МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра

«Математическая кибернетика и информационные технологии»

Лабораторная работа №1

**«Методы сортировки»**

по дисциплине

**«**Структуры и алгоритмы обработки данных**»**

Выполнил: студент гр. БСТ2004 Шадюк М.Р.

Вариант №22

Москва, 2022 г.

Оглавление

[**Задание №1 3**](#_Toc100081154)

[**Выполнение 3**](#_Toc100081155)

[**Задание №2 4**](#_Toc100081156)

[**Выполнение 4**](#_Toc100081157)

[**Задание №3 6**](#_Toc100081158)

[**Выполнение 6**](#_Toc100081159)

[**Выбором 6**](#_Toc100081160)

[**Вставкой 7**](#_Toc100081161)

[**Обменом 8**](#_Toc100081162)

[**Шелла 9**](#_Toc100081163)

[**Турнирная 10**](#_Toc100081164)

[**Быстрая 11**](#_Toc100081165)

[**Пирамидальная 13**](#_Toc100081166)

[**Тайминг 14**](#_Toc100081167)

# Задание №1

1. Создать Jupyter Notebook со следующим наименованием: Lab1\_Группа\_ФИО
2. Создать новую ячейку с помощью кнопки «Добавить»
3. . В созданной ячейке по указанной ниже форме заполните оглавление файла, заменив наименование группы и вписав свое ФИО, после чего создайте еще одну ячейку и напишите следующий код: print(“Hello World”)
4. С помощью кнопки “Run” запустите выполнение всех ячеек.
5. После выполнения у вас должна отформатироваться ячейка с оглавлением и должен выполниться “Hello, World!” (см. рис. 2).

## Выполнение

Изображение выглядит как текст, снимок экрана, монитор, экран

Автоматически созданное описание

Рисунок 1 – Оформление шапки лабораторной работы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Итоговый вид заголовка программы

# 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)

## Выполнение

m=50

n=50

min\_limit=-250

max\_limit=1022

m2=input("\n\nВведите m: ")

if len(m2)>0: m=int(m2)

n2=input("Введите n: ")

if len(n2)>0: n=int(n2)

min\_limit2=input("Введите минимальное значение: ")

if len(min\_limit2)>0: min\_limit=int(min\_limit2)

max\_limit2=input("Введите максимальное значение: ")

if len(max\_limit2)>0: max\_limit=int(max\_limit2)

arr=np.random.randint(min\_limit, max\_limit,size=(m, n))

print(arr)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Результат выполнения программы по созданию массива

# 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: Выбором, Вставкой, Обменом, Шелла, Турнирная, Быстрая, Пирамидальная

## Выполнение

### Выбором

# Сортировка выбором

def selection\_sort(arr):

    for row in range(len(arr)):

        for i in range(0, len(arr[row]) - 1):

            min\_idx = i

            for j in range(i+1, len(arr[row])):

                if arr[row][min\_idx] > arr[row][j]:

                    min\_idx = j

            arr[row][i], arr[row][min\_idx] = arr[row][min\_idx], arr[row][i]

selection\_sort(arr)

print(arr)

Изображение выглядит как текст

Автоматически созданное описание  
Рисунок 4 – Результат сортировки выбором

### Вставкой

# Сортировка вставкой

def insertion\_sort(arr):

    for row in range(len(arr)):

        for i in range(1, len(arr[row])):

            temp = arr[row][i]

            j = i - 1

            while (j>=0 and temp < arr[row][j]):

                arr[row][j + 1] = arr[row][j]

                j -= 1

            arr[row][j+1]=temp

insertion\_sort(arr)

print(arr)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Результат сортировки вставкой

### Обменом

def bubble\_sort(arr):

    for row in range(len(arr)):

        for i in range(len(arr[row])):

            for j in range(len(arr[row])-1):

                if arr[row][j + 1] < arr[row][j]:

                    arr[row][j],  arr[row][j + 1] =  arr[row][j + 1],  arr[row][j]

bubble\_sort(arr)

print (arr)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Результат сортировки обменом

### Шелла

def shellSort(array):

    for row in range(len(array)):

        n = len(array[row])

        k = int(math.log2(n))

        interval = 2\*\*k -1

        while interval > 0:

            for i in range(interval, n):

                temp = array[row][i]

                j = i

                while j >= interval and array[row][j - interval] > temp:

                    array[row][j] = array[row][j - interval]

                    j -= interval

                array[row][j] = temp

            k -= 1

            interval = 2\*\*k -1

    return array

shellSort(arr)

print(arr)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Результат сортировки Шелла

### Турнирная

def TournamentSort(arr):

    for i in range(len(arr)):

        tournamentSort(arr[i])

    return arr

def tournamentSort(arr):

    tree = [None] \* 2 \* (len(arr) + len(arr) % 2)

    index = len(tree) - len(arr) - len(arr) % 2

    for i, v in enumerate(arr):

        tree[index + i] = (i, v)

    for j in range(len(arr)):

        n = len(arr)

        index = len(tree) - len(arr) - len(arr) % 2

        while index > -1:

            n = (n + 1) // 2

            for i in range(n):

                i = max(index + i \* 2, 1)

                if tree[i] != None and tree[i + 1] != None:

                    if tree[i][1] < tree[i + 1][1]:

                        tree[i // 2] = tree[i]

                    else:

                        tree[i // 2] = tree[i + 1]

                else:

                    tree[i // 2] = tree[i] if tree[i] != None else tree[i + 1]

            index -= n

        index, x = tree[0]

        arr[j] = x

        tree[len(tree) - len(arr) - len(arr) % 2 + index] = None

TournamentSort(arr)

print(arr)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 – Результат турнирной сортировки

### Быстрая

def QuickSort(arr):

    for i in range(len(arr)):

        quickSort(0,len(arr[i])-1,arr,i)

    return arr

def quickSort(\_first, \_last, array, row):

    first=int(\_first)

    last=int(\_last)

    middle=int((first+last)/2)

    while (first<last):

        while( array[row][first] < array[row][middle]):

            first+=1

        while( array[row][last] > array[row][middle]):

            last-=1

        if(first<=last):

            array[row][first],array[row][last]=array[row][last],array[row][first]

            first+=1

            last-=1

    if(\_first<last):

        quickSort(\_first,last,array,row)

    if(first<\_last):

        quickSort(first,\_last,array,row)

QuickSort(arr)

print(arr)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9 – Результат быстрой сортировки

### Пирамидальная

#Пирамидальная сортировка

def heapify(arr, n, i):

    largest = i

    l = 2 \* i + 1

    r = 2 \* i + 2

    if l < n and arr[i] < arr[l]:

        largest = l

    if r < n and arr[largest] < arr[r]:

        largest = r

    if largest != i:

        arr[i],arr[largest] = arr[largest],arr[i]

        heapify(arr, n, largest)

def HeapSort(arr):

    for i in range(len(arr)):

        n = len(arr[i])

        for j in range(n, -1, -1):

            heapify(arr[i], n, j)

        for j in range(n-1, 0, -1):

            arr[i][j], arr[i][0] = arr[i][0], arr[i][j]

            heapify(arr[i], j, 0)

    return arr

HeapSort(arr)

print(arr)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 – Результат Пирамидальной сортировки

### Тайминг

print("Сортировка выбором выполнилась за ")

arr=np.random.randint(min\_limit, max\_limit,size=(m, n))

start\_time = timeit.default\_timer()

selection\_sort(arr)

time\_2 = timeit.default\_timer() - start\_time

print(time\_2)

print("Сортировка вставкой выполнилась за ")

arr=np.random.randint(min\_limit, max\_limit,size=(m, n))

start\_time = timeit.default\_timer()

insertion\_sort(arr)

time\_2 = timeit.default\_timer() - start\_time

print(time\_2)

print("Сортировка обменом выполнилась за ")

arr=np.random.randint(min\_limit, max\_limit,size=(m, n))

start\_time = timeit.default\_timer()

bubble\_sort(arr)

time\_2 = timeit.default\_timer() - start\_time

print(time\_2)

print("Сортировка Шелла выполнилась за ")

arr=np.random.randint(min\_limit, max\_limit,size=(m, n))

start\_time = timeit.default\_timer()

shellSort(arr)

time\_2 = timeit.default\_timer() - start\_time

print(time\_2)

print("Сортировка Турнирная выполнилась за ")

arr=np.random.randint(min\_limit, max\_limit,size=(m, n))

start\_time = timeit.default\_timer()

TournamentSort(arr)

time\_2 = timeit.default\_timer() - start\_time

print(time\_2)

print("Сортировка Быстрая выполнилась за ")

arr=np.random.randint(min\_limit, max\_limit,size=(m, n))

start\_time = timeit.default\_timer()

QuickSort(arr)

time\_2 = timeit.default\_timer() - start\_time

print(time\_2)

print("Сортировка Пирамидальная выполнилась за ")

arr=np.random.randint(min\_limit, max\_limit,size=(m, n))

start\_time = timeit.default\_timer()

HeapSort(arr)

time\_2 = timeit.default\_timer() - start\_time

print(time\_2)

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 – Вывод времени работы каждой из сортировок