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.В. Сусликов

(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 вставками;
3) сортировка выбором.

В лабораторной работе требуется:

1) изучить алгоритмы сортировки массивов;
2) дать теоритическую оценку алгоритмам сортировки: пузырьком, вставками, выбором;
4) реализовать три алгоритма сортировки массивов;
5) сравнить алгоритмы сортировок.

1 Аналитический раздел

В данном разделе представлены математические описания алгоритмов сортировки массивов.

1.1 Алгоритм сортировки пузырьком

Сортировка пузырьком – простейший алгоритм сортировки, применяемый чисто для учебных целей. Практического применения этому алгоритму нет, так как он не эффективен, особенно если необходимо отсортировать массив большого размера. К плюсам сортировки пузырьком относится простота реализации алгоритма.

Алгоритм сортировки пузырьком сводится к повторению проходов по элементам сортируемого массива. Проход выполняет внутренний цикл. За каждый проход сравниваются два соседних элемента, и если порядок неверный элементы меняются местами. Внешний цикл будет работать до тех пор, пока массив не будет отсортирован. Таким образом внешний цикл контролирует количество срабатываний внутреннего цикла Когда при очередном проходе по элементам массива не будет совершено ни одной перестановки, то массив будет считаться отсортированным.

1.2 Алгоритм сортировки вставками

Сортировка вставками — достаточно простой алгоритм. Как в и любом другом алгоритме сортировки, с увеличением размера сортируемого массива увеличивается и время сортировки.

Сортируемый массив можно разделить на две части — отсортированная часть и неотсортированная. В начале сортировки первый элемент массива считается отсортированным, все остальные — не отсортированные. Начиная со второго элемента массива и заканчивая последним, алгоритм вставляет неотсортированный элемент в нужную позицию в отсортированной части. Таким

образом, за один шаг сортировки отсортированная часть массива увеличивается на один элемент, а неотсортированная часть уменьшается на один.

1.3 Алгоритм сортировки выбором

Сотрировка выбором - один из простоый алгоритмов. удя по названию сортировки, необходимо что-то выбирать (максимальный или минимальный элементы массива). Алгоритм сортировки выбором находит в исходном массиве максимальный или минимальный элементы, в зависимости от того как необходимо сортировать массив, по возрастанию или по убыванию. Если массив должен быть отсортирован по возрастанию, то из исходного массива необходимо выбирать минимальные элементы. Если же массив необходимо отсортировать по убыванию, то выбирать следует максимальные элементы.

Допустим необходимо отсортировать массив по возрастанию. В исходном массиве находим минимальный элемент, меняем его местами с первым элементом массива. Уже, из всех элементов массива один элемент стоит на своём месте. Теперь будем рассматривать не отсортированную часть массива, то есть все элементы массива, кроме первого. В неотсортированной части массива опять ищем минимальный элемент. Найденный минимальный элемент меняем местами со вторым элементом массива и т. д. Таким образом, суть алгоритма сортировки выбором сводится к многократному поиску минимального (максимального) элементов в неотсортированной части массива.

Вывод

Можно сделать вывод, что, помимо приведенных выше алгоритмов сортировки массива, есть множество других не менее или даже более эффективных.

2 Конструкторский раздел

В данном разделе представлены схемы алгоритмов и дана оценка их трудоемкости.

2.1 Схемы алгоритмов

Ниже на Рисунке 1 представлена схема алгоритма сортировки пузырьком.

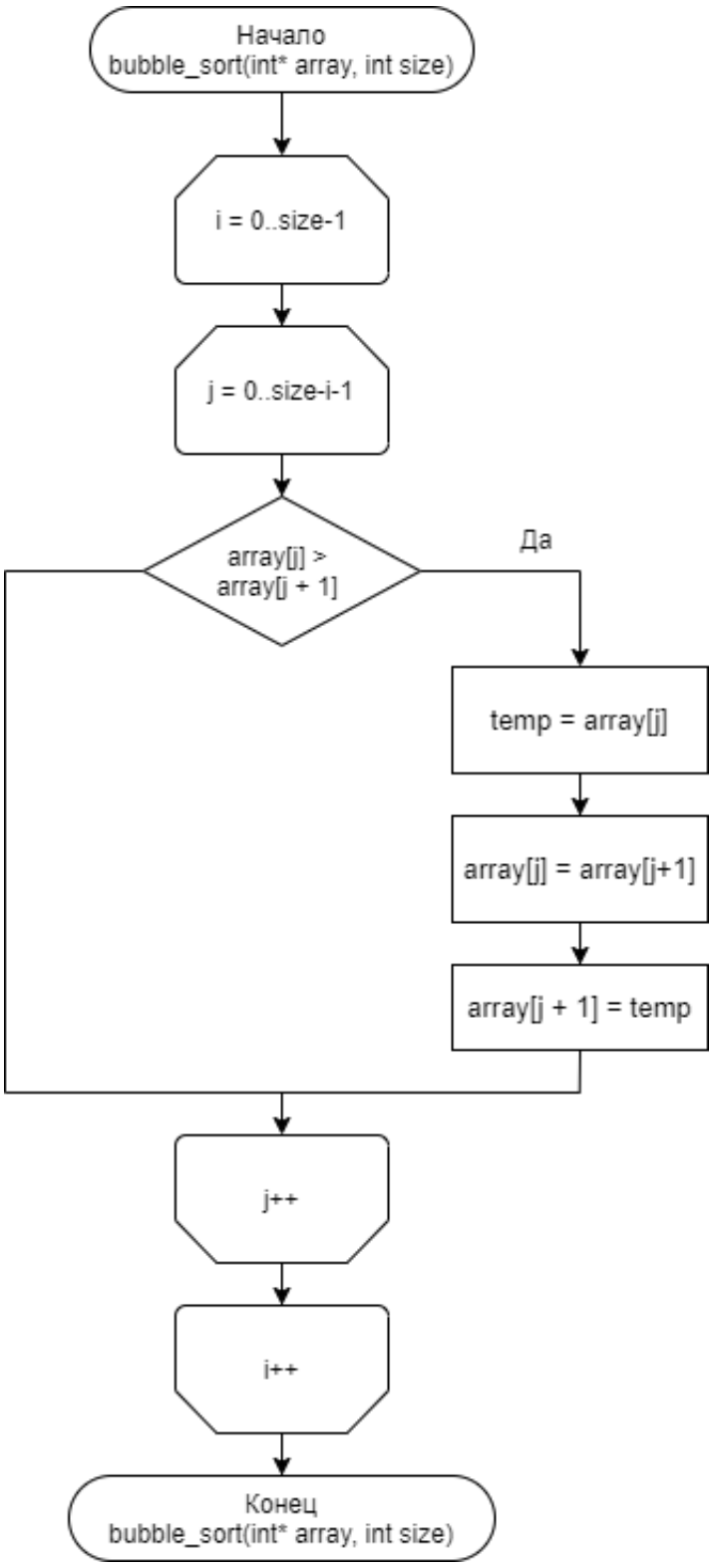

Рисунок 1 - Схема алгоритма сортировки пузырьком

Ниже на Рисунке 2 изображена схема алгоритма сортировки вставками.

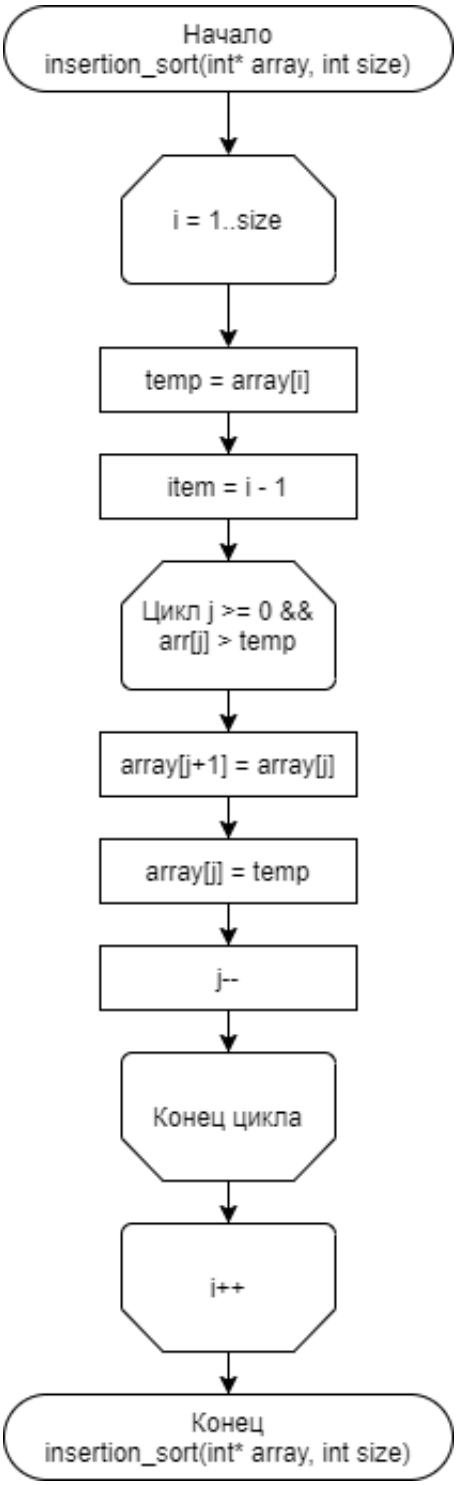

Рисунок 2 - Схема алгоритма сортировки вставками

Ниже на Рисунке 3 показана схема алгоритма сортировки выбором.

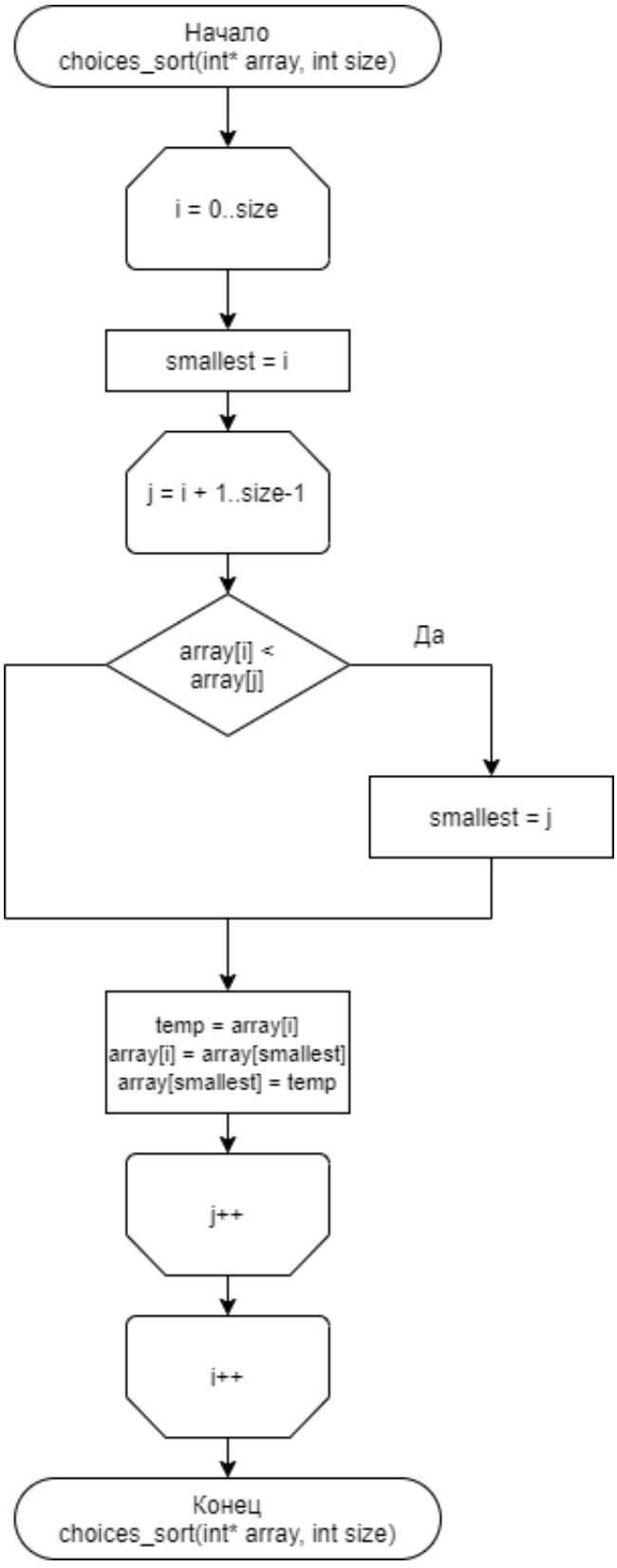

Рисунок 3 - Схема алгоритма сортировки выбором

2.2 Трудоемкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов:

1) Операции, чья стоимость 1: $=, +, *, \simeq, <, >, \geq, \leq, ==, !=, [], +=, -=, *=, /=, ++, --$;

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

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

   Оценим трудоемкость алгоритмов.

2.2.1 Трудоемкость общей первичной проверки

Трудоемкость проверки $if(size <= 0)return$; - 1.

2.2.2 Трудоемкость алгоритма сортировки 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:

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

Расчет лучшего случая (массив отсортирован):

3 Технологическая часть

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

3.1 Общие требования к программе

Требования к вводу:

1) возможность ввода размера массива;

2) ввод или автоматическая генерация массива.

Требования к программе:

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

2) корректная сортировка массива.

3.2 Средства реализации

В лабораторной работе был использован язык C++[1], так как он известен, и на нём было написано множество предыдущих работ.

Среда разработки - Qt[2].

Для замеров процессорного времени была использована функция $clock()$[3].

3.3 Реализация алгоритмов

В Листинге 1 показана реализация алгоритма сортировки пузырьком.

Листинг 1 - Алгоритма сортировки пузырьком

```
void bubble_sort(int* array, int size)
{
  if (size <= 0)
  {
    std::cout << "Incorrect array" << std::endl;
    return;
  }

  for (int i = 0; i < size - 1; i++)
  {
    for (int j = 0; j < size - i - 1; j++)
    {
      if (array[j] > array[j + 1])
      {
        int temp = array[j];
        array[j] = array[j + 1];
        array[j + 1] = temp;
      }
    }
  }
}
```

В Листинге 2 описана реализация алгоритма сортировки вставками.

Листинг 2 - Алгоритм сортировки вставками

```
void insertion_sort(int *array, int size)
{
  if (size <= 0)
  {
    std::cout << "Incorrect array" << std::endl;
    return;
  }

  for (int i = 1; i < size; i++)
  {
    int temp = array[i];
    int item = i - 1;
    while(item >= 0 && array[item] > temp)
    {
      array[item + 1] = array[item];
      array[item] = temp;
      item--;
    }
  }
}
```

В Листинге 3 описана реализация алгоритма сортировки выбором.

Листинг 3 - Алгоритм сортировки выбором

```
void choices_sort(int* array, int size)
{
  if (size <= 0)
  {
    std::cout << "Incorrect array" << std::endl;
    return;
  }

  for (int i = 0; i < size - 1; i++)
  {
    int smallest = i;
    for (int j = i + 1; j < size; j++)
    {
      if (array[j] < array[smallest])
      smallest = j;
    }
    int temp = array[i];
    array[i] = array[smallest];
    array[smallest] = temp;
  }
}
```

Вывод

По итогу, написанная программа соотвествует всем описанным выше требованиям, алгоритмы были реализованы на C++, так как данный язык известен, выполнено на нём много прошлых работ.

4 Экспериментальный раздел

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

4.1 Примеры работы программы

На Рисунке 4 представлены меню и ввод массива.

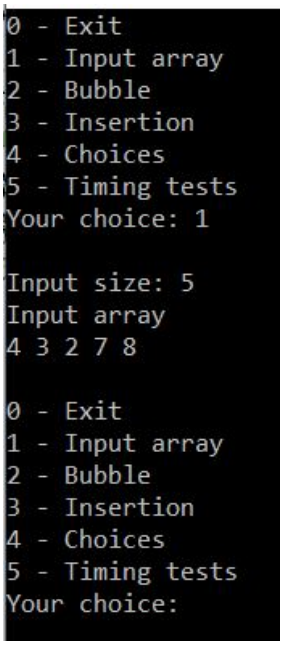

Рисунок 4 - Меню выбора и ввод массива

На Рисунке 5 можно увидеть пример работы всех алгоритмов

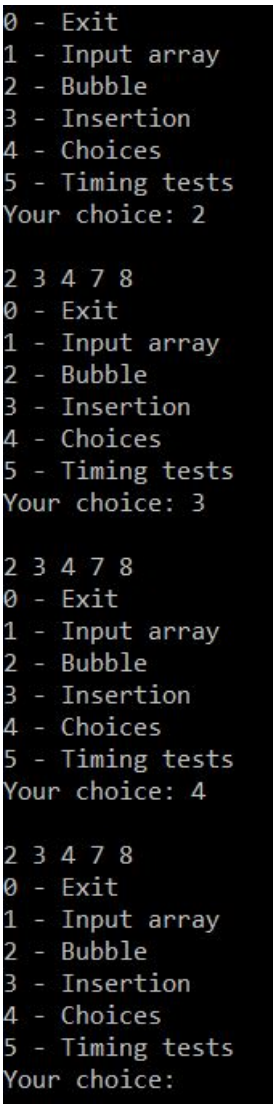

Рисунок 5 - Пример работы всех алгоритмов

4.2 Анализ времени работы алгоритмов

Эксперементы проводятся на массивах длин 100, 200, 300, 400, 500 элементов. Элементы массива заполняются произвольно.

В первом случае берутся заранее отсортированные массивы. Графики зависимости времени работы алгоритмов от размеров массивов изображены на Рисунке 6.

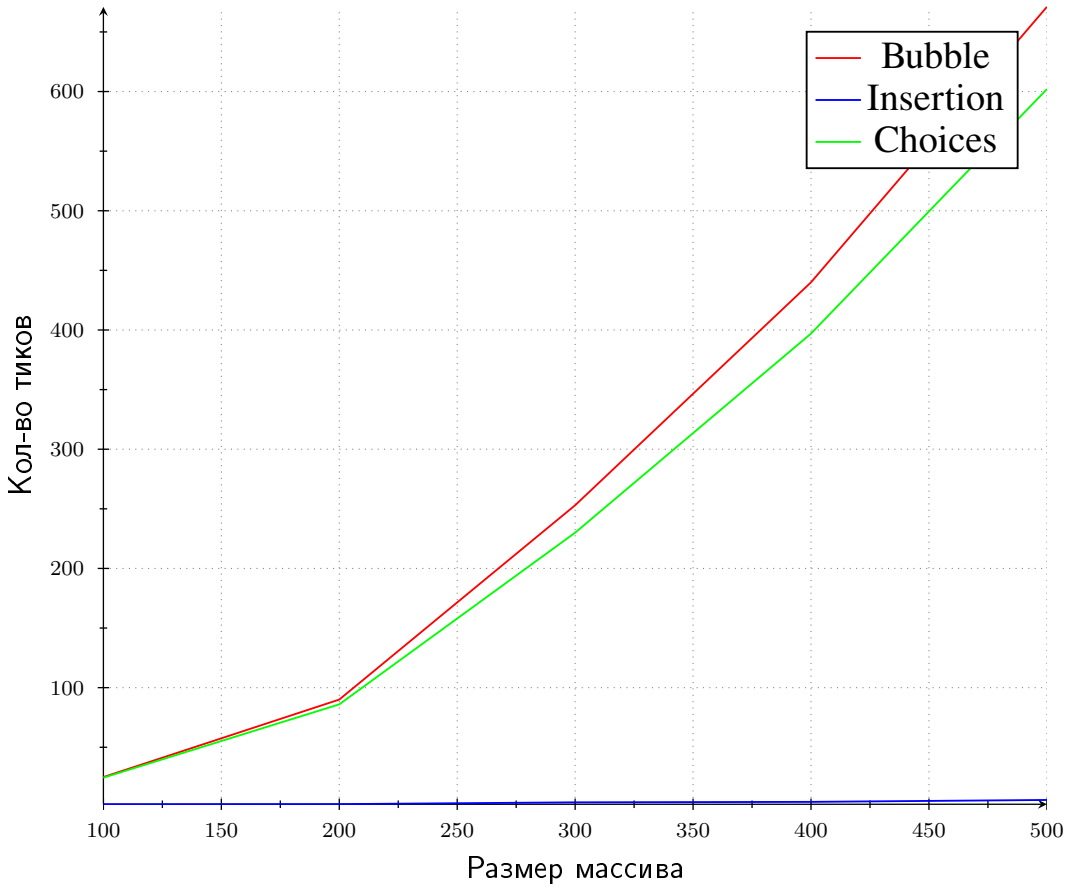

Рисунок 6 - Графики зависимости времени работы алгоритмов от размеров при заранее отсортированных массивах

Во втором случае берутся обратно отсортированные массивы. Графики зависимости времени работы алгоритмов от размеров массивов изображены на Рисунке 7.

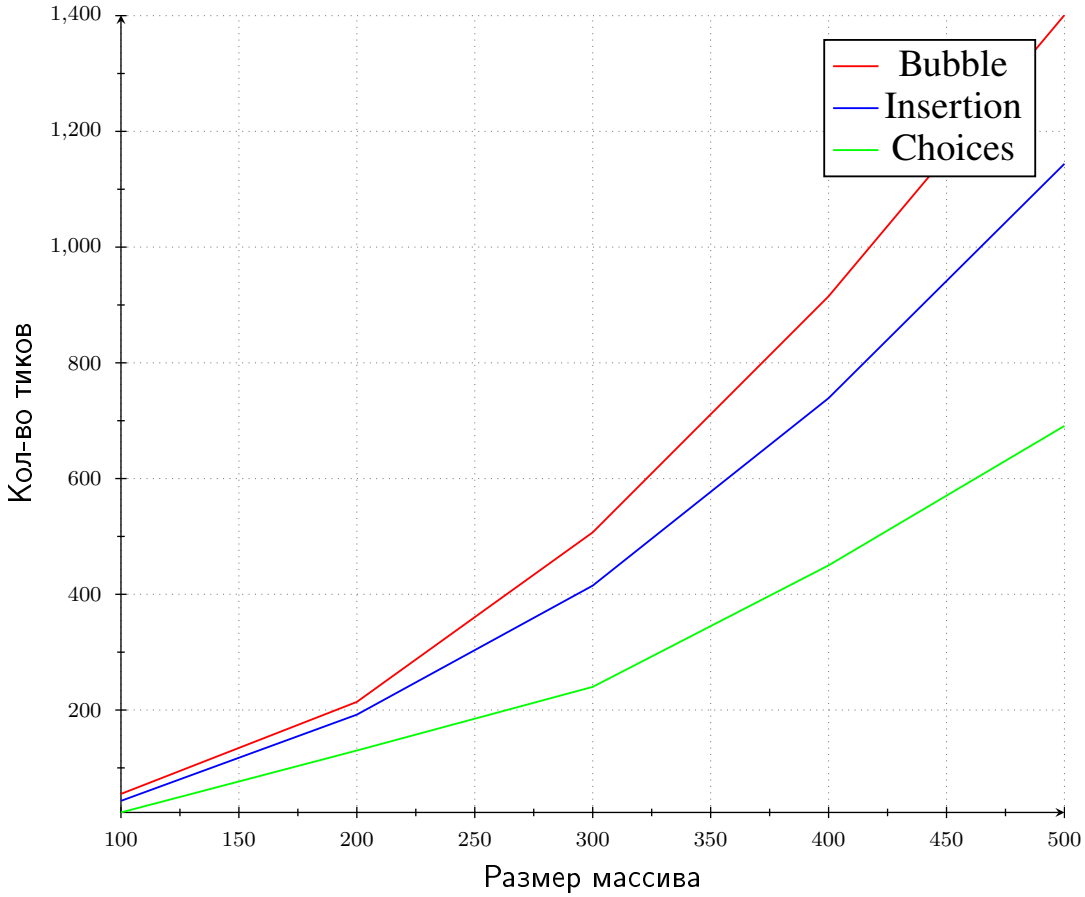

Рисунок 7 - Графики зависимости времени работы алгоритмов от размеров при заранее обратно отсортированных массивах

В третьем случае берутся произвольные массивы. Графики зависимости времени работы алгоритмов от размеров массивов изображены на Рисунке 8.

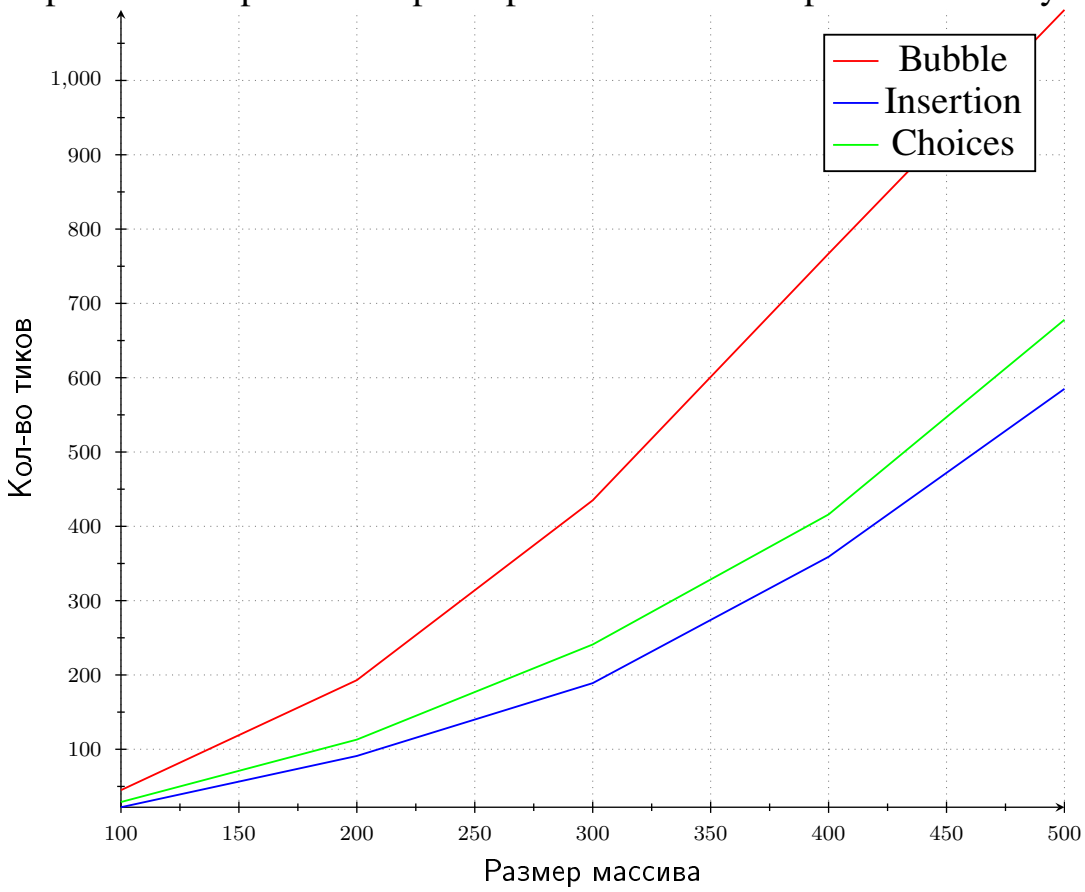

Рисунок 8 - Графики зависимости времени работы алгоритмов от размеров при произвольно заданных массивах

Вывод

По итогу, все алгоритмы дают верный результат. По значениям замеров можно сделать вывод, что алгоритм сортировки вставками работает быстрее других алгоритмов в большинстве случаях. При обратно отсортированных массивах алгоритм сортировки вставками работает медленнее сортировки выбором. Алгоритм сортировки пузырьком оказался медленнее всех.

Заключение

В ходе выполнения лабораторной работы были изучены алгоритмы сортировки массивов: пузырьком, вставками и выбором. Были даны теоритические оценки алгоритмов сортировки массиов. Была дана оценека трудоемкости алгоритмов. Также было произведено сравнение времени работы алгоритмов, в результате которого стало очевидно, что алгоритм сортировки вставками примерно в 2 раза быстрее сортировки пузырьком и в 1.2 раза - сортировки выбором (за исключением ситуации обратно отсортированного массива).

Литература

1) Бьерн Страуструп. Язык программирования C++. -URL:

   https://codernet.ru/books/c_plus/bern_straustrup_yazyk_programmirovaniya_c_specialnoe_izdanie/

   (дата обращения: 01.10.2020). Текст: электронный.

2) Qt. -URL:

   https://www.qt.io/ (дата обращения: 01.10.2020). Текст: электронный.

3) Функция $clock$. -URL:

   https://docs.microsoft.com/ru-ru/cpp/c-runtime-library/reference/clock?view=vs-2019 (дата обращения: 01.10.2020). Текст: электронный.

4) Дж. Макконнелл. Основы современных алгоритмов.
   2-е дополненное издание
   Москва: Техносфера, 2004. - 368с. ISBN 5-94836-005-9
   с. 130 - 133