**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра ІПІ**

**Звіт**

з лабораторної роботи № 1 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

„ **Проектування і аналіз алгоритмів внутрішнього сортування**”

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-12 Титаренко Данило Олегович Олегович*

**Перевірив**

(прізвище, ім'я, по батькові)

Київ 2022

**Лабораторна робота № 1**

**Проектування і аналіз алгоритмів внутрішнього сортування**

**Мета роботи** – вивчити основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування і оцінити поріг їх ефективності.

**Виконання**

1. *Сортування бульбашкою*

Аналіз алгоритму на відповідність властивостям

|  |  |
| --- | --- |
| **Властивість** | **Сортування бульбашкою** |
| Стійкість | + |
| «Природність» поведінки (Adaptability) | + |
| Базуються на порівняннях | + |
| Необхідність в додатковій пам'яті (об'єм) | О(1) |
| Необхідність в знаннях про структури даних | Масив |

1. *Псевдокод алгоритму*

**Повторити**

**Для i від 1 до n-1**

**Повторити**

**Для j від 1 до n-i-1**

**Якщо(a[j] > a[j + 1])**

k=a[j]

a[j]=a[j+1]

a[j+1]=k

**все якщо**

**все повторити**

**все повторити**

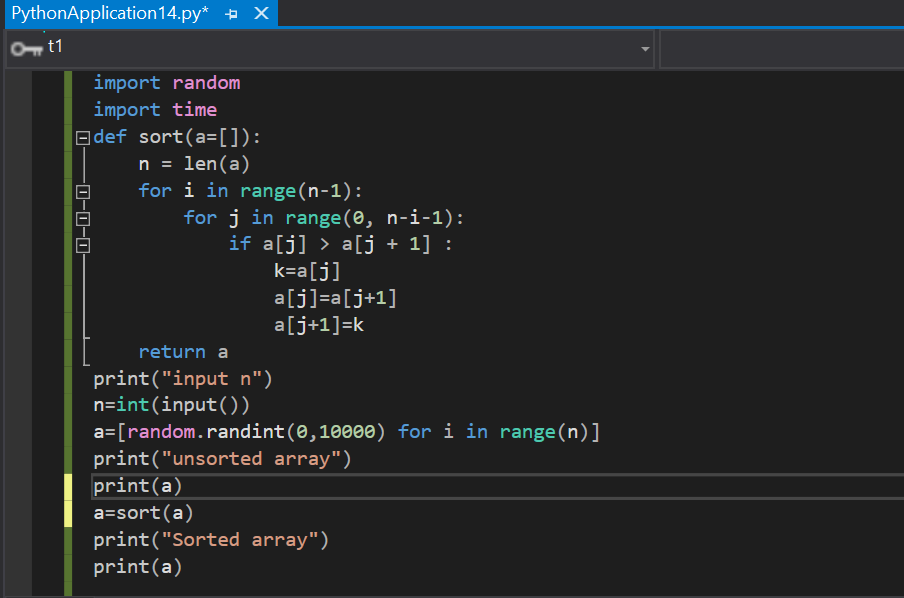
1. *Аналіз часової складності алгоритму*

**О(n2) – у найгіршому випадку**

**Θ(n2) –** у середньому випадку

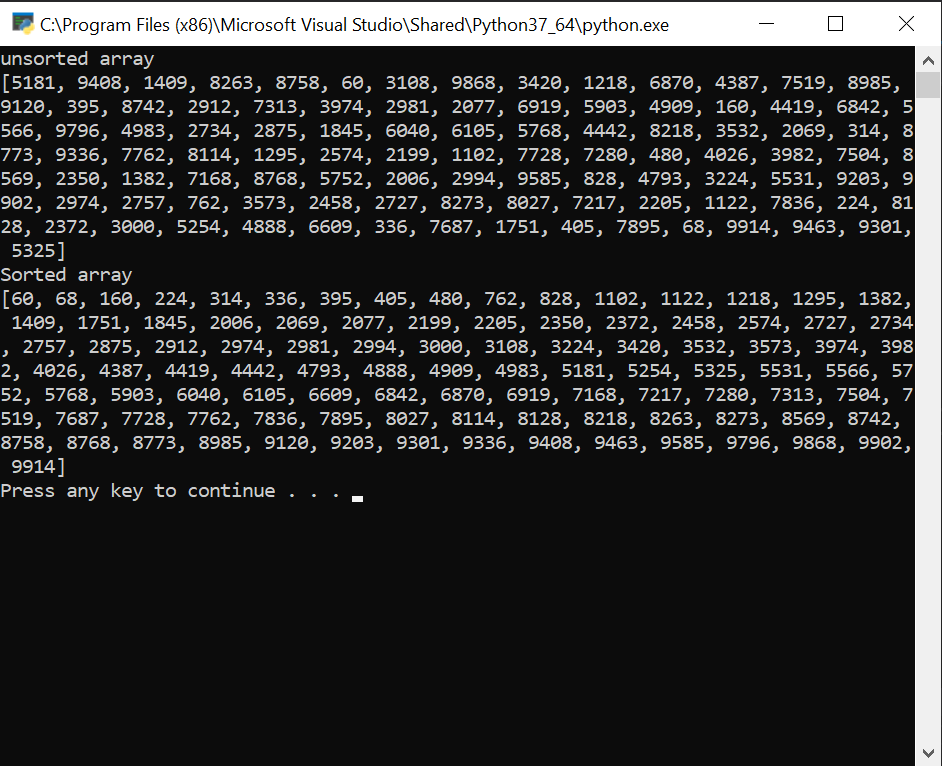
**Ω(n) –** у найкращому випадку

1. *Вихідний код*

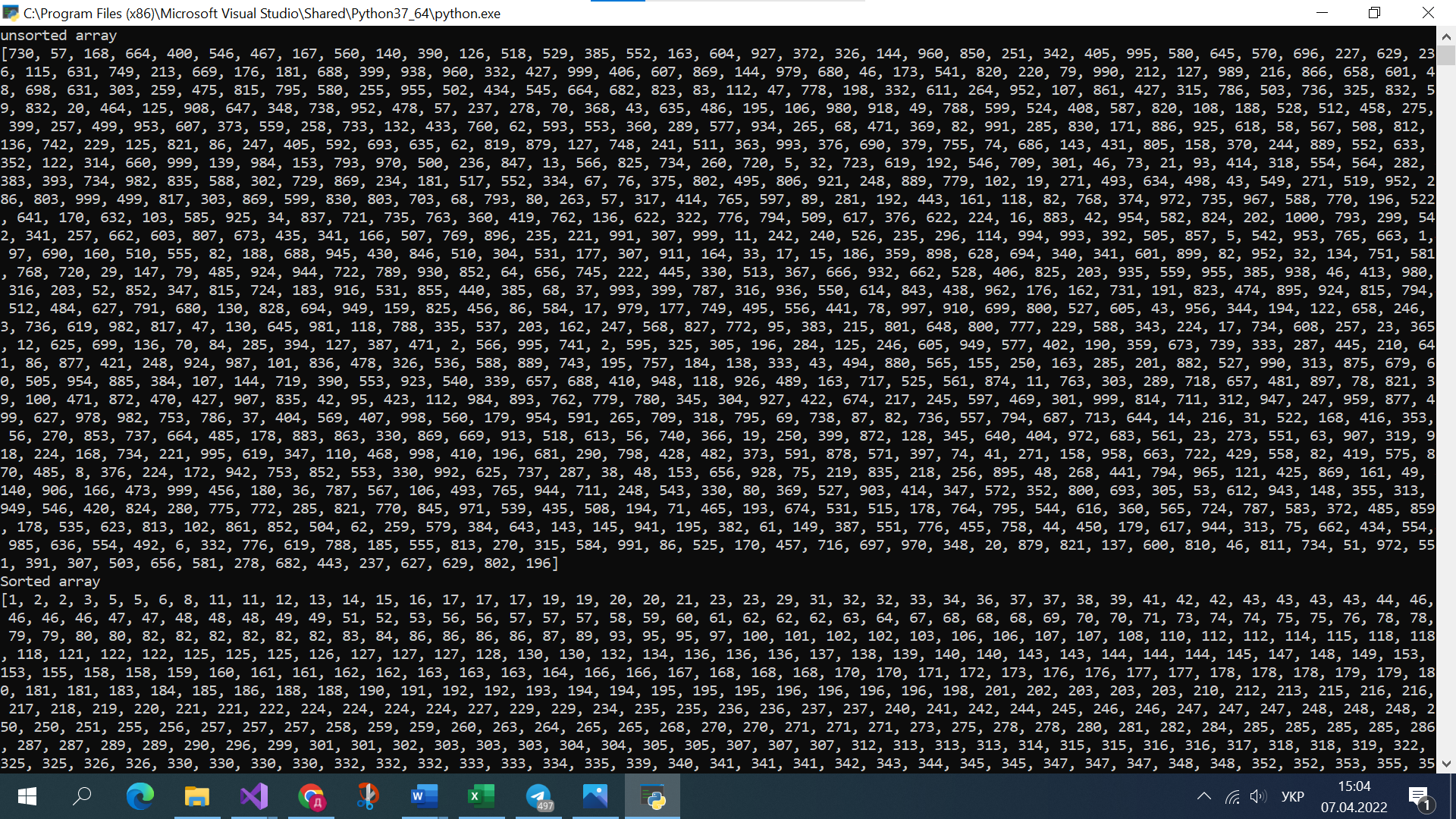
****

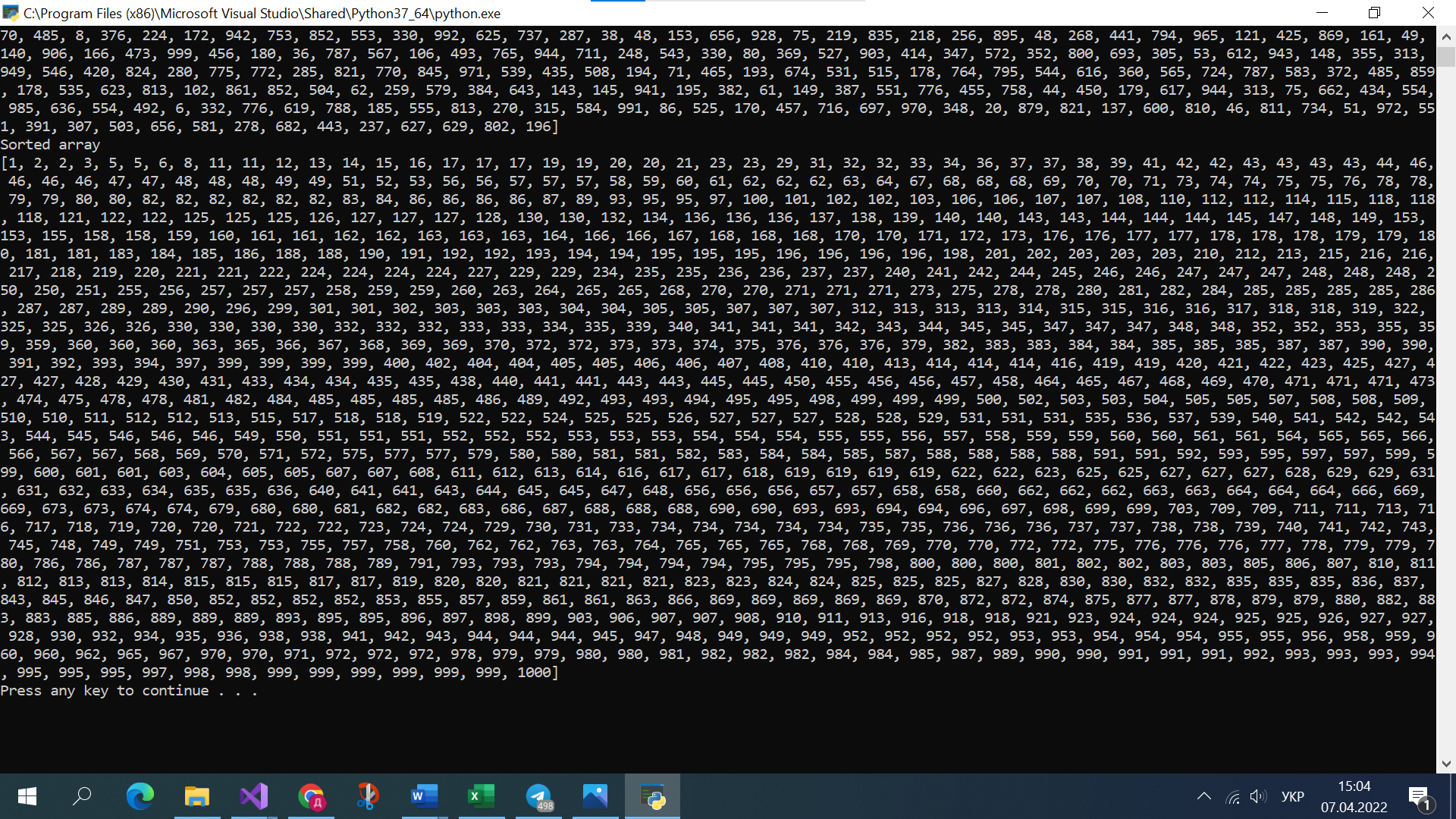
1. *Приклад роботи на 100 та 1000 елементів*

Сортування на 100 елементів



Сортування на 1000 елементів





1. *Тестування алгоритму*

Характеристики оцінювання алгоритму сортування бульбашки для упорядкованої послідовності елементів у масиві

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45 | 0 |
| 100 | 4950 | 0 |
| 1000 | 499500 | 0 |
| 5000 | 12497500 | 0 |
| 10000 | 49995000 | 0 |
| 20000 | 199990000 | 0 |
| 50000 | 1249975000 | 0 |

Характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45 | 45 |
| 100 | 4950 | 4950 |
| 1000 | 499500 | 499500 |
| 5000 | 12497500 | 12497500 |
| 10000 | 49995000 | 49995000 |
| 20000 | 199990000 | 199990000 |
| 50000 | 1249975000 | 124975000 |

Характеристика оцінювання алгоритму сортування бульбашки для випадкової послідовності елементів у масиві.

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45 | 33 |
| 100 | 4950 | 2400 |
| 1000 | 499500 | 245793 |
| 5000 | 12497500 | 6352994 |
| 10000 | 49995000 | 25138696 |
| 20000 | 199990000 | 100862595 |
| 50000 | 1249975000 | 693529940 |

### *Графіки залежності часових характеристик оцінювання від розмірності масив*



*Сортування гребінцем*

1. Аналіз алгоритму на відповідність властивостям

|  |  |
| --- | --- |
| **Властивість** | **Сортування гребінцем** |
| Стійкість | - |
| «Природність» поведінки (Adaptability) | + |
| Базуються на порівняннях | + |
| Необхідність в додатковій пам'яті (об'єм) | О(1) |
| Необхідність в знаннях про структури даних | Масив |

1. *Псевдокод алгоритму*

gap = n

swapped = True

**Повторити**

**Поки (**while gap !=1 or swapped == 1):

swapped = False

gap = max(1, int(gap **/** 1.25)

**Повторити**

**Для і від 0 до n-gap**

**Якщо** (arr[i] > arr[i + gap]):

**То**

arr[i], arr[i + gap]=arr[i + gap], arr[i]

swapped = True

**Все якщо**

**Все повторити**

**Все повторити**

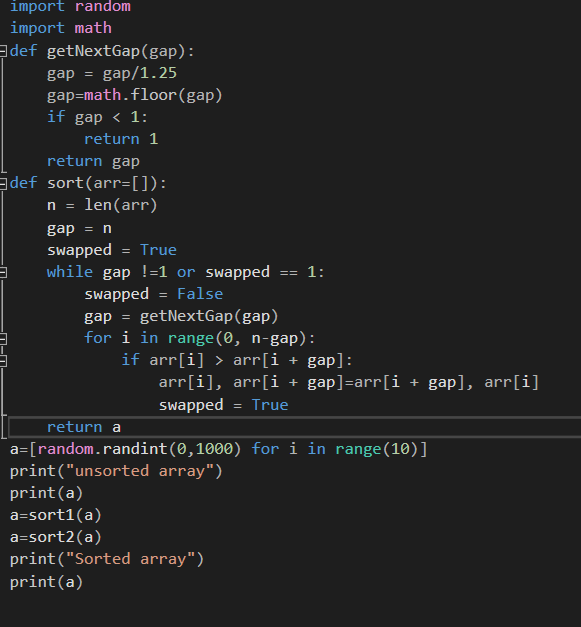
1. *Аналіз часової складності алгоритму*

**О(n2) – у найгіршому випадку**

**Θ(n\*log(n)) –** у середньому випадку

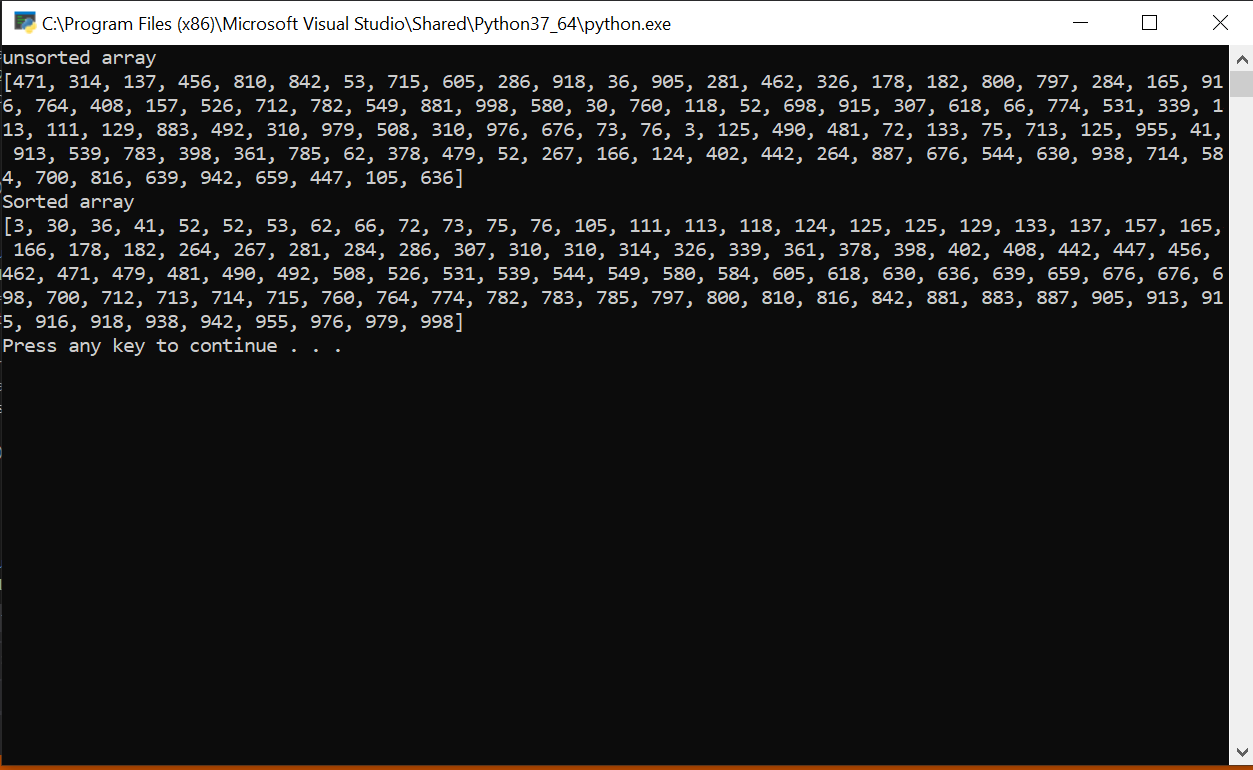
**Ω(n) –** у найкращому випадку

1. *Вихідний код*

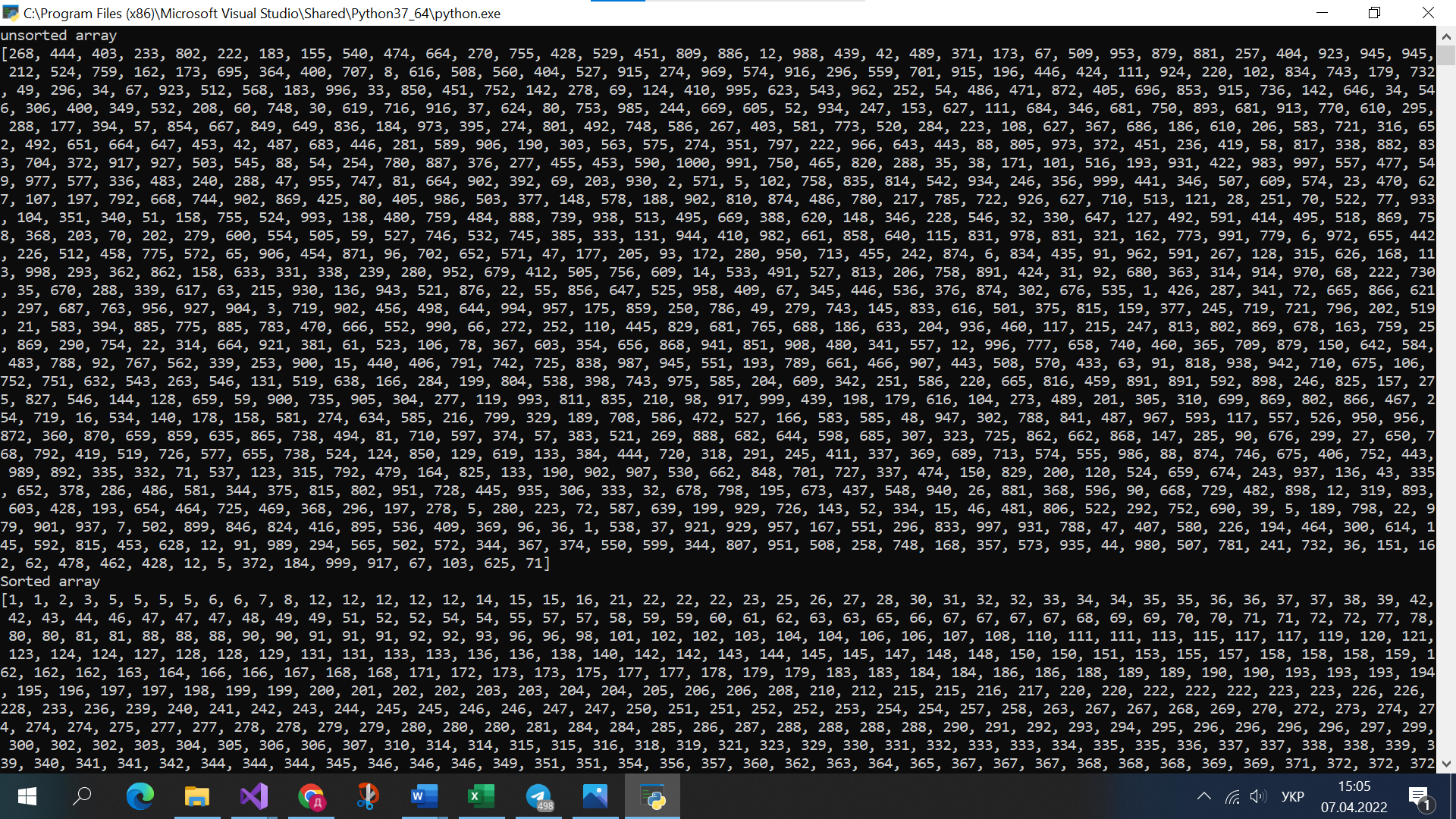
****

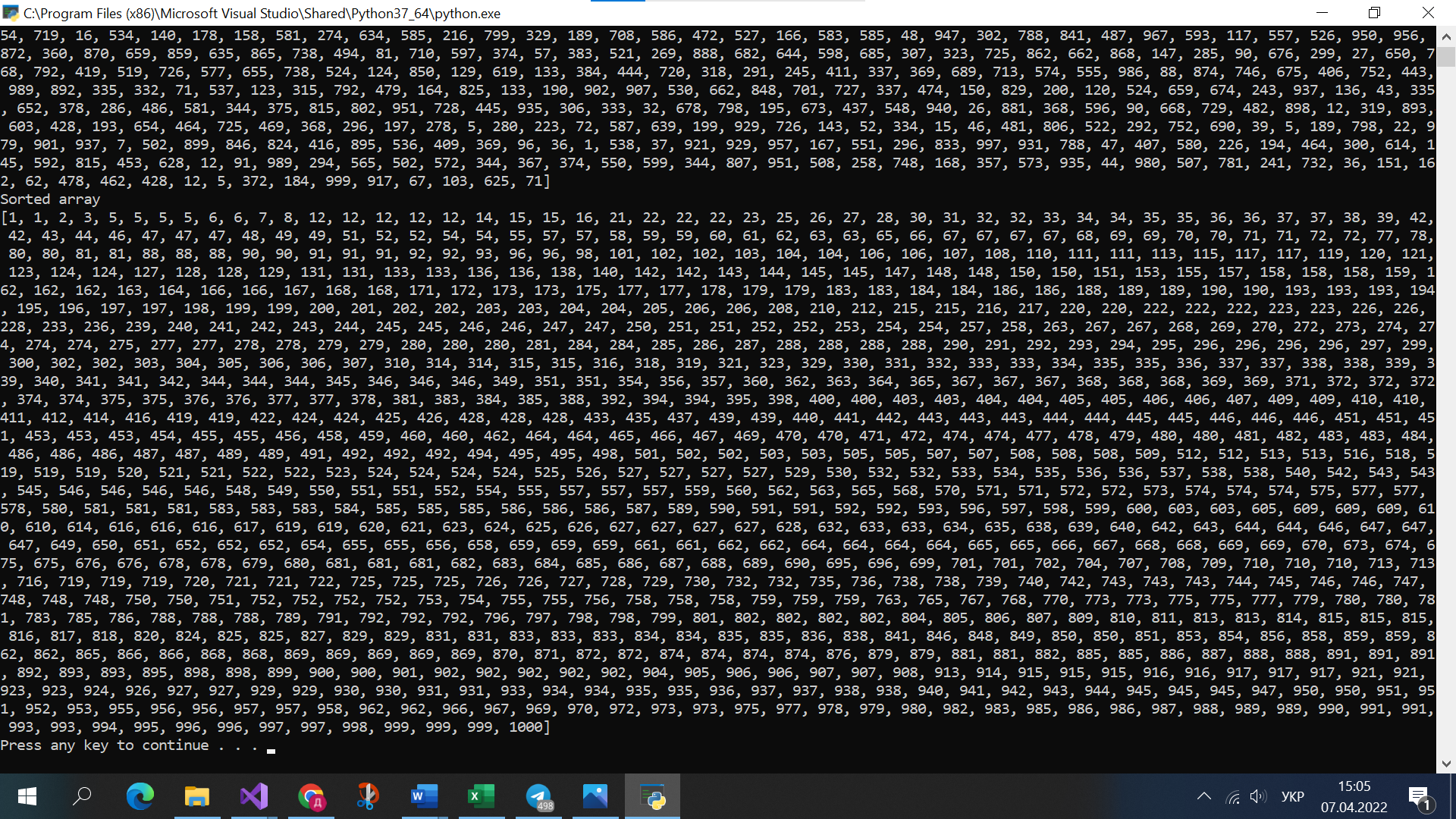
1. *Приклад роботи на 100 та 1000 елементів*

На 100 елементів



На 1000 елементів





1. *Тестування алгоритму*

Характеристики оцінювання алгоритму сортування гребінцем для упорядкованої послідовності елементів у масиві

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 36 | 0 |
| 100 | 1229 | 0 |
| 1000 | 22056 | 0 |
| 5000 | 145060 | 0 |
| 10000 | 320071 | 0 |
| 20000 | 700092 | 0 |
| 50000 | 2000058 | 0 |

Характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45 | 9 |
| 100 | 1328 | 110 |
| 1000 | 23055 | 1508 |
| 5000 | 150059 | 9135 |
| 10000 | 330070 | 18791 |
| 20000 | 720091 | 38797 |
| 50000 | 2050057 | 97745 |

Характеристика оцінювання алгоритму сортування гребінцем для випадкової послідовності елементів у масиві.

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45 | 11 |
| 100 | 1328 | 233 |
| 1000 | 23055 | 4288 |
| 5000 | 155058 | 26802 |
| 10000 | 350068 | 57918 |
| 20000 | 760089 | 122408 |
| 50000 | 2100056 | 318132 |

### *Графіки залежності часових характеристик оцінювання від розмірності масив*

1. *Висновок*

Під час виконання лабораторної роботи було досліджено два методи сортування масиву: бульбашкою та гребінцем. В ході цієї роботи визначив, що сортування гребінцем набагато ефективніший спосіб сортування, ніж бульбашка, тому що в нього менша кількість порівнянь та перестановок, а отже потрібен менший час для виконання алгоритму. Також було досліджено часову складність обох алгоритмів.