Министерство цифрового развития и массовых коммуникаций

Российской Федерации Ордена Трудового Красного Знамени

федеральное государственное образовательное бюджетное

учреждение высшего профессионального образования

«Московский технический университет связи и информатики»

**Кафедра «Математическая кибернетика и**

**информационные технологии»**

**Отчет по лабораторной работе №1**

по дисциплине«Структуры и алгоритмы обработки данных»

на тему: «**Методы сортировки**»

Выполнил: студент

группы БСТ2001

Ибодуллоев У.Х.

Проверил:

Ст. преп. Чайка А.Д.

Москва, 2022

Оглавление

[**1. Цель работы 3**](#_Toc102087707)

[**2. Задания 3**](#_Toc102087708)

[**Задание №1: 3**](#_Toc102087709)

[**Задание №2: 3**](#_Toc102087710)

[**Задание №3: 3**](#_Toc102087711)

[**Задание №4: 3**](#_Toc102087712)

[**3. Ход работы 3**](#_Toc102087713)

[**Задание №1: 3**](#_Toc102087714)

[**Задание №2: 3**](#_Toc102087715)

[**Задание №3: 3**](#_Toc102087716)

[**1. Выбором 3**](#_Toc102087717)

[**2. Вставкой 3**](#_Toc102087718)

[**3. Обменом (пузырьковая) 3**](#_Toc102087719)

[**4. Шелла 3**](#_Toc102087720)

[**5. Турнирная (слияние) 3**](#_Toc102087721)

[**6. Быстрая сортировка 3**](#_Toc102087722)

[**7. Пирамидная 3**](#_Toc102087723)

[**Задание №4 3**](#_Toc102087724)

[**4. Вывод 3**](#_Toc102087725)

[**5. Ссылка на удалённый репозиторий 3**](#_Toc102087726)

[**6. Список использованных источников 3**](#_Toc102087727)

# Цель работы

Научится использовать различные способы сортировки на языке программирования python, а также научиться работать с такими сервисами как JupyterLab и GitHub.

# Задания

## Задание №1:

1. Создать Jupyter Notebook со следующим наименованием: Lab1\_Группа\_ФИО

2. Создать новую ячейку с помощью кнопки

3. В созданной ячейке по указанной ниже форме заполните оглавление файла, заменив наименование группы и вписав свое ФИО, Рисунок 1 - Форма оглавления файла в ячейке после чего создайте еще одну ячейку и напишите следующий код:

4. С помощью кнопки запустите выполнение всех ячеек.

5. После выполнения у вас должна отформатироваться ячейка с оглавлением и должен выполниться "Hello, World!"

## Задание №2:

Написать генератор случайных матриц(многомерных), который принимает опциональные параметры m, n, min\_limit, max\_limit, где m и n указывают размер матрицы, а min\_lim и max\_lim - минимальное и максимальное значение для генерируемого числа . По умолчанию при отсутствии параметров принимать следующие значения:

m = 50

n = 50

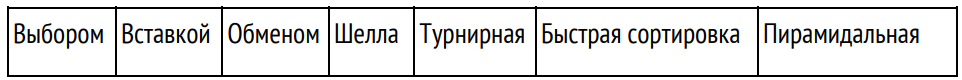
min\_limit = -250

max\_limit = 1000 + (номер своего варианта)

## Задание №3:

Реализовать методы сортировки строк числовой матрицы в соответствии с заданием. Оценить время работы каждого алгоритма сортировки и сравнить его со временем стандартной функции сортировки. Испытания проводить на сгенерированных матрицах.

Методы:



## Задание №4:

Создать публичный репозиторий на github, и запушить выполненное задание в .ipynb формате.

# Ход работы

## Задание №1:

Для начала установим jupyter notebook с помощью командной строки, после установки прописываем в командную строку «jupyter notebook», выбираем нужную папку и создаем python 3 (ipykenernel).

Теперь создадим комментарий выбрав «Markdown» и прописываем комментарий в ячейку, далее выбираем «Code» и прописываем код в ячейку: print (“Hello, World!”). На рисунке 1 изображен результат:

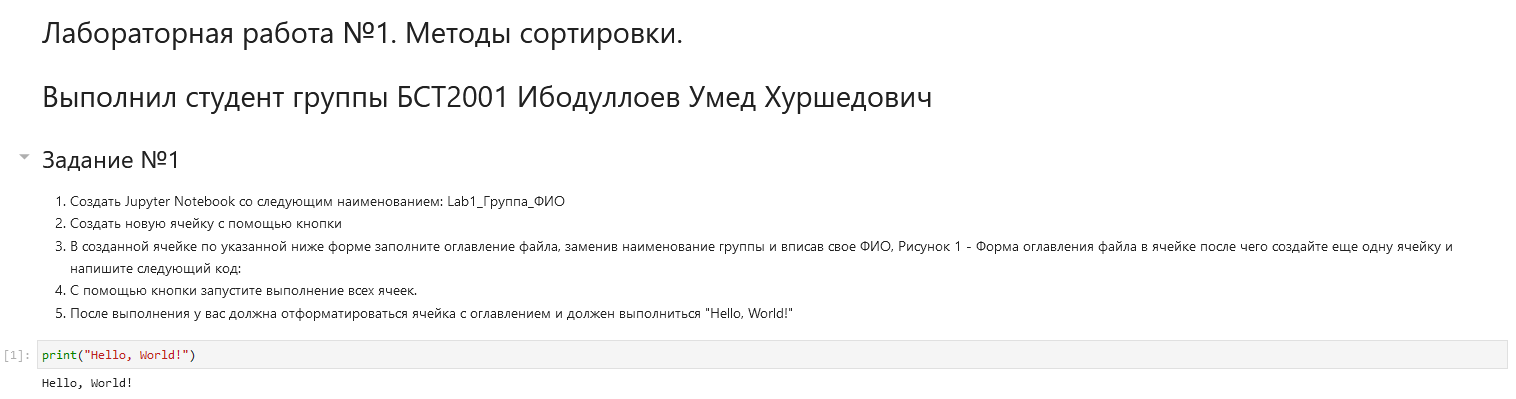
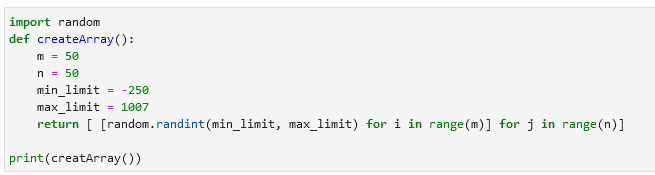


Рисунок 1 - Ячейки после выполнения

Для выполнения заданий 2 и 3 пропишем код в новой ячейке кода. Код можно будет посмотреть в удалённом репозитории github.

## Задание №2:

Создаем функцию createArray(), которая создает двумерный массив с заданными параметрами:



## Задание №3:

### Выбором

В этом алгоритме мы создаем два сегмента нашего списка: один отсортированный, а другой несортированный. В процессе выполнения алгоритма мы каждый раз удаляем самый маленький элемент из несортированного сегмента списка и добавляем его в отсортированный сегмент. Мы не меняем местами промежуточные элементы. Следовательно, этот алгоритм сортирует массив с минимальным количеством перестановок.

***Алгоритм сортировки выбором:***

* нерекурсивный;
* может быть как устойчивым, так и неустойчивым;
* преобразует входные данные без использования вспомогательной структуры данных (in place);
* имеет сложность O(n2);

### Вставкой

Подобно алгоритму сортировки выбором, мы делим наш список на две части. Далее мы перебираем неотсортированную часть и вставляем каждый элемент из данного сегмента на его правильное место в отсортированной части списка.

***Алгоритм сортировки вставками:***

* нерекурсивный;
* устойчивый;
* преобразует входные данные без использования вспомогательной структуры данных (in place);
* имеет сложность O(n2);

### Обменом (пузырьковая)

Это самый простой алгоритм сортировки. В процессе его выполнения мы перебираем наш список и на каждой итерации сравниваем элементы попарно. При необходимости элементы меняются местами, чтобы больший элемент отправлялся в конец списка.

***Алгоритм сортировки позырьком:***

* нерекурсивный;
* устойчивый;
* преобразует входные данные без использования вспомогательной структуры данных (in place);
* имеет сложность O(n2);

### Шелла

Сортировка Шелла является оптимизированным вариантом сортировки вставками. Оптимизация достигается путем сравнения не только соседних элементов, но и элементов на определенном расстоянии, которое в течении работы алгоритма уменьшается. На последней итерации это расстояние равно 1. После этого алгоритм становится обычным алгоритмом сортировки вставками, что гарантирует правильный результат сортировки. Но следует отметить один момент: к тому времени, когда это произойдет, наш массив будет почти отсортирован, поэтому итерации будут выполнятся очень быстро.

* Алгоритм сортировки Шелла:
* нерекурсивный;
* устойчивый;
* преобразует входные данные без использования вспомогательной структуры данных (in place);
* имеет сложность O(n2), но это также зависит от выбора длины интервала;

### Турнирная (слияние)

Этот алгоритм работает по принципу «разделяй и властвуй». Здесь мы делим список ровно пополам и продолжаем это делать, пока в нем не останется только один элемент. Затем мы объединяем уже упорядоченные части нашего списка. Мы продолжаем это делать, пока не получим отсортированный список со всеми элементами несортированного входного списка.

* Алгоритм сортировки слиянием:
* рекурсивный;
* устойчивый;
* требует дополнительной памяти;
* имеет сложность O(nlog(n));

### Быстрая сортировка

В этом алгоритме мы разбиваем список при помощи опорного элемента, сортируя значения вокруг него. В нашей реализации мы выбрали опорным элементом последний элемент массива. Наилучшая производительность достигается тогда, когда опорный элемент делит список примерно пополам.

* Алгоритм быстрой сортировки:
* рекурсивный;
* неустойчивый;
* преобразует входные данные без использования вспомогательной структуры данных (in place);
* имеет сложность O(nlog(n));

### Пирамидная

Как и в двух предыдущих алгоритмах, мы создаем два сегмента списка: отсортированный и несортированный. В данном алгоритме для эффективного нахождения максимального элемента в неотсортированной части списка мы используем структуру данных «куча». Метод heapify в примере кода использует рекурсию для получения элемента с максимальным значением на вершине.

* Алгоритм пирамидальной сортировки:
* нерекурсивный;
* неустойчивый;
* преобразует входные данные без использования вспомогательной структуры данных (in place);
* имеет сложность O(nlog(n));

После реализации сортировок сравниваем затраченное время на обработку каждой сортировкой:

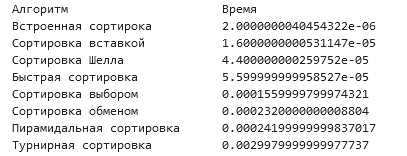


Рисунок 2 – Сравнение времени обработки алгоритмами

## Задание №4

Для загрузки в github данный файл, нужно создать репозиторий с помощью кода, который выдал сам github после создания можно просто загрузить данный файл или же с помощью git-а добавить файл в локальный репозиторий далее закоммитить и уже только запушить и файл загрузиться на удаленный репозиторий. Результат находиться на удалённом репозитории.

# Вывод

Таким образом, я научился использовать различные способы сортировки на языке программирования python. А также работать с такими сервисами как JupyterLab и GitHub.

# Ссылка на удалённый репозиторий

<https://github.com/1Double/MTUCI/tree/main/Term_4/SAOD/Lab1>

# Список использованных источников

1. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа, 2006.
2. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004.
3. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчёт о научно-исследовательской работе. Структура и правила оформления