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) *ІП-13 Вальчишен Ярослав Олександрович*
(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 гребінцем на відповідність властивостям наведено в таблиці 3.1.

Таблиця 3.1 – Аналіз алгоритму на відповідність властивостям

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

3.2 Псевдокод алгоритму

1) Сортування бульбашкою

Функція Sort(int[] array)

```
temp = 0
для i від 0 до array.Length повторити
    для j від 0 до array.Length – i повторити
        temp = array[j – 1]
        array[j – 1] = array[j]
        array[j] = temp
    все повторити
все повторити
```

2) Сортування гребінцем

Функція Sort(int[] array)

```
Temp = 0
gap = array.Length
```

поки gap > 1

gap = (int)(gap / 1.3)

якщо gap < 1 то

gap = 1

все якщо

для **i** від 0 до **array.Length - gap** повторити

якщо array[i] > array[i + gap] то

temp = array[j – 1]

array[j – 1] = array[j]

array[j] = temp

все якщо

все повторити

3.3 Аналіз часової складності

	Сортування бульбашкою	**Сортування гребінцем**
Найгірший випадок	$O(n^2)$	$O(n^2)$
Середній випадок	$O(n^2)$	$O(n^2)$
Найкращий випадок	$O(n^2)$	O(nlog(n))

3.4 Програмна реалізація алгоритму

3.4.1 Вихідний код

1) Сортування бульбашкою

```
void Sort(int[] array)
{
    int temp = 0;
    for (int i = 0; i < array.Length; i++)
    {
        for (int j = 1; j < array.Length - i; j++)
        {
            if (array[j - 1] > array[j])
            {
```

```
                    temp = array[j – 1]
                    array[j – 1] = array[j]
                    array[j] = temp
                }
            }
        }
    }
```

2) Сортування гребінцем

```
    void Sort(int[] array)
    {
        int temp = 0;
        int gap = array.Length;

        while (gap > 1)
        {
            gap = (int)(gap / 1.3);
            if (gap < 1)
                gap = 1;

            for (int i = 0; i < array.Length - gap; i++)
            {

                if (array[i] > array[i + gap])
                {
                    temp = array[j – 1]
                    array[j – 1] = array[j]
                    array[j] = temp
                }
            }
        }
    }
```

3.4.2 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

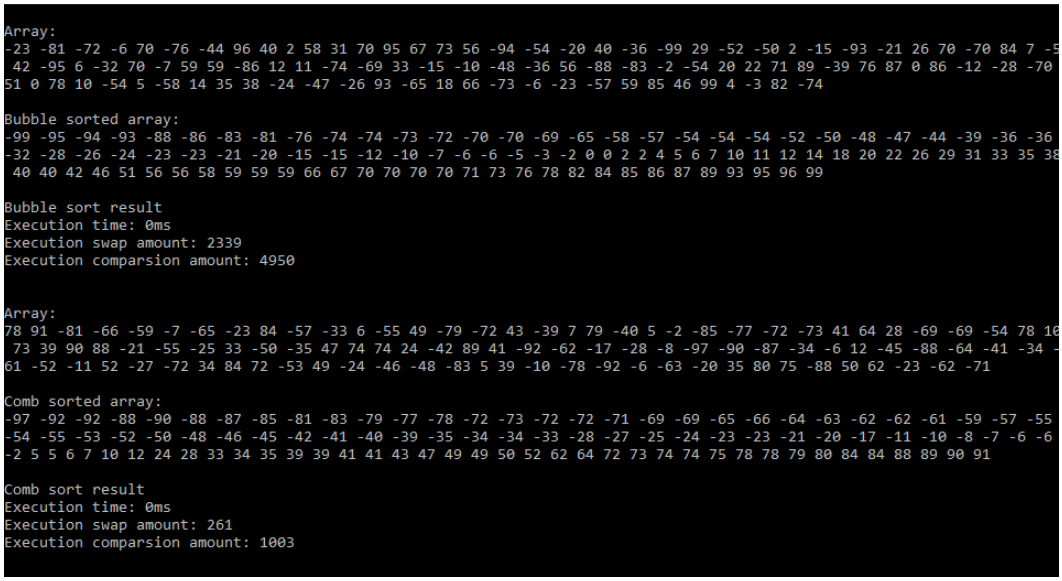
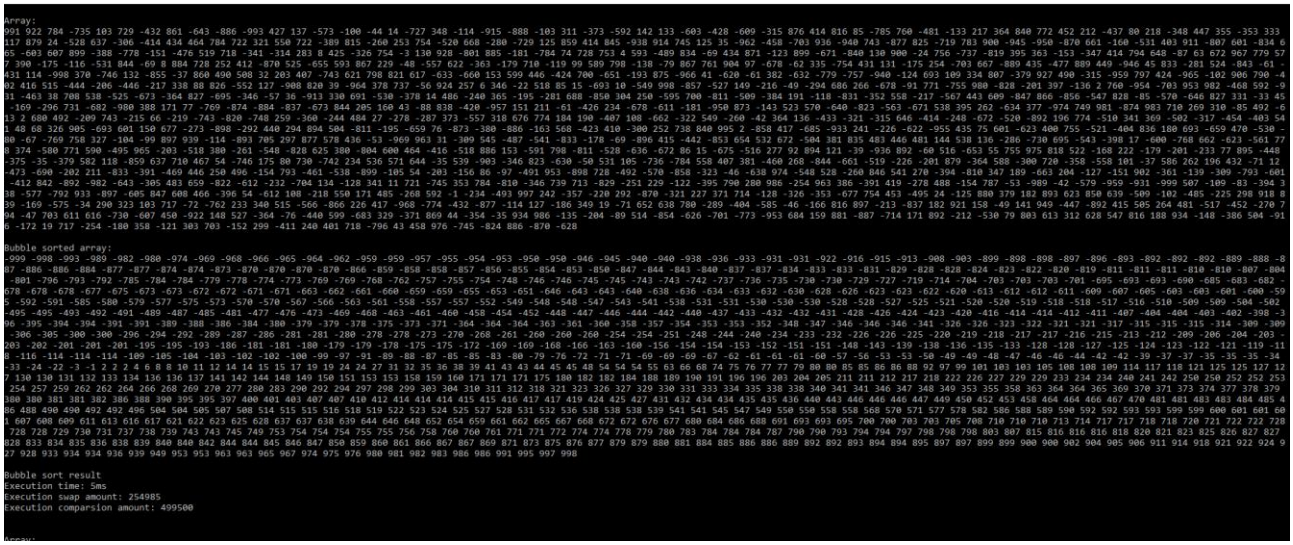


Рисунок 3.2 – Сортування масиву на 1000 елементів



3.5 Тестування алгоритму

3.5.1 Часові характеристики оцінювання.

Характеристики оцінювання алг оритмів сортування для упорядкованої послідовності елементів у масиві

Сортування бульбашкою

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

Сортування гребінцем

Розмірність масиву	Число порівнянь	Число перестановок
10	32	0
100	1003	0
1000	18713	0
5000	123386	0
10000	276739	0
20000	613402	0
50000	1683412	0

Характеристики оцінювання алгоритмів сортування для зворотно упорядкованої послідовності елементів у масиві.

Сортування бульбашкою

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

Сортування гребінцем

Розмірність масиву	Число порівнянь	Число перестановок
10	32	7
100	1003	122
1000	18713	1582
5000	123386	9572
10000	276739	20078
20000	613402	42634
50000	1683412	116838

Характеристика оцінювання алгоритмів сортування для випадкової послідовності елементів у масиві.

Сортування бульбашкою

Розмірність масиву	Число порівнянь	Число перестановок
10	45	25
100	4950	2892
1000	499500	253227
5000	12497500	6248996
10000	49995000	24837938
20000	199990000	100322407
50000	1249975000	625235373

Сортування гребінцем

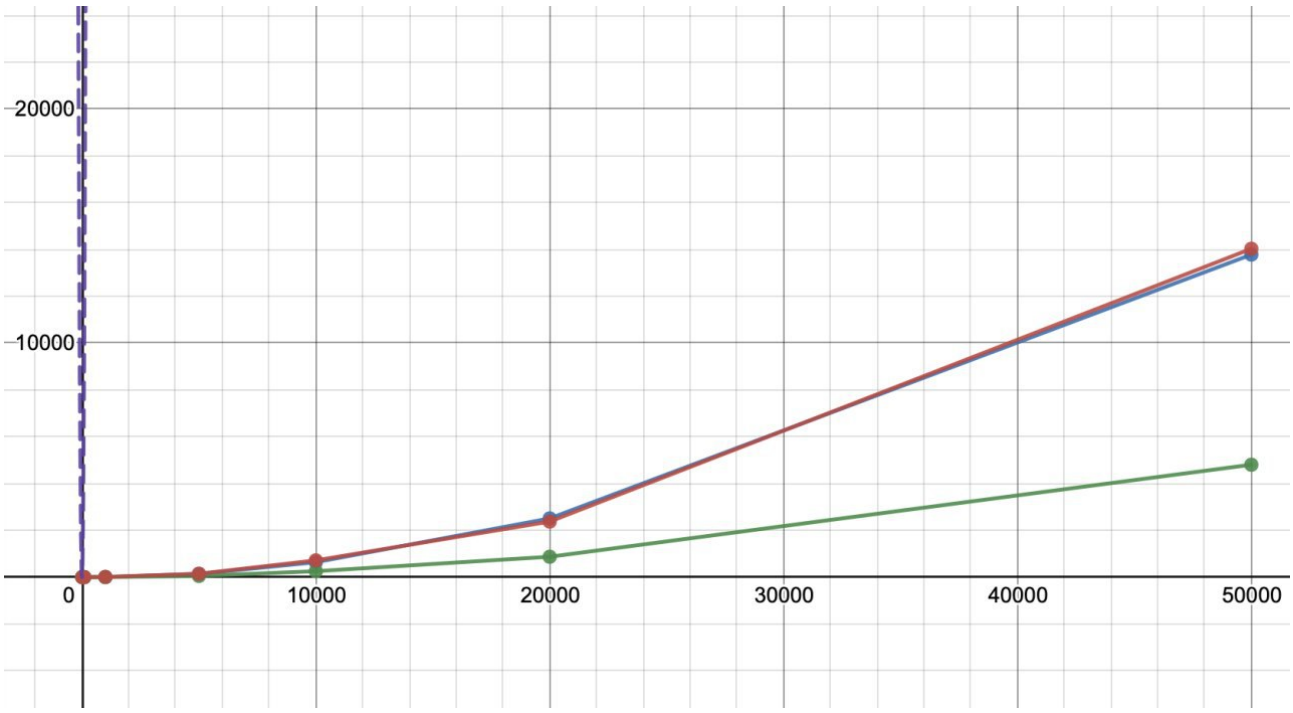
Розмірність масиву	Число порівнянь	Число перестановок
10	32	11
100	1003	250
1000	18713	4306
5000	123386	27786
10000	276739	61784
20000	613402	133151
50000	1683412	383067

3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву

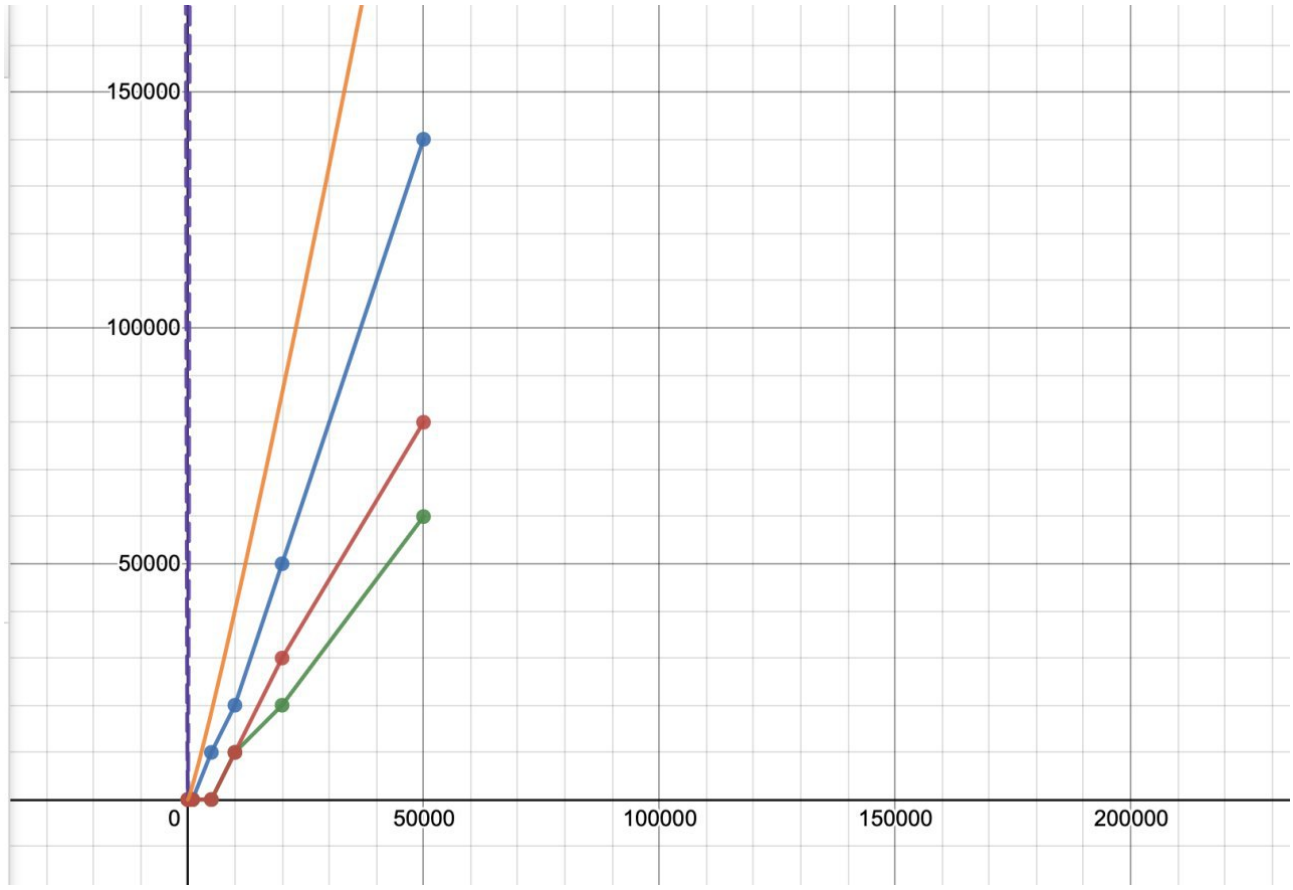
На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (зелений графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять випадкову 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

Сортування бульбашкою



Сортування гребінцем



ВИСНОВОК

При виконанні даної лабораторної роботи я вивчив основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування і оцінив поріг їх ефективності.

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%.