Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Катедра ІПІ

Звіт

з лабораторної роботи №1 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

„Проєктування і аналіз алгоритмів внутрішнього сортування”

Виконав *ІП-11 Лесів Владислав Ігорович* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Халус Олена Андріївна* _____
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

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

2 ЗАВДАННЯ

Виконати аналіз алгоритму внутрішнього сортування на відповідність наступним властивостям (таблиця 2.1):

– стійкість;

– «природність» поведінки (Adaptability);

– базуються на порівняннях;

– необхідність додаткової пам'яті (об'єму);

– необхідність в знаннях про структуру даних.

Записати алгоритм внутрішнього сортування за допомогою псевдокоду (чи іншого способу по вибору).

Провести аналіз часової складності в гіршому, кращому і середньому випадках та записати часову складність в асимптотичних оцінках.

Виконати програмну реалізацію алгоритму на будь-якій мові програмування з фіксацією часових характеристик оцінювання (кількість порівнянь, кількість перестановок, глибина рекурсивного поглиблення та інше в залежності від алгоритму).

Провести ряд випробувань алгоритму на масивах різної розмірності (10, 100, 1000, 5000, 10000, 20000, 50000 елементів) і різних наборів вхідних даних (впорядкований масив, зворотно упорядкований масив, масив випадкових чисел) і побудувати графіки залежності часових характеристик оцінювання від розмірності масиву, нанести на графік асимптотичну оцінку гіршого і кращого випадків для порівняння.

Зробити порівняльний аналіз двох алгоритмів.

Зробити узагальнений висновок з лабораторної роботи.

Таблиця 2.1 – Варіанти алгоритмів

№	**Алгоритм сортування**
1	Сортування бульбашкою
2	Сортування гребінцем («розчіскою»)

3 ВИКОНАННЯ

3.1 Аналіз алгоритму на відповідність властивостям

Аналіз алгоритму сортування бульбашкою та сортування гребінцем на відповідність властивостям наведено в таблиці 3.1.

Таблиця 3.1 – Аналіз алгоритму на відповідність властивостям

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

3.2 Псевдокод алгоритму сортування бульбашкою.

```
bubbleSort(arr):
    for i ← 0 to length [arr]
        for j ← 0 to length [arr] - i -1
            if arr[j] > arr[j + 1]
                arr[j], arr[j+1] ← arr[j+1], arr[j]
```

Псевдокод алгоритму сортування гребінцем.

```
combSort(arr):
    gap ← len(arr)
    swapped ← True
    while gap > 1 or swapped:
        gap ← max(1, int(gap / 1.24733))
```

swapped ← False

for i ← 0 to length [arr] - gap

if arr[i] > arr[i + gap]

arr[i], arr[i + gap] ← arr[i + gap], arr[i]

swapped ← True

3.3 Аналіз часової складності

У випадку найгіршої швидкодії часова складність алгоритму сортування бульбашкою становить $O(n^2)$, гребінцем – $O(n^2)$. Найкраща швидкодія: бульбашка – O(n), гребінець – O(n). Середня швидкодія алгоритмів: бульбашка – $O(n^2)$, гребінець – O(n*log(n))

Програмна реалізація алгоритму

3.3.1 Вихідний код

```
from random import randint
from datetime import datetime
def bubbleSort(arr):
    swaps, comparations=0,0
    start=datetime.now()
    for i in range(len(arr)):
        for j in range(len(arr)-i-1):
            comparations+=1
            if arr[j]>arr[j+1]:
                swaps+=1
                arr[j],arr[j+1]=arr[j+1],arr[j]
    time=datetime.now()-start
    return comparations, swaps, time

def combSort(arr):
    swaps, comparations=0,0
    gap=len(arr)
    swapped=True
    start=datetime.now()
    while gap>1 or swapped:
        gap=max(1,int(gap/1.24733))
        swapped=False
        for i in range(len(arr)-gap):
            comparations+=1
            if arr[i]>arr[i+gap]:
                swaps+=1
                arr[i],arr[i+gap]=arr[i+gap],arr[i]
                swapped=True
    time=datetime.now()-start
    return comparations, swaps, time

def cases(n):
    best=[i for i in range(n)]
    worst=[i for i in range(n,0,-1)]
    rand=[randint(0,n*2) for i in range(n)]
    return best,worst,rand
```

```
n=int(input("Введіть кількість елементів масиву: "))
bb, bw, br=cases(n)
cb, cw, cr =bb[:], bw[:], br[:]
for i in range(3):
    if i==0:
        print("Найкращий випадок:")
        b,c=bb,cb
    elif i==1:
        print("\nНайгірший випадок:")
        b,c=bw,cw
    else:
        print("\nВипадковий випадок:")
        b,c=br,cr
    compBub, swapBub, timeB=bubbleSort(b)
    print("Сортування бульбашкою: порівнянь -",compBub, ", перестановок -",swapBub, ", час
сортування -", timeB)
    compComb, swapComb, timeC=combSort(c)
          print("Сортування гребінцем: порівнянь -",compComb, ", перестановок -",swapComb,
", час сортування -", timeC)
```

3.3.2 Приклад роботи

На рисунках 3.1 і 3.2 показані приклади роботи програми сортування масивів на 100 і 1000 елементів відповідно.

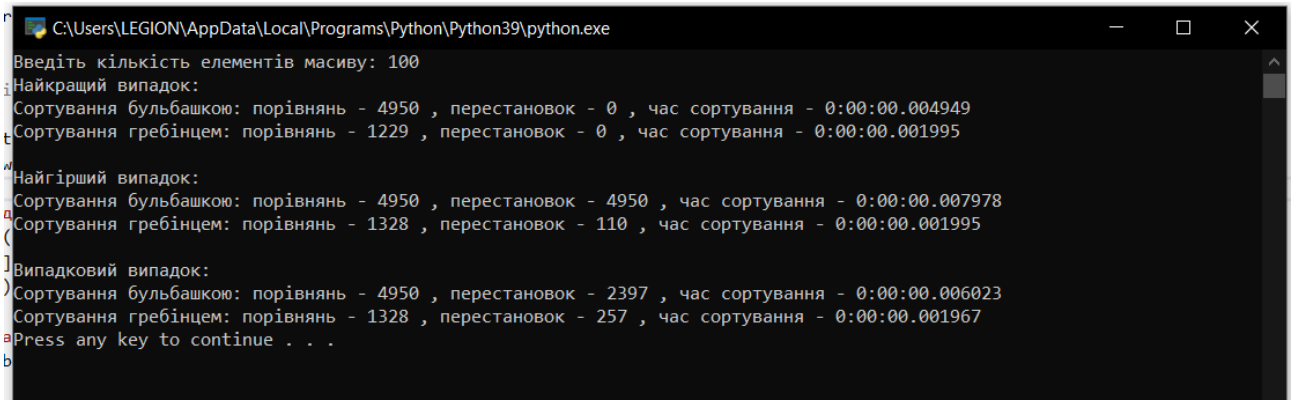


Рисунок 3.1 – Сортування масиву на 100 елементів

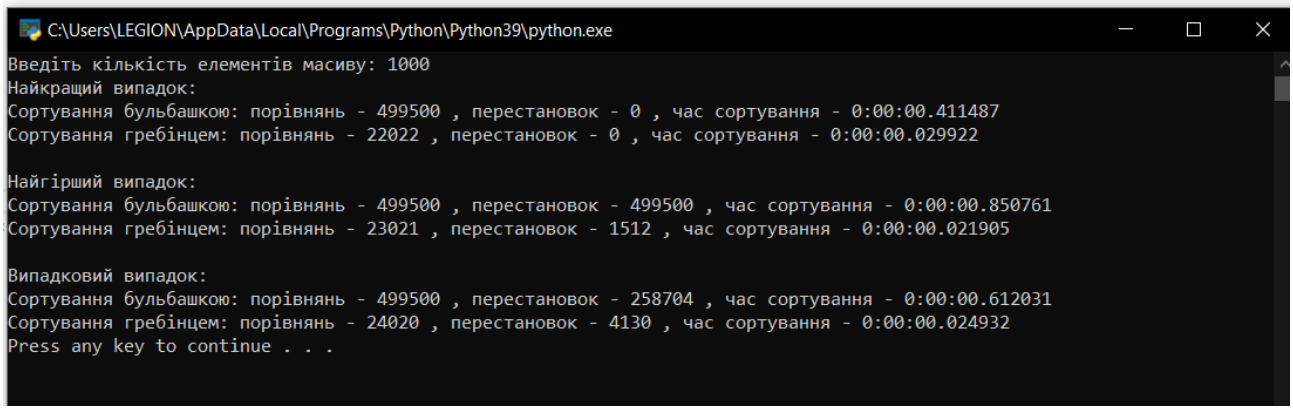


Рисунок 3.2 – Сортування масиву на 1000 елементів

3.4 Тестування алгоритму

3.4.1 Часові характеристики оцінювання

В таблиці 3.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.2 – 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

Для упорядкованої послідовності гребінцем:

Розмірність масиву	Число порівнянь	Число перестановок
10	36	0
100	1229	0
1000	22022	0
5000	144862	0
10000	329644	0
20000	719241	0
50000	1997958	0

В таблиці 3.3 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

Таблиця 3.3 – Характеристики оцінювання алгоритму сортування бульбашки для зворотно упорядкованої послідовності елементів у масиві.

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	1249975000

Для зворотно упорядкованої послідовності гребінцем:

Розмірність масиву	Число порівнянь	Число перестановок
10	45	9
100	1328	110
1000	23021	1512
5000	149861	9016
10000	339643	19132
20000	739240	40852
50000	2047957	109958

У таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4 – Характеристика оцінювання алгоритму сортування бульбашки для випадкової послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	45	26
100	4950	2445
1000	499500	239969

5000	12497500	6253988
10000	49995000	24848095
20000	199990000	100123539
50000	1249975000	624600655

Для випадкової послідовності гребінцем:

Розмірність масиву	Число порівнянь	Число перестановок
10	45	9
100	1328	243
1000	23021	4209
5000	164858	26668
10000	349642	58036
20000	759239	127213
50000	2097956	361438

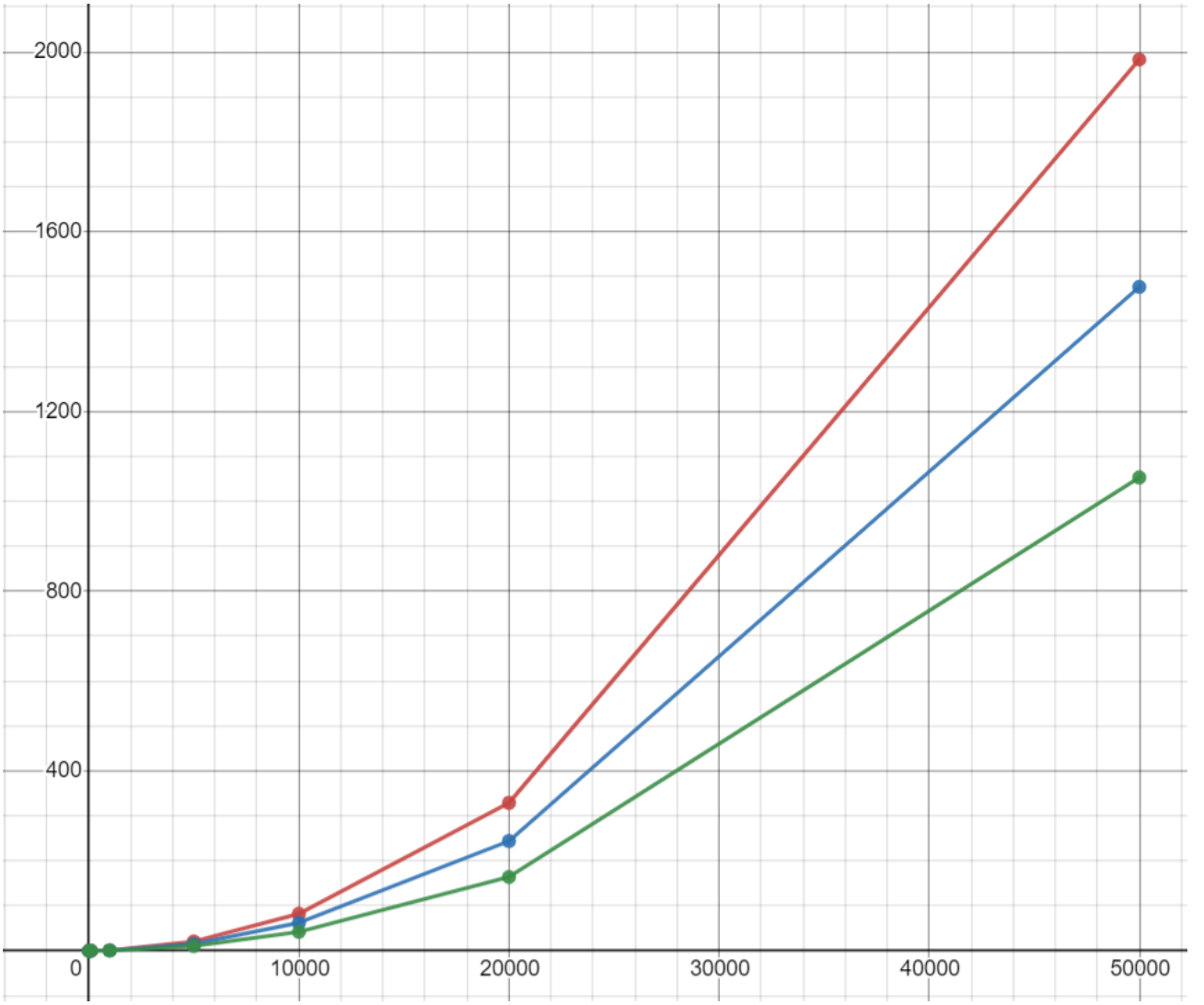
3.4.2 Графіки залежності часових характеристик оцінювання від розмірності масиву

На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (зелений графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять випадкову послідовність елементів (синій графік), також показані асимптотичні оцінки гіршого (фіолетовий графік) і кращого (жовтий графік) випадків для порівняння.

Рисунок 3.3 – Графіки залежності часових характеристик оцінювання

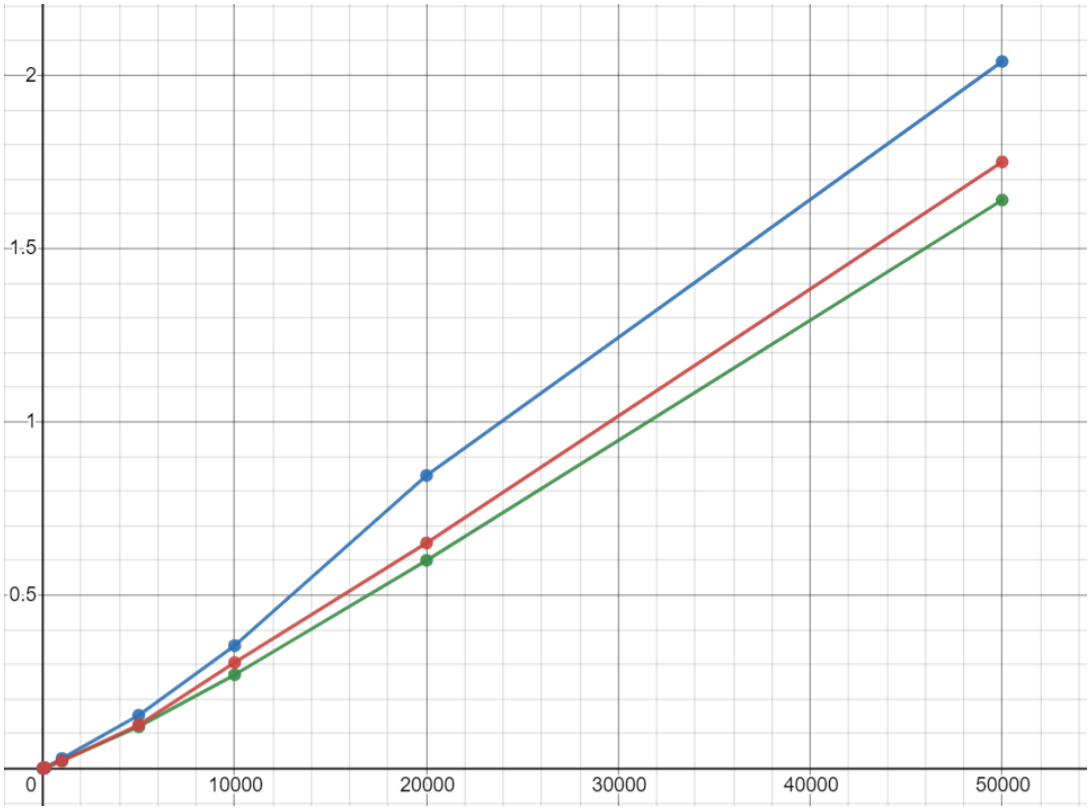
Графік залежності для алгоритму **бульбашки**.

Вісь Ox – кількість елементів масиву, вісь Oy – час сортування (с).

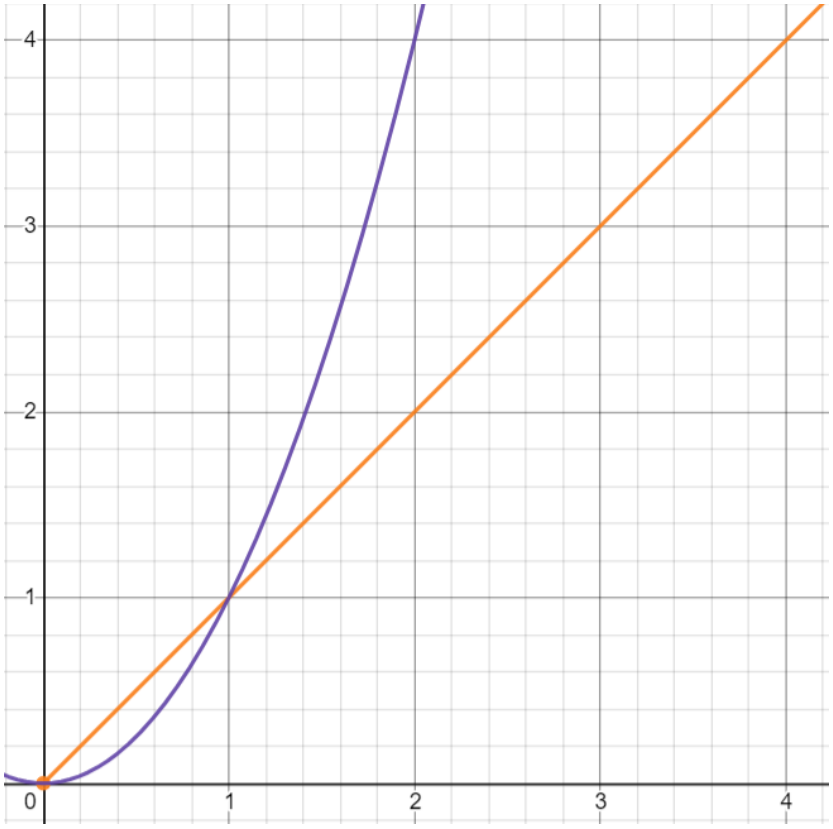


Графік залежності для алгоритму **гребінця**.

Вісь Ox – кількість елементів масиву, вісь Oy – час сортування (с).



Асимптотичні оцінки гіршого і кращого випадків для порівняння (як для бульбашки, так і для гребінця це $O(n^2)$ і $O(n)$ відповідно).



ВИСНОВОК

При виконанні даної лабораторної роботи я вивчив основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування – бульбашкою та гребінцем - і оцінив поріг їх ефективності. У результаті лабораторної роботи було проаналізовано властивості алгоритмів, записано їх псевдокод, програмно реалізовано алгоритми мовою Python, які виконують задачу відповідно до постановки. Використовуючи написану програму та порівнюючи результати таблично й графічно, отримуємо такий результат: алгоритм сортування гребінцем є значно ефективнішим за алгоритм сортування бульбашкою. Ефективність підтверджується набагато меншою кількістю як перевірок й перестановок всередині масиву чисел, так і меншою кількістю часу, витраченого на це сортування. Особливо результат помітний під час застосування алгоритму на масивах з великою кількістю елементів.

КРИТЕРІЇ ОЦІНЮВАННЯ

У випадку здачі лабораторної роботи до 21.02.2022 включно максимальний бал дорівнює – 5. Після 21.02.2022 – 28.02.2022 максимальний бал дорівнює – 2,5. Після 28.02.2022 робота не приймається

Критерії оцінювання у відсотках від максимального балу:

- аналіз алгоритму на відповідність властивостям – 10%;
- псевдокод алгоритму – 15%;
- аналіз часової складності – 25%;
- програмна реалізація алгоритму – 25%;
- тестування алгоритму – 20%;
- висновок – 5%.