Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Алгоритми та методи обчислення

Лабораторна робота №2

**«Обчислювальна складність алгоритмів сортування»**

Виконав:

Студент групи ІВ-82

Троценко Д.А.

Номер залікової книжки 8227

Номер у списку 25

Перевірив:

ст.вик. Порєв В. М.

Київ

2020 р.

**Мета:** Закріплення навичок практичної оцінки алгоритмічної складності логічних алгоритмів на прикладі алгоритмів сортування.

**Завдання:** Використовуючи відповідний до варіанту алгоритм сортування

написати програму сортування масиву даних. Застосовуючи дану програму,

дослідити часову складність алгоритму сортування та порівняти її з

теоретичною алгоритмічною складністю.

**Алгоритм за варіантом:** сортування Боуза-Нельсона

**Роздруківка тексту програми:**

**Algo.py**

def bose\_nelson(data):

    m = 1

    while m < len(data):

        j = 0

        while j + m < len(data):

            bose\_nelson\_merge(j, m, m, data)

            j = j + m + m

        m = m + m

    return data

def bose\_nelson\_merge(j, r, m, data):

    if j + r < len(data):

        if m == 1:

            if data[j] > data[j + r]:

                data[j], data[j + r] = data[j + r], data[j]

        else:

            m = m // 2

            bose\_nelson\_merge(j, r, m, data)

            if j + r + m < len(data):

                bose\_nelson\_merge(j + m, r, m,data)

            bose\_nelson\_merge(j + m, r - m, m,data)

    return data

**main.py**

import sys

from PyQt5.QtWidgets import (QInputDialog, QApplication, QLineEdit, QGridLayout, QLabel, QWidget, QToolTip, QTextEdit, QPushButton, QApplication, QMessageBox, QDesktopWidget, QMainWindow)

import algo

import numpy as np

import time

import matplotlib.pyplot as plt

import math

test\_arrays = [10, 100, 500, 1000, 2000, 5000, 10000, 50000, 100000, 500000]

test\_time = [0.0, 0.005987, 0.0643, 0.108,0.203, 0.7748, 2.137, 26.36, 73.45, 73.45\*5.1]

class MainWindow(QWidget):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

        self.initUI()

    def initUI(self):

        self.setGeometry(300, 300, 250, 150)

        lbl1 = QLabel('Лабораторна робота №2 - Сортування Боуза-Нельсона - 25 варіант')

        lbl2 = QLabel('Студент групи ІВ-82')

        lbl3 = QLabel('Троценко\nДанііл\nАнатолійович')

        lbl4 = QLabel('Розміри масивів - 10 100 500 1000 2000 5000 10000 50000 100000 500000')

        lbl5 = QLabel('ВСього - 10 рандомних тестових масивів')

        ell1 = QLabel('Введіть розмір бажаного масиву:')

        self.el1 = QLineEdit()

        btn1 = QPushButton('Ввести та показати час')

        btn1.clicked.connect(self.show\_own)

        ell2 = QLabel('Тестові рандомні масиви:')

        btn2 = QPushButton('Показати графіки теоретичних масивів')

        btn2.clicked.connect(self.show\_theorical)

        self.label\_time = QLabel('Час:')

        self.label\_time2 = QLabel()

        grid = QGridLayout()

        grid.setSpacing(5)

        grid.addWidget(lbl1,0,0)

        grid.addWidget(lbl2,0,1)

        grid.addWidget(lbl3,0,2)

        grid.addWidget(lbl4,1,0)

        grid.addWidget(lbl5,1,2)

        grid.addWidget(ell1, 2,0)

        grid.addWidget(self.el1, 2,1)

        grid.addWidget(btn1, 2,2)

        grid.addWidget(ell2, 3,0)

        grid.addWidget(btn2, 3,2)

        grid.addWidget(self.label\_time, 4,0)

        grid.addWidget(self.label\_time2, 4,1)

        self.setLayout(grid)

        self.setGeometry(300, 300, 500, 500)

        self.setWindowTitle('Review')

        self.show()

    def show\_own(self):

        if not self.el1.text().isdigit():

            return

        tests = np.random.sample(int(self.el1.text())) \* 200 - 100

        start\_time = time.time()

        algo.bose\_nelson(tests)

        end\_time = time.time() - start\_time

        print(float(end\_time))

        self.label\_time2.setText(str(end\_time))

    def show\_theorical(self):

        fig, ax = plt.subplots()

        ax.plot(test\_arrays, test\_time)

        ax.grid()

        ax.set\_xlabel(' Розмір масиву',

              fontsize = 15,    *#  размер шрифта*

              color = 'red',    *#  цвет шрифта*

*#  параметры области с текстом*

              bbox = {'boxstyle': 'rarrow',    *#  вид области*

                      'pad': 0.1,     *#  отступы вокруг текста*

                      'facecolor': 'white',    *#  цвет области*

                      'edgecolor': 'red',    *#  цвет крайней линии*

                      'linewidth': 3})

        ax.set\_ylabel('Швидкість сортування',

              fontsize = 15,

              color = 'red',

              bbox = {'boxstyle': 'rarrow',

                      'pad': 0.1,

                      'facecolor': 'white',

                      'edgecolor': 'red',

                      'linewidth': 3})

        fig.set\_figwidth(9)

        fig.set\_figheight(9)

        plt.title(label='Сортування. Фактичний час')

        plt.show()

        tests = np.linspace(0, 500000)

        fig, ax = plt.subplots()

        ax.plot(tests, tests \* np.log2(tests)  )

        ax.grid()

        ax.set\_xlabel(' Розмір масиву',

              fontsize = 15,    *#  размер шрифта*

              color = 'red',    *#  цвет шрифта*

*#  параметры области с текстом*

              bbox = {'boxstyle': 'rarrow',    *#  вид области*

                      'pad': 0.1,     *#  отступы вокруг текста*

                      'facecolor': 'white',    *#  цвет области*

                      'edgecolor': 'red',    *#  цвет крайней линии*

                      'linewidth': 3})

        ax.set\_ylabel('Швидкість сортування',

              fontsize = 15,

              color = 'red',

              bbox = {'boxstyle': 'rarrow',

                      'pad': 0.1,

                      'facecolor': 'white',

                      'edgecolor': 'red',

                      'linewidth': 3})

        fig.set\_figwidth(9)

        fig.set\_figheight(9)

        plt.title(label='Сортування. Теоретичний час')

        plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    app = QApplication(sys.argv)

    ex = MainWindow()

    sys.exit(app.exec\_())

**Висновки:** У ході виконання лабораторної роботи я закріпив знання з базових понять алгоритмів, навички практичної оцінки алгоритмічної складності логічних алгоритмів на прикладі алгоритмів сортування. Для того, щоб не чекати на виконання та замір часу для тестових масивів, я виміряв дані на своїй машині та заніс до масиву, тому що на мові Python обчислення сортування випадкових даних для 500000 займає ~6-7хв. Але можна ввести розмір для будь-якого масиву, та зачекати. Графік виводиться.