Государственное образовательное учреждение высшего профессионального образования
«Московский государственный технический университет имени Н. Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Дисциплина: анализ алгоритмов

Лабораторная работа №3

Студент: Власова Е. В., ИУ7-54

2019 г.

Содержание

Введение

Порядок – это гармоничное и предсказуемое состояние и расположение чего-либо. Когда человек что-то ищет, ему будет проще искать в уже отсортированной по какому-либо признаку сущности. Если программисту требуется работать с какой-либо структурой, то он может облегчить себе жизнь алгоритмами сортировки. Алгоритмы сортировки – это алгоритмы для упорядочивания элементов в какой-либо структуре. Существует огромное количество разных сортировок и их вариаций: как совсем неэффективных, так и немного полезных. Так как данные могут храниться в разных структурах, то и сами алгоритмы могут в зависимости от этого немного отличаться друг от друга. Но это неважно, так как главное понять саму суть способа сортировки, и вы сможете отсортировать всё что угодно, вы будете всемогущ, и никто не будет преградой для вас. В этой лабораторной работе будут рассмотрены и разработаны алгоритмы сортировки массивов.

1 Аналитическая часть

Алгоритмы сортировки – важная часть жизни любого программиста. В этом разделе будут рассмотрены алгоритмы сортировки, использующиеся в этой работе: сортировка пузырьком, сортировками вставками и коктейльная сортировка.

1.1 Описание алгоритмов

Описание рассматриваемых алгоритмов сортировки.

Сортировка пузырьком

Один самых простых алгоритмов сортировки в плане реализации и понимания. Эффективен лишь для массивов малых размеров.

Ссылаясь на [1], идея данной сортировки заключается в попарном сравнении соседних элементов, начиная с нулевого в массиве. Больший элемент при этом в конце первой итерации оказывается на месте последнего элемента массива, и в следующих итерациях мы его уже не сравниваем его с остальными элементами (то есть у нас будет n-1 сравнений). Затем таким же образом мы находим второй по максимальности элемент и ставим его на предпоследнее место, и т. д. После всех итераций получится, что на месте нулевого элемента окажется элемент с наименьшим числовым значением, а на месте последнего – с наибольшим числовым значением. Таким образом у нас как бы “всплывают” элементы от большего к меньшему.

Сортировка вставками

Согласно [2], это алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов.

Полагаем, что массив из первого элемента, то есть из одного элемента, упорядочен. Действительно, одно число всегда является упорядоченным. Затем выбираем следующий элемент i, и вставляем его в нужную позицию в уже отсортированном массиве, и так до тех пор, пока набор входных данных не будет исчерпан.

Коктейльная сортировка (Шейкер сортировка)

Является усовершенствованным алгоритмом сортировки пузырьком. Его описание возьмём из [1].

Анализируя метод пузырьковой сортировки, можно заметить два факта:

- если при движении по части массива перестановки не происходят, то эта часть массива уже отсортирована и, следовательно, ее можно исключить из рассмотрения.
- 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 модификациям в методе пузырьковой сортировки.
- От последней перестановки до конца (начала) массива находятся отсортированные элементы. Учитывая данный факт, просмотр осуществляется не до конца (начала) массива, а до конкретной позиции. Границы сортируемой части массива сдвигаются на 1 позицию на каждой итерации.
- Массив просматривается поочередно справа налево и слева направо.
- Просмотр массива осуществляется до тех пор, пока все элементы не встанут в порядке возрастания (убывания).
- Количество просмотров элементов массива определяется моментом упорядочивания его элементов.

1.2 Задание на выполнение лабораторной работы

- Реализовать три алгоритма сортировки;
- Оценить их трудоёмкость;
- Исследовать время работы алгоритмов.

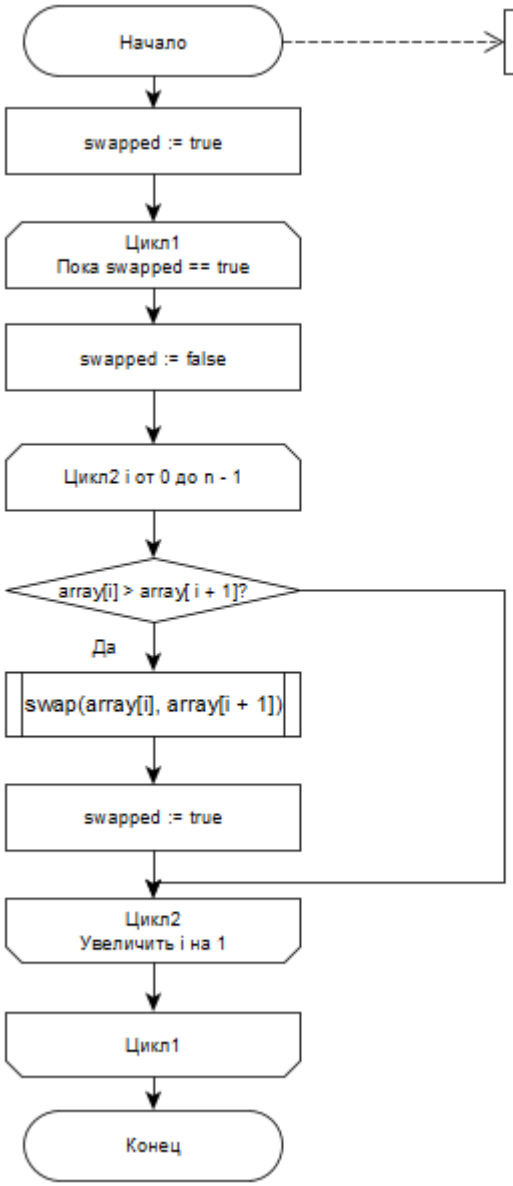
2 Конструкторская часть

В данном разделе представлены блок-схемы рассматриваемых алгоритмов.

2.1 Разработка алгоритмов

Рассматриваемые алгоритмы представлены в виде блок-схем, оценена их трудоёмкость.

2.1.1 Сортировка пузырьком



Fтело = 1 + Fвнеш

Fвнеш = 1 + N * (1 + 1) + Fвнут

Fвнут = N – 1 (2 +

+1 + 2 + 1 +

+ 2 +

+ 1 +

+ 1)

Оператор	Количество
==	N

=	2N
<	2N+1
>	2N+1
+	3N + 1

Трудоёмкость алгоритма

Тип операции	Стоимость
Арифметические операции	1
Операторы сравнения	1
Логические операторы	1
Операторы присваивания	1
Другие операторы	1
Функция swap(x, y)	2

Лучший случай

Массив полностью отсортирован, не произошло ни одного обмена за один поход – выход из цикла.

Оценим трудоёмкость:

Fвнут = N – 1(2 + 1) = 3N - 3

Fвнеш = 1 + (1 + 1) + Fвнут = 3N

Fтело = 1 + Fвнеш = 3N + 1

Общая сложность алгоритма: O(N)

Худший случай

Массив отсортирован в обратном порядке, при каждом проходе происходит обмен.

Оценим трудоёмкость:

Fвнут = N^2 / 2 * (4 + 2 + 1 + 1 + 2) = $5N^2$

Fвнеш = N * (1 + 1) +Fвнут = 2N + $5N^2$

Fтело = 1 + Fвнеш = 2N + $5N^2$ + 1

Общая сложность алгоритма: O(N^2)

2.1.2 Сортировка вставками

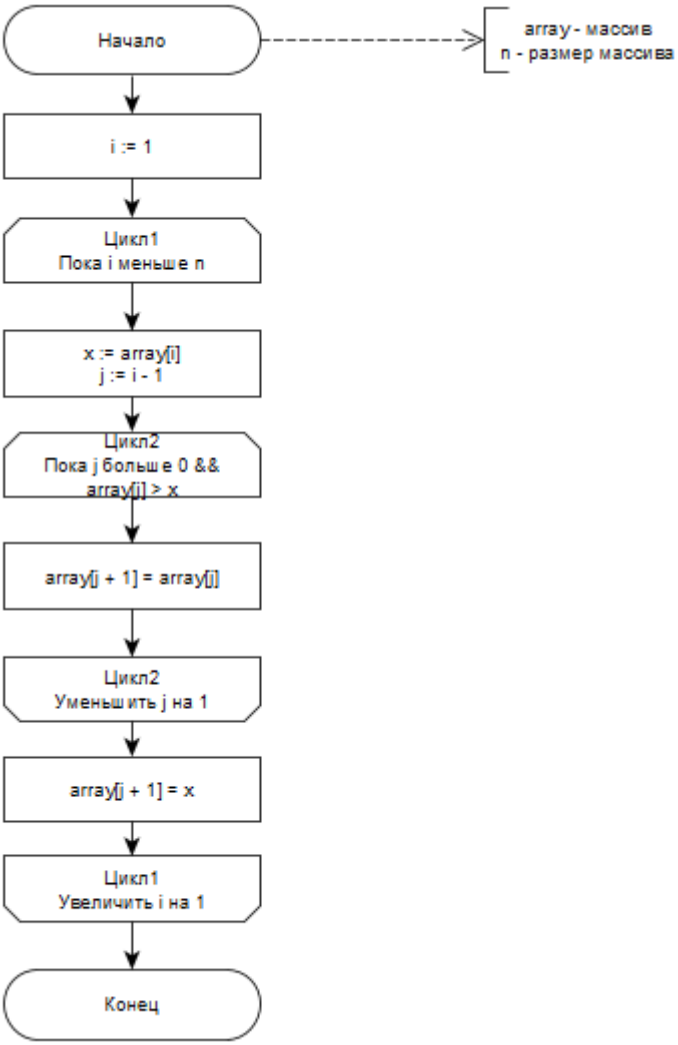


Рисунок 2.1.2 Сортировка вставками

Анализ трудоёмкости

Лучший случай

Сложность сортировки вставками для лучшего случая оценивается как O(N).

Худший случай

Сложность сортировки вставками для худшего случая оценивается как $O(N^2)$.

2.1.3 Шейкер сортировка

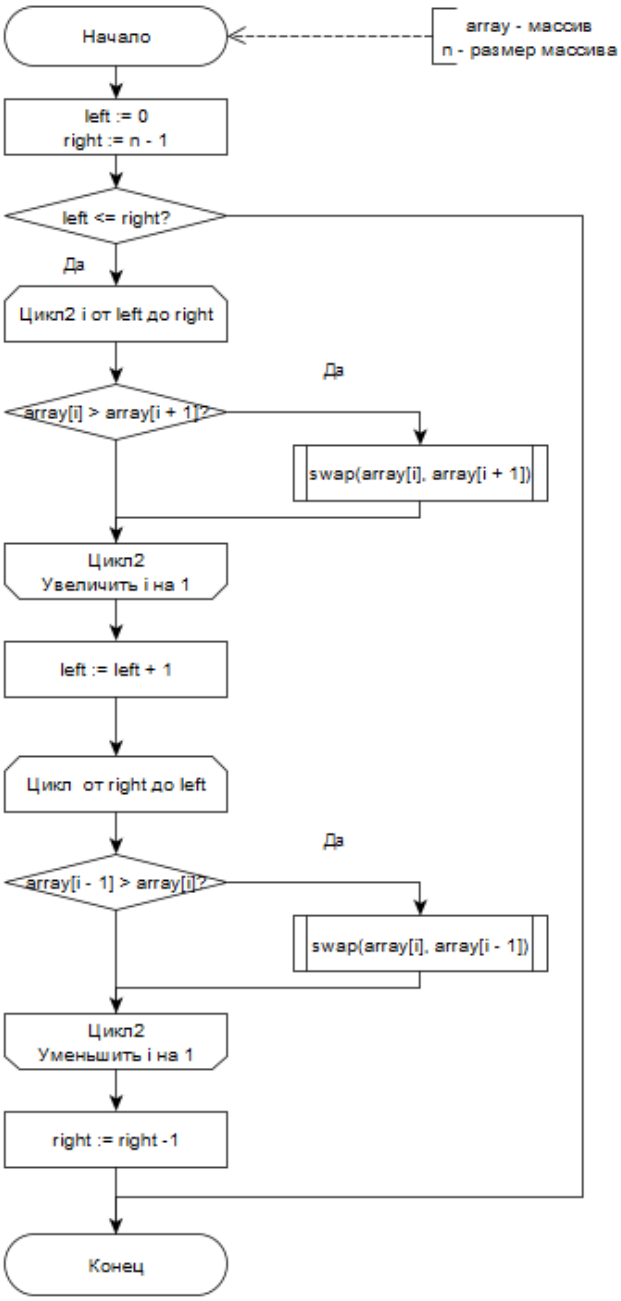


Рисунок 2.1.3 Шейкер-сортировка

Анализ трудоёмкости

Лучший случай

Сложность шейкер-сортировки для лучшего случая оценивается как O(N).

Худший случай

Сложность шейкер-сортировки для худшего случая оценивается как $O(N^2)$.

2.2 Выводы по конструкторскому разделу

Были сделаны блок-схемы реализуемых алгоритмов. Для каждого алгоритма был рассмотрен лучший и худший случаи их работы, оценена сложность алгоритмов. Для сортировки пузырьком была оценена трудоёмкость.

3 Технологическая часть

В данном разделе был сделан выбор используемого языка программирования, предоставлен код алгоритмов.

3.1 Требования к программному обеспечению

Программа должна поддерживать пользовательский режим, в котором пользователь может с клавиатуры задать размер массива и значения его элементов; а также операционный режим, в котором пользователь вводит размер массива, он автоматически заполняется, и для каждой сортировки выводится время работы её функции.

3.2 Средства реализации

В качестве языка программирования в данной работе был выбран Forth – конкатентативный язык программирования, в котором программы записываются последовательностью лексем. При написании программы использовался интерпретатор Gforth. Данный ЯП был выбран в целях ознакомления с его особенностями. Стандарт языка описан в [6].

3.3 Листинг кода

Листинг 0: функция для измерения времени в секундах

```
: time: ( "word" -- )
utime 2>R ' EXECUTE
utime 2R> D-
<# # # # # # # [CHAR] . HOLD #S #> TYPE ." seconds" ;
```

Листинг 1: сортировка пузырьком.

```
: [!] ( value index array -- ) swap cells + ! ;
: bubble_sort ( -- )
0
BEGIN
0= ( 0 true -1 false)
WHILE
-1
array_size 1 - 0 ?DO
array I cells + @ array I 1 + cells + @ >
IF
array I cells + @
array I 1 + cells + @
I array [!]
I 1 + array [!]
DROP 0
THEN
LOOP
REPEAT
;
```

Листинг 2: сортировка вставками

```
: [!] ( value index array -- ) swap cells + ! ;
: insertion_sort ( -- )
1
BEGIN
DUP array_size <
WHILE
DUP array SWAP cells + @
OVER 1 -
BEGIN
OVER OVER DUP array SWAP cells + @
SWAP 0>= rot rot < AND
WHILE
dup array SWAP cells + @
OVER 1 + array [!]
1 -
REPEAT
DUP 1 + ROT SWAP array [!] drop
1 +
REPEAT
;
```

Листинг 3: Шейкер-сортировка

```
: [!] ( value index array -- ) swap cells + ! ;
: shaker_sort ( -- )
  0 array_size 1 -
  BEGIN
  OVER OVER <=
  WHILE
  OVER OVER SWAP ?DO
   array I cells + @
   array I 1 + cells + @
   > IF
   array I cells + @
   array I 1 + cells + @
   I array [!]
   I 1 + array [!]
   THEN
   LOOP
  1 -
  OVER OVER ?DO
   array I 1 - cells + @
   array I cells + @
   > IF
   array I 1 - cells + @
   array I cells + @
   I 1 - array [!]
   I array [!]
   THEN
  -1 +LOOP
  SWAP 1 + SWAP
  REPEAT
;
```

3.4 Выводы по технологическому разделу

Был выбран язык программирования Forth, средство реализации языка Gforth. Была определена функция для замера времени в секундах, а также реализованы функции сортировок. Описаны требования к ПО.

4 Экспериментальная часть

В данном разделе предоставлены примеры работы программы и тесты для функций.

4.1 Примеры работы

Пользователь задаёт размер массива, после чего вводит каждый элемент через кнопку Enter.

```
Lenght of array:4
Input numbers: ( number - enter )
Input: 4
Input: 3
Input: 2
Input: 1

Array: 4 3 2 1

INSERTION SORT: 1 2 3 4
BUBBLE SORT:    1 2 3 4
SHAKER SORT:    1 2 3 4
```

4.1 Пример работы программы

```
Lenght of array:3
Input numbers: ( number - enter )
Input: 1
Input: 0
Input: 3

Array: 1 0 3

INSERTION SORT: 0 1 3
BUBBLE SORT:    0 1 3
SHAKER SORT:    0 1 3
```

4.2 Пример работы программы

4.2 Постановка эксперимента

Будем измерять время работы алгоритмов на отсортированных, отсортированных в обратном порядке и случайно сгенерированных массивах. Для замера времени будем использовать раннее приведённую функцию.

4.2.1 Тестирование времени работы функций

Сортировка пузырьком

Размер массива	Время сортировки сортированного массива, сек.	Время сортировки отсортированного в обратном порядке массива, сек.	Время сортировки массива со случайными значениями, сек.
100	0.000	0.000	0.000
200	0.000	0.002	0.001
300	0.000	0.003	0.002
400	0.000	0.005	0.004
500	0.000	0.007	0.006
600	0.000	0.014	0.007
700	0.000	0.018	0.011
800	0.000	0.020	0.015
900	0.000	0.023	0.017
1000	0.000	0.028	0.020

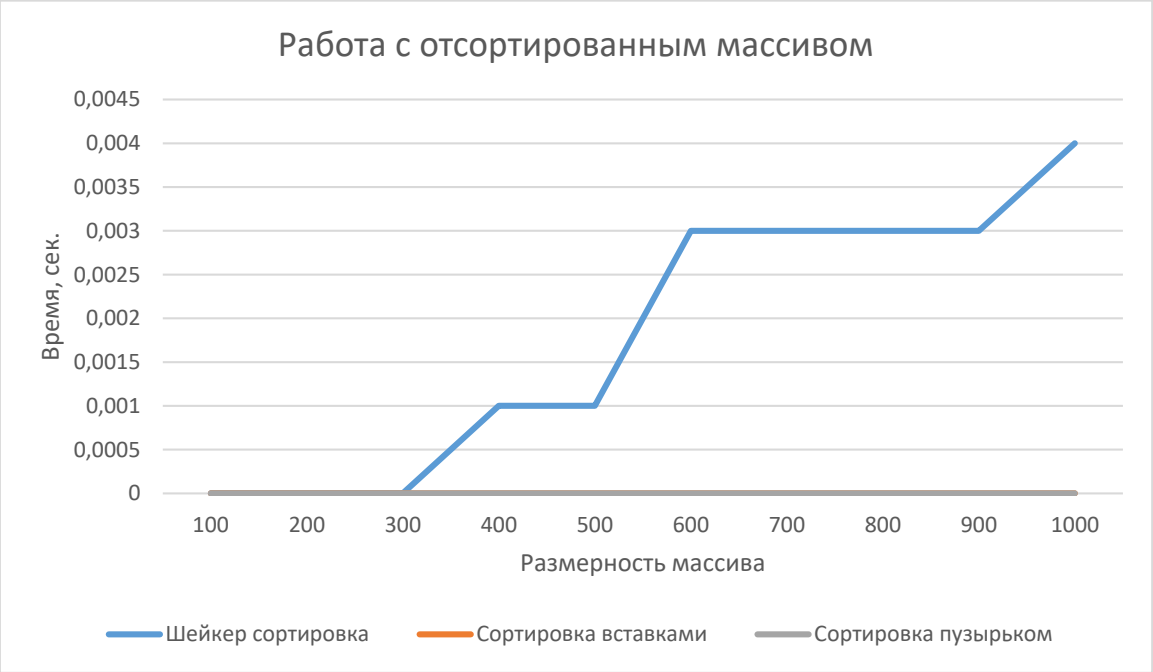
Сортировка вставками

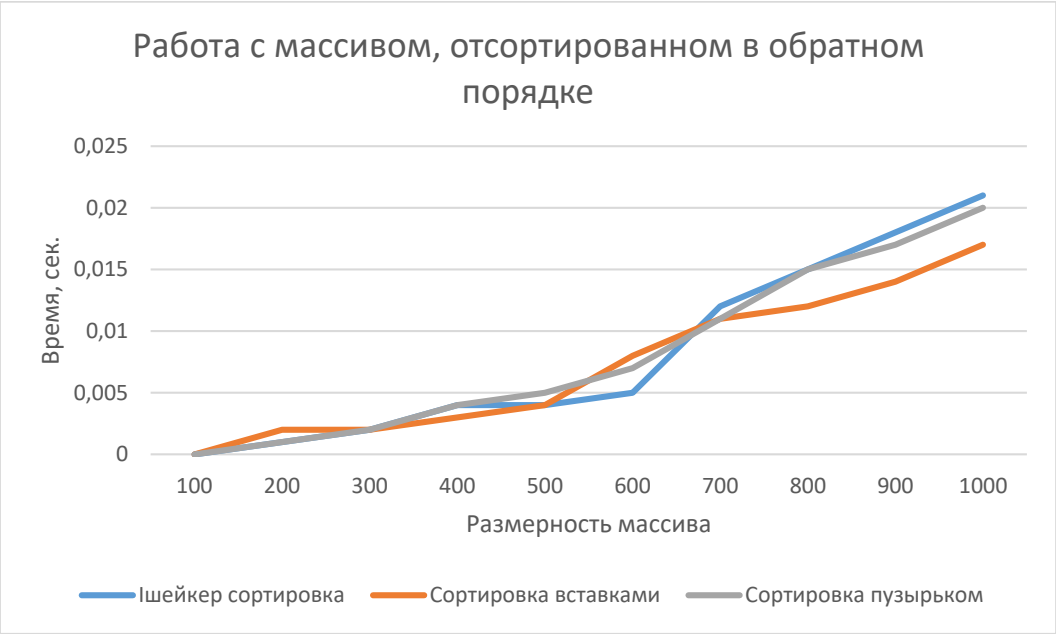
Размер массива	Время сортировки сортированного массива, сек.	Время сортировки отсортированного в обратном порядке массива, сек.	Время сортировки массива со случайными значениями, сек.
100	0.000	0.000	0.000
200	0.000	0.002	0.000
300	0.000	0.002	0.001
400	0.000	0.003	0.001
500	0.000	0.004	0.002
600	0.000	0.008	0.003
700	0.000	0.011	0.005
800	0.000	0.012	0.006
900	0.000	0.014	0.007
1000	0.000	0.017	0.009

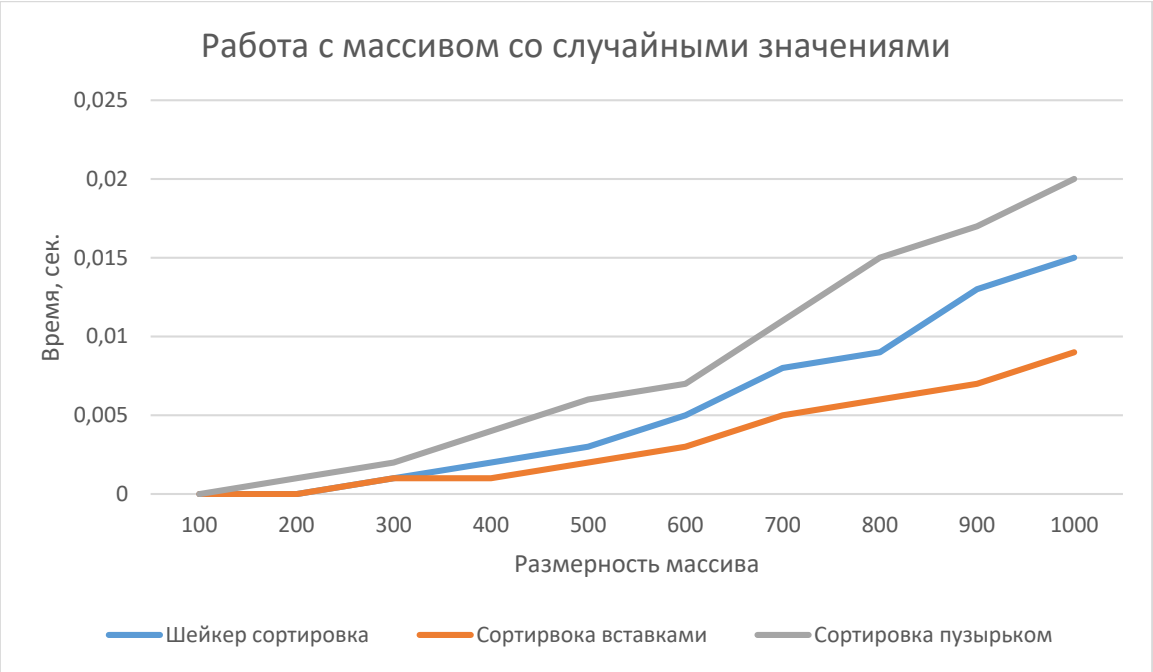
Коктейльная сортировка

Размер массива	Время сортировки сортированного массива, сек.	Время сортировки отсортированного в обратном порядке массива, сек.	Время сортировки массива со случайными значениями, сек.
100	0.000	0.000	0.000
200	0.000	0.001	0.000
300	0.000	0.002	0.001
400	0.001	0.004	0.002
500	0.001	0.004	0.003
600	0.003	0.005	0.005
700	0.003	0.012	0.008
800	0.004	0.015	0.009
900	0.003	0.018	0.130
1000	0.004	0.021	0.150

4.3 Сравнительный анализ на основе экспериментальных данных







4.4 Выводы по экспериментальной части

Были протестированы алгоритмы сортировки на массивах размерами 100…1000 с шагом 100. Рассмотрены отсортированные, отсортированные в обратном порядке массивы и массивы со случайными значениями.

В результате тестирования было получено, что лучшее время сортировки показывают на отсортированных массивах: так, даже при размерности массива в 1000 элементов сортировка пузырьком и сортировка вставками показали значения 0.000 секунд, а коктейльная сортировка – 0.006 секунд. Худшие значения сортировки показывают на обратно отсортированных массивах, причём чем больше размер такого массива, тем медленнее работают сортировки. Худший результат работы на обратно отсортированном массиве в 1000 элементов показала сортировка пузырьком: 0.028 сек, далее шейкер сортировка – 0.021 сек. Лучшее время работы у сортировки вставками – 0.017 сек. При сортировке массивов со случайными значениями время работы алгоритмов такое: массив в 1000 элементов сортировка пузырьком отсортировала за 0.02 сек, сортировка вставками – за 0.006, а коктейльная сортировка за 0.015 сек.

При сравнении времени работы алгоритмов можно сделать вывод, что на размерах массива до 300 элементов время работы алгоритмов приближено к 0 сек. При работе с массивами больших размеров при худшем случае сортировка вставками работает быстрее других алгоритмов – 0.017 сек. Коктейльная сортировка для этого же массива показала время в 0.021 сек., а сортировка пузырьком показала худший результат – 0.028 сек.

Заключение

В ходе выполнения данной лабораторной работы были реализованы три алгоритма сортировки: сортировка пузырьком, сортировка вставками и шейкер-сортировка. Был проведён анализ каждого алгоритма и измерено время работы алгоритмов для массивов разных размеров. Была оценена трудоёмкость алгоритмов.

Были сделаны выводы о времени работы каждого алгоритма – лучшее время работы все сортировки показывают на отсортированных массивах: при обработке массива в 1000 элементов сортировка пузырьком и вставками отработали за 0.000 сек, а шейкер-сортировка – за 0.006 сек. Худшее время алгоритмы сортировки показали при работе с массивами, отсортированными в обратном порядке. Причём чем больше размерность такого массива, тем медленнее работает алгоритм.

Был сделан вывод, что на массивах размерностью до 300 элементов время работы всех трёх алгоритмов близится к 0 сек. При обработке массивов больших размеров, например, 1000, лучший результат показала сортировка вставками - 0.017 сек., худшее – сортировка пузырьком, 0.028 сек. Шейкер-сортировка показала время в 0.021 сек.

Список использованной литературы

1. Knuth, Donald E., Sorting and Searching, The Art of Computer Programming, 3 (2nd ed.), 1998.
2. Алгоритмы.[Электронный ресурс] URL: Sequential and parallel sorting algorithms
3. Т. Кормен. Алгоритмы. Построение и анализ. / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн . – Издательство «Вильямс», 2013.
4. Дж. Макконелл Анализ алгоритмов. Активный обучающий подход. — 3-е дополненное издание. М: Техносфера, 2009.
5. Скиена С. Алгоритмы. Руководство по разработке. 2-е изд.: Пер. с англ. — СПб.: БХВ-Петербург. 2011. — 720 с.: ил.
6. Стандарт Forth, [Электронный ресурс], URL: https://forth-standard.org/
7. Л. Броуди, Начальный курс программирования на языке ФОРТ, 1990.
8. Stephen Pelc, Programming Forth Stephen Pelc, 2005.
9. Leo Brodie, Thinking Forth, 1984.
10. J. L. Bezemer, And so forth, 2004.
11. С. Н. Баранов, Н. Р. Ноздрунов, Язык Форт и его реализации, 1988.
12. Язык Форт. .[Электронный ресурс] URL: https://www.forth.org.ru/
13. Gforth Manual.[Электронный ресурс] URL: http://www.complang.tuwien.ac.at/forth/gforth/Docs-html/