**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2

по дисциплине **«Алгоритмы и структуры данных»**

на тему: **«Сортировка и поиск»**

Выполнил: студент гр. ИП-21

М. С. Есис

Принял: преподаватель

В. Н. Шибеко

Дата сдачи отчета: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата допуска к защите: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гомель 2024

**Цель работы**: ознакомиться с наиболее эффективным и оптимальными алгоритмами сортировки и поиска.

**Задания к лабораторной работе:**

Тема. Сортировка и поиск элементов.

1. Подготовительный этап. Создание кода приложения

1.1.Создать структуру (тип данных) – (для краткости обозначим как S) из 3-элементов - числовой тип - дата - строка

1.2.Разработать функцию генерации массив из N строк, где каждая строка имеет тип S

1.3.Реализовать функции сортировки в соответствии с заданием (по одному, двум и трем ключам и заданными направлениями)

1.4.Реализовать функцию поиска в соответствии с заданным условием

1.5.Разработать функцию съема времени измерения

2. Расчет и измерения Для каждого N из списка количества строк в массиве N= [100,1000,10000,1000000]

2.1. Выполнить сортировку по каждому ключу и снять измерения времени

2.2.Выполнить поиск по каждому ключу

2.3.Оценить эффективность каждой сортировки по времени и количеству перестановок и сравнений

2.4.Построить график зависимости по времени от значения N для каждого ключа

2.5. Дополнительно построить график зависимости количества перестановок от значения N для каждого ключа.

2.6.Дать оценку полученным результатам

3. Расчет и измерения

3.1.Провести четвертую сортировку, включив проверку (на каждом шаге), что массив уже отсортирован.

3.2.Дать оценку полученным результатам

4. Оформить отчет, включив в него

• код приложения,

• результаты измерений

• графики

• анализ и выводы

****

**Код программы:**

from datetime import datetime

import random

import time as t

import matplotlib.pyplot as plt

class S:

    def \_\_init\_\_(self, float\_data, date, string):

        self.string = string

        self.float\_data = float\_data

        self.date = date

    def \_\_str\_\_(self):

        return f"float\_data: {self.float\_data}, date: {self.date}, string: {self.string}"

def Generate(N):

    array = []

    for \_ in range(N):

        num = random.randint(0, 100)

        year = random.randint(1970, datetime.now().year)

        month = random.randint(1, 12)

        day = random.randint(1, 28)

        hour = random.randint(0, 23)

        minute = random.randint(0, 59)

        second = random.randint(0, 59)

        random\_date = datetime(year, month, day, hour, minute, second)

        string = "".join(

            random.choices("abcdefghijklmnopqrstuvwxyz", k=10)

        )  # создание строки из 10 символос допуском пробелов

        s = S(num, random\_date, string)

        array.append(s)

    return array

def Sort1(array, key):

    n = len(array)

    swaps = 0

    for i in range(1, n):

        j = i

        while j > 0 and getattr(array[j - 1], key) > getattr(array[j], key):

            array[j], array[j - 1] = array[j - 1], array[j]

            swaps += 1

            j -= 1

    return swaps

def Sort2(array, key1, key2):

    n = len(array)

    swaps = 0

    for i in range(1, n):

        j = i

        while j > 0 and (

            getattr(array[j - 1], key1) > getattr(array[j], key1) or (

                getattr(array[j - 1], key1) == getattr(array[j], key1) and

                getattr(array[j - 1], key2) > getattr(array[j], key2)

            )

        ):

            array[j], array[j - 1] = array[j - 1], array[j]

            swaps += 1

            j -= 1

    return swaps

def Sort3(array, key1, key2, key3):

    n = len(array)

    swaps = 0

    for i in range(1, n):

        j = i

        while j > 0 and (

            getattr(array[j - 1], key1) < getattr(array[j], key1) or (

                getattr(array[j - 1], key1) == getattr(array[j], key1) and (

                    getattr(array[j - 1], key2) < getattr(array[j], key2) or (

                        getattr(array[j - 1], key2) == getattr(array[j], key2) and

                        getattr(array[j - 1], key3) > getattr(array[j], key3)

                    )

                )

            )

        ):

            array[j], array[j - 1] = array[j - 1], array[j]

            swaps += 1

            j -= 1

    return swaps

def InterpolationSearch(array, key, value):

    low = 0

    high = len(array) - 1

    while low <= high and array[low].date <= value <= array[high].date:

        pos = low + int(((high - low) / (array[high].date - array[low].date).total\_seconds()) \* (value - array[low].date).total\_seconds())

        if array[pos].date == value:

            return array[pos]

        if array[pos].date < value:

            low = pos + 1

        else:

            high = pos - 1

    return None

N = [100, 1000, 10000]

times1, times2, times3, swaps1, swaps2, swaps3 = [], [], [], [], [], []

for n in N:

    starttime = t.perf\_counter\_ns()

    data, count = Generate(n), 0

    copyData1 = data.copy()

    copyData2 = data.copy()

    copyData3 = data.copy()

    swaps1.append(Sort1(copyData1, 'string'))

    times1.append((t.perf\_counter\_ns() - starttime) \* 10 \*\* (-9))

    swaps2.append(Sort2(copyData2, 'string', 'float\_data'))

    times2.append((t.perf\_counter\_ns() - starttime) \* 10 \*\* (-9))

    swaps3.append(Sort3(copyData3, 'string', 'float\_data', 'date'))

    times3.append((t.perf\_counter\_ns() - starttime) \* 10 \*\* (-9))

    print("Объем: %d" %n)

    print("--------------------------")

    print("Время на сортировку 1: %.10f c\nВремя на сортировку 2: %.10f c\nВремя на сортировку 3: %.10f c\n\n" % (times1[count], times2[count], times3[count]))

    count += 1

search\_date = datetime(2023, 5, 10, 12, 30, 0)

result = InterpolationSearch(data, 'date', search\_date)

if result is not None:

    print(f"Found: {result}")

else:

    print("Not found")

plt.subplot(1, 2, 1)

plt.plot(N, times1, marker='\*', label='Сортировка 1')

plt.plot(N, times2, marker='\*', label='Сортировка 2')

plt.plot(N, times3, marker='\*', label='Сортировка 3')

plt.xlabel('Объем данных')

plt.ylabel('Время (секунды)')

plt.title('Зависимость времени выполнения от объема данных')

plt.legend()

plt.subplot(1, 2, 2)

plt.plot(N, swaps1, marker='\*', label='Сортировка 1')

plt.plot(N, swaps2, marker='\*', label='Сортировка 2')

plt.plot(N, swaps3, marker='\*', label='Сортировка 3')

plt.xlabel('Объем данных')

plt.ylabel('Количество обменов')

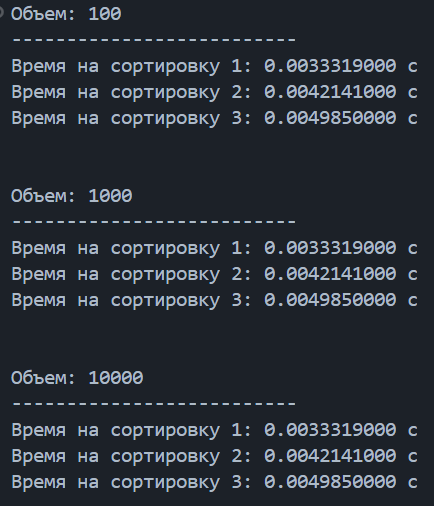
plt.title('Зависимость количества обменов от объема данных')

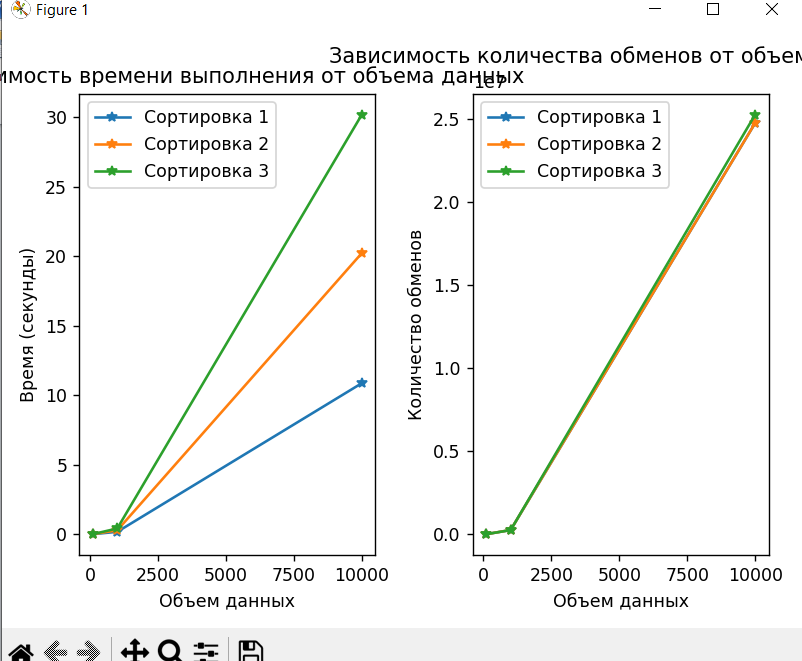
plt.legend()

plt.tight\_layout()

plt.show()

**Результат:**

****

****

**Вывод**: ознакомилась с наиболее эффективным и оптимальными алгоритмами сортировки и поиска.