

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

**Кафедра ІІІ**

**Звіт**

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

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

**Виконав(ла)**

**ІІІ-13 Павленко Микита Андрійович**  
(шифр, прізвище, ім'я, по батькові)

**Перевірив**

\_\_\_\_\_  
(прізвище, ім'я, по батькові)

Київ 2022

## ЗМІСТ

<b>1</b>	<b>МЕТА ЛАБОРАТОРНОЇ РОБОТИ .....</b>	<b>3</b>
<b>2</b>	<b>ЗАВДАННЯ .....</b>	<b>4</b>
<b>3</b>	<b>ВИКОНАННЯ .....</b>	<b>5</b>
3.1	АНАЛІЗ АЛГОРИТМУ НА ВІДПОВІДНІСТЬ ВЛАСТИВОСТЯМ .....	5
3.2	ПСЕВДОКОД АЛГОРИТМУ .....	6
3.3	АНАЛІЗ ЧАСОВОЇ СКЛАДНОСТІ.....	7
3.4	ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ.....	7
3.4.1	<i>Вихідний код.....</i>	<i>7</i>
3.4.2	<i>Приклад роботи.....</i>	<i>8</i>
3.5	ТЕСТУВАННЯ АЛГОРИТМУ .....	10
3.5.1	<i>Часові характеристики оцінювання.....</i>	<i>10</i>
3.5.2	<i>Графіки залежності часових характеристик оцінювання від розмірності масиву .....</i>	<i>12</i>
	<b>ВИСНОВОК .....</b>	<b>15</b>
	<b>КРИТЕРІЇ ОЦІНЮВАННЯ .....</b>	<b>16</b>

## 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.

Таблиця 3.1 – Аналіз алгоритму на відповідність властивостям

Властивість	<b>Сортування бульбашкою</b>
Стійкість	Алгоритм стійкий
«Природність» поведінки (Adaptability)	Неприродна поведінка
Базуються на порівняннях	Сортування за допомогою порівнянь
Необхідність в додатковій пам'яті (об'єм)	Не потребується
Необхідність в знаннях про структури даних	Потребується

Таблиця 3.1.2 – Аналіз алгоритму на відповідність властивостям

Властивість	<b>Сортування гребінцем</b>
Стійкість	Алгоритм нестійкий
«Природність» поведінки (Adaptability)	Неприродна поведінка
Базуються на порівняннях	Сортування за допомогою порівнянь
Необхідність в додатковій пам'яті (об'єм)	Не потребується
Необхідність в знаннях про структури даних	Потребується

### 3.2 Псевдокод алгоритму

#### Функція **BubbleSort(array)**

Для  $i$  від 0 до  $\text{array.Length}$  **повторити**

Для  $j$  від 0 до  $\text{array.Length}-1$  **повторити**

**Якщо**  $\text{array}[j] > \text{array}[j+1]$  **то**

$\text{tmp} = \text{array}[j+1]$

$\text{array}[j+1] = \text{array}[j]$

$\text{array}[j] = \text{tmp}$

**все якщо**

**все повторити**

**все повторити**

**все функція**

#### Функція **CombSort(array)**

$\text{interval} = \text{array.Length}$

**Поки**  $\text{interval} > 1$  **повторити**

$\text{interval} = \text{interval} / 1.3$

**Якщо**  $\text{interval} < 1$  **то**

$\text{interval} = 1$

**все якщо**

Для  $i$  від 0 до  $\text{array.Length}-\text{interval}$  **повторити**

**Якщо**  $\text{array}[i] > \text{array}[i + \text{interval}]$  **то**

$\text{tmp} = \text{array}[i]$

$\text{array}[i] = \text{array}[i + \text{interval}]$

$\text{array}[i + \text{interval}] = \text{tmp}$

**все якщо**

**все повторити**

**все повторити**

**все функція**

### 3.3 Аналіз часової складності

Для сортування бульбашкою:

Найкращий випадок (впорядкований масив)	$\Omega(n^2)$
Найгірший випадок (обернено впорядкований масив)	$O(n^2)$
Середній випадок	$\Theta(n^2)$

Для сортування гребінцем:

Найкращий випадок (впорядкований масив)	$\Omega(n \log(n))$
Найгірший випадок	$O(n^2)$
Середній випадок	$\Theta(n^2)$

### 3.4 Програмна реалізація алгоритму на мові C#

#### 3.4.1 Вихідний код

1) Сортування бульбашкою

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

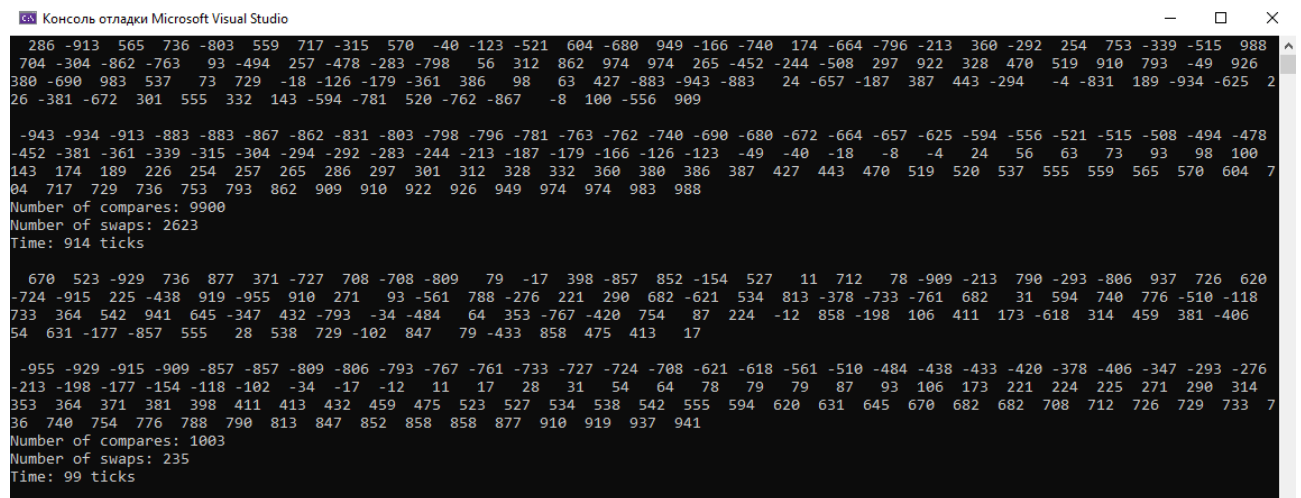
## 2) Сортуння гребінцем

```
public static void CombSort(int[] array)
{
    int tmp;
    int interval = array.Length;
    while (interval > 1)
    {
        interval = (int)(interval / 1.3);
        if (interval < 1)
        {
            interval = 1;
        }

        for (int i = 0; i < array.Length - interval; i++)
        {
            if (array[i] > array[i + interval])
            {
                tmp = array[i];
                array[i] = array[i + interval];
                array[i + interval] = tmp;
            }
        }
    }
}
```

### 3.4.2 Приклад роботи

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



```
Консоль отладки Microsoft Visual Studio
286 -913 565 736 -803 559 717 -315 570 -40 -123 -521 604 -680 949 -166 -740 174 -664 -796 -213 360 -292 254 753 -339 -515 988
704 -304 -862 -763 93 -494 257 -478 -283 -798 56 312 862 974 974 265 -452 -244 -508 297 922 328 470 519 910 793 -49 926
380 -690 983 537 73 729 -18 -126 -179 -361 386 98 63 427 -883 -943 -883 24 -657 -187 387 443 -294 -4 -831 189 -934 -625 2
26 -381 -672 301 555 332 143 -594 -781 520 -762 -867 -8 100 -556 909

-943 -934 -913 -883 -883 -867 -862 -831 -803 -798 -796 -781 -763 -762 -740 -690 -680 -672 -664 -657 -625 -594 -556 -521 -515 -508 -494 -478
-452 -381 -361 -339 -315 -304 -294 -292 -283 -244 -213 -187 -179 -166 -126 -123 -49 -40 -18 -8 -4 24 56 63 73 93 98 100
143 174 189 226 254 257 265 286 297 301 312 328 332 360 380 386 387 427 443 470 519 520 537 555 559 565 570 604 7
04 717 729 736 753 793 862 909 910 922 926 949 974 974 983 988
Number of compares: 9900
Number of swaps: 2623
Time: 914 ticks

670 523 -929 736 877 371 -727 708 -708 -809 79 -17 398 -857 852 -154 527 11 712 78 -909 -213 790 -293 -806 937 726 620
-724 -915 225 -438 919 -955 910 271 93 -561 788 -276 221 290 682 -621 534 813 -378 -733 -761 682 31 594 740 776 -510 -118
733 364 542 941 645 -347 432 -793 -34 -484 64 353 -767 -420 754 87 224 -12 858 -198 106 411 173 -618 314 459 381 -406
54 631 -177 -857 555 28 538 729 -102 847 79 -433 858 475 413 17

-955 -929 -915 -909 -857 -857 -809 -806 -793 -767 -761 -733 -727 -724 -708 -621 -618 -561 -510 -484 -438 -433 -420 -378 -406 -347 -293 -276
-213 -198 -177 -154 -118 -102 -34 -17 -12 11 17 28 31 54 64 78 79 79 87 93 106 173 221 224 225 271 290 314
353 364 371 381 398 411 413 432 459 475 523 527 534 538 542 555 594 620 631 645 670 682 682 708 712 726 729 733 7
36 740 754 776 788 790 813 847 852 858 858 877 910 919 937 941
Number of compares: 1003
Number of swaps: 235
Time: 99 ticks
```

Рисунок 3.1 – Сортуння масиву на 100 елементів



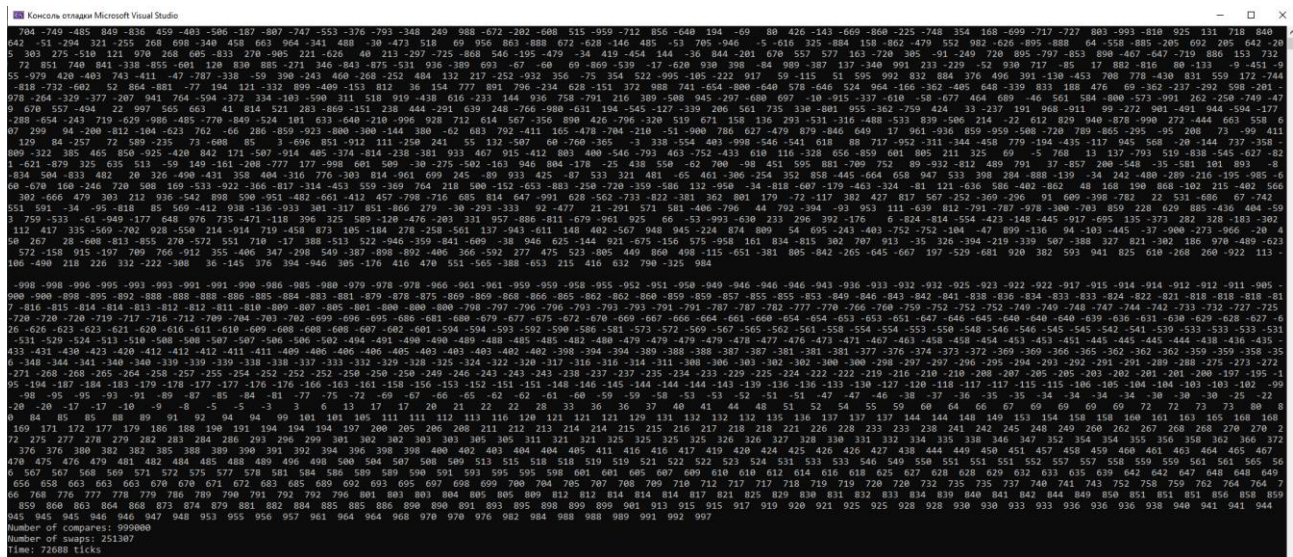


Рисунок 3.2 – Сортунання масиву на 1000 елементів

### 3.5 Тестування алгоритму

#### 3.5.1 Часові характеристики оцінювання

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

Таблиця 3.2 – Характеристики оцінювання алгоритму сортування для упорядкованої послідовності елементів у масиві

Для сортування бульбашкою

Розмірність масиву	Число порівнянь	Число перестановок
10	110	0
100	10100	0
1000	1001000	0
5000	25005000	0
10000	100010000	0
20000	400020000	0
50000	1624950000	0

Для сортування гребінцем

Розмірність масиву	Число порівнянь	Число перестановок
10	42	0
100	1103	0
1000	18730	0
5000	123414	0
10000	276767	0
20000	613430	0
50000	1683441	0

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

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

Для сортування бульбашкою

Розмірність масиву	Число порівнянь	Число перестановок
10	90	45
100	9900	4950
1000	999000	499500
5000	24995000	12497500
10000	99990000	49995000
20000	399980000	199990000
50000	1795017000	1249975000

Для 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

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** для випадкової послідовності елементів у масиві.

Для сортування бульбашкою

Розмірність масиву	Число порівнянь	Число перестановок
10	90	90
100	9900	2385
1000	999000	243371
5000	24995000	6272862
10000	99990000	25121547
20000	399980000	99698401
50000	1805017200	624976528

Для сортування гребінцем

Розмірність масиву	Число порівнянь	Число перестановок
10	32	11
100	1003	218
1000	18713	4084
5000	123386	27306
10000	276739	60305
20000	613402	130300
50000	1683412	376819

### 3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву

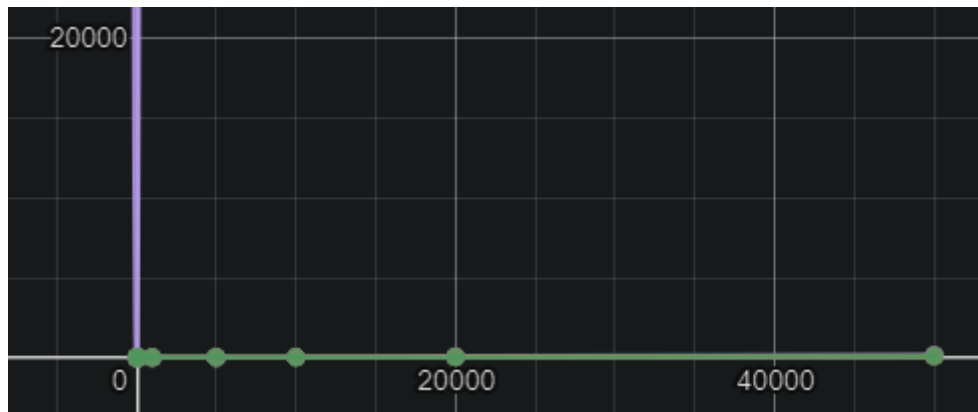
На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (зелений графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять випадкову 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. Перевірів здобуті знання шляхом програмної реалізації і оцінок.

## 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%.