



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:

Ле Ни Куанг

Группа:

ИУ7и-56Б

Преподаватель:

Волкова Л. Л.

Строганов Ю. В.

Москва

2020

Оглавление

Введение	3
1 Аналитический раздел	4
1.1 Описание алгоритмов	4
1.1.1 Сортировка пузырьком	4
1.1.2 Сортировка вставками	4
1.1.3 Сортировка слиянием	5
1.1.4 Модель вычислений	5
1.2 Вывод	6
2 Конструкторский раздел	7
2.1 Разработка алгоритмов	7
2.1.1 Сортировка пузырьком	7
2.1.2 Сортировка вставками	8
2.1.3 Сортировка слиянием	9
2.2 Оценка трудоемкости	10
2.2.1 Сортировка пузырьком	10
2.2.2 Сортировка вставками	10
2.2.3 Сортировка слиянием	10
2.3 Вывод	10
3 Технологический раздел	11
3.1 Средства реализации	11
3.2 Листинг кода	11
3.3 Описание тестирования	13
3.4 Вывод	13
4 Экспериментальный раздел	14
4.1 Примеры работы	14
4.2 Результаты тестирования	14
4.3 Постановка эксперимента по замеру времени	14

4.3.1	Сортировка пузырьком	15
4.3.2	Сортировка вставками	15
4.3.3	Сортировка слиянием	16
4.3.4	Среднее время	16
4.3.5	Лучшее время	17
4.4	Вывод	17
Заключение		18
Литература		18

Введение

Алгоритм сортировки - это алгоритм для упорядочивания элементов в списке. В случае, когда элемент списка имеет несколько полей, поле, служащее критерием порядка, называется ключом сортировки. На практике в качестве ключа часто выступает число, а в остальных полях хранятся какие-либо данные, никак не влияющие на работу алгоритма.

Целью работы: изучение алгоритмов сортировки массивов, сравнительный анализ времени работы данных алгоритмов, анализ трудоемкости алгоритмов.

Задачи работы:

1. реализовать три различных алгоритма сортировки;
2. теоретически вычислить эффективность алгоритмов;
3. сравнить эффективности алгоритмов по времени.

1 Аналитический раздел

В данном разделе будет приведено описание алгоритмов и модель вычислений для оценок трудоемкости.

1.1 Описание алгоритмов

Алгоритмы сортировки оцениваются по скорости выполнения и эффективности использования памяти. В этом разделе будет приведено три алгоритма: сортировка пузырьком, сортировка вставками и сортировка слиянием.

1.1.1 Сортировка пузырьком

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 или до тех пор, пока на очередном проходе не окажется, что обмены больше не нужны, что означает - массив отсортирован.

Сложность по времени

- лучшее время: $O(n)$
- среднее время: $O(n^2)$
- худшее время: $O(n^2)$

Затраты памяти: $O(1)$

1.1.2 Сортировка вставками

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных

данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

Сложность по времени

- лучшее время: $O(n)$
- среднее время: $O(n^2)$
- худшее время: $O(n^2)$

Затраты памяти: $O(1)$

1.1.3 Сортировка слиянием

Сортируемый массив разбивается на две части примерно одинакового размера. Каждая из получившихся частей сортируется отдельно. Два упорядоченных массива половинного размера соединяются в один.

Сложность по времени

- лучшее время: $O(n \log(n))$
- среднее время: $O(n \log(n))$
- худшее время: $O(n \log(n))$

Затраты памяти: $O(n)$

1.1.4 Модель вычислений

В данной работе используется следующая модель вычислений:

1. Стоимость базовых операций: $F = 1$

(=, *, +, -, /, %, <, <=, >, >=, ==, !=, [], + =, - =, * =, / =)

2. Стоимость цикла *for*

$$F_{for} = f_{init} + f_{compare} + N_{loop} \cdot (f_{body} + f_{inc} + f_{compare}) \quad (1.1)$$

3. Трудоемкость условного оператора *if*

$$F_{if} = f_{compare} + f_{body} = f_{compare} + \begin{cases} f_{min}, & \text{лучший случай} \\ f_{max}, & \text{худший случай} \end{cases} \quad (1.2)$$

1.2 Вывод

Были приведено описание алгоритмов сортировка пузырьком, сортировка вставками и сортировка слиянием, также рассмотрено модель вычислений для оценок трудоемкости.

2 Конструкторский раздел

В данном разделе будет приведено описание схем алгоритмов сортировка пузырьком, сортировка вставками, сортировка слиянием и вычислены их трудоемкости.

2.1 Разработка алгоритмов

На рисунках 2.1, 2.2, 2.3 показаны схемы алгоритмов сортировки.

2.1.1 Сортировка пузырьком

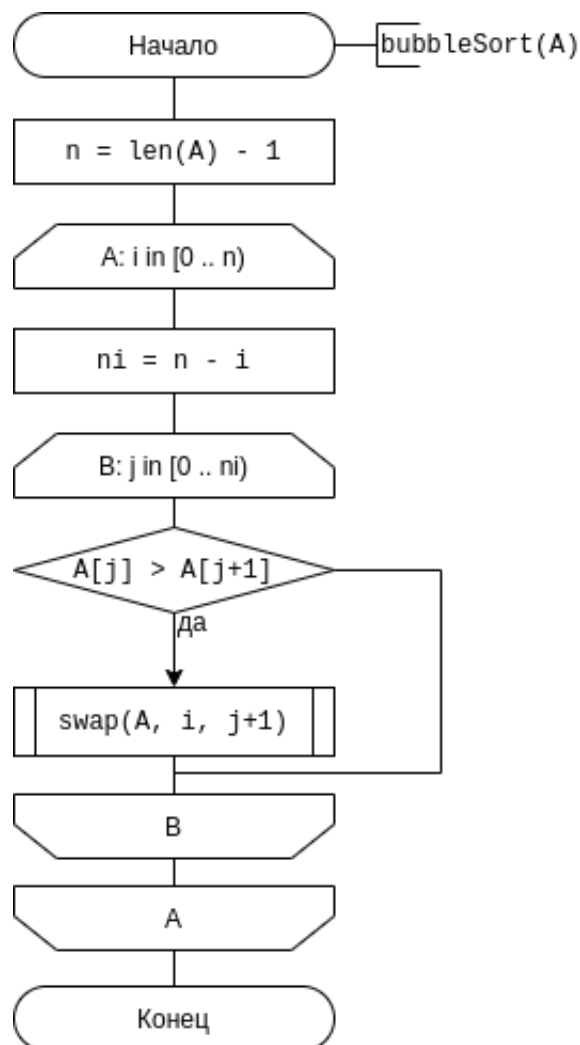


Рис. 2.1: Схема алгоритма сортировка пузырьком

2.1.2 Сортировка вставками

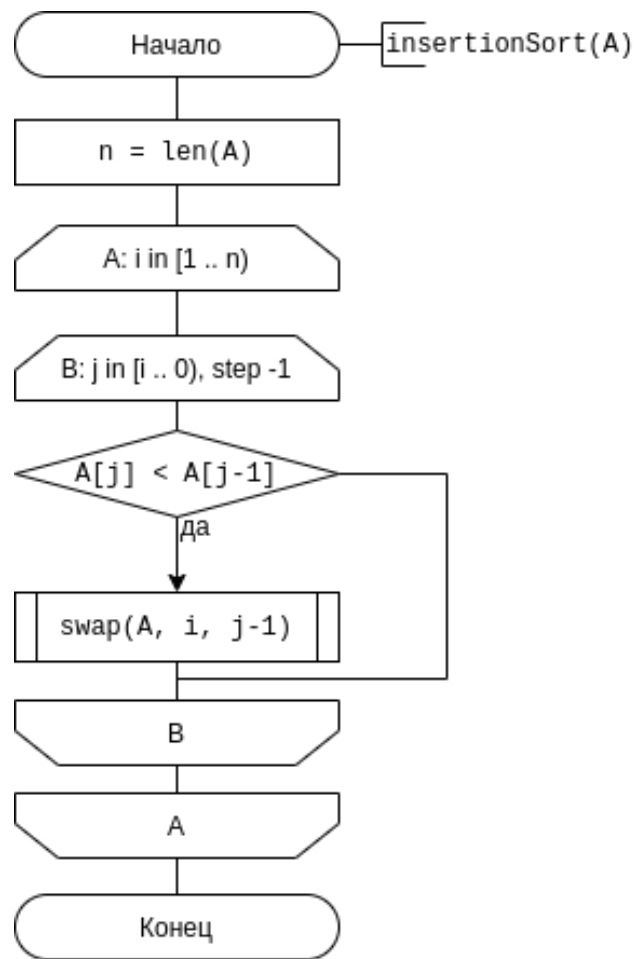


Рис. 2.2: Схема алгоритма сортировка вставками

2.1.3 Сортировка слиянием

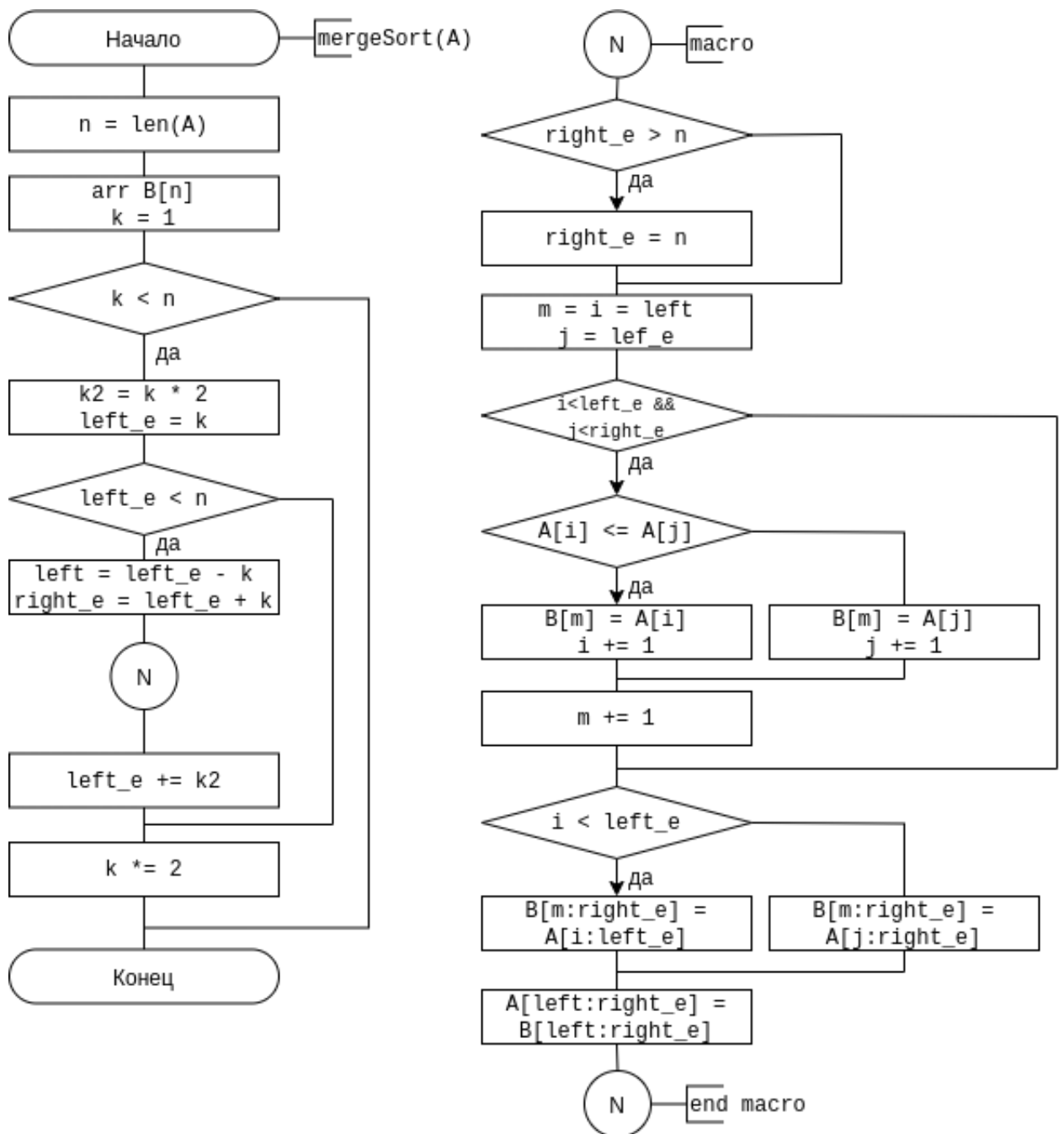


Рис. 2.3: Схема алгоритма сортировка слиянием

2.2 Оценка трудоемкости

Применить формулы 1.1 и 1.2. Предположим, что сложность $swar()$ равна 5, сложность $len()$ равна n .

2.2.1 Сортировка пузырьком

F_{if}	$\begin{cases} 4, & \text{л.с.} \\ 9, & \text{х.с.} \end{cases}$
F_{forA}	$\begin{cases} 4n + 6 \cdot \frac{(n-2)(n-1)}{2} = 3n^2 - 5n + 6, & \text{л.с.} \\ 4n + 11 \cdot \frac{(n-2)(n-1)}{2} = 5.5n^2 - 12.5n + 11, & \text{х.с.} \end{cases}$
F_{bubble}	$\begin{cases} 3n^2 - 4n + 10, & \text{л.с.} \\ 5.5n^2 - 11.5n + 15, & \text{х.с.} \end{cases}$

2.2.2 Сортировка вставками

F_{if}	$\begin{cases} 4, & \text{л.с.} \\ 9, & \text{х.с.} \end{cases}$
F_{forA}	$\begin{cases} 2n + 6 \cdot \frac{(n-2)(n-1)}{2} = 3n^2 - 7n + 6, & \text{л.с.} \\ 2n + 11 \cdot \frac{(n-2)(n-1)}{2} = 5.5n^2 - 14.5n + 11, & \text{х.с.} \end{cases}$
F_{insert}	$\begin{cases} 3n^2 - 6n + 9, & \text{л.с.} \\ 5.5n^2 - 13.5n + 14, & \text{х.с.} \end{cases}$

2.2.3 Сортировка слиянием

В любом случае сложность сортировки слиянием $n \log_2 n$ [4].

2.3 Вывод

В данном разделе было приведено описание схем алгоритмов и вычислены их трудоемкости.

3 Технологический раздел

3.1 Средства реализации

Язык программирования: Python (IPython)

Библиотеки: unittest, timeit, matplotlib, ...

Редактор: Jupyter-Lab

Я использую эти инструменты потому, что они мощные, широко используемые и знакомые мне.

3.2 Листинг кода

Листинг 3.1: Сортировка пузырьком

```
1 def bubbleSort(A):
2     n = len(A) - 1
3     for i in range(n):
4         ni = n-i
5         for j in range(ni):
6             if A[j] > A[j+1]:
7                 A[j], A[j+1] = A[j+1], A[j]
```

Листинг 3.2: Сортировка вставками

```
1 def insertionSort(A):
2     n = len(A)
3     for i in range(1, n):
4         for j in range(i, 0, -1):
5             if A[j] < A[j-1]:
6                 A[j], A[j-1] = A[j-1], A[j]
7         else:
8             break
```

Листинг 3.3: Сортировка слиянием

```
1 def mergeSort(A):
2     n = len(A)
3     B = [None] * n
4     k = 1
5     while k < n:
6         k2 = k * 2
7         left_end = k
8         while left_end < n:
9             left = left_end - k
10            right_end = left_end + k
11
12            if right_end > n: right_end = n
13
14            m = i = left
15            j = left_end
16
17            while i < left_end and j < right_end:
18                if A[i] <= A[j]:
19                    B[m] = A[i]
20                    i += 1
21                else:
22                    B[m] = A[j]
23                    j += 1
24                m += 1
25
26            if i < left_end:
27                B[m:right_end] = A[i:left_end]
28            else:
29                B[m:right_end] = A[j:right_end]
30
31            A[left:right_end] = B[left:right_end]
32
33            left_end += k2
34
35            k *= 2
```

3.3 Описание тестирования

В таблице 3.1 приведен функциональные тесты для алгоритмов сортировки.

Массив	Результат
1	1
1 1 1 1 1	1 1 1 1 1
1 2 3 4 5	1 2 3 4 5
5 4 3 2 1	1 2 3 4 5
2 4 5 1 3	1 2 3 4 5
2 4 2 2 4	2 2 2 4 4
2 4 8 6 4 0	0 2 4 4 6 8

Таблица 3.1: Функциональные тесты

3.4 Вывод

В этом разделе было рассмотрено код программы и описание тестирования.

4 Экспериментальный раздел

4.1 Примеры работы

На рисунке 4.1 приведен пример работы программы.

```
Input:  [157  8  6 172 169 ... 68 20  7 95 142]
Output: [ 1  2  3  4  5 ... 196 197 198 199 200]
```

n = 200	Best	Average	Worst
=====			
Bubble sort	0.001582	0.002635	0.004127
Insertion sort	0.000065	0.002377	0.004868
Merge sort	0.000249	0.000359	0.000333

n = 400	Best	Average	Worst
=====			
Bubble sort	0.007427	0.011859	0.013976
Insertion sort	0.000129	0.007216	0.013042
Merge sort	0.000481	0.000677	0.000699

Рис. 4.1: Примеры работы алгоритмов умножения матриц

4.2 Результаты тестирования

На рисунке 4.2 приведен результат теста с использованием фреймворка google test.

```
test_bubble (__main__.TestSorting) ... ok
test_insertion (__main__.TestSorting) ... ok
test_merge (__main__.TestSorting) ... ok

-----
Ran 3 tests in 0.003s

OK
```

Рис. 4.2: Примеры работы алгоритмов умножения матриц

4.3 Постановка эксперимента по замеру времени

Операционная система - Ubuntu 20.04.1 LTS

Процессор - Intel® Core™ i5-7300HQ CPU @ 2.50GHz × 4

4.3.1 Сортировка пузырьком

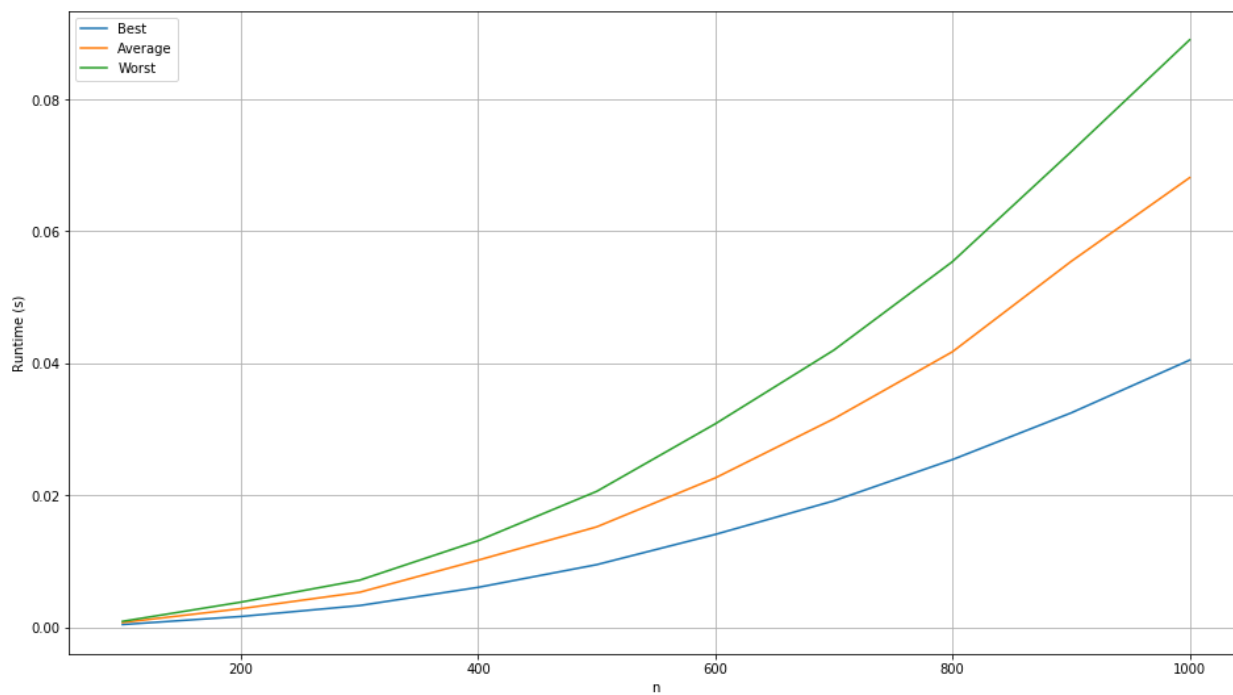


Рис. 4.3: Времени сортировки пузырьком

4.3.2 Сортировка вставками

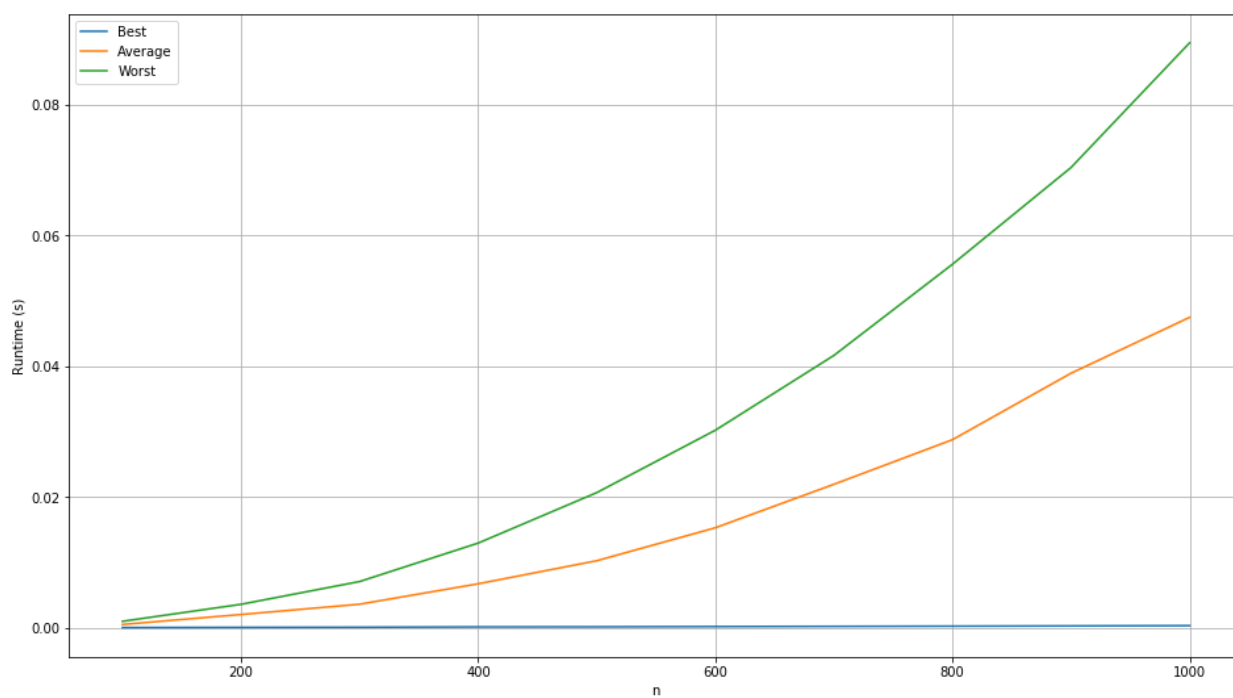


Рис. 4.4: Времени сортировки вставками

4.3.3 Сортировка слиянием

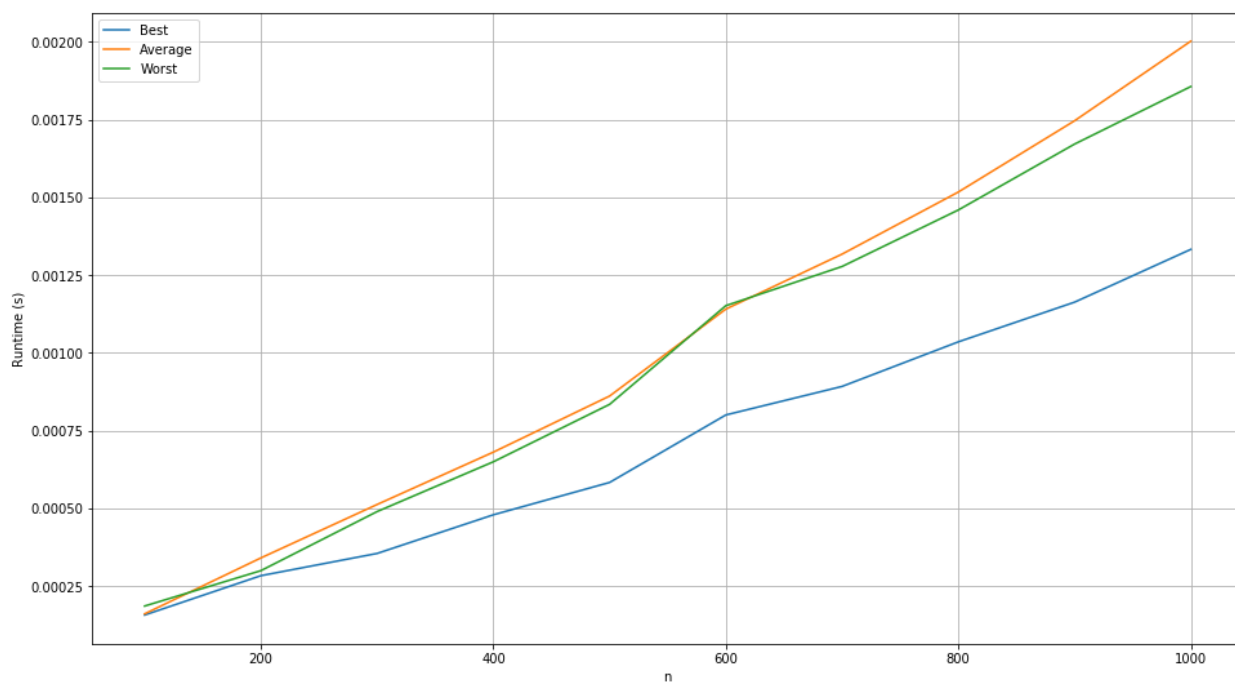


Рис. 4.5: Времени сортировки слиянием

4.3.4 Среднее время

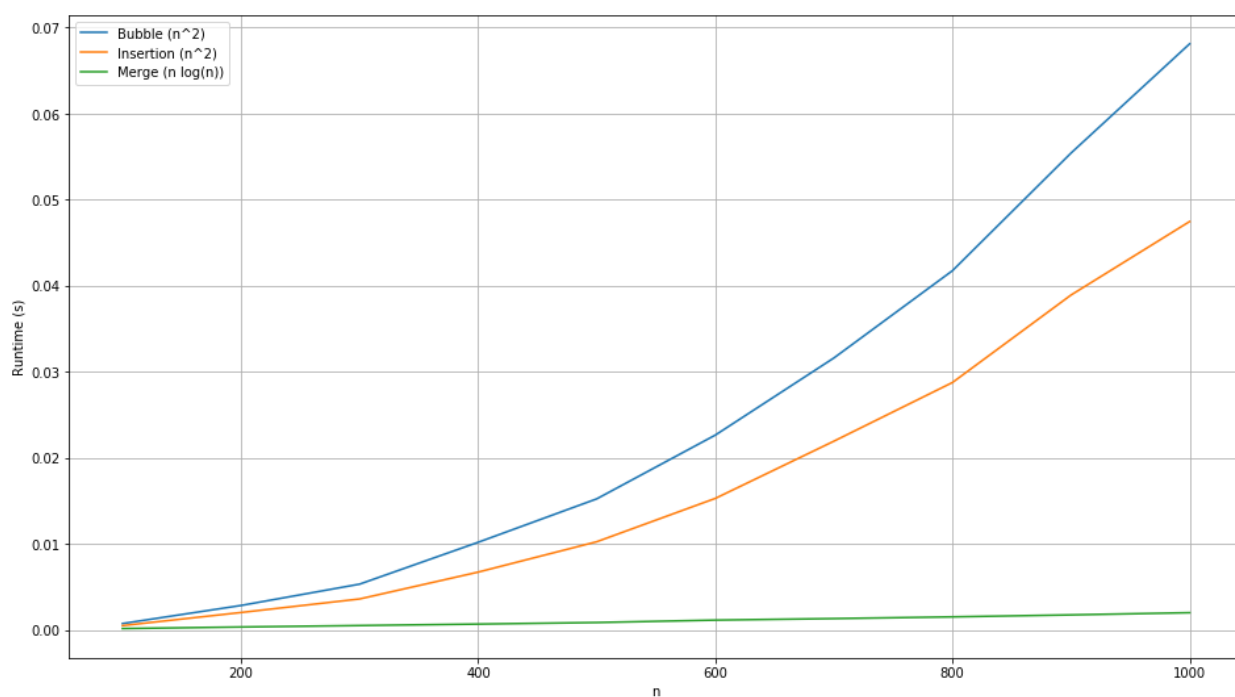


Рис. 4.6: Сравнение среднее времени работы алгоритмов

4.3.5 Лучшее время

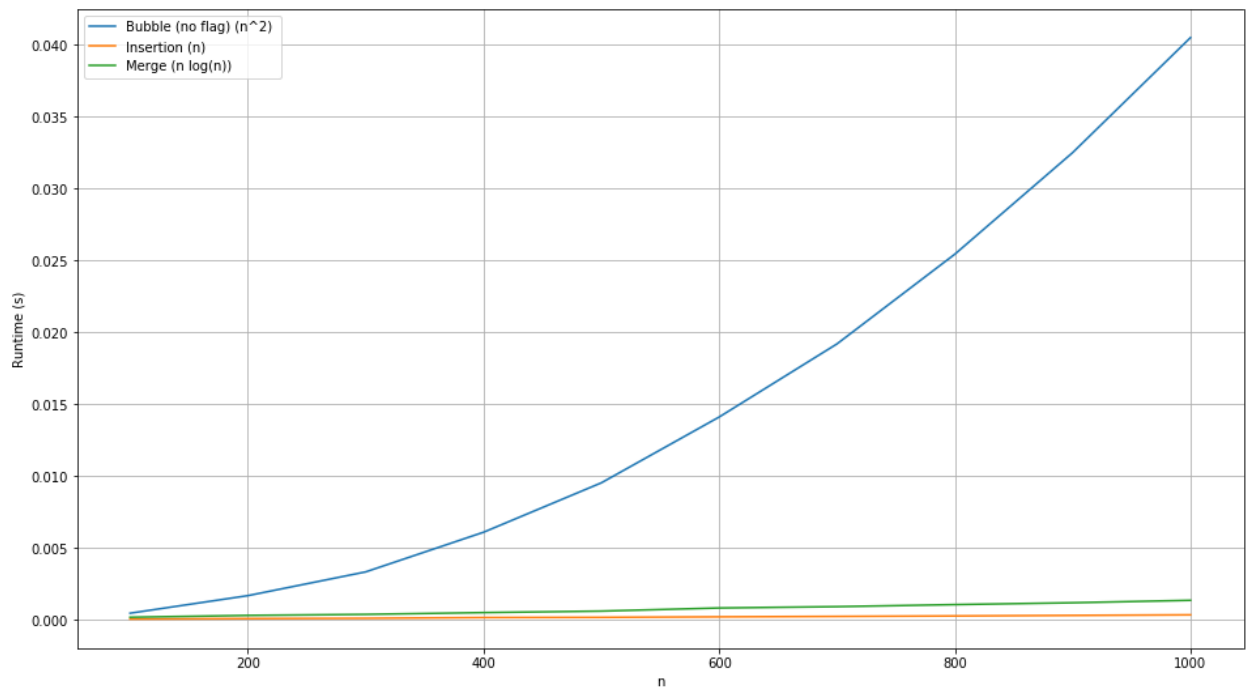


Рис. 4.7: Сравнение лучшее времени работы алгоритмов

4.4 Вывод

Из графика мы видим, что алгоритм сортировки слиянием работает со сложностью $O(n \log(n))$ во всех случаях и является наиболее оптимальным, алгоритм сортировки вставками является лучшим, если массив уже отсортирован или почти отсортирован. Сортировка пузырьком используется в основном как обучающий инструмент. На массив длины 1000, сортировка слиянием быстрее сортировки пузырьком в 33% и быстрее сортировки вставками в 23%.

Заключение

В ходе лабораторной работы было проведено сравнение трех алгоритмов сортировки: сортировка пузырьком, сортировка вставками и сортировка слиянием. Были сделаны следующие выводы:

- сортировка слиянием на порядок быстрее сортировки пузырьком и вставками (использует дополнительную $O(n)$ память);
- сортировка вставками быстрее сортировки пузырьком (без флаг) во всех случаях;
- сортировка вставками самый быстрый для почти отсортирован массив;
- на массив длины 1000, сортировка слиянием быстрее сортировки пузырьком в 33% и быстрее сортировки вставками в 23%.

Литература

- [1] Кнут Д. Э. Искусство программирования. Том 3. Сортировка и поиск = The Art of Computer Programming. Volume 3. Sorting and Searching / под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). — 2-е изд. — Москва: Вильямс, 2007. — Т. 3. — 832 с.
- [2] Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. Алгоритмы: построение и анализ = Introduction to algorithms. — 2-е изд. — М.: «Вильямс», 2006. — С. 1296.
- [3] The Python Language Reference
<https://docs.python.org/3/reference/>
- [4] Analysis of merge sort
<https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/analysis-of-merge-sort>