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 Жмайло Дмитро Олександрович* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Сопов Олексій Олександрович* _____
(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 алгоритмів на відповідність властивостям

Властивість	Сортування бульбашкою	Сортування гребінцем
Стійкість	Так	Так
«Природність» поведінки (Adaptability)	Ні	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:**

```
підпрограма BubbleSort(int [] arr)
    для і від 0 до arr.Length повторити
        для j від 1 до arr.Length - 1 повторити
            якщо arr[j-1] > arr[j]
                то
                temp := arr[j - 1]
                arr[j - 1] := arr[j]
                arr[j] := temp
            все якщо
        все повторити
    все повторити
все підпрограма
```

2) Сортування **гребінцем:**

```
підпрограма CombSort(int[]arr)
        factor := 1.3
        step := arr.Length - 1;
        поки step > 1 повторити
           step := (int)(step / factor);
           якщо step < 1
              то
              step := 1;
           все якщо

           повторити для i від 0 до i + step < arr.Length
              якщо arr[i] > arr[i+step]
                    то
                    int temp := arr[i + step]
                     arr[i + step] := arr[i]
                    arr[i] := temp
             все якщо
          все повторити
       все повторити
все підпрограма
```

3.3 Аналіз часової складності

	Сортування бульбашкою	**Сортування гребінцем**
Найгірший випадок *(масив відсортований у зворотньому порядку)*	$O(n^2)$	$O(n^2)$
Середній випадок *(масив з випадковими Значеннями)*	$O(n^2)$	$O(n^2)$
Найкращий випадок *(вже відсортований масив)*	$O(n^2)$	O(nlog(n))

3.4 Програмна реалізація алгоритму

3.4.1 Вихідний код

Сортування **бульбашкою**:

```
public static void BubbleSort(int[]arr)
{
    int temp;
    for (int i = 0; i < arr.Length; i++)
    {
        for (int j = 1; j < arr.Length - i; j++)
        {
            if (arr[j-1] > arr[j])
            {
                temp = arr[j - 1];
                arr[j - 1] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

Сортування **гребінцем**:

```
public static void CombSort(int[]arr)
        {
            double factor = 1.3;
            int step = arr.Length - 1;
            while (step > 1)
            {
                step = (int)(step / factor);
                if (step < 1)
                    step = 1;
                for (int i = 0; i + step < arr.Length; i++)
                {
                    if (arr[i] > arr[i+step])
                    {
                        int temp = arr[i + step];
                        arr[i + step] = arr[i];
                        arr[i] = 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

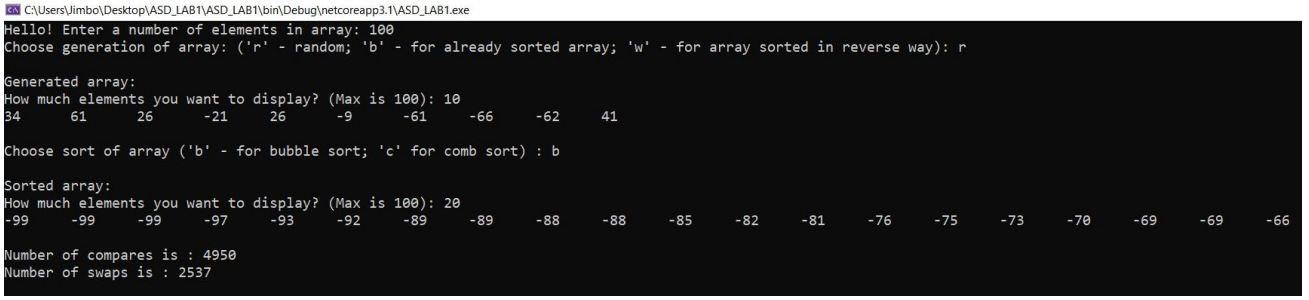
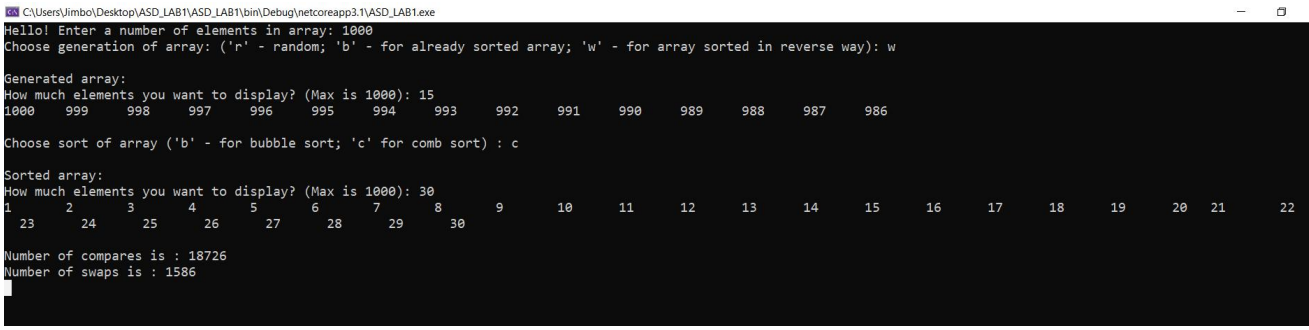


Рисунок 3.2 – Сортування масиву на 1000 елементів



3.5 Тестування алгоритму

3.5.1 Часові характеристики оцінювання

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

Таблиця 3.2 – Характеристики оцінювання алгоритмів сортування бульбашкою та гребінцем для упорядкованої послідовності елементів у масиві

	Сортування бульбашкою		Сортування гребінцем	
Розмірність масиву	Число порівнянь	Число перестановок	Число порівнянь	Число перестановок
10	45	0	34	0
100	4950	0	1003	0
1000	499500	0	18726	0
5000	12497500	0	123394	0
10000	49995000	0	276740	0
20000	199990000	0	613403	0
50000	1249975000	0	1683418	0

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

Таблиця 3.3 – Характеристики оцінювання алгоритмів сортування бульбашкю та гребінцем для зворотно упорядкованої послідовності елементів у масиві.

	Сортування бульбашкою		Сортування гребінцем	
Розмірність масиву	Число порівнянь	Число перестановок	Число порівнянь	Число перестановок
10	45	45	34	9
100	4950	4950	1003	122
1000	499500	499500	18726	1586
5000	12497500	12497500	123394	9490
10000	49995000	49995000	276740	20040
20000	199990000	199990000	613403	42338
50000	1249975000	1249975000	1683418	117028

У таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритмів сортування бульбашкою та гребінцем для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4 – Характеристика оцінювання алгоритмів сортування бульбашкою та гребінцем для випадкової послідовності елементів у масиві.

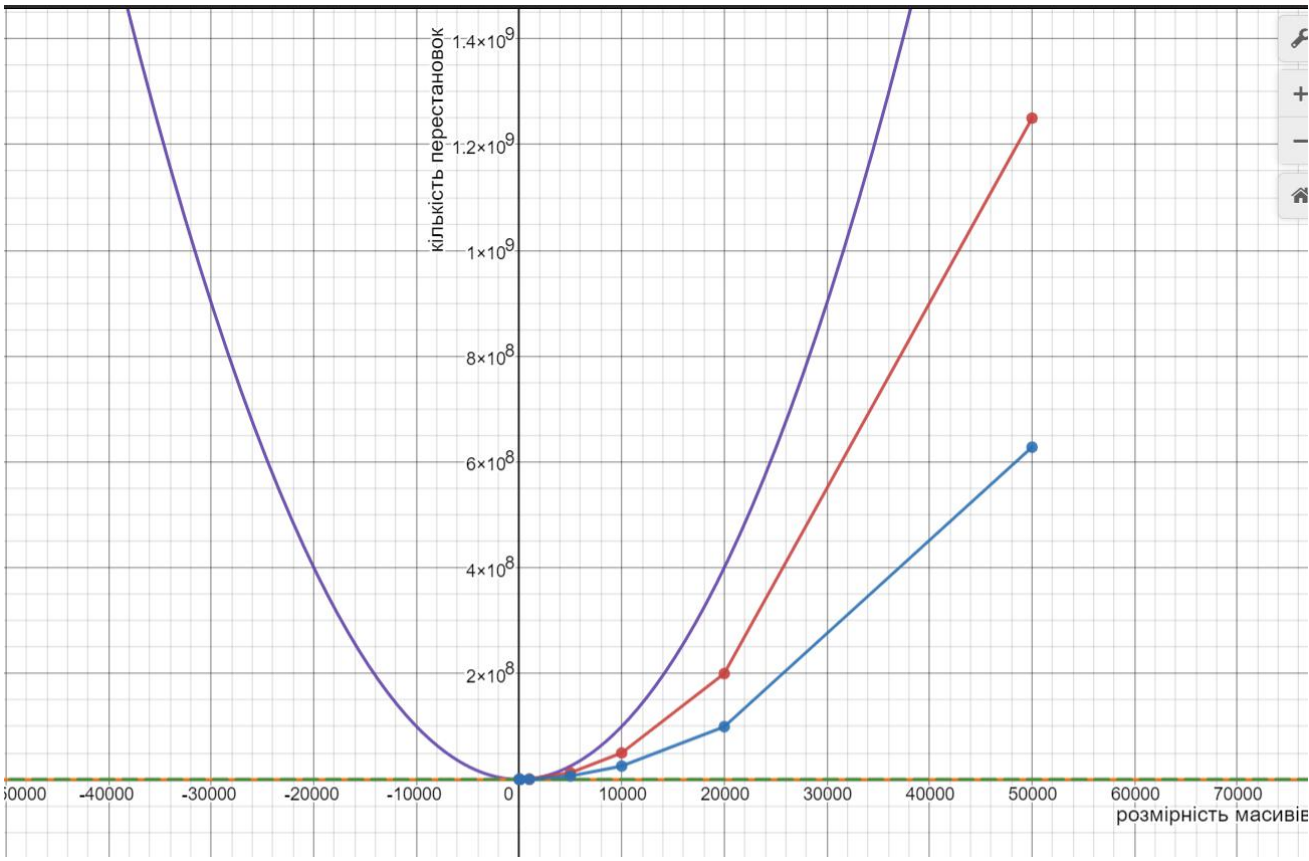
	Сортування бульбашкою		Сортування гребінцем	
Розмірність масиву	Число порівнянь	Число перестановок	Число порівнянь	Число перестановок
10	45	19	34	9
100	4950	2585	1003	234
1000	499500	243393	18726	4529
5000	12497500	6343574	123394	27733
10000	49995000	25162957	276740	61591
20000	199990000	99556670	613403	132864
50000	1249975000	627921151	1683418	381690

3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву

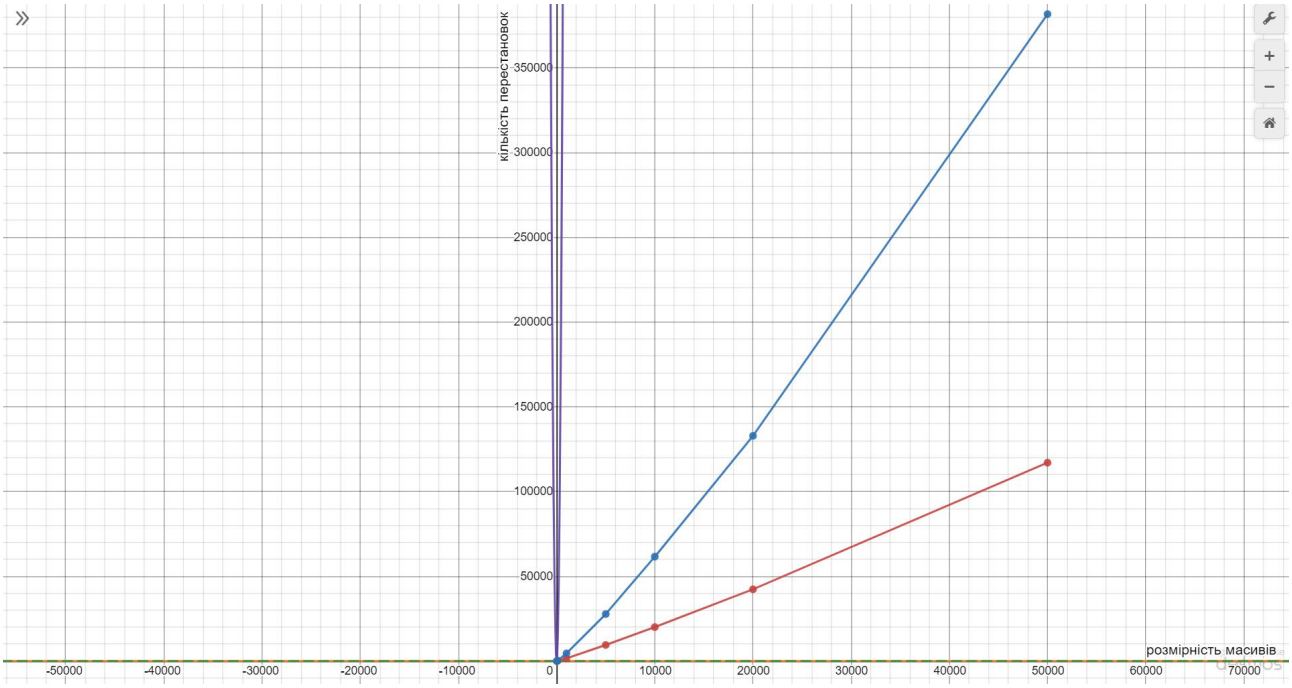
На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (зелений графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять випадкову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

1) Сортування бульбашкою:



2) Сортування гребінцем:



ВИСНОВОК

При виконанні даної лабораторної роботи я дослідив два алгоритми сортування (сортування бульбашкою і сортування гребінцем), дізнався про їх програмну реалізацію, спосіб роботи; дослідив їх часову складність та наочно порівняв роботу алгоритмів сортування при різних вхідних масивах за допомогою графіків та таблиць. Створив програму на основі цих алгоритмів сортування та перевірив її працездатність. Побачив значну перевагу сортування гребінцем як в часі виконання, так і в кількості порівнянь та перестановок. Порівняв характеристики алгоритмів сортування у найгіршому, середньому та найкращому випадках і зробив висновки.

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%.