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»

ДИСЦИПЛИНА «Анализ алгоритмов»

Лабораторная работа № 3

Тема «Алгоритмы сортировки»

Студент Чалый А.А.

Группа ИУ7 – 52Б

Оценка (баллы) _____

Преподаватели Волкова Л.Л.
Строганов Ю.В.

Москва.
2020 г.

Оглавление

Введение

Целью данной работы является изучение алгоритмов сортировки, получение практических навыков реализации алгоритмов сортировки, приобретение навыков расчета трудоемкости алгоритмов, а также получение навыков в сравнении алгоритмов.

Требуется реализовать описанные ниже алгоритмы:

1. сортировка «пузырьком»;
2. сортировка «вставками»;
3. сортировка «выбором».

1. Аналитическая часть

В данном разделе будет рассмотрено описание алгоритмов сортировки.

1.1 Описание алгоритмов

Сортировка массива — одна из самых популярных операций над массивом. Алгоритмы реализуют упорядочивание 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 [2].

Эти алгоритмы активно применяются во всех областях, в которых необходимо хранить последовательность чисел, такие как:

- математика;
- физика;
- экономика;
- и так далее.

1.2 Алгоритм сортировки «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. 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).

1.3 Алгоритм сортировки «вставками»

В сортировке вставками массив делится на 2 части — отсортированную и неотсортированную. Из неотсортированной части извлекается любой элемент. Поскольку другая часть массива отсортирована, то в ней достаточно быстро можно найти своё место для этого извлечённого элемента. Элемент вставляется куда нужно, в результате чего отсортированная часть массива увеличивается, а неотсортированная уменьшается. По данному принципу и работает сортировка «вставками».

1.4 Алгоритм сортировки «выбором»

Пусть имеется массив A размером N, тогда сортировка выбором сводится к следующему:

1. берем первый элемент последовательности A[i], здесь i – номер элемента, для первого i равен 1;
2. находим минимальный (максимальный) элемент последовательности и запоминаем его номер в переменную key;
3. если номер первого элемента и номер найденного элемента не совпадают, т. е. если key ≠ 1, тогда два этих элемента обмениваются значениями, иначе никаких манипуляций не происходит;
4. увеличиваем i на 1 и продолжаем сортировку оставшейся части массива, а именно с элемента с номером 2 по N, так как элемент A[1] уже занимает свою позицию;

1.5 Вывод

Были рассмотрены алгоритмы сортировки «пузырьком», «вставками», «выбором», принципиальная разница которых – обход массива разными способами, вследствие чего сортировка происходит по-разному.

2. Конструкторская часть

В данном разделе будут рассмотрены схемы алгоритмов:

- сортировки «пузырьком»;
- сортировки «вставками»;
- сортировки «выбором».

2.1 Разработка алгоритмов

На рис. 1-3 приведены схемы указанных алгоритмов.

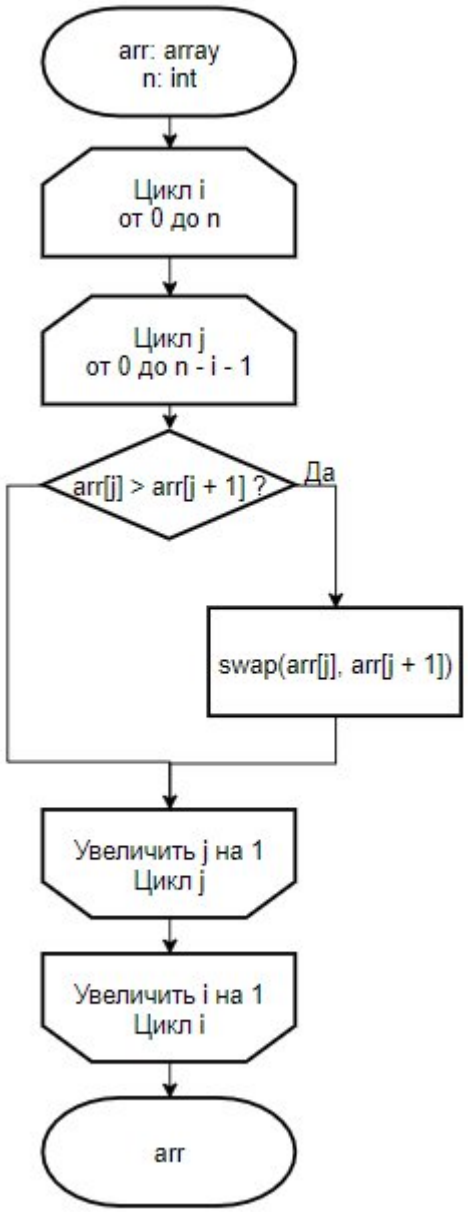

Рисунок 1. Схема алгоритма сортировки «пузырьком»

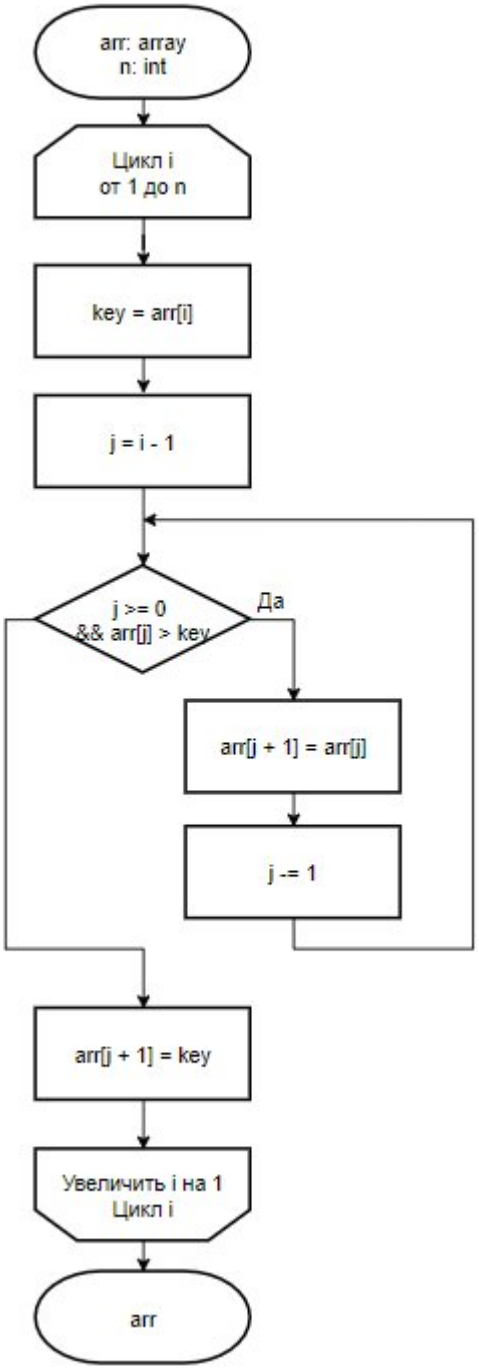

Рисунок 2. Схема алгоритма сортировки «вставками»

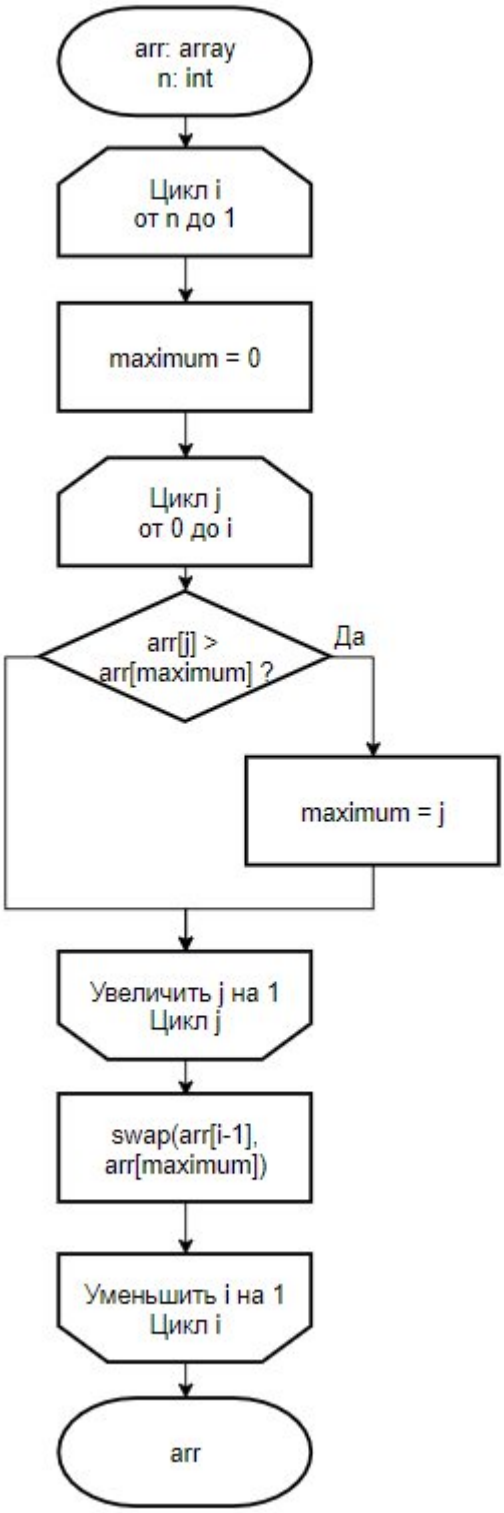

Рисунок 3. Схема алгоритма сортировки «выбором»

2.3 Трудоемкость алгоритмов

Введем модель трудоемкости для оценки алгоритмов:

1. базовые операции стоимостью 1 — +, -, *, /, =, ==, <=, >=, !=, +=, [];
2. оценка трудоемкости цикл $F_{\text{ц}} = F_{\text{иниц}} + N * (F_{\text{срав}} + F_{\text{тела}})$

3. Стоимость условного перехода возьмем за 0, стоимость вычисления условия остаётся.

В таблицах 1–3 приведены оценки трудоемкости алгоритмов.

Таблица 1. Оценка трудоемкости алгоритма сортировки «пузырьком»

Трудоемкость	Оценка трудоемкости
Fbubble наил.	2 + (N - 1) * (3 + 1 + 2 + (N / 2) * (2 + 3))
Fbubble наил. итог	2.5 * N * N + 2.5 * N - 3
Fbubble наих.	2 + (N - 1) * (2 + 1 + 2 + (N / 2) * (2 + 3 + 7))
Fbubble наих. итог	6 * N * N - 1 * N - 3

Таблица 2. Оценка трудоемкости алгоритма сортировки «вставками»

Трудоемкость	Оценка трудоемкости
Finsert наил.	2 + N * (2 + 2 + 3 + 6)
Finsert наил. итог	13 * N + 2
Finsert наих.	2 + N * (2 + 2 + 3 + 6 + (N - 1) / 4 * (5 + 4))
Finsert наих. итог	4.5 * N * N + 8.5 * N + 2

Таблица 3. Оценка трудоемкости алгоритма сортировки «выбором»

Части алгоритма	Трудоемкость
Fselect наил.	2 + (N - 1) * (3 + 11 + 3 + (N / 2) * (2 + 2))
Fselect наил. итог	2 * N * N + 15 * N - 15
Fselect наих.	2 + (N - 1) * (3 + 11 + 3 + (N / 2) * (2 + 2 + 3))
Fselect наих. итог	3.5 * N * N + 13.5 * N - 15

2.3 Вывод

В данном разделе были построены схемы алгоритмов:

1. сортировки «пузырьком»;
2. сортировки «вставками»;
3. сортировки «выбором».

Была проведена оценка трудоемкости алгоритмов, в ходе которой было выяснено, что сортировка вставками показывает наилучший результат при уже отсортированном массиве. Если брать наихудший итог, то сортировка выбором пусть и ненамного, но выигрывает по скорости работы у остальных алгоритмов сортировки.

3.Технологическая часть

В данном разделе будут рассмотрены требования к программному обеспечению, средства реализации и представлен листинг кода.

3.1 Требования к программному обеспечению

Входные данные: arr – массив чисел, n – длина массива.

Выходные данные: отсортированный массив чисел.

Функциональная схема процесса перемножения двух матриц в нотации IDEF0 представлена на рис. 4.

Рисунок 4. Функциональная схема процесса сортировки массива

3.2 Средства реализации

В данной работе был использован язык программирования Python. Проект выполнен в IDE PyCharm с использованием сторонней библиотеки для создания массивов numpy.

Для замера процессорного времени была использована python библиотека time [1].

3.3 Листинг кода

В данном пункте представлен листинг кода функций:

1) алгоритм сортировки «пузырьком»;
2) алгоритм сортировки «вставками»;
3) алгоритм сортировки «выбором».

Листинг 1. Функция сортировки «пузырьком»

```
def bubble_sort(arr, n):
    i = 0
    while i < n:
        j = 0
        while j < n - i - 1:
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

```
            j += 1
        i += 1
    return arr
```

Листинг 2. Функция сортировки «вставками»

```
def insertion_sort(arr, n):
    i = 1
    while i < n:
        key = arr[i]
        j = i - 1
        while j >=0 and key < arr[j]:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key
        i += 1
    return arr
```

Листинг 3. Функция сортировки «выбором»

```
def select_sort(arr, n):
    i = n
    while i > 1:
        maximum = 0
        j = 0
        while j < i:
            if arr[j] > arr[maximum]:
                maximum = j
            j += 1
        arr[i-1], arr[maximum] = arr[maximum], arr[i-1]
        i -= 1
    return arr
```

3.4 Тестирование

В тестах будет рассмотрена проверка работы алгоритмов при:

- отсортированном массиве;
- отсортированном в обратном порядке массиве;
- массиве, содержащим отрицательные числа;
- массиве, содержащем одинаковые элементы.

Тестирование производится методом черного ящика.

На рисунках ниже будут приведены тесты с целью демонстрации корректности работы программы.

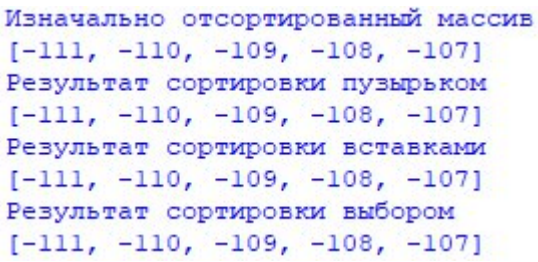

Рисунок 5. Проверка работы алгоритмов при отсортированном массиве

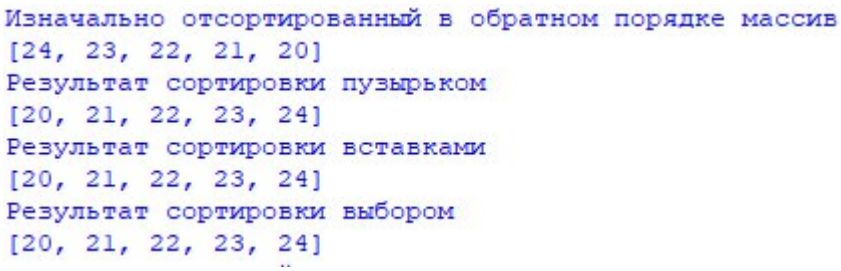

Рисунок 6. Проверка работы алгоритмов при отсортированном в обратном порядке массиве

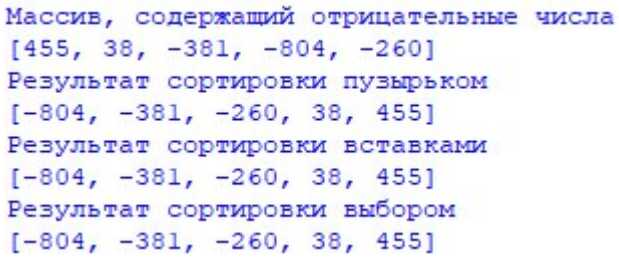

Рисунок 7. Проверка работы алгоритмов при массиве, содержащим отрицательные числа

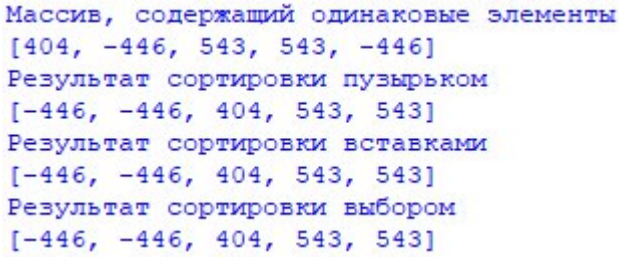

Рисунок 8. Проверка работы алгоритмов при массиве, содержащим повторяющиеся элементами

Все тесты пройдены успешно.

3.5 Вывод

В данном разделе была представлена реализация алгоритмов сортировки «пузырьком», сортировки «вставками», сортировки «выбором».

Также проведено тестирование разработанных методов по методу чёрного ящика.

4. Экспериментальная часть

В данном разделе будет проведен эксперимент и сравнительный анализ полученных данных.

4.1 Постановка эксперимента

В рамках данной части были проведены эксперименты, описанные ниже.

1. Сравнение времени работы трех алгоритмов при массивах случайных чисел. Создаются массивы размерностью 5 и 10, заполненные случайными числами. Эксперимент по измерению одной размерности ставился 100000 раз;
2. Сравнение времени работы трех алгоритмов при уже отсортированных массивах. Создаются массивы размерностью 5 и 10, заполненные уже отсортированными числами. Эксперимент по измерению одной размерности ставился 100000 раз;
3. Сравнение времени работы трех алгоритмов при числах отсортированных в обратном порядке. Создаются массивы размерностью 5 и 10, заполненные числами отсортированными в обратном порядке. Эксперимент по измерению одной размерности ставился 100000 раз;

4.2 Сравнительный анализ на материале экспериментальных данных

Сравнение времени работы алгоритмов при массивах, заполненных случайными числами, с длинами 5 и 10 и количестве итераций 100000 изображены на рис. 9 - 10.

```
Количество итераций: 100000
Длина массива: 5
Наполнение: случайные числа
Сортировка пузырьком =  4.21875e-05
Сортировка вставками =  2.1875e-05
Сортировка выбором =  3.59375e-05
```

Рисунок 9. Время работы алгоритмов при массиве длиной 5

```
Количество итераций: 100000
Длина массива: 10
Наполнение: случайные числа
Сортировка пузырьком =  0.0001578125
Сортировка вставками =  7.1875e-05
Сортировка выбором =  0.0001
```

Рисунок 10. Время работы алгоритмов при массиве длиной 10

Сравнение времени работы алгоритмов при массивах, заполненных уже отсортированными числами, с длинами 5 и 10 и количестве итераций 100000 изображены на рис. 11 - 12.

```
Количество итераций: 100000
Длина массива: 5
Наполнение: отсортированные числа
Сортировка пузырьком =  2.8125e-05
Сортировка вставками =  1.5625e-05
Сортировка выбором =  3.59375e-05
```

Рисунок 11. Время работы алгоритмов при массиве длиной 5

```
Количество итераций: 100000
Длина массива: 10
Наполнение: отсортированные числа
Сортировка пузырьком =  0.0001046875
Сортировка вставками =  2.5e-05
Сортировка выбором =  0.0001109375
```

Рисунок 12. Время работы алгоритмов при массиве длиной 10

Сравнение времени работы алгоритмов при массивах, заполненных числами отсортированными в обратном порядке, с длинами 5 и 10 и количестве итераций 100000 изображены на рис. 13 - 14.

```
Количество итераций: 100000
Длина массива: 5
Наполнение: отсортированные в обратном порядке числа
Сортировка пузырьком =  4.84375e-05
Сортировка вставками =  2.96875e-05
Сортировка выбором =  3.4375e-05
```

Рисунок 13. Время работы алгоритмов при массиве длиной 5

```
Количество итераций: 100000
Длина массива: 10
Наполнение: отсортированные в обратном порядке числа
Сортировка пузырьком =  0.00019375
Сортировка вставками =  0.0001109375
Сортировка выбором =  0.0001046875
```

Рисунок 14. Время работы алгоритмов при массиве длиной 10

4.3 Вывод

В данном разделе был поставлен эксперимент по замеру времени выполнения каждого алгоритма. По итогам замеров можно сделать вывод о том, что при любых входных данных алгоритм сортировки «вставками» является самым быстрым среди всех. Тем не менее, все три алгоритма имеют одинаковый характер трудоемкости — квадратный от длины массива.

Заключение

В ходе работы были изучены алгоритмы сортировки: пузырьком, вставками, выбором. Выполнена оценка трудоемкости всех вышеперечисленных алгоритмов в ходе которой был сделан вывод, что алгоритм сортировки «вставками» является самым быстрым алгоритмом по трудоемкости. Также были произведены замеры времени выполнения алгоритмов, в котором алгоритм сортировки «вставками» оказался самым быстрым. Изучены зависимости времени выполнения алгоритмов от длин массивов. Все поставленные задачи были выполнены. Целью лабораторной работы являлось изучение алгоритмов сортировки массивов, что также было достигнуто.

Список использованных источников

1. time – Time access and conversions: сайт. – URL: https://docs.python.org/3/library/time.html (дата обращения 16.09.2020). – Текст: электронный.
2. Дж. Макконнелл. Анализ алгоритмов. Активный обучающий подход. – М.: Техносфера, 2017. – 267 с.