



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»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3 по курсу «Анализ алгоритмов»

«Трудоёмкость алгоритмов сортировки»

Студент _____ Маслова Марина Дмитриевна

Группа _____ ИУ7-53Б

Оценка (баллы) _____

Преподаватель _____ Волкова Лилия Леонидовна

2021 г.

Содержание

Введение	4
1 Аналитическая часть	5
1.1 Алгоритм сортировки вставками	5
1.2 Алгоритм сортировки перемешиванием	5
1.3 Алгоритм сортировки выбором	5
1.4 Вывод	6
2 Конструкторская часть	7
2.1 Разработка алгоритмов	7
2.2 Оценка трудоемкости алгоритмов	11
2.2.1 Модель вычислений	11
2.2.2 Оценка трудоемкости алгоритма сортировки вставками	12
2.2.3 Оценка трудоемкости алгоритма сортировки перемешиванием	12
2.2.4 Оценка трудоемкости алгоритма сортировки выбором .	12
2.3 Структура разрабатываемого ПО	12
2.4 Классы эквивалентности при тестировании	12
2.5 Вывод	12
3 Технологическая часть	14
3.1 Требования к программному обеспечению	14
3.2 Средства реализации	14
3.3 Листинги кода	15
3.4 Описание тестирования	16
3.5 Вывод	16
4 Исследовательская часть	17
4.1 Технические характеристики	17
4.2 Примеры работы программы	17
4.3 Результаты тестирования	17
4.4 Постановка эксперимента по замеру времени	18
4.5 Результаты эксперимента	18

4.6 Вывод	18
Заключение	19
Список литературы	20

Введение

Сортировка – процесс перестановки объектов данного множества в определенном порядке. Сортировка служит для последующего облегчения поиска элементов в отсортированном множестве. При этом сортировка является примером огромного разнообразия алгоритмов, выполняющих одну и ту же задачу. Несмотря на большое количество алгоритмов сортировки, любой из них можно разбить на три основные части:

- сравнение, определяющее порядок элементов в паре;
- перестановка, меняющая элементы в паре местами;
- сам сортирующий алгоритм, который выполняет два предыдущих шага до полного упорядочивания.

Ввиду большого количества алгоритмов сортировки одни из них имеют преимущества над другими, что приводит к необходимости их сравнительного анализа [1].

Целью данной работы является получение навыков анализа алгоритмов сортировок.

Для достижения поставленной цели необходимо выполнить следующие **задачи**:

- изучить три алгоритма сортировки: вставками, перемешиванием, выбором;
- разработать каждый из алгоритмов;
- дать теоретическую оценку трудоемкости алгоритмов сортировки;
- реализовать каждый алгоритм;
- провести тестирование реализованных алгоритмов;
- провести сравнительный анализ алгоритмов по процессорному времени работы реализации;

1 Аналитическая часть

В данном разделе представлено теоретическое описание алгоритмов сортировки вставками, перемешиванием и выбором.

1.1 Алгоритм сортировки вставками

В алгоритме сортировки вставками сортируемая последовательность условно делится на входную неотсортированную часть и выходную отсортированную часть. На каждом шаге из неотсортированной части выбирается элемент и помещается на нужную позицию в уже отсортированной части. В начале алгоритма считается, что первый элемент последовательности является отсортированной частью, поэтому вставка элементов в отсортированную часть начинается со второго элемента [2].

1.2 Алгоритм сортировки перемешиванием

Алгоритм сортировки перемешиванием является модификацией алгоритма сортировки пузырьком. В отличие от сортировки пузырьком, где происходит обход последовательности только в одном направлении, в алгоритме сортировки перемешиванием после достижения одной из границ рабочей части последовательности (то есть той части, которая еще не отсортирована и в которой происходит смена элементов) меняет направление движения. При этом при движении в одном направлении алгоритм перемещает к границе рабочей области максимальный элемент, а в другом направлении – минимальный элемент. Границы рабочей части последовательности устанавливаются в месте последнего обмена [2].

1.3 Алгоритм сортировки выбором

Алгоритм сортировки выбором в текущей последовательности находит минимальный/максимальный элемент, производит обмен этого элемента со значением первой неотсортированной позиции и сортирует оставшуюся часть последовательности, исключив из рассмотрения уже отсортированные элементы [2].

1.4 Вывод

В данном разделе были рассмотрены алгоритмы сортировки вставками, перемешиванием и выбором. Из представленных описаний можно предъявить ряд требований к разрабатываемому программному обеспечению:

- на вход подается последовательность, которую необходимо отсортировать;
- на выходе должна выдаваться отсортированная последовательность.

2 Конструкторская часть

В данном разделе разрабатываются алгоритмы сортировки, структура программы и способы её тестирования, также проводится оценка трудоемкости каждого из алгоритмов.

2.1 Разработка алгоритмов

На рисунке 2.1 представлена схема алгоритма сортировки вставками, на рисунке 2.2 — схема алгоритма сортировки перемешиванием, на рисунке 2.3 — схема алгоритма сортировки выбором.

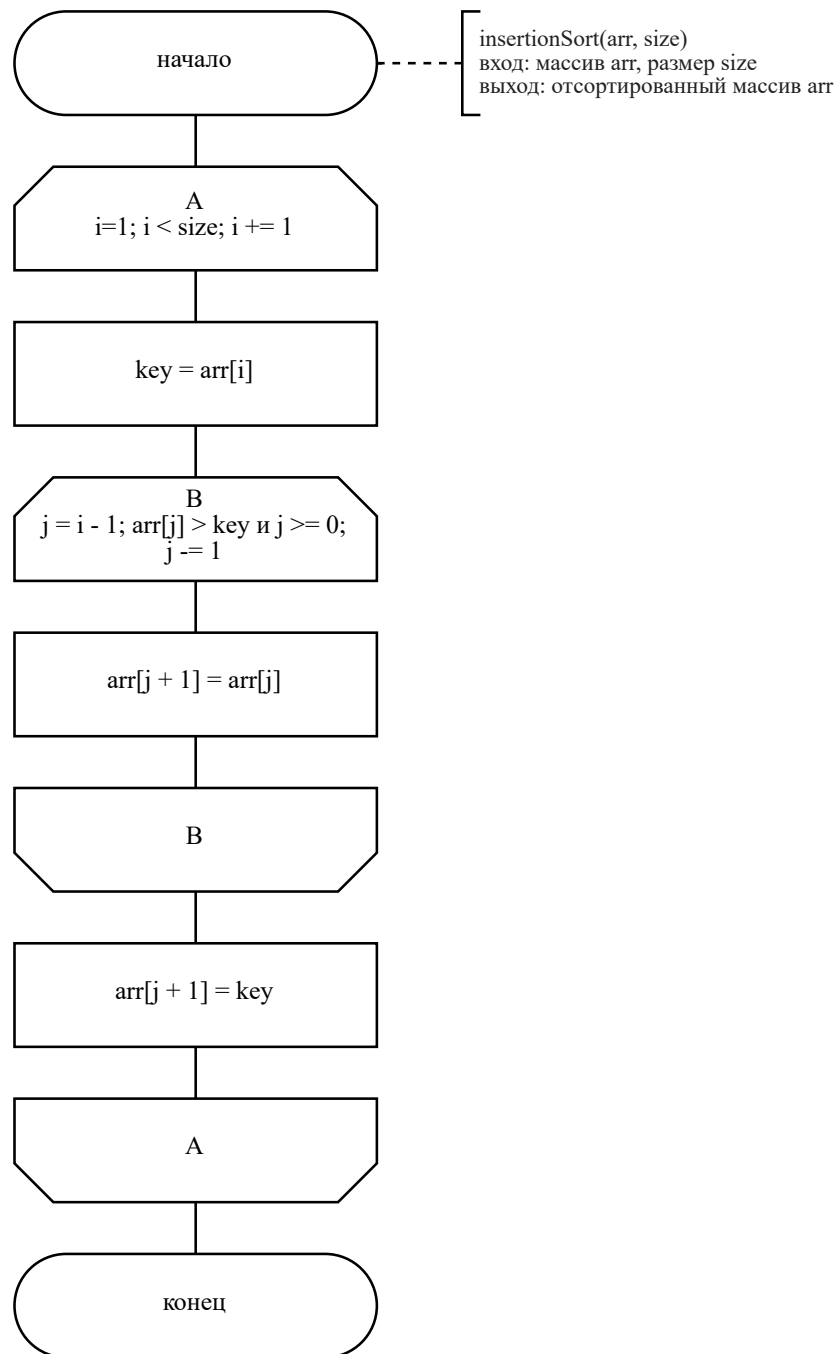


Рисунок 2.1 – Схема алгоритма сортировки вставками

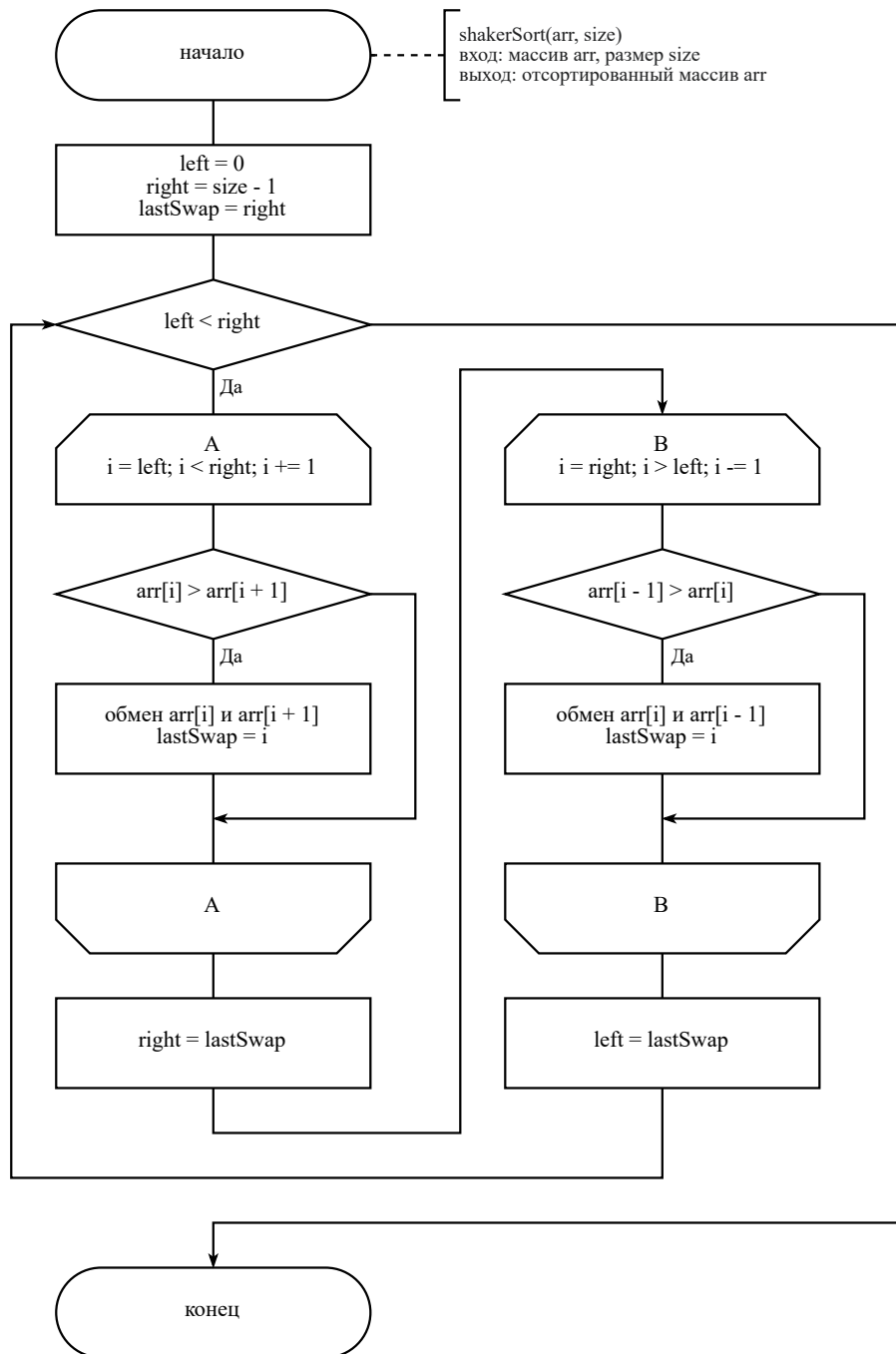


Рисунок 2.2 – Схема алгоритма сортировки перемешиванием

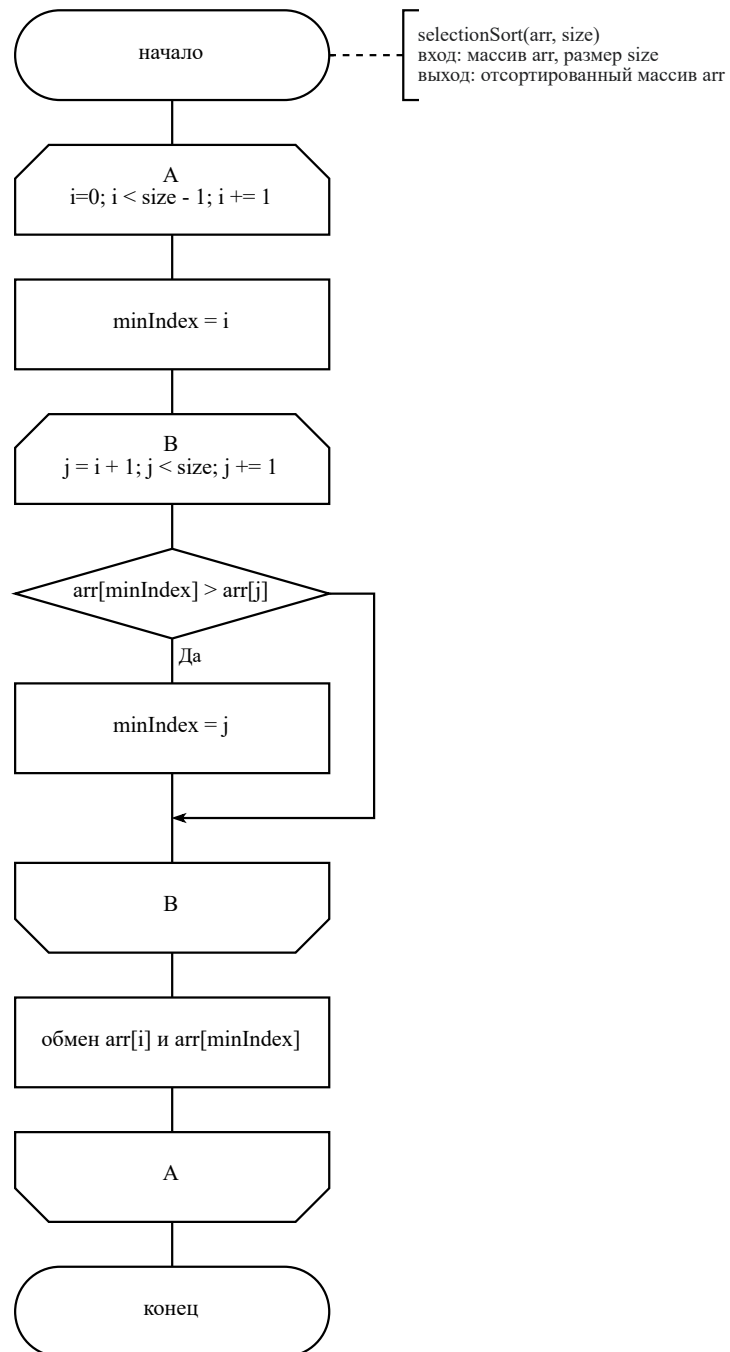


Рисунок 2.3 – Схема алгоритма сортировки выбором

2.2 Оценка трудоемкости алгоритмов

В данном подразделе производится оценка трудоемкости каждого из алгоритмов.

2.2.1 Модель вычислений

Введем модель вычислений для оценки трудоемкости алгоритмов:

- операции из списка 2.1 имеют трудоемкость 2;

$$*, /, //, \%, *=, /=, //= \quad (2.1)$$

- операции из списка 2.2 имеют трудоемкость 1;

$$\begin{aligned} &=, +, -, +=, -=, <, >, ==, !=, \\ &>=, <=, [], <<, >>, ++, --, and, or \end{aligned} \quad (2.2)$$

- трудоемкость оператора выбора `if условие then A else B` рассчитывается по формуле 2.3:

$$f_{if} = f + \begin{cases} f_A, & \text{если условие выполняется;} \\ f_B, & \text{иначе} \end{cases} \quad (2.3)$$

- трудоемкость оператора цикла рассчитывается по формуле 2.4:

$$f_{for} = f_{инициализации} + f_{сравнения} + N(f_{тела} + f_{инкремента} + f_{сравнения}) \quad (2.4)$$

- трудоемкость вызова функции равна 0.

2.2.2 Оценка трудоемкости алгоритма сортировки вставками

2.2.3 Оценка трудоемкости алгоритма сортировки перемешиванием

2.2.4 Оценка трудоемкости алгоритма сортировки выбором

2.3 Структура разрабатываемого ПО

Для реализации разрабатываемого программного обеспечения будет использоваться метод структурного программирования. Каждый из алгоритмов будет представлен отдельной функцией, при необходимости будут выделены подпрограммы для каждой из них. Также будут реализованы функции для ввода-вывода и функция, вызывающая все подпрограммы для связности и полноценности программы.

2.4 Классы эквивалентности при тестировании

Для тестирования программного обеспечения во множестве тестов будут выделены следующие классы эквивалентности:

- пустой массив;
- упорядоченный массив четной длины;
- упорядоченный массив нечетной длины;
- упорядоченный в обратном порядке массив четной длины;
- упорядоченный в обратном порядке массив нечетной длины;
- случайный массив;
- массив из одного элемента.

2.5 Вывод

В данном разделе были разработаны алгоритмы сортировки вставками, перемешиванием и выбором, также была произведена оценка трудоемкостей

алгоритмов. Для дальнейшей проверки правильности работы программы были выделены классы эквивалентности тестов.

3 Технологическая часть

В данном разделе описаны требования к программному обеспечению, средства реализации, приведены листинги кода и данные, на которых будет проводиться тестирование.

3.1 Требования к программному обеспечению

Программа должна предоставлять следующие возможности:

- выбор режима работы: для единичного эксперимента и для массовых экспериментов;
- в режиме единичного эксперимента ввод последовательности для сортировки и вывод результатов работы каждого из алгоритмов сортировки (отсортированной последовательности);
- в режиме массовых экспериментов измерение времени работы каждого из алгоритмов сортировки при различных длинах сортируемой последовательности и различных её первоначальных состояниях: отсортированная, отсортированная в обратном порядке, случайная последовательности.

3.2 Средства реализации

Для реализации данной лабораторной работы выбран интерпретируемый язык программирования высокого уровня Python[3], так как он позволяет реализовывать сложные задачи за кратчайшие сроки за счет простоты синтаксиса и наличия большого количества подключаемых библиотек.

В качестве среды разработки выбран текстовый редактор Vim[4] с установленными плагинами автодополнения и поиска ошибок в процессе написания, так как он реализует быстрое перемещение по тексту программы и простое взаимодействие с командной строкой.

Замеры времени проводились при помощи функции `process_time_ns` из библиотеки `time`[5].

3.3 Листинги кода

В данном подразделе представлены листинги кода ранее описанных алгоритмов:

- алгоритм сортировки вставками (листинг 3.1);
- алгоритм сортировки перемешиванием (листинг 3.2);
- алгоритм сортировки выбором (листинги 3.3).

Листинг 3.1 – Реализация алгоритма сортировки вставками

```
1 def insertionSort(arr, size):
2     for i in range(1, size):
3         key = arr[i]
4
5         j = i - 1
6         while j >= 0 and arr[j] > key:
7             arr[j + 1] = arr[j]
8             j -= 1
9
10        arr[j + 1] = key
11
12    return arr
```

Листинг 3.2 – Реализация алгоритма сортировки перемешиванием

```
1 def shakerSort(arr, size):
2     left = 0
3     right = size - 1
4     lastSwap = 0
5
6     while left < right:
7         for i in range(left, right):
8             if arr[i] > arr[i + 1]:
9                 arr[i], arr[i + 1] = arr[i + 1], arr[i]
10                lastSwap = i
11
12        right = lastSwap
13
14        for i in range(right, left, -1):
15            if arr[i - 1] > arr[i]:
16                arr[i], arr[i - 1] = arr[i - 1], arr[i]
17                lastSwap = i
18
19        left = lastSwap
20
21    return arr
```

Листинг 3.3 – Реализация алгоритма сортировки выбором

```
1 def selectionSort(arr, size):
2     for i in range(size - 1):
3         minIndex = i
4
5         for j in range(i + 1, size):
6             if arr[minIndex] > arr[j]:
7                 minIndex = j
8
9         arr[minIndex], arr[i] = arr[i], arr[minIndex]
10
11     return arr
```

3.4 Описание тестирования

В таблице 3.1 приведены функциональные тесты для алгоритмов сортировки.

Таблица 3.1 – Функциональные тесты

Последовательность	Ожидаемый результат
[]	[]
[1, 2, 3, 4]	[1, 2, 3, 4]
[-1, -2, 0, 3, 4]	[-1, -2, 0, 3, 4]
[100, 50, -20, -99]	[-99, -20, 50, 100]
[13, 7, 1, -1, -17]	[-17, -1, 1, 7, 13]
[4, 2, -44, 76, 0, 11]	[-44, 0, 2, 4, 11, 76]
[113]	[133]

3.5 Вывод

В данном разделе были реализованы алгоритмы сортировки вставками, перемешиванием, выбором. Также были написаны тесты для каждого класса эквивалентности, описанного в конструкторском разделе.

4 Исследовательская часть

4.1 Технические характеристики

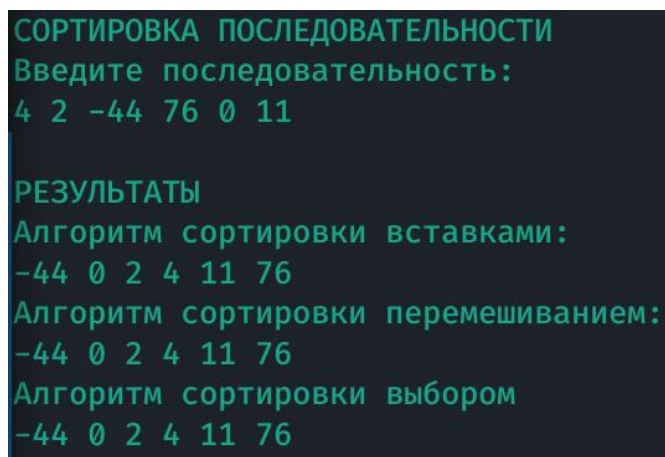
Технические характеристики устройства, на котором выполнялось тестирование:

- Операционная система: Manjaro [6] Linux x86_64.
- Память: 8 GiB.
- Процессор: Intel® Core™ i5-8265U[7].

Тестирование проводилось на ноутбуке, включенном в сеть электропитания. Во время тестирования ноутбук был нагружен только встроенными приложениями окружения, окружением, а также непосредственно системой тестирования.

4.2 Примеры работы программы

На рисунке 4.1 представлены результаты работы программы на случайной последовательности.



```
СОРТИРОВКА ПОСЛЕДОВАТЕЛЬНОСТИ
Введите последовательность:
4 2 -44 76 0 11

РЕЗУЛЬТАТЫ
Алгоритм сортировки вставками:
-44 0 2 4 11 76
Алгоритм сортировки перемешиванием:
-44 0 2 4 11 76
Алгоритм сортировки выбором
-44 0 2 4 11 76
```

Рисунок 4.1 – Пример работы программы

4.3 Результаты тестирования

Программа была протестирована на входных данных, приведенных в таблице 3.1. Полученные результаты работы программы совпали с ожидаемыми результатами.

4.4 Постановка эксперимента по замеру времени

Для оценки времени работы реализаций алгоритмов сортировки был проведен эксперимент, в котором определялось влияние размеров последовательности на время работы каждого из алгоритмов. Тестирование проводилось на размерах последовательностей от 100 до 1000 с шагом 100. Так как от запуска к запуску процессорное время, затрачиваемое на выполнение алгоритма, менялось в определенном промежутке, необходимо было усреднить вычисляемые значения. Для этого каждый алгоритм запускался по K раз, и для полученных K значений определялось среднее арифметическое, которое заносилось в таблицу результатов.

Так как трудоемкость алгоритмов сортировки зависит от состояния входной последовательности, эксперимент проводился на трех видах последовательностей: отсортированных, отсортированных в обратном порядке и случайных.

Результаты эксперимента были представлены в виде таблиц и графиков, приведенных в следующем подразделе.

4.5 Результаты эксперимента

4.6 Вывод

По результатам эксперимента можно сделать следующие выводы:

—

Заключение

В ходе выполнения лабораторной работы:

—

Список литературы

- [1] Вирт Н. Алгоритмы и структуры данных. М.: Мир, 1989. с. 360.
- [2] Шагбазян Д. В., Штанюк А. А., Малкина Е. В. Алгоритмы сортировки. Анализ, реализация, применение. Нижний Новгород: Нижегородский госуниверситет, 2019. с. 42.
- [3] Welcome to Python [Электронный ресурс]. Режим доступа: <https://www.python.org> (дата обращения: 12.10.2021).
- [4] welcome home : vim online [Электронный ресурс]. Режим доступа: <https://www.vim.org/> (дата обращения: 12.10.2021).
- [5] time — Time access and conversions [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/time.html#time.process_time_ns (дата обращения: 04.10.2021).
- [6] Manjaro — enjoy the simplicity [Электронный ресурс]. Режим доступа: <https://manjaro.org/> (дата обращения: 17.10.2021).
- [7] Процессор Intel® Core™ i5-8265U [Электронный ресурс]. Режим доступа: <https://ark.intel.com/content/www/ru/ru/ark/products/149088/intel-core-i5-8265u-processor-6m-cache-up-to-3-90-ghz.html> (дата обращения: 17.10.2021).