Державний торговельно-економічний університет  
Кафедра інженерії програмного забезпечення та кібербезпеки

**ЛАБОРАТОРНА РОБОТА №6**

**З ДИСЦИПЛІНИ «АЛГОРИТМИ ТА СТРУКТУРИ ДАНИХ»**

**НА ТЕМУ «АНАЛІЗ АЛГОРИТМІВ СОРТУВАННЯ»**

**Виконала**: студентка факультету

інформаційних технологій

групи\_курсу 3-4

Авєріна Наталія Ігорівна

**Перевірила**: Палагута К. О.

Київ 2024

**Лабораторна робота №6**

**Тема**: Аналіз алгоритмів сортування.

**Мета**: познайомитись з алгоритмами сортування, навчитись створювати програми із застосуванням алгоритмів сортування на мові програмування Python, аналізувати ефективність алгоритмів.

**Програмне забезпечення**: PyCharm, MS Word.

**1 варіант**

**Хід роботи:**

1. Створюємо функцію, що буде генерувати випадковий список N елементів у діапазоні від 1 до 50 і повертає його.

def generateRandomIntegerList(n):  
 list = []  
 for i in range(n):  
 list.append(random.randint(1, 50))  
 return list

1. Реалізуємо функцію сортування вибором

def selectionSort(list):  
 for i in range(len(list)):  
 min\_ind = i  
  
 for j in range(i + 1, len(list)):  
 if list[j] < list[min\_ind]:  
 min\_ind = j  
 if min\_ind != i:  
 value = list[i]  
 for l in range(min\_ind, i - 1, -1):  
 list[l] = list[l - 1]  
 list[i] = value

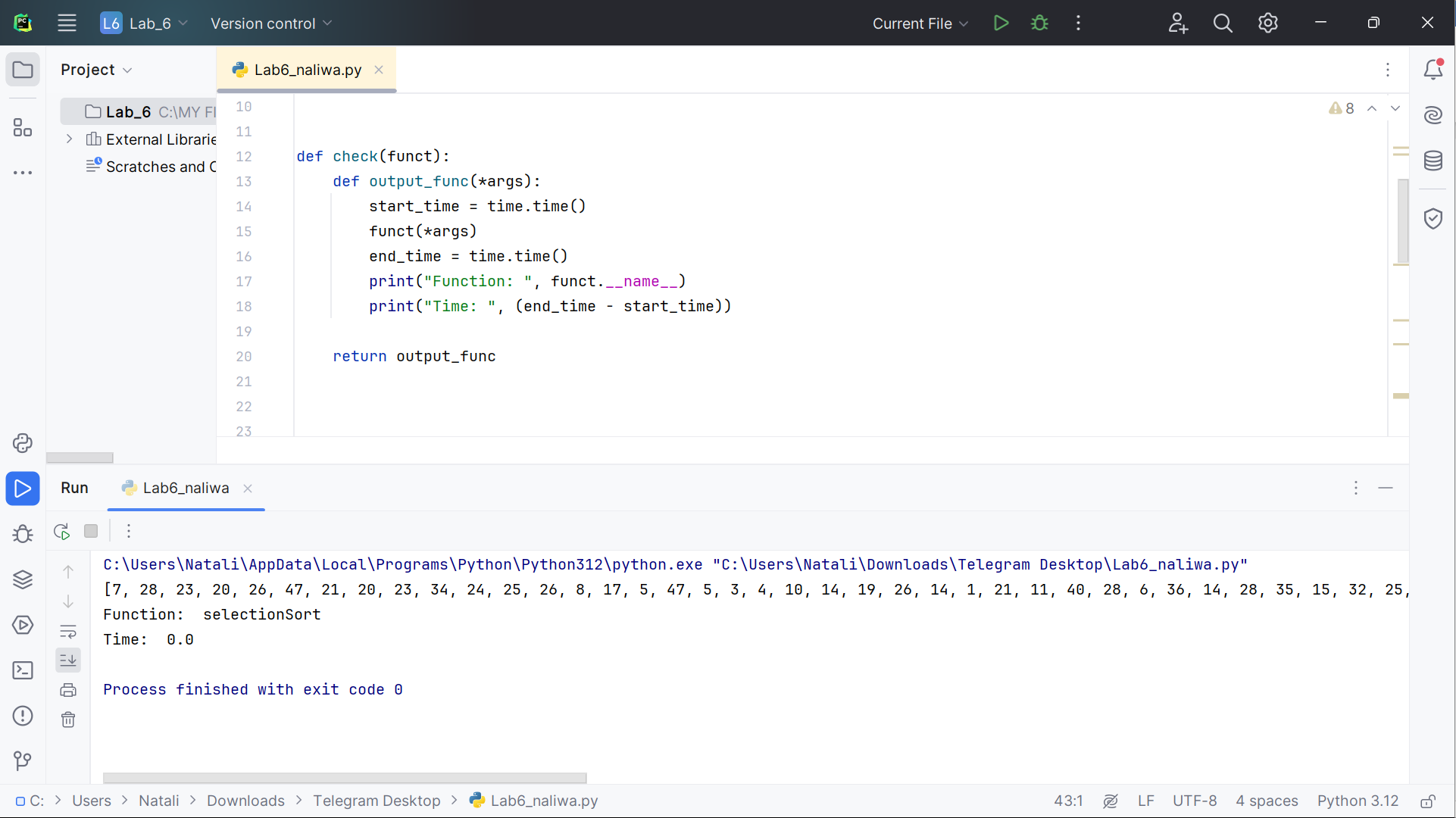
1. Створюємо перевірку для визначення часу виконання сортування.

def check(funct):  
 def output\_func(\*args):  
 start\_time = time.time()  
 funct(\*args)  
 end\_time = time.time()  
 print("Function: ", funct.\_\_name\_\_)  
 print("Time: ", (end\_time - start\_time))  
  
 return output\_func

1. Використовуємо функції

decorated\_selectionSort = check(selectionSort)  
a = generateRandomIntegerList(100)  
print(a)  
decorated\_selectionSort(a)

1. Результат виконання пункту 1.



1. Створюємо метод для розрахунку часу сортування для списку N елементів.

def measure\_sorting\_time(n\_values):  
 results = []  
 for n in n\_values:  
 a = generateRandomIntegerList(n)  
 start\_time = time.time()  
 selectionSort(a)  
 end\_time = time.time()  
 execution\_time = end\_time - start\_time  
 results.append((n, execution\_time))  
 return results

1. Визначаємо діапазон значень від 100 до 2000 та крок 100. Виводимо результати у вигляді таблиці.

n\_values = list(range(100, 2001, 100))  
print("\nResults:")  
print("Number of Elements | Time (seconds)")  
print("----------------------------------")  
for result in selection\_sort\_results:  
 print("{:<18} | {:<13}".format(result[0], result[1]))

1. Створюємо функцію сортування злиттям mergeSort.

def mergeSort(lst):  
 if len(lst) <= 1:  
 return lst  
mid = len(lst) // 2left\_half = mergeSort(lst[:mid])right\_half = mergeSort(lst[mid:])return merge(left\_half, right\_half)

1. Зливаємо два відсортованих списки

def merge(left, right):  
 merged = []  
 left\_index = 0  
 right\_index = 0  
while left\_index < len(left) and right\_index < len(right):  
 if left[left\_index] <= right[right\_index]:  
 merged.append(left[left\_index])  
 left\_index += 1  
 else:  
 merged.append(right[right\_index])  
 right\_index += 1  
while left\_index < len(left):  
 merged.append(left[left\_index])  
 left\_index += 1  
  
 while right\_index < len(right):  
 merged.append(right[right\_index])  
 right\_index += 1  
  
 return merged

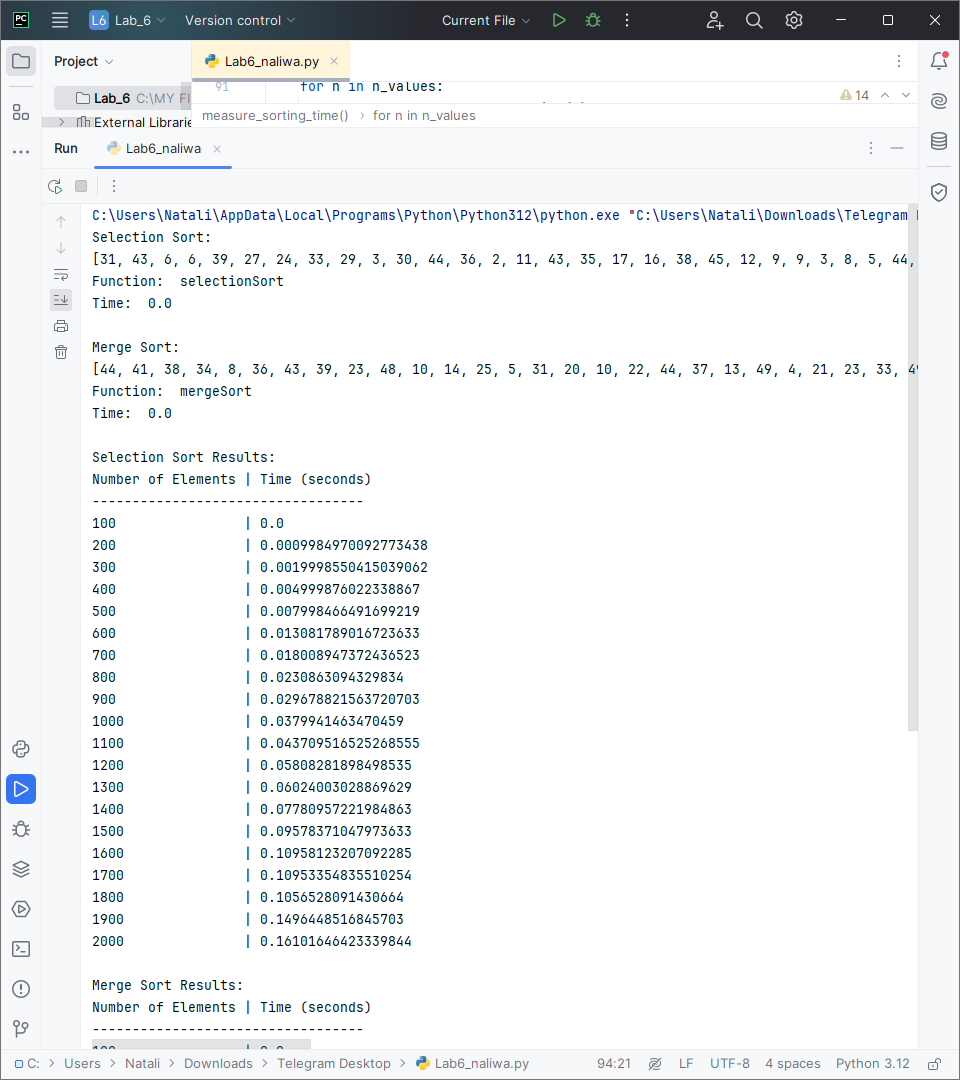
1. В функцію для вимірювання часу measure\_sorting\_time додаємо параметр sort\_func.

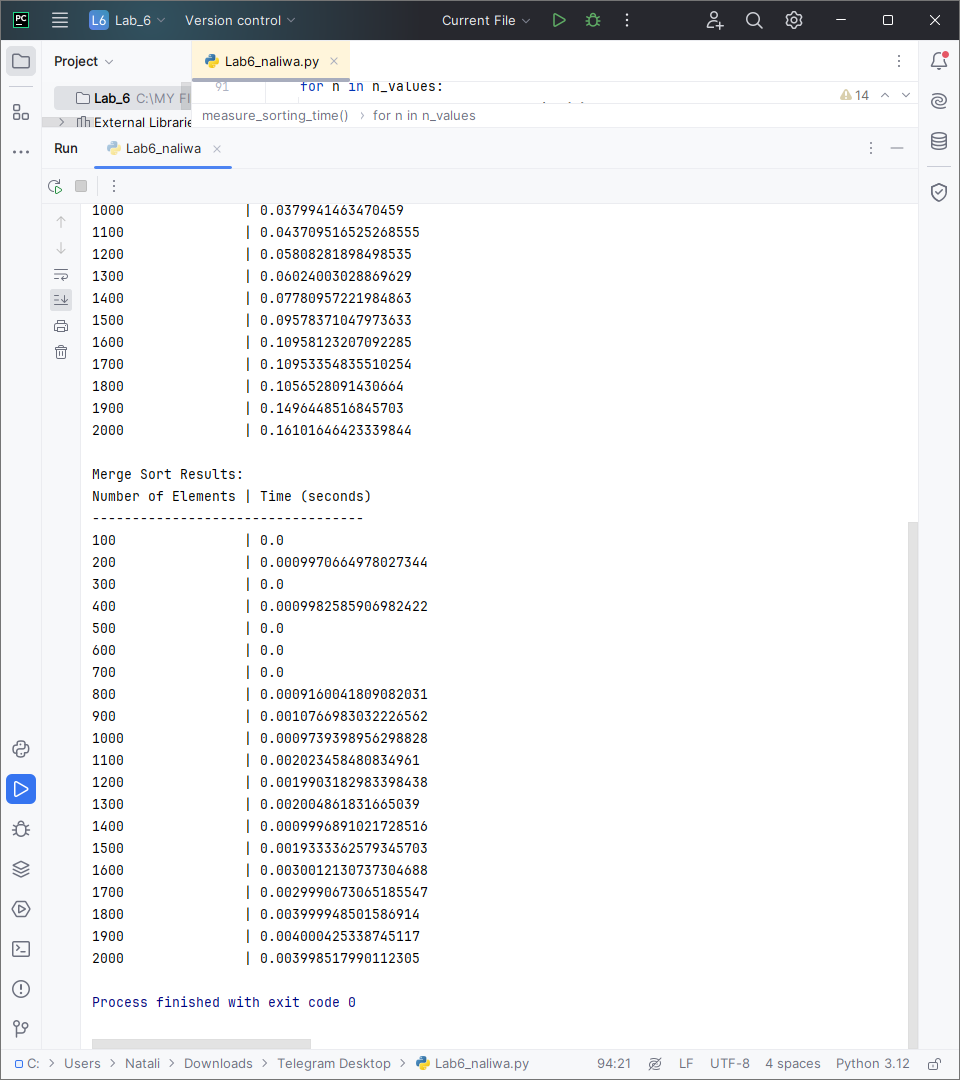
def measure\_sorting\_time(n\_values, sort\_func):  
 results = []  
 for n in n\_values:  
 a = generateRandomIntegerList(n)  
 start\_time = time.time()  
 sort\_func(a)  
 end\_time = time.time()  
 execution\_time = end\_time - start\_time  
 results.append((n, execution\_time))  
 return results

1. Викликаємо всі функції та виводимо результати

n\_values = list(range(100, 2001, 100))  
print("\nSelection Sort Results:")  
print("Number of Elements | Time (seconds)")  
print("----------------------------------")  
selection\_sort\_results = measure\_sorting\_time(n\_values, selectionSort)  
for result in selection\_sort\_results:  
 print("{:<18} | {:<13}".format(result[0], result[1]))  
  
print("\nMerge Sort Results:")  
print("Number of Elements | Time (seconds)")  
print("----------------------------------")  
merge\_sort\_results = measure\_sorting\_time(n\_values, mergeSort)  
for result in merge\_sort\_results:  
 print("{:<18} | {:<13}".format(result[0], result[1]))

Результат виконання:





**Висновок**: під час виконання даної лабораторної роботи було набуто практичних навичок роботи з сортуванням даних. Була створена програма із застосуванням алгоритмів сортуванням вибору та злиттям. Судячи з результатів, виведених програмою, можна визначити, що сортування злиттям займає менше часу.