МГТУ им. Баумана

Лабораторная работа №3

По курсу: "Анализ алгоритмов"

# Исследование алгоритмов сортировки

Студент: Аникин И. А., ИУ7-51Б

Преподаватель: Волкова Л.Л.

*Москва, 2020*

# Содержание

# Введение

В современном мире программы оперируют большим количеством данных. Одной из распространенных задач при работе с данными является задача сортировки.

Сортировка - это расположение элементов в наборе по порядку. Порядок может быть не только числовой или лексикографический, но и задаваться какой-либо функцией, которая возвращает результат для каждого элемента набора.

Существует множество различных алгоритмов сортировки: пузырьком, слиянием, быстрая, Шелла и другие. Эти алгоритмы сортировки отличаются между собой временем работы и потребляемой памятью. Не существует одного самого лучшего алгоритма сортировки, каждый алгоритм хорош или плох в зависимости от исходной задачи.

В данной лабораторной работе рассматриваются три обменные сортировки: пузырьком, выбором и вставками. Их идея наиболее понятна, а сами сортировки просты в реализации.

# 1. Аналитическая часть

Рассмотрим цель и задачи данной лабораторной работы.

Цель лабораторной работы - разработать и провести сравнительный анализ алгоритмов сортировки.
Для достижения данной цели можно выделить следующие задачи:

- разработать алгоритмы сортировки пузырьком, выбором и вставками;
- реализовать разработанные алгоритмы;
- провести тестирование разработанных алгоритмов по методу черного ящика;
- провести замеры времени работы реализаций алгоритмов на наборе данных;
- провести теоретическую оценку трудоемкости алгоритмов;
- провести сравнительный анализ временной эффективности разработанных алгоритмов.

## 1.1. Обзор существующих алгоритмов сортировки

Существует множество различных алгоритмов сортировки. Алгоритмы сортировки отличаются друг от друга по характеристикам. Существуют сортировки, которые работают за квадратичное время, например: сортировка пузырьком, выбором. Эти сортировки используют тривиальные идеи упорядочивания элементов. Такие алгоритмы просты для понимания и реализации. А также не требуют дополнительной памяти, что может быть удобно, например, для микроконтроллеров, где память ограничена.

Вышеописанные сортировки являются обменными, т. е. используют принцип обмена элементов между собой. Существуют более быстрые обменные сортировки, которые работают за $nlog(n)$. Например, сортировка слиянием, быстрая сортировка[1]. Однако, ценой такой производительности является использование дополнительной памяти, что может оказаться невозможным, если данные являются слишком большими.

Существуют также сортировки, которые в своей основе содержат построение структуры данных, с помощью которой можно легко получить отсортированный набор. К таким сортировкам можно отнести пирамидальную сортировку и сортировку двоичным деревом. Эти сортировки работают также за $nlog(n)$

и требуют дополнительную память для построения структуры данных[2].

## 1.2. Сортировка пузырьком

Это алгоритм сортировки, который сравнивает текущий элемент со следующим и если текущий больше следующего, то происходит их обмен.

Таким образом, при каждом пробеге массива самый большой элемент "всплывает"наверх массива.

## 1.3. Сортировка выбором

В этом алгоритме сортировки на каждой итерации цикла ищется минимальный элемент из оставшихся в неотсортированной части массива и обменивается с текущим элементом.

## 1.4. Сортировка вставками

Этот алгоритм сортировки вставляет текущий элемент из неотсортированной части на свое место в отсортированной части массива.

# 2. Конструкторская часть

## 2.1. Требования к ПО

Разработанное должно обладать следующей функциональностью:

- считывание массива целых чисел;
- выбор метода сортировки;
- вывод отсортированного массива;
- осуществление замеров процессорного времени работы реализаций алгоритмов.

## 2.2. Разработка алгоритмов

В ходе данной лабораторной работы будет реализовано 3 различных алгоритма сортировки массива:

- сортировка пузырьком;
- сортировка выбором;
- сортировка вставками.

На рисунке 4 в приложении 1 представлена схема сортировки пузырьком.

На рисунке 5 в приложении 2 представлена схема сортировки выбором.

На рисунке 6 в приложении 3 представлена схема сортировки вставками.

## 2.3. Определение трудоемкости алгортимов

Определим трудоемкость каждого алгоритма. Будем использовать следующую модель вычислений:

- все базовые операции (+, -, *, /, [], =, += , -=, >, <, ==, !=) стоят 1;
- проверка условия стоит 1;
- цикл стоит $f_{initialization} + f_{condition} + n(f_{condition} + f_{increment} + f_{body})$, где $n$ - количество итераций.

### 2.3.1. Сортировка пузырьком

Рассмотрим сортировку пузырьком.

$f_{bubble} = 1 + f_1$, где $f_1$ - трудоемкость внешнего цикла,
$f_1 = 2 + \frac{2n(n+1)}{2} f_2$, где $f_2$ - трудоемкость тела внутреннего цикла,
$f_2^\vee = 4$ - происходит проверка, но обмена не происходит,
$f_2^\wedge = 13$ - происходит проверка и обмен.
$f_{bubble}^\vee = 1 + 2 + 4n(n+1) = 4n^2 + 4n + 3$
$f_{bubble}^\wedge = 13n^2 + 13n + 3$

### 2.3.2. Сортировка выбором

Рассмотрим сортировку выбором.

$f_{selection} = 1 + f_1$, где $f_1$ - трудоемкость внешнего цикла,
$f_1 = 2 + 3n + \frac{2n(n+1)}{2} f_2$, где $f_2$ - трудоемкость тела внутреннего цикла,
$f_2^\vee = 2$ - происходит проверка, но минимум не обновляется,
$f_2^\wedge = 5$ - происходит проверка и обновление минимума.
$f_{selection}^\vee = 2n^2 + 7n + 3$
$f_{selection}^\wedge = 10n^2 + 13n + 3$

### 2.3.3. Сортировка вставками

Рассмотрим сортировку вставками.

$f_{insertion} = 1 + f_1$, где $f_1$ - трудоемкость внешнего цикла,
$f_1^\vee = 2 + (n-1) + 7n$, где $f_2$ - условие внутреннего цикла выполняется один раз,
$f_1^\wedge = 2 + (n-1) + (7 + f_2)\frac{n(n+1)}{2}$
$f_2 = 9$ - трудоемкость тела внутреннего цикла (обмен элементов).
$f_{insertion}^\vee = 8n + 1$
$f_{insertion}^\wedge = 16n^2 + 17n + 1$

# 3. Технологическая часть

В ходе выполнения лабораторной работы были реализованы следующие модули:

- модуль sorts.py, в котором реализованы алгоритмы сортировки;
- модуль test.py, предназначенный для тестирования реализованных алгоритмов;
- модуль analysis.py, предназначенный для замеров времени выполнения алгоритмов на различных входных данных и построения графиков;
- модуль main.py, предоставляющий пользователю возможности работы с алгоритмами сортировки.

В качестве языка разработки был выбран язык python, т. к. он имеет ряд преимуществ для достижения цели лабораторной работы:

- высокая скорость разработки программ;
- простой синтаксис языка;
- наличие большого количества библиотек, что позволяет легко анализировать полученные результаты и проводить тестирование.

## 3.1. Листинги алгоритмов

На листинге 1 приведен алгоритм сортировки пузырьком.

Листинг 1: Алгоритм сортировки пузырьком

```
def bubble_sort(a):
    n = len(a)
    for i in range(n-1):
        for j in range(n-i-1):
            if a[j] > a[j+1]:
                a[j], a[j+1] = a[j+1], a[j]
    return a
```

На листинге 2 приведен алгоритм сортировки выбором.

Листинг 2: Алгоритм сортировки выбором

```
def selection_sort(a):
    n = len(a)
    for i in range(n):
```

```
        m, i_m = a[i], i
        for j in range(i, n):
            if a[j] < m:
                m, i_m = a[j], j
        a[i], a[i_m] = a[i_m], a[i]
    return a
```

На листинге 3 приведен алгоритм сортировки вставками.

Листинг 3: Алгоритм сортировки вставками

```
def insertion_sort(a):
    n = len(a)
    for i in range(1, n):
        j = i
        while j > 0 and a[j] < a[j-1]:
            a[j], a[j-1] = a[j-1], a[j]
            j -= 1
    return a
```

## 3.2. Тестирование

Тестирование разработанных алгоритмов проводилось на наборе тестов, который представлен в таблице 1.

Таблица 1: Набор тестов

| Входные данные | Ожидаемые выходные данные |
|---|---|
| 0 1 2 3 4 5 6 7 8 9 | 0 1 2 3 4 5 6 7 8 9 |
| 2 0 1 6 4 3 9 7 5 8 | 0 1 2 3 4 5 6 7 8 9 |
| 9 8 7 6 5 4 3 2 1 0 | 0 1 2 3 4 5 6 7 8 9 |

Все тесты пройдены успешно.

# 4. Исследовательская часть

## 4.1. Сравнение временных характеристик алгоритмов

Проведем исследование временных характеристик реализованных алгоритмов.

Замеры времени происходили на следующей архитектире: Intel Xeon E3-1270 V2, 16 Гб ОЗУ. Ниже представлены графики, показывающие временя работы реализаций алгоритмов на различных входных данных. На вход подается массив.

На рисунке 1 изображено время работы в лучшем случае, где входные данные - отсортированный массив.

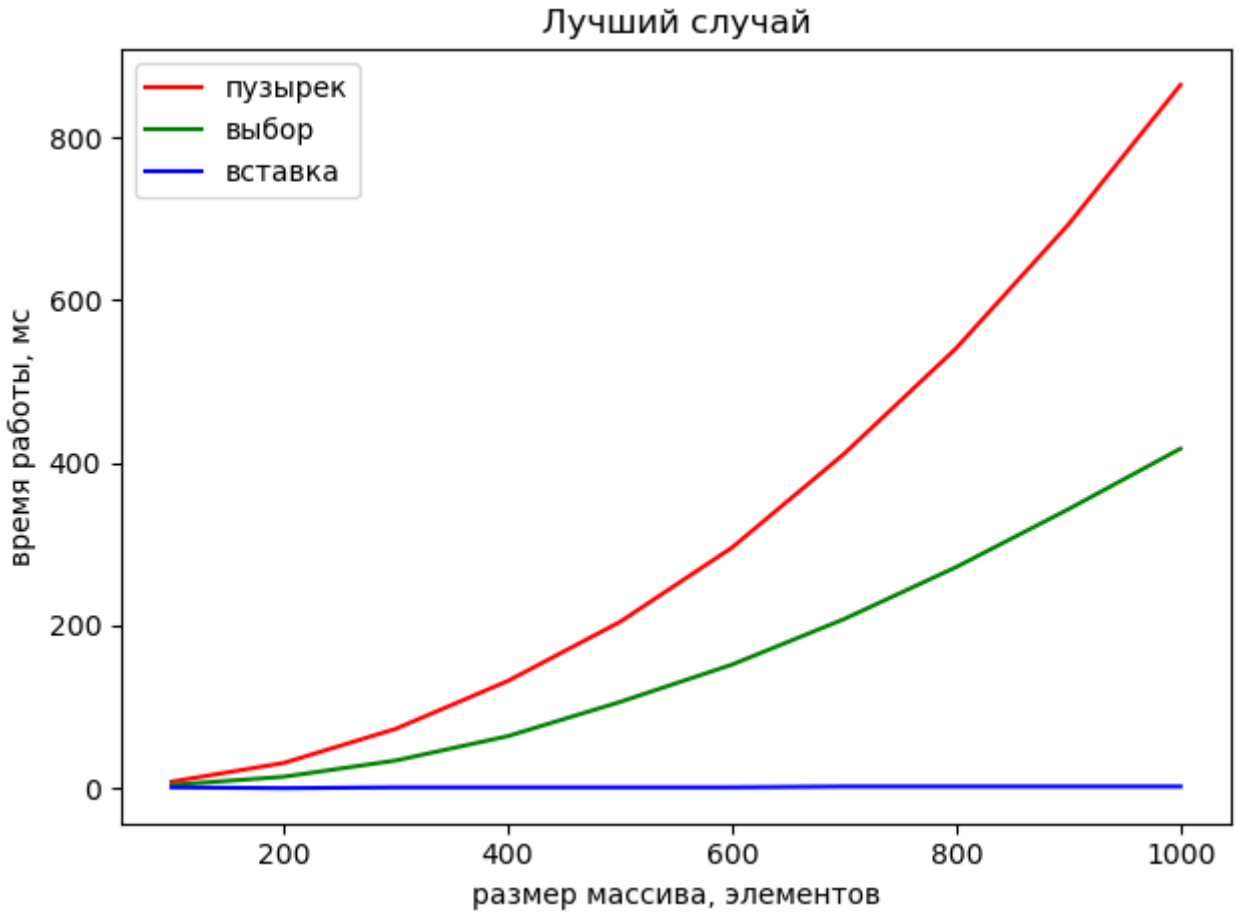


Рис. 1: Лучший случай для трех сортировок

На рисунке 2 изображено время работы в произвольном случае, где входные данные - неупорядоченный массив.

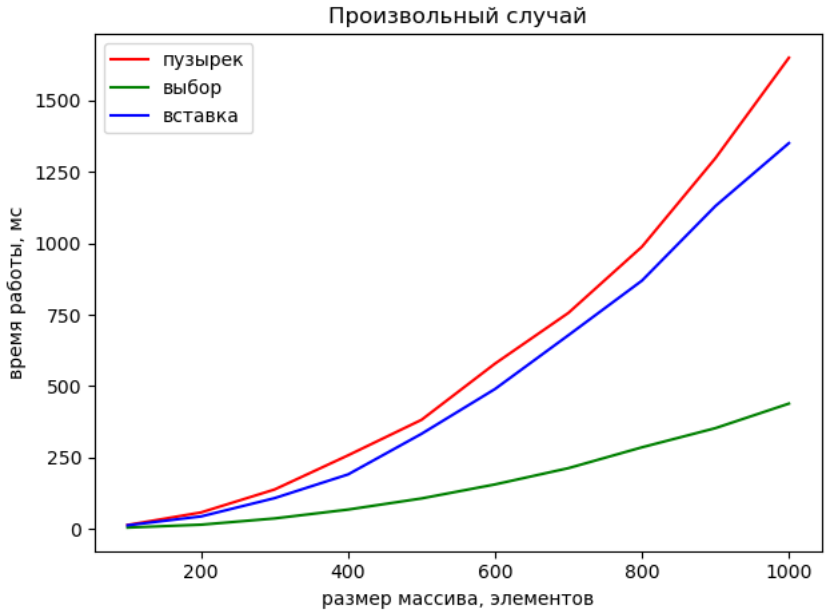


Рис. 2: Произвольный случай для трех сортировок

На рисунке 3 изображено время работы в худшем случае, где входные данные - обратно отсортированный массив.

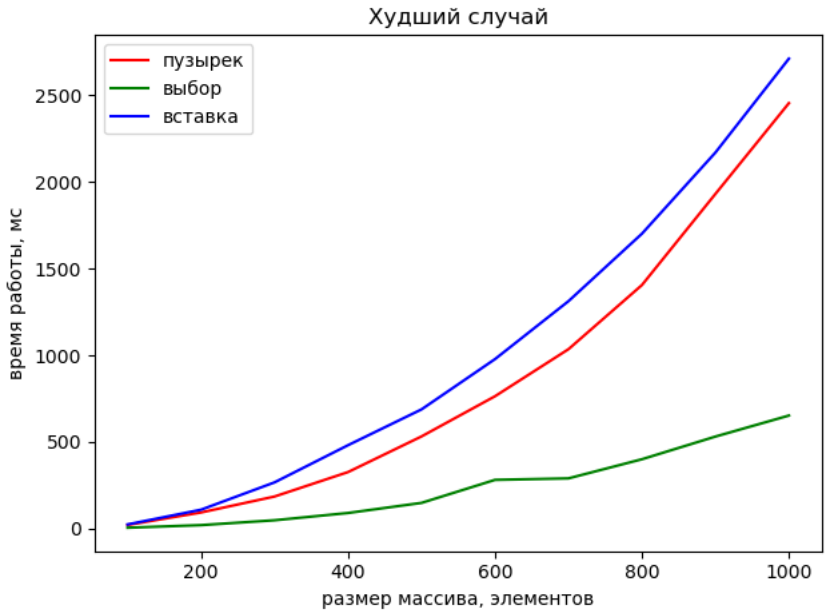


Рис. 3: Худший случай для трех сортировок

Исходя из данных графиков видно, что в лучшем случае наиболее быстрым является алгоритм сортировки вставками. Это происходит потому, что

внутренний цикл алгоритма не будет выполняться в том случае, если текущий элемент меньше следующего. Полученные графики подтверждают теоретические оценки трудоемкости.

В худшем и произвольном случаях наиболее быстрой является сортировка выбором. Это происходит потому, что во внутреннем цикле не происходит обменов, а лишь поиск минимума и его индекса. Обмены происходят только во внешнем цикле. Самой медленной является сортировка вставками, т. к. каждая итерация внутреннего цикла требует много действий на проверку условий и совершение обмена.

К применению рекомендуется сортировка выбором как наименее трудоемкая и наименее затратная по времени в большинстве возможных случаев (проигрывает она при наступлении лучшего случая).

# Заключение

В ходе выполнения лабораторной работы были выполнены следующие задачи:

- разработаны алгоритмы сортировки пузырьком, выбором и вставками;
- реализованы разработанные алгоритмы;
- проведено тестирование разработанных алгоритмов по методу черного ящика;
- проведены замеры времени работы реализаций алгоритмов на наборе данных;
- проведена теоретическую оценку трудоемкости алгоритмов;
- проведен сравнительный анализ временной эффективности разработанных алгоритмов.

Цель лабораторной работы достигнута, все задачи выполнены.

По итогам достигнутых задач можно сделать следующие выводы.

Наиболее быстрой в произвольном и худшем случаях является сортировка выбором. Она является наиболее быстрой т. к. во внутреннем цикле производится минимальное количество операций.

Наиболее медленной в худшем случае является сортировка вставками, т. к. требует большого количества обменов и проверок на каждой итерации внутреннего цикла.

# Список литературы

[1] Кнут, Д. Э. Искусство программирование, том 3. Сортировка и поиск. -Изд. 2. -М.: ООО "И. Д. Вильямс 2007. -832с.

[2] Бхаргава, А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. -СПб.: "Питер 2018. -288с.

# Приложения

Приложение 1

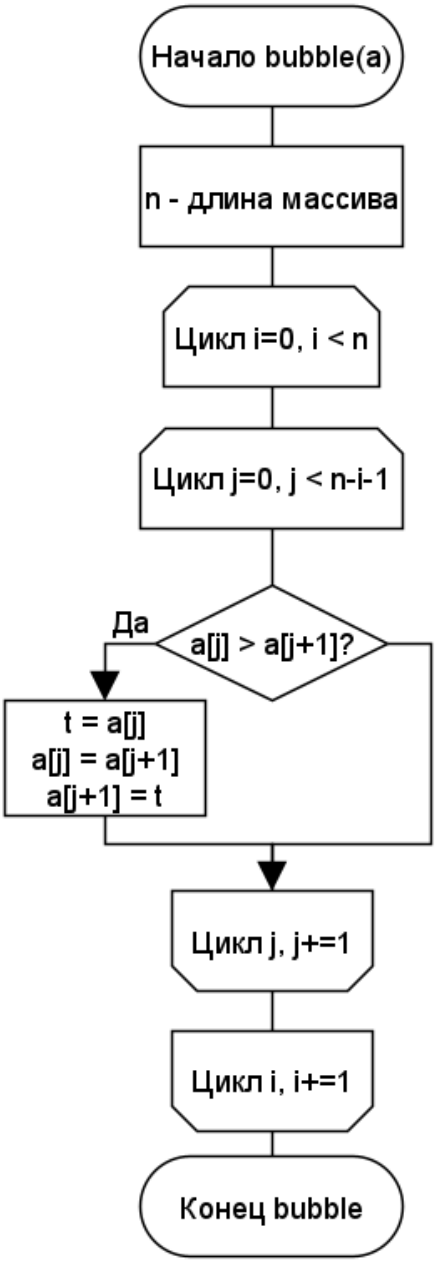


Рис. 4: Схема алгоритма сортировки пузырьком

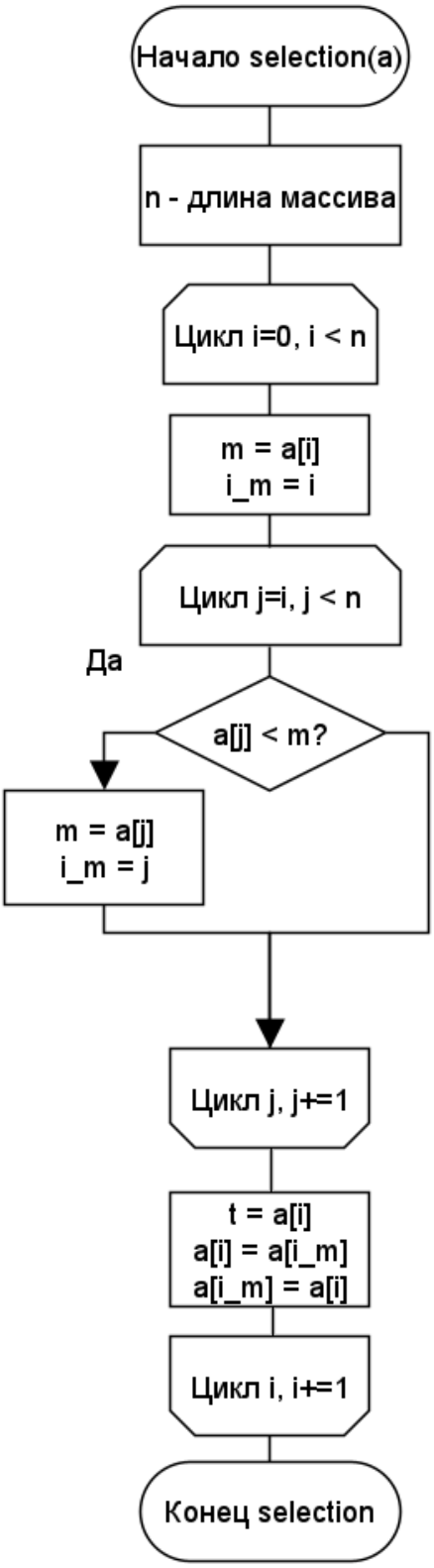


Рис. 5: Схема алгоритма сортировки выбором

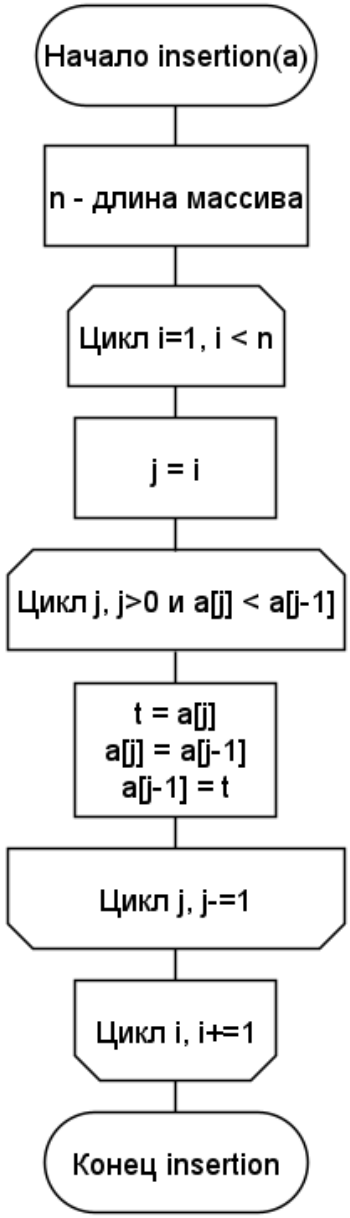


Рис. 6: Схема алгоритма сортировки вставками