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 Пархомчук Ілля Вікторович* _____
(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 на відповідність властивостям наведено в таблиці 3.1.1

Таблиця 3.1.1 – Аналіз алгоритму на відповідність властивостям

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

Аналіз алгоритму сортування гребінцем на відповідність властивостям наведено в таблиці 3.1.2

Таблиця 3.1.2 – Аналіз алгоритму на відповідність властивостям

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

3.2 Псевдокод алгоритму

Сортування бульбашкою

BubbleSort(arr[], size)

для i **від** 0 **до** *size -1* , **з кроком** 1 **повторити**
для j **від** 0 **до** *size -1*, **з кроком** 1 **повторити**
якщо arr[j] > arr[j + 1]
то
temp := arr[j]
arr[j] := arr[j + 1]
arr[j+1] := temp
все якщо
все повторити
все повторити

Сортування гребінцем

GetGap(gap)
gap := gap / 1.3
якщо gap < 1
то
gap := 1
все якщо
повернути gap

```
CombSort(arr[], size)
  gap := size
  swapped := істина
  повторити поки gap > 1 або  swapped
       swapped := хиба
       gap = GetGap(gap)

       для i від 0 до size -gap , з кроком 1 повторити
          якщо arr[i] > arr[i + gap]
            то
              temp := arr[i]
              arr[i] := arr[i + gap]
              arr[i+gap] := temp
              swapped := істина
            все якщо
       все повторити
  все повторити
```

3.3 Аналіз часової складності

Сортування бульбашкою

Випадок	Складність
Найгірший	n^2
Середній	n^2
Найкращий	n^2

Сортування гребінцем

Випадок	Складність
Найгірший	n^2
Середній	$n \log n$
Найкращий	$n \log n$

3.4 Програмна реалізація алгоритму

3.4.1 Вихідний код

Сортування бульбашкою

```
void BubbleSort(int* Arr, int size)
{
	int temp;
	for (int i = 0; i < size-1; i++)
	{
		for (int j = 0; j < size-1; j++)
		{
			if (Arr[j] > Arr[j + 1])
			{
				temp = Arr[j];
				Arr[j] = Arr[j + 1];
				Arr[j + 1] = temp;
			}
		}
	}
}
```

Сортування гребінцем

```
int GetGap(int gap)
{
	gap /= 1.3;
	if (gap < 1)
	{
	gap = 1;
	}
	return gap;
}

void CombSort(int* arr, int size)
{
	int temp, gap = size;
	bool swapped = true;
	while (gap > 1 || swapped)
	{
		swapped = false;
		gap = GetGap(gap);
		for (int i = 0; i < size - gap; ++i)
		{
			if (arr[i] > arr[i + gap])
			{
				temp = arr[i];
				arr[i] = arr[i + gap];
				arr[i + gap] = temp;
				swapped = true;
			}
		}
	}
}
```

3.4.2 Приклад роботи

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

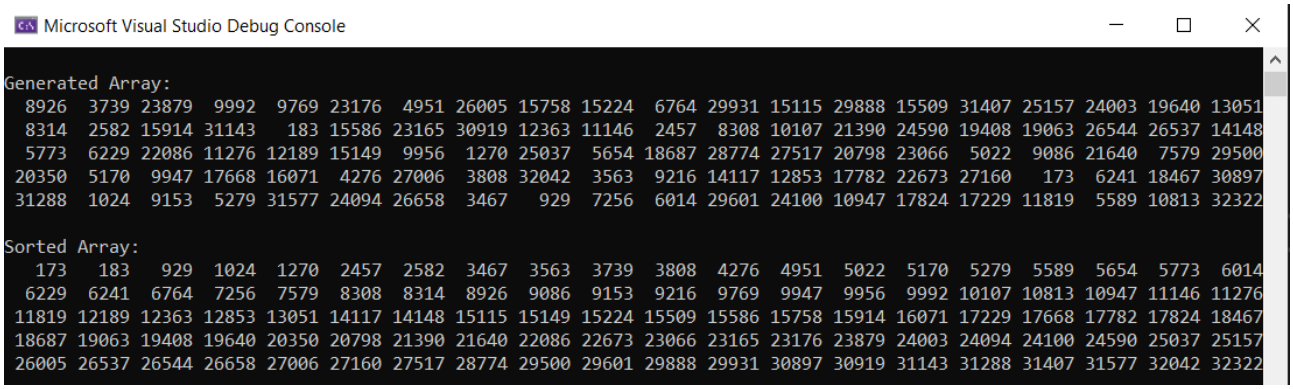


Рисунок 3.1 – Сортування масиву на 100 елементів

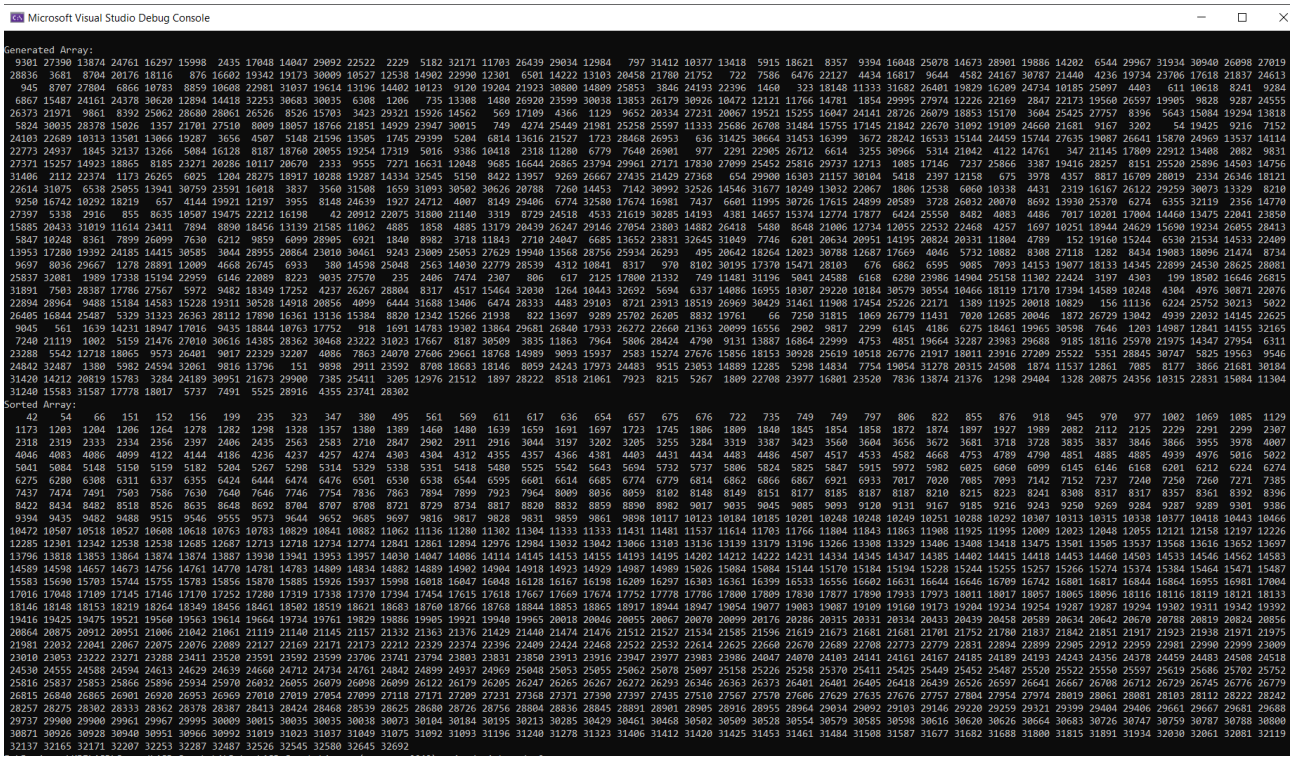


Рисунок 3.2 – Сортування масиву на 1000 елементів

3.5 Тестування алгоритму

3.5.1 Часові характеристики оцінювання

В таблиці 3.2.1 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.2.1 – Характеристики оцінювання алгоритму сортування бульбашки для упорядкованої послідовності елементів у масиві

Розмірність масиву	Число порівнянь	Число перестановок
10	81	0
100	9801	0
1000	998001	0
5000	24990001	0
10000	99980001	0
20000	399960001	0
50000	2499900001	0

В таблиці 3.2.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.2.2 – Характеристики оцінювання алгоритму сортування гребінцем для упорядкованої послідовності елементів у масиві

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

В таблиці 3.3.1 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

Таблиця 3.3.1 – Характеристики оцінювання алгоритму сортування бульбашки для зворотно упорядкованої послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	81	45
100	9801	4950
1000	998001	499500
5000	24990001	12497500
10000	99980001	49995000
20000	399960001	199990000
50000	2499900001	1249975000

В таблиці 3.3.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

Таблиця 3.3.2 – Характеристики оцінювання алгоритму сортування гребінцем для зворотно упорядкованої послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	41	7
100	1102	122
1000	19712	1582
5000	128385	9572
10000	286738	20078
20000	633401	42634
50000	1733411	116838

У таблиці 3.4.1 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4.1 – Характеристика оцінювання алгоритму сортування бульбашки для випадкової послідовності елементів у масиві.

Розмірність масиву	Число порівнянь	Число перестановок
10	81	14
100	9801	2401
1000	998001	249351
5000	24990001	6344409
10000	99980001	25124386
20000	399960001	100130948
50000	2499900001	624639797

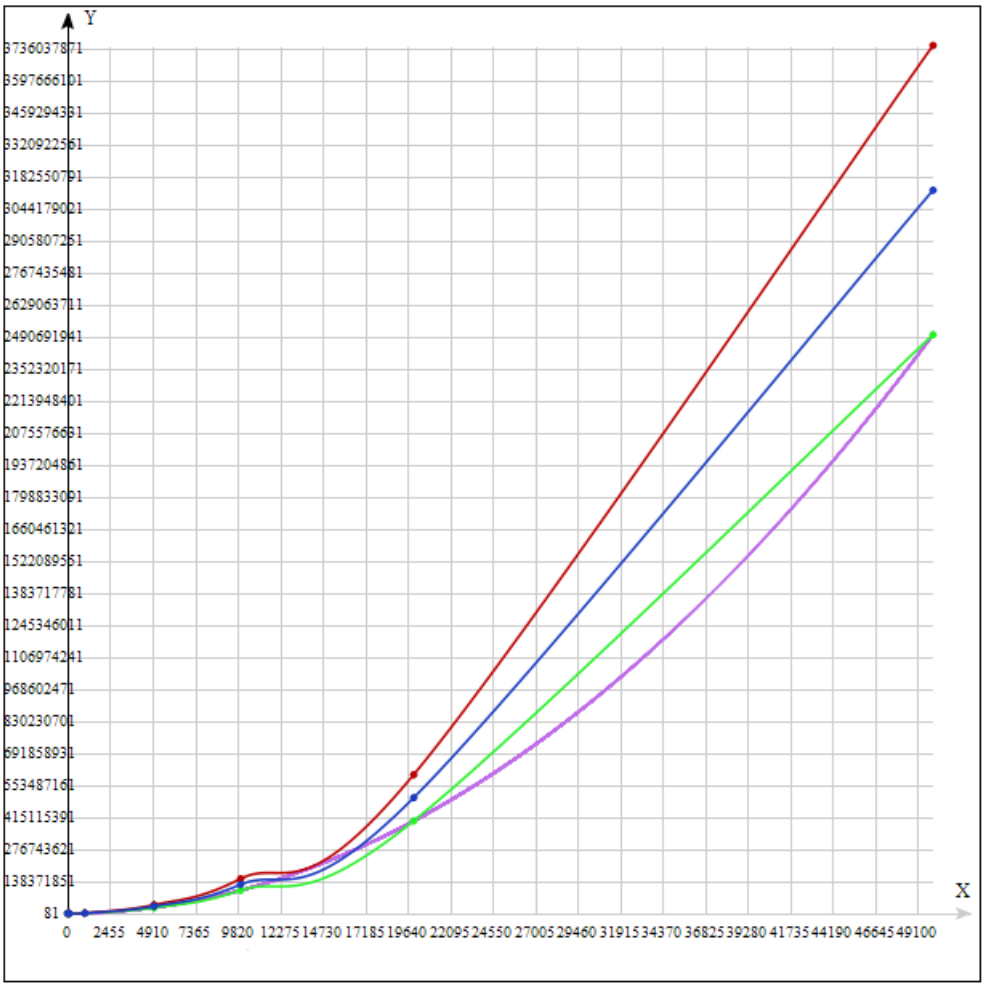
У таблиці 3.4.2 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4.2 – Характеристика оцінювання алгоритму сортування гребінцем для випадкової послідовності елементів у масиві.

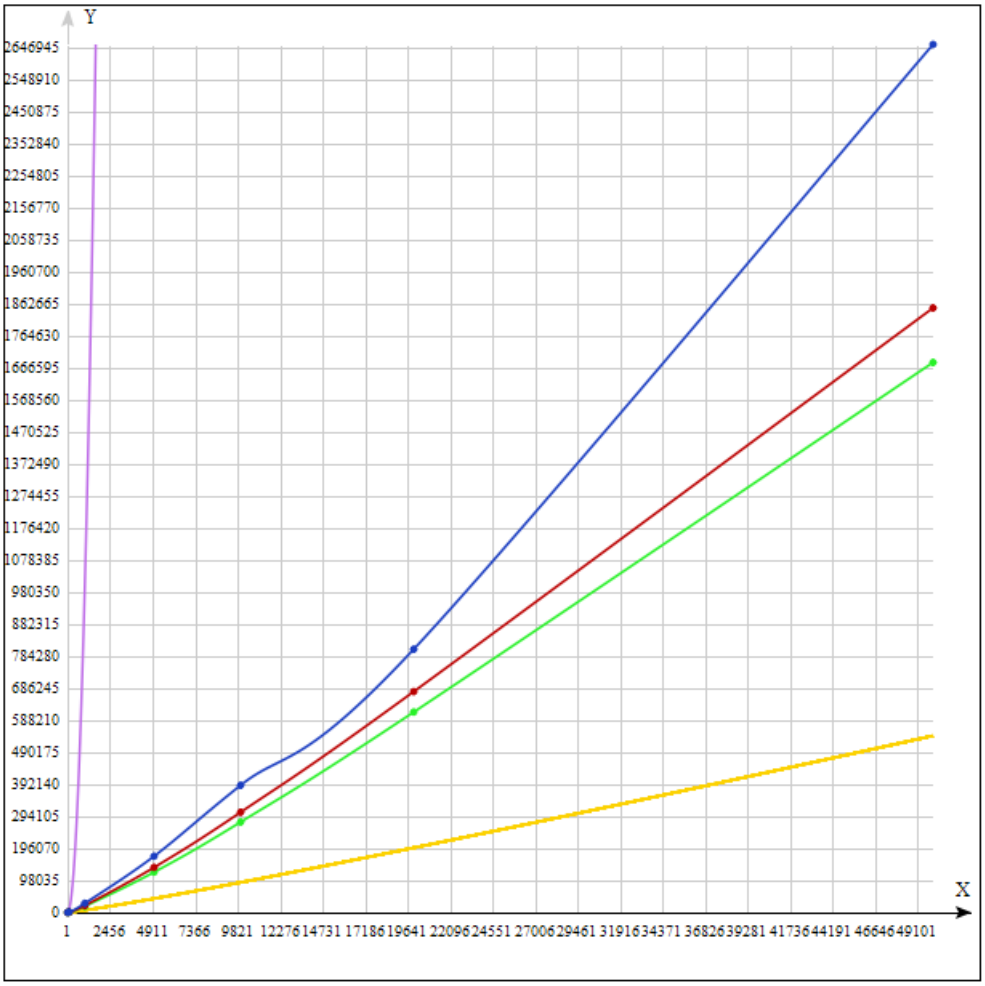
Розмірність масиву	Число порівнянь	Число перестановок
10	41	6
100	1300	267
1000	23708	4322
5000	143382	28741
10000	326734	62564
20000	673399	132277
50000	2283400	373305

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.



Сортування бульбашкою



Сортування гребінцем

Рисунок 3.3 – Графіки залежності часових характеристик оцінювання

ВИСНОВОК

При виконанні даної лабораторної роботи я ознайомився з нотацією великого О. Навчився оцінювати швидкодію алгоритмів бульбашки та гребінцем. Мною був написаний програмний код та проведено тестування алгоритмів з фіксуванням кількості порівнянь та кількості перестановок. На основі отриманих даних було побудовано графіки із залежністю кількості операцій від кількості елементів.

Я помітив, що сортування бульбашкою у всіх випадках веде себе майже однаково й очікувано. Однак, сортування гребінцем на прямо та зворотно упорядкованій послідовності елементів веде себе краще, чим на випадковій послідовності. Це пояснюється основою ідеєю алгоритму, щоб великі елементи зразу ставити ближче до кінця, а маленькі – ближче до початку.