Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп’ютерно-інформаційних систем та програмної інженерії

Кафедра програмної інженерії

ЗВІТ

До модульного контролю №6

з навчальної дисципліни

Алгоритми і структури даних

Підготував:

студент групи СП-11

Великанич Максим

Варіант №14

Тернопіль 2023

IDE: Visual Studio Code

Мова програмування: Python

***Мета роботи:*** набути практичних навичок по реалізації різноманітних алгоритмів сортування та порівняти швидкості виконання простих і швидких алгоритмів

Завдання №1:

Посортувати масив із 10ти елементів типу int, використовуючи:  
 – сортування вставками

Лістинг коду:

size = int(input("Enter the array size "))

array = []

for i in range(size):

value = int(input(f"Enter the array element {i + 1}: "))

array.append(value)

print("Array:", array)

def insertionSort(array):

for i in range(len(array)):

cursor = array[i]

pos = i

while pos > 0 and array[pos - 1] > cursor:

array[pos] = array[pos - 1]

pos = pos - 1

array[pos] = cursor

return array

sortedArray = insertionSort(array)

print("Sorted array:", sortedArray)

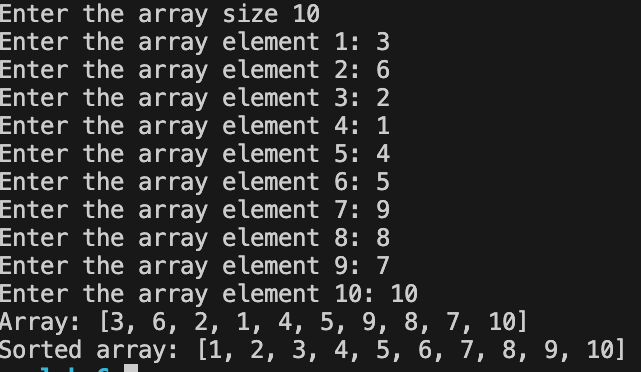


Рисунок 1 - Результат виконання завдання №1

Завдання №2:

Реалізувати алгоритм швидкого сортування для впорядкування в неспадній послідовності масив структур:

Лістинг коду:

key = [

{"key" :7, "info" : "info 7"},

{"key" :3, "info" : "info 3"},

{"key" :1, "info" : "info 1"},

{"key" :2, "info" : "info 2"},

{"key" :5, "info" : "info 5"},

{"key" :4, "info" : "info 4"},

{"key" :6, "info" : "info 6"},

{"key" :8, "info" : "info 8"},

{"key" :10, "info" : "info 10"},

{"key" :9, "info" : "info 9"}

]

def quicSort(key):

if len(key) <= 1:

return key

else:

pivot = key[0]

less = []

great = []

for i in key[1:]:

if i["key"] < pivot["key"]:

less.append(i)

else:

great.append(i)

return quicSort(less) + [pivot] + quicSort(great)

print(quicSort(key))

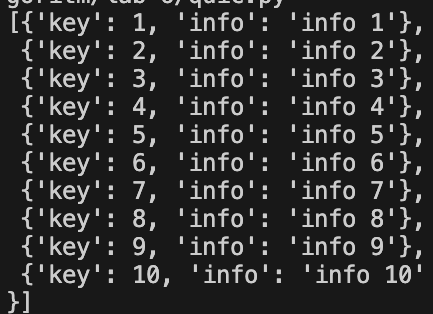


Рисунок 2 - Результат виконання завдання №2

Завдання №3:

Провести порівняльний аналіз часу сортування масиву з 50000 елементів типу int для алгоритму швидкого сортування, та для алгоритму  
2 – сортування методом вибору

Провести п’ять контрольних замірів для різних вхідних даних і в якості результуючого значення взяти їх середнє арифметичне.

Лістинг коду:

import random

import time

NUM = 50000

def createArray():

random.seed(1)

A = [random.randint(1, NUM) for \_ in range(NUM)]

B = A.copy()

return A, B

def selectionSort(arr):

n = len(arr)

for i in range(n - 1):

minIndex = i

for j in range(i + 1, n):

if arr[j] < arr[minIndex]:

minIndex = j

arr[i], arr[minIndex] = arr[minIndex], arr[i]

def quickSort(arr, low, high):

if low < high:

pivot = arr[high]

i = low - 1

for j in range(low, high):

if arr[j] <= pivot:

i += 1

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

arr[i + 1], arr[high] = arr[high], arr[i + 1]

pi = i + 1

quickSort(arr, low, pi - 1)

quickSort(arr, pi + 1, high)

totalSelectionSortTime = 0.0

totalQuickSortTime = 0.0

for \_ in range(5):

A, B = createArray()

start\_time = time.time()

selectionSort(A)

selection\_sort\_time = time.time() - start\_time

totalSelectionSortTime += selection\_sort\_time

print("Selection sort time: {:.6f} sec".format(selection\_sort\_time))

A = B.copy()

start\_time = time.time()

quickSort(A, 0, NUM - 1)

quick\_sort\_time = time.time() - start\_time

totalQuickSortTime += quick\_sort\_time

print("Quicksort time: {:.6f} sec".format(quick\_sort\_time))

averageSelectionSortTime = totalSelectionSortTime / 5

averageQuickSortTime = totalQuickSortTime / 5

print("Average selection sort time: {:.6f} sec".format(averageSelectionSortTime))

print("Average quicksort time: {:.6f} sec".format(averageQuickSortTime))

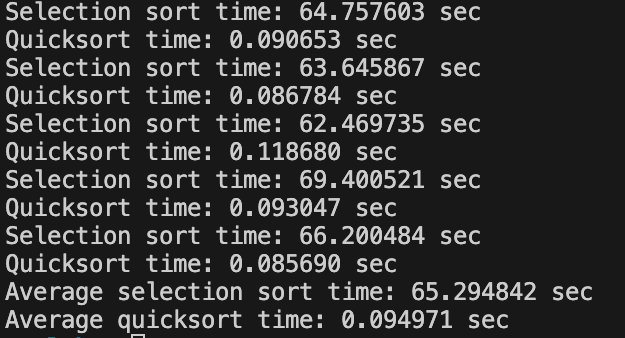


Рисунок 3 - Результат виконання завдання №3

Висновки:набув практичних навичок по реалізації різноманітних алгоритмів сортування та порівняти швидкості виконання простих і швидких алгоритмів