



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"

Тема Алгоритмы сортировки

Студент Зайцева А.А.

Группа ИУ7-52Б

Преподаватели Волкова Л.Л.

Москва — 2021 г.

Оглавление

Введение	3
1 Аналитическая часть	4
1.1 Сортировка вставками	4
1.2 Сортировка пузырьком	4
1.3 Шейкерная сортировка	5
2 Конструкторская часть	6
2.1 Схема алгоритма сортировки вставками	6
2.2 Схема алгоритма сортировки пузырьком	7
2.3 Схема алгоритма шейкерной сортировки	7
3 Технологическая часть	9
3.1 Требования к ПО	9
3.2 Выбор средств реализации	9
3.3 Листинги кода	9
3.4 Тестирование	10
4 Исследовательская часть	12
4.1 Пример работы	12
4.2 Технические характеристики	13
4.3 Сравнение времени выполнения реализаций алгоритмов . . .	13
4.4 Сравнение трудоемкости реализаций алгоритмов	14
Заключение	19
Список использованной литературы	20

Введение

Целью данной лабораторной работы является получение навыков оценки трудоемкости алгоритмов на примере трех алгоритмов сортировок.

Под сортировкой понимается упорядочивание элементов по какому-либо признаку. Сортировка относится к важнейшему классу алгоритмов обработки данных и осуществляется большим количеством способов [1].

Алгоритмы сортировки имеют большое практическое применение и часто встречаю там, где речь идет об обработке и хранении больших объемов информации. Некоторые задачи обработки данных решаются проще, если данные заранее упорядочить. Упорядоченные объекты содержатся в телефонных книгах, ведомостях налогов, в библиотеках, в оглавлениях, в словарях [2].

В настоящее время, в связи с постоянно растущими объемами данных, вопрос эффективности сортировки данных не теряет свою актуальность.

В рамках выполнения работы необходимо решить следующие задачи:

- 1) изучить три алгоритма сортировок: сортировку вставками, сортировку пузырьком и шейкерную сортировку;
- 2) разработать и реализовать изученные алгоритмы;
- 3) провести сравнительный анализ трудоёмкости реализаций алгоритмов на основе теоретических расчетов;
- 4) провести сравнительный анализ процессорного времени выполнения реализаций алгоритмов на основе экспериментальных данных.

1 Аналитическая часть

В данном разделе будут рассмотрены основные идеи трех алгоритмов сортировок - сортировки вставками, сортировки пузырьком и шейкерной сортировки.

1.1 Сортировка вставками

Сортировка вставками — достаточно простой алгоритм. Сортируемый массив можно разделить на две части — отсортированную и неотсортированную. В начале сортировки первый элемент массива считается отсортированным, все остальные — не отсортированные. Начиная со второго элемента массива и заканчивая последним, алгоритм вставляет очередной элемент из неотсортированной части массива в нужную позицию в отсортированной части.

Таким образом, за один шаг сортировки отсортированная часть массива увеличивается на один элемент, а неотсортированная часть массива уменьшается на один элемент. Данный процесс вставки продолжается до тех пор, пока все элементы исходного списка не окажутся в расширяющейся отсортированной части списка [3].

1.2 Сортировка пузырьком

Сортировка пузырьком — один из самых известных алгоритмов сортировки. 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, где N - длина сортируемого массива. 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].

1.3 Шейкерная сортировка

Алгоритм шейкерной сортировки является модификацией пузырьковой сортировки по направлению движения. Отличия первой от второй заключаются в том, что при прохождении части массива происходит проверка, были ли перестановки. Если их не было, значит эта часть массива уже упорядочена, и она исключается из дальнейшей обработки. Кроме того, при прохождении массива от начала к концу минимальный элемент перемещается в самое начало, а максимальный элемент сдвигается к концу массива [4].

Вывод

В данном разделе были рассмотрены идеи, лежащие в основе рассматриваемых алгоритмов сортировок - сортировки вставками, сортировки пузырьком и шейкерной сортировки.

2 Конструкторская часть

В данном разделе будут рассмотрены схемы описанных в предыдущем разделе алгоритмов сортировок.

2.1 Схема алгоритма сортировки вставками

На рисунке 2.1 приведена схема алгоритма сортировки вставками.

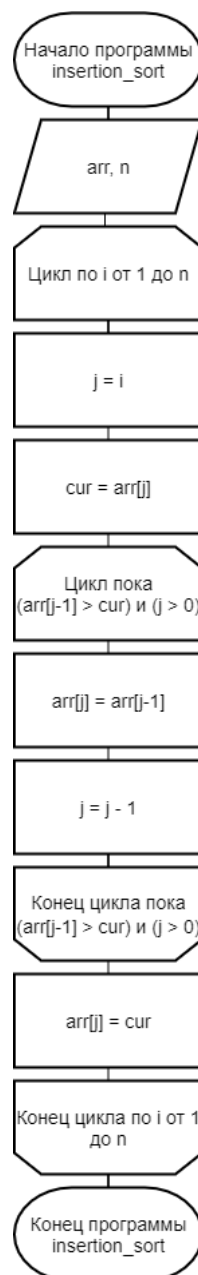


Рисунок 2.1 – Схема алгоритма сортировки вставками

2.2 Схема алгоритма сортировки пузырьком

На рисунке 2.2 приведена схема алгоритма сортировки пузырьком.

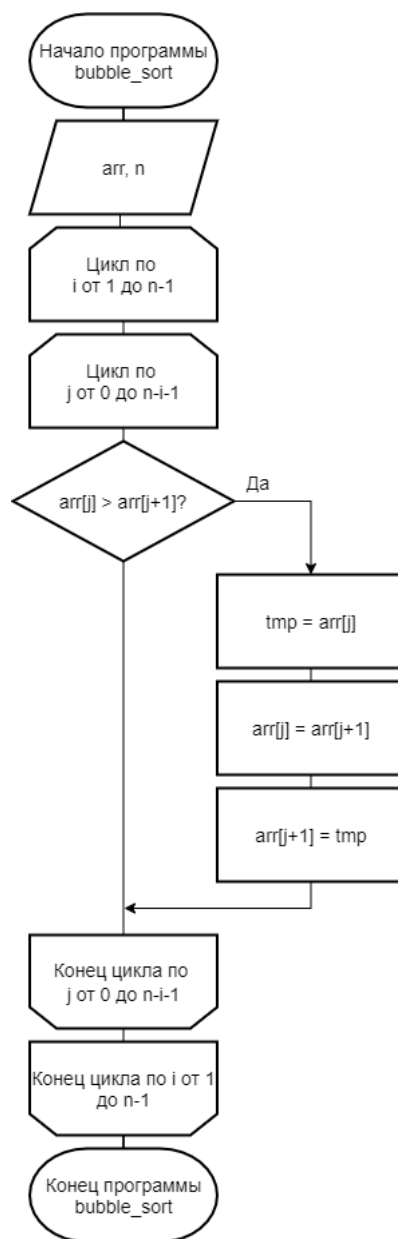


Рисунок 2.2 – Схема алгоритма сортировки пузырьком

2.3 Схема алгоритма шейкерной сортировки

На рисунке 2.3 приведена схема алгоритма шейкерной сортировки.

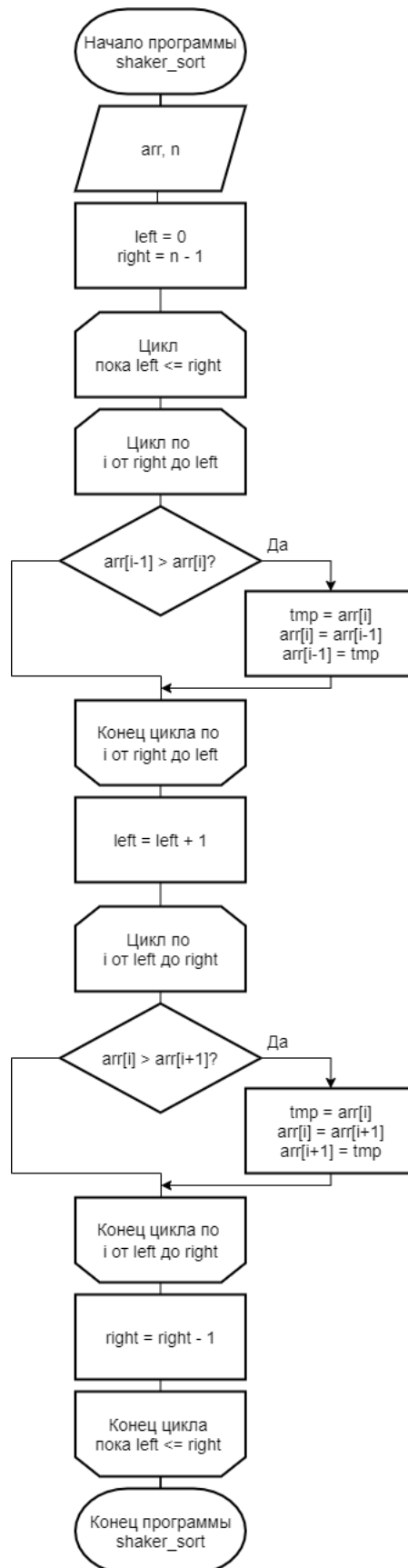


Рисунок 2.3 – Схема алгоритма шейкерной сортировки

Вывод

Были разработаны схемы трех алгоритмов сортировки.

3 Технологическая часть

В данном разделе производится выбор средств реализации, а также приводятся требования к программному обеспечению (ПО), листинги реализованных алгоритмов и тесты для программы.

3.1 Требования к ПО

На вход программе подается массив целых чисел, а на выходе должен быть получен массив, отсортированный с помощью каждого реализованного алгоритма сортировки: вставками, пузырьком и шейкерной. Также необходимо вывести затраченное каждой реализацией процессорное время.

3.2 Выбор средств реализации

В качестве языка программирования для реализации данной лабораторной работы был выбран язык Python [5]. Он позволяет быстро реализовывать различные алгоритмы без выделения большого времени на проектирование структуры программы и выбор типов данных.

Кроме того, в Python есть библиотека `time`, которая предоставляет функцию `process_time` для замера процессорного времени [6].

В качестве среды разработки выбран PyCharm. Он является кросс-платформенным, а также предоставляет удобный и функциональный отладчик и средства для рефакторинга кода, что позволяет быстро находить и исправлять ошибки [7].

3.3 Листинги кода

В листингах 3.1 - 3.3 представлены реализации рассматриваемых алгоритмов.

Листинг 3.1 – Реализация алгоритма сортировки вставками

```
1 def insertion_sort(arr, n):
```

```

2   for i in range(1, n):
3       j = i
4       cur = arr[j]
5       while arr[j - 1] > cur and j > 0:
6           arr[j] = arr[j - 1]
7           j -= 1
8       arr[j] = cur
9   return arr

```

Листинг 3.2 – Реализация алгоритма сортировки пузырьком

```

1   def bubble_sort(arr, n):
2       for i in range(n - 1):
3           for j in range(n - i - 1):
4               if arr[j] > arr[j + 1]:
5                   arr[j], arr[j + 1] = arr[j + 1], arr[j]
6   return arr

```

Листинг 3.3 – Реализация алгоритма шейкерной сортировки

```

1
2   def shaker_sort(arr, n):
3       left = 0
4       right = n - 1
5       while left <= right:
6           for i in range(right, left, -1):
7               if arr[i - 1] > arr[i]:
8                   arr[i - 1], arr[i] = arr[i], arr[i - 1]
9           left += 1
10          for i in range(left, right, 1):
11              if arr[i] > arr[i + 1]:
12                  arr[i + 1], arr[i] = arr[i], arr[i + 1]
13          right -= 1
14
15   return arr

```

3.4 Тестирование

В таблице 3.1 приведены функциональные тесты для реализаций алгоритмов сортировок. Все тесты пройдены успешно каждой реализацией.

Таблица 3.1 – Тесты

Исходный массив	Ожидаемый результат
1	1
4, 4, 4, 4	4, 4, 4, 4
-1, 1, 2, 2, 10	-1, 1, 2, 2, 10
9, 8, 0	0, 8, 9
4, 5, 2, -1, 5, 6	-1, 2, 4, 5, 5, 6

Вывод

Был произведен выбор средств реализации, реализованы и протестированы алгоритмы сортировок (вставками, пузырьком и шейкером), а также приведены требования к ПО.

4 Исследовательская часть

4.1 Пример работы

На рисунке 4.1 приведен пример работы программы.

```
Введите массив целых чисел (в одну строку, через пробелы):  
-1 9 83 193 1 3 5 -45 6  
  
Пузырьковая сортировка:  
До:  
-1 9 83 193 1 3 5 -45 6  
Ответ:  
-45 -1 1 3 5 6 9 83 193  
Время: 0.0  
  
Шейкерная сортировка:  
До:  
-1 9 83 193 1 3 5 -45 6  
Ответ:  
-45 -1 1 3 5 6 9 83 193  
Время: 0.0  
  
Сортировка вставками:  
До:  
-1 9 83 193 1 3 5 -45 6  
Ответ:  
-45 -1 1 3 5 6 9 83 193  
Время: 0.0
```

Рисунок 4.1 – Пример работы программы

4.2 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование:

- операционная система: Windows 10;
- оперативная память: 16 Гб;
- процессор: Intel® Core™ i5-8259U.

Во время тестирования ноутбук был включен в сеть питания и нагружен только встроенными приложениями окружения и системой тестирования.

4.3 Сравнение времени выполнения реализаций алгоритмов

Все реализации алгоритмов сравнивались на трех видах массивов: упорядоченный (лучший случай - все реализации выдают наименьшее время работы), упорядоченный в обратном порядке (худший случай - все реализации выдают наибольшее время работы), случайно сгенерированный массив (произвольный случай). При этом для каждого класса массивов создавались массивы длиной от 0 до 10000 с шагом, увеличивающимся по мере роста длины массива.

Так как сортировки выполняются достаточно быстро, а замеры времени имеют некоторую погрешность, они для каждого класса и длины массивы и каждой реализации алгоритма сортировки выполнялись 10 раз, а затем вычислялось среднее время работы.

На рисунке 4.2 приведены результаты сравнения времени работы всех реализаций на всех массивах.

На рисунках 4.3 - 4.5 приведены результаты сравнения времени работы всех реализаций на отдельных классах массивов.

На рисунках 4.6 - 4.8 приведены результаты сравнения времени работы каждой реализации на всех классах массивов.

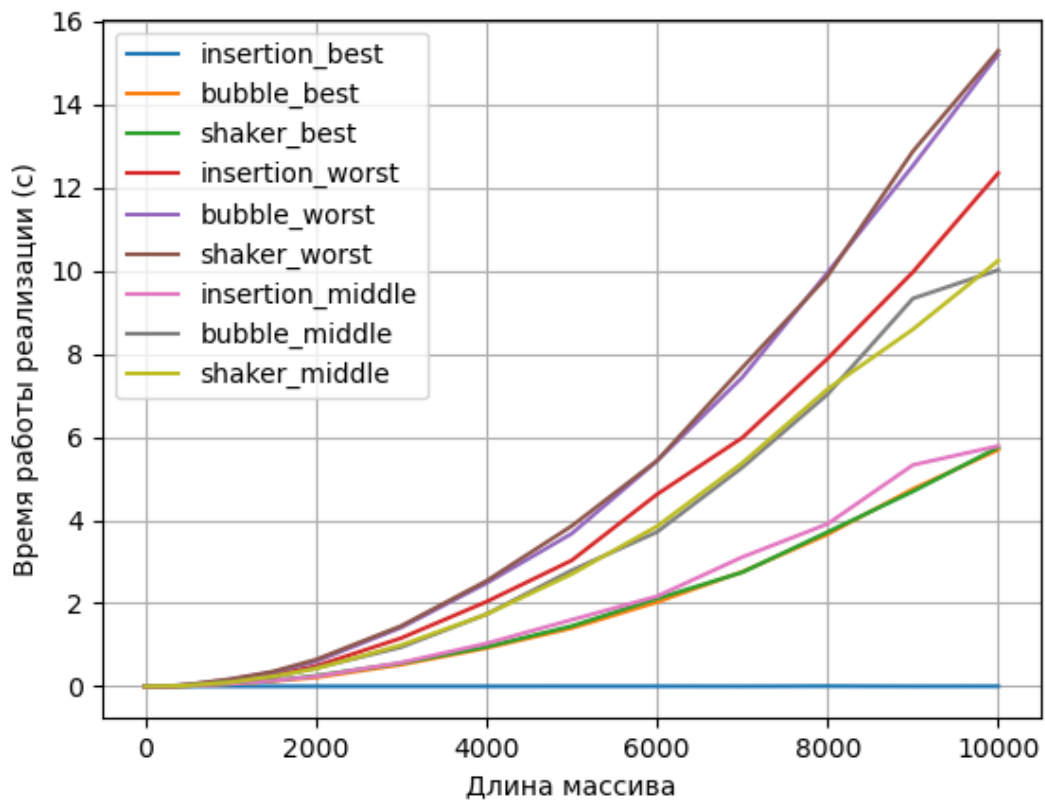


Рисунок 4.2 – Сравнение времени работы реализаций всех алгоритмов на всех массивах

4.4 Сравнение трудоемкости реализаций алгоритмов

Введем модель вычисления трудоемкости для оценки алгоритмов:

- базовые операции стоимостью 1: -, -, -=, +, ++, +=, ==, !=, <, >, <=, >=, », «, [];
- базовые операции стоимостью 2: *, *=, /, /=, %, %=;
- оценка трудоемкости цикла: $F_{\text{ц}} = F_{\text{init}} + F_{\text{сравн}} + N * (F_{\text{тела}} + F_{\text{finc}} + F_{\text{сравн}})$, где F_{init} , $F_{\text{сравн}}$, $F_{\text{тела}}$, F_{finc} - трудоемкости инициализации, проверки условия цикла, инкрементирования и тела цикла, соответственно, а N - количество итераций;

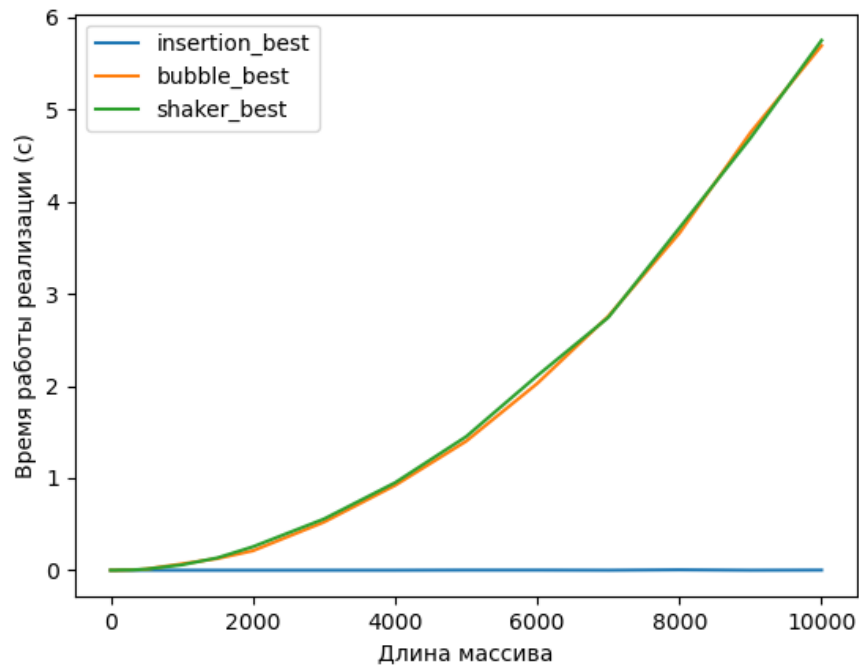


Рисунок 4.3 – Сравнение времени работы реализаций всех алгоритмов на упорядоченных массивах

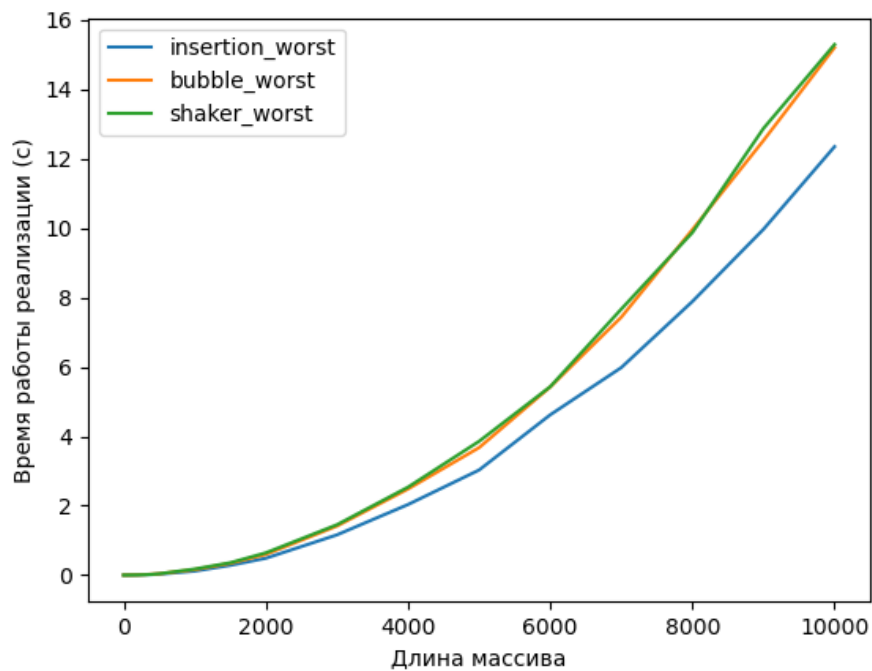


Рисунок 4.4 – Сравнение времени работы реализаций всех алгоритмов на упорядоченных в обратном порядке массивах

- стоимость условного оператора: $F_{yo} = F_{сравн} +$

$$+ \begin{bmatrix} \min(f1, f2) \\ \max(f1, f2) \end{bmatrix}, \quad (4.1)$$

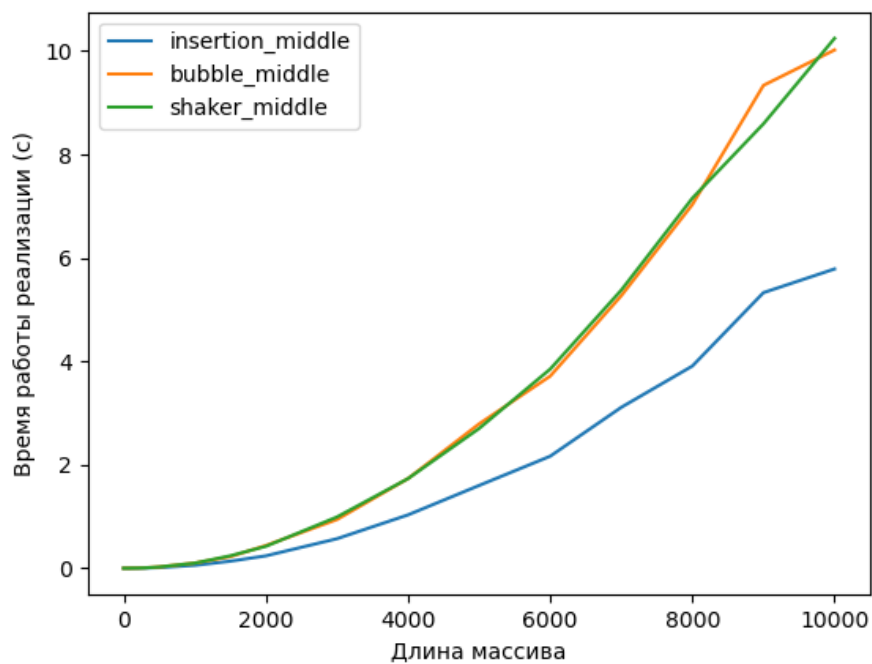


Рисунок 4.5 – Сравнение времени работы реализаций всех алгоритмов на произвольных массивах

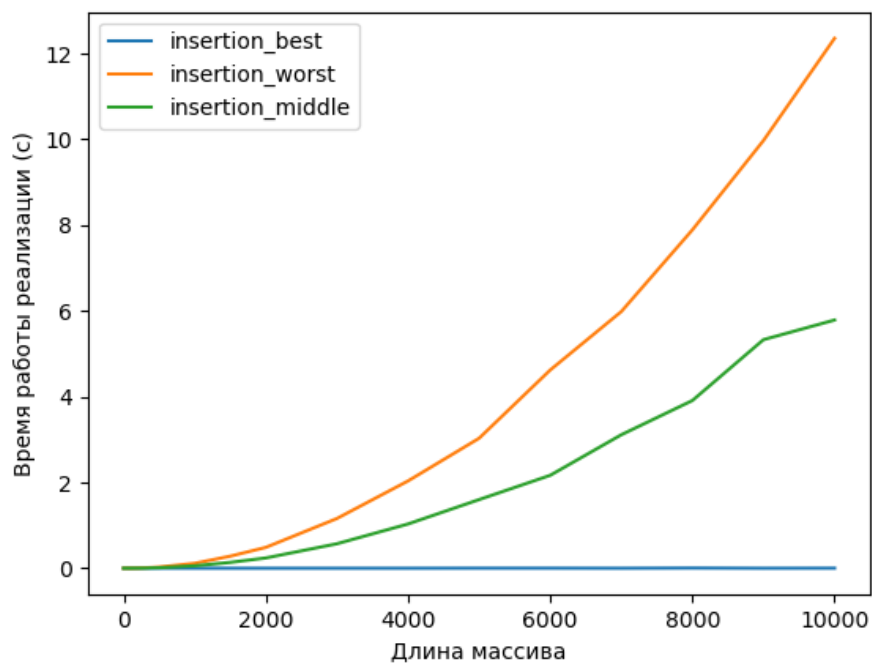


Рисунок 4.6 – Сравнение времени работы реализации алгоритма сортировки вставками на всех классах массивов

где $F_{\text{сравн}}$, f_1 , f_2 - трудоемкости проверки условия, первого блока и второго блока, соответственно.

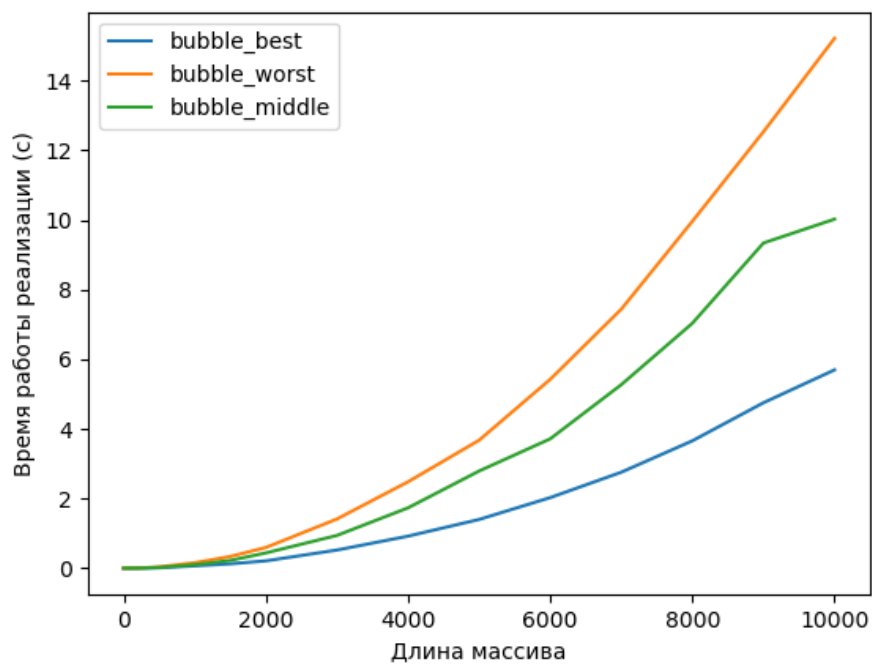


Рисунок 4.7 – Сравнение времени работы реализации алгоритма сортировки пузырьком на всех классах массивов

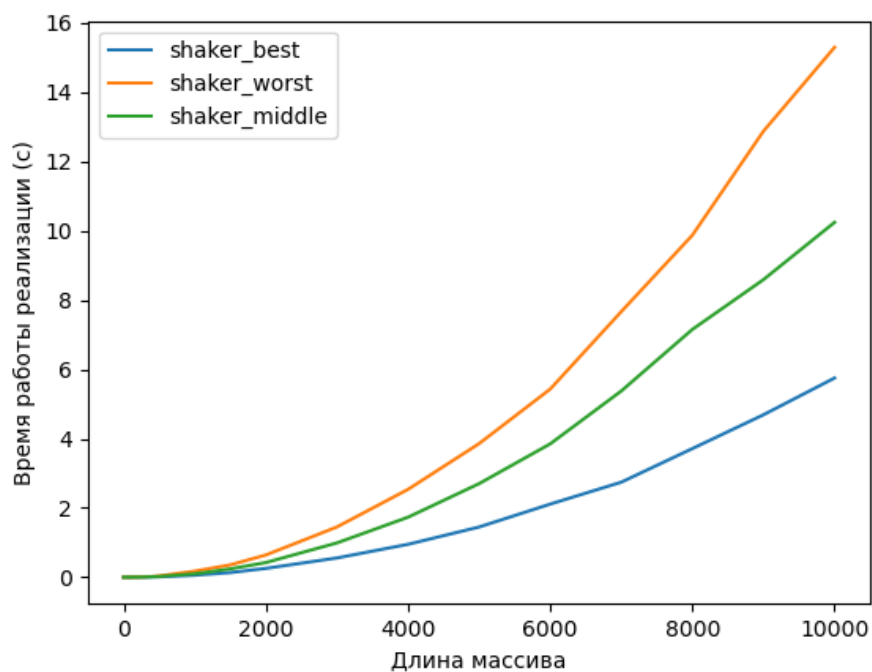


Рисунок 4.8 – Сравнение времени работы реализации алгоритма шейкерной сортировки на всех классах массивов

Оценим трудоемкость реализаций алгоритмов сортировки.
Рассмотрим трудоемкость реализации алгоритма сортировки вставка-

ми.

Лучший случай (отсортированный массив): алгоритм ни разу не войдет во внутренний цикл. Трудоемкость: $f(n) = 2 + (n - 1)(2 + 1 + 2 + 4 + 2) = 11n - 9 = O(n)$.

Худший случай (отсортированный в обратном порядке массив): алгоритм будет полностью (пока не дойдет до начала массива) проходить внутренний цикл каждый раз. Трудоемкость: $f(n) = 2 + (n - 1)(2 + 1 + 2 + 4 + 2 + \frac{n}{2}(4 + 4 + 1)) = \frac{9n^2}{2} + \frac{13n}{2} - 9 = O(n^2)$.

Рассмотрим трудоемкость алгоритма сортировки пузырьком.

Лучший случай (отсортированный массив): не будет произведено ни одного обмена. Трудоемкость: $f(n) = 3 + (n - 1)(3 + \frac{n}{2}4) = 2n^2 + n = O(n^2)$.

Худший случай (отсортированный в обратном порядке массив): замена во внутреннем цикле будет производиться каждый раз. Трудоемкость: $f(n) = 3 + (n - 1)(3 + \frac{n}{2}(4 + 9)) = \frac{13n^2}{2} - \frac{7n}{2} = O(n^2)$.

Вывод

Реализация сортировки пузырьком квадратично зависит от количества элементов в массиве (как в лучшем, так и в худшем случае) и работает дольше всех остальных протестированных реализаций. Ее модификация - шейкерная сортировка - показывает схожие результаты, так как является модификацией только лишь по направлению движения.

Реализацией, которая работает меньшее количество времени, чем вышеупомянутые, является реализация алгоритма сортировки вставками. Для нее в худшем случае время сортировки массива также квадратично зависит от количества элементов в нем, как и для алгоритмов пузырьковой и шейкерной сортировки. Но для лучшего случая время работы реализации алгоритма сортировки вставками зависит от количества элементов линейно.

Таким образом, реализацией с наименьшим временем выполнения (по экспериментальным данным - во всех случаях) является сортировка вставками.

Заключение

В результате выполнения лабораторной работы были получены навыки оценки трудоемкости алгоритмов на примере трех алгоритмов сортировок.

В ходе выполнения лабораторной работы были выполнены следующие задачи:

- 1) изучены три алгоритма сортировок: сортировка вставками, сортировка пузырьком и шейкерная сортировка;
- 2) разработаны и реализованы изученные алгоритмы;
- 3) проведен сравнительный анализ трудоёмкости реализаций алгоритмов на основе теоретических расчетов;
- 4) проведен сравнительный анализ процессорного времени выполнения реализаций алгоритмов на основе экспериментальных данных.

Литература

- [1] Д.В. Шагбазян. Алгоритмы сортировки. Анализ, реализация, применение. Нижний Новгород: Нижегородский госуниверситет, 2019. Т. 42.
- [2] Лекция 43: Алгоритмы сортировки массивов. Внутренняя сортировка [Электронный ресурс]. Режим доступа: <https://intuit.ru/studies/courses/648/504/lecture/11472> (дата обращения: 13.09.2021).
- [3] Сортировка вставками [Электронный ресурс]. Режим доступа: <http://cppstudio.com/post/462/> (дата обращения: 13.09.2021).
- [4] Алгоритм сортировки перемешиванием [Электронный ресурс]. Режим доступа: <https://space-base.ru/library/algorithm/algorithm-sortirovki-peremeshivaniem-shejkernaya-dvunapravlennaya-pu> (дата обращения: 13.09.2021).
- [5] Лутц Марк. Изучаем Python, том 1, 5-е изд. Пер. с англ. — СПб.: ООО “Диалектика”, 2019. Т. 832.
- [6] time — Time access and conversions [Электронный ресурс]. Режим доступа: <https://docs.python.org/3/library/time.html> (дата обращения: 05.09.2021).
- [7] Python и Pycharm [Электронный ресурс]. Режим доступа: <https://py-charm.blogspot.com/2017/09/pycharm.html> (дата обращения: 05.09.2021).