинг 1 – Класс Array для работы с массивами

```
class Array
{
    private int[] array;
    private int n;
    public Array() { }

    public Array(int n)
    {
        this.n = n;
        array = new int[n];
    }

    public int N
    {
        get { return n; }
        set { if (value > 0) n = 0; }
    }

    public int this[int i]
    {
        get { return array[i]; }
        set { array[i] = value; }
    }

    public void Copy(Array arr)
    {
        for (int i = 0; i < n; i++)
        {
            arr[i] = array[i];
```

```
30              }
31          }
32
33          public void Read()
34          {
35              for (int i = 0; i < n; i++)
36              {
37                  Console.Write(array[i] + "\t");
38              }
39              Console.WriteLine();
40          }
41
42          public void Fill()
43          {
44              Random rand = new Random();
45              for (int i = 0; i < n; i++)
46              {
47                  array[i] = rand.Next(100);
48              }
49          }
50      }
```

В листинге 2 реализован алгоритм сортировки пузырьком.

Листинг 2 – Алгоритм сортировки пузырьком

```
1       public static void BubbleSort(Array arr)
2       {
3           int len = arr.N;
4           for (int indI = 0; indI + 1 < len; indI++)
5           {
6               for (int indJ = 0; indJ + 1 < len - indI; indJ++)
7               {
8                   if (arr[indJ + 1] < arr[indJ])
9                   {
10                      int tmp = arr[indJ];
```

```
                    arr[indJ] = arr[indJ + 1];
                    arr[indJ + 1] = tmp;
                }
            }
        }
    }
```

В листинге 3 реализован алгоритм сортировки выбором.

Листинг 3 – Алгоритм сортировки выбором

```
    public static void SelectionSort(Array arr)
    {
        int len = arr.N;
        for (int i = 0; i < len - 1; i++)
        {
            int select = i;
            for (int j = i + 1; j < len; j++)
            {
                if (arr[j] < arr[select])
                {
                    select = j;
                }

            }
            int tmp = arr[select];
            arr[select] = arr[i];
            arr[i] = tmp;
        }
    }
```

В листинге 4 реализован алгоритм сортировки вставками.

Листинг 4 – Алгоритм сортировки вставками

```
    public static void InsertionSort(Array arr)
    {
        int len = arr.N;
```

```
4           for (int i = 1; i < len; i++)
5           {
6               int x = arr[i];
7               int j = i - 1;
8               for (; j >= 0 && arr[j] > x; j--)
9               {
10                  arr[j + 1] = arr[j];
11              }
12              arr[j + 1] = x;
13          }
14      }
```

3.5 Вывод

В данном разделе были представлены сведения о модулях программы, а также реализованы три алгоритма сортировки массивов.

4 Экспериментальный раздел

В данном разделе представлены результаты работы программы и приведен анализ времени работы каждого из алгоритмов.

4.1 Примеры работы программы

На рисунке 4 представлен результат работы алгоритмов.

```
1 - Сортировать
2 - Тест
Ввод: 1
Введите количество элементов массива: 10
10      98      20      45      96      56      24      78      98      64
Bubble:         10      20      24      45      56      64      78      96      98      98
Selection:      10      20      24      45      56      64      78      96      98      98
Insertion:      10      20      24      45      56      64      78      96      98      98
```

Рисунок 4 – Первый результат работы программы

На рисунке 5 представлен результат работы алгоритмов.

```
1 - Сортировать
2 - Тест
Ввод: 1
Введите количество элементов массива: 10
10      98      20      45      96      56      24      78      98      64
Bubble:         10      20      24      45      56      64      78      96      98      98
Selection:      10      20      24      45      56      64      78      96      98      98
Insertion:      10      20      24      45      56      64      78      96      98      98
```

Рисунок 5 – Второй результат работы программы

4.2 Анализ времени работы алгоритмов

Выполняется первый эксперимент. Берутся заранее отсортированные масивы размерами 100, 200, 300, 400 и 500. Элементы массива заполняются произвольно. Результат можно увидеть на рисунке 6.

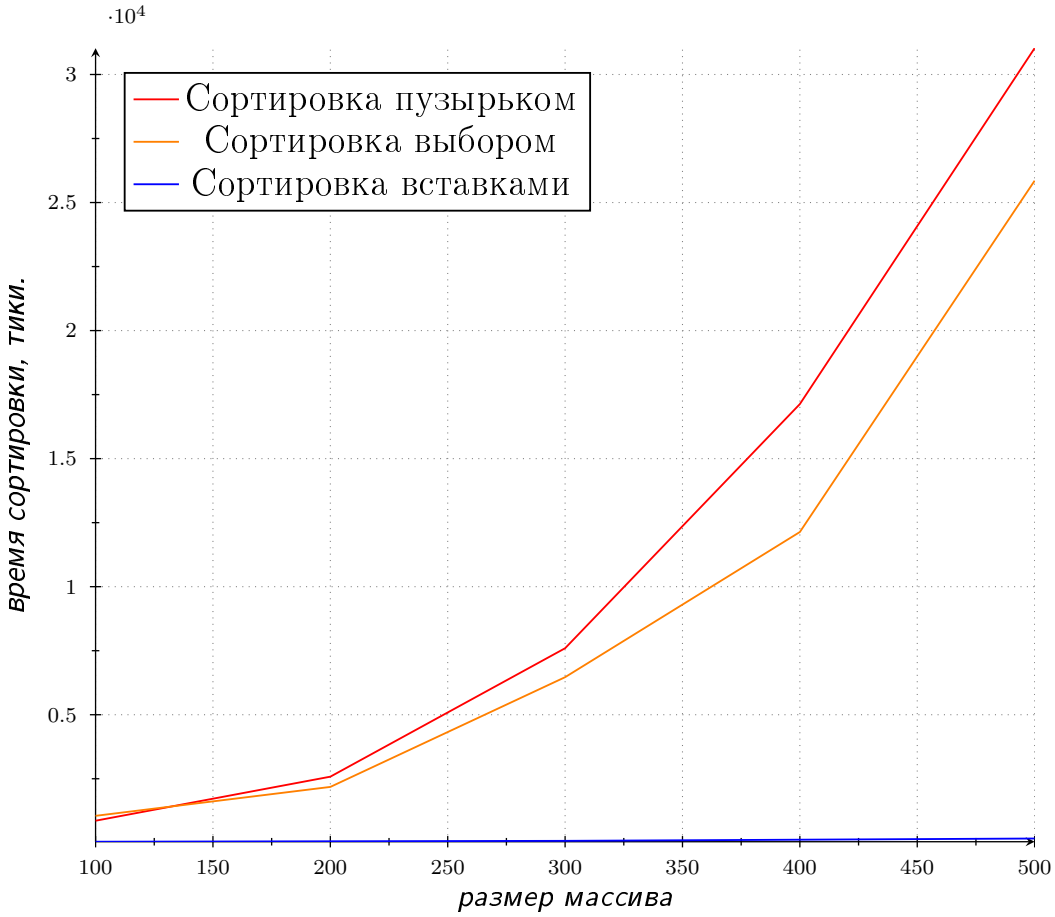

Рисунок 6 – Результаты замеров процессорного времени сортировки уже упорядоченных массивов.

Выполняется второй эксперимент. Берутся заранее отсортированные в обратном порядке масивы размерами 100, 200, 300, 400 и 500. Элементы массива заполняются произвольно. Результат можно увидеть на рисунке 7.

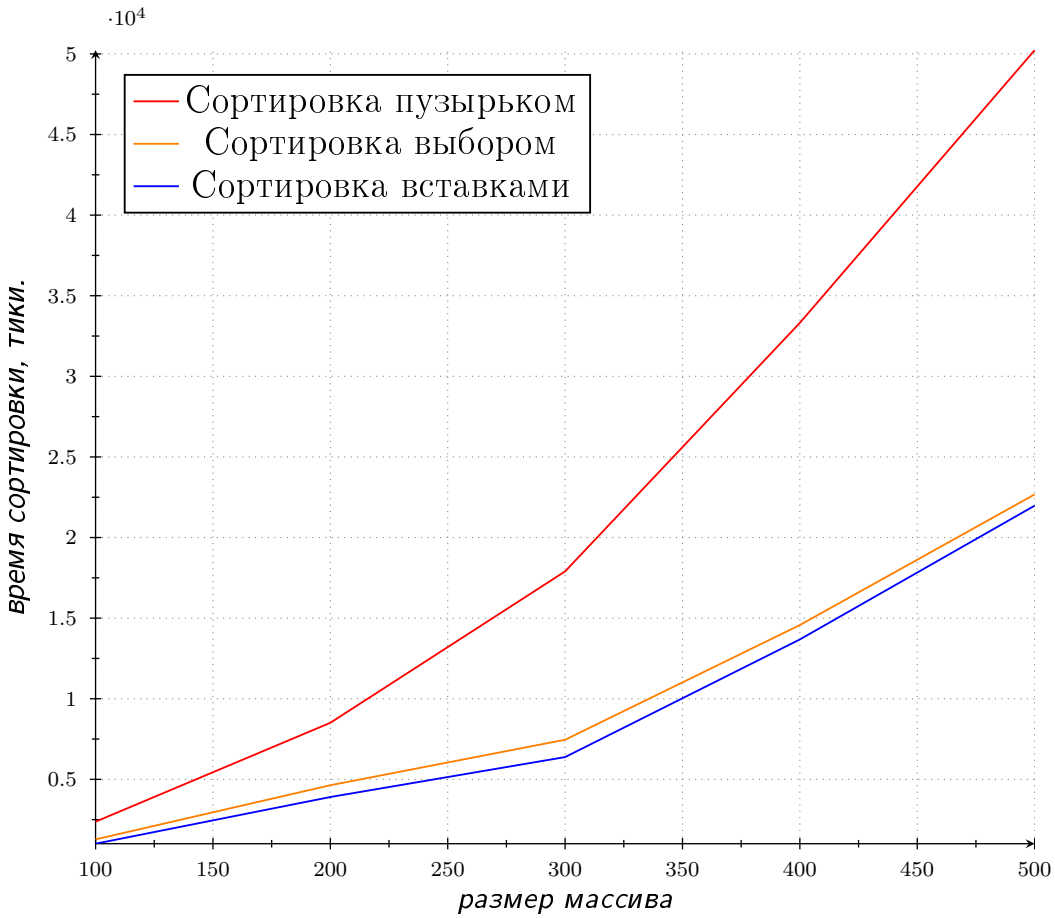

Рисунок 7 – Результаты замеров процессорного времени сортировки масивов, отсортированных в обратном порядке.

Выполняется третий эксперимент. Берутся масивы с произвольным порядком и размерами 100, 200, 300, 400 и 500. Элементы массива заполняются произвольно. Результат можно увидеть на рисунке 8.

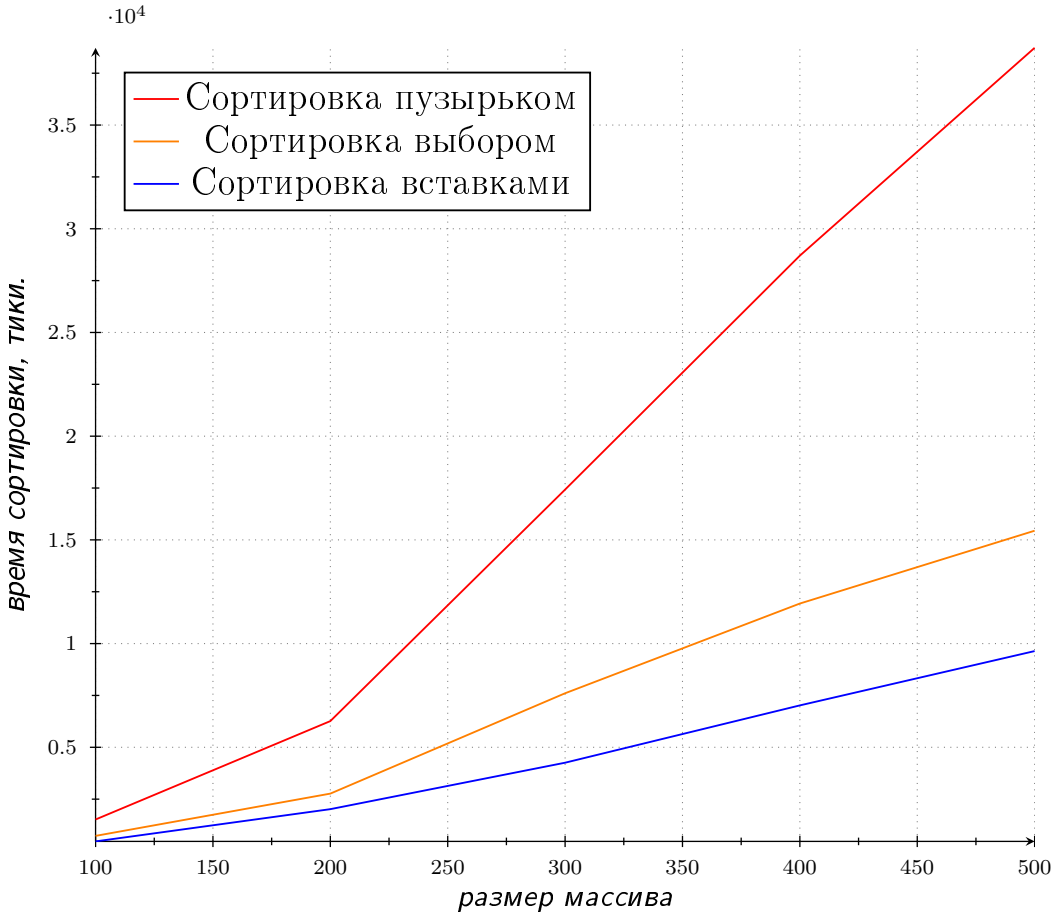

Рисунок 8 – Результаты замеров процессорного времени сортировки масивов.

4.3 Вывод

Результаты тестирования показывают, что во всех трес случаях порядка массива самым быстрым является алгоритм сортировки вставками. Самым медленным оказался алгоритм сортировки пузырьком. Сортировка выбором немного медленнее, чем сортировка вставками, в случаях упорядоченного и неупорядоченного заполнения массива. В случае обратного порядка массива - время работы сортировки выбором схоже с временем работы сортировки вставками.

Заключение

В ходе выполнения лабораторной работы были изучены алгоритмы сортировки массивов: пузырьком, выбором и вставками. Были даны теоритические оценки алгоритмов умножения матриц. Была оценена трудоемкость алгоритмов. Также сравнили время работы алгоритмов, в результате которого стало понятно, что алгоритм сортировки вставками примерно в 3 раза эффективнее по времени сортировки пузырьком и в 1.5 раза быстрее, что сортировка выбором.

Литература

1. Описание алгоритмов сортировки и сравнение их производительности. -URL: https://habr.com/ru/post/335920/ (дата обращения: 22.10.2020). -Текст: электронный.

2. Документация по C#. -URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 01.10.2020). -Текст: электронный.

3. Документация по семейству продуктов Visual Studio. -URL: https://docs.microsoft.com/ru-ru/visualstudio/?view=vs-2019 (дата обращения: 01.10.2020). -Текст: электронный.

4. Stopwatch Класс. -URL: https://goo.su/2e99 (дата обращения: 01.10.2020). -Текст: электронный.

5. Под капотом у Stopwatch. -URL: https://habr.com/ru/post/226279/ (дата обращения: 01.10.2020). Текст: электронный.