**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)**

(шифр, прізвище, ім'я, по батькові)

*ІП-12 Єльчанінов Артем Юрійович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Сопов О.О.*

Київ 2022

Зміст

[1 Мета лабораторної роботи 3](#_Toc69772242)

[2 ЗаВдання 4](#_Toc69772243)

[3 Виконання 7](#_Toc69772244)

[3.1 Аналіз алгоритму на відповідність властивостям 7](#_Toc69772245)

[3.2 Псевдокод алгоритму 7](#_Toc69772246)

[3.3 Аналіз часової складності 7](#_Toc69772247)

[3.4 Програмна реалізація алгоритму 7](#_Toc69772248)

[3.4.1 Вихідний код 7](#_Toc69772249)

[3.4.2 Приклад роботи 8](#_Toc69772250)

[3.5 Тестування алгоритму 9](#_Toc69772251)

[3.5.1 Часові характеристики оцінювання 9](#_Toc69772252)

[3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву 11](#_Toc69772253)

[Висновок 12](#_Toc69772254)

[Критерії оцінювання 13](#_Toc69772255)

# Мета лабораторної роботи

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

# 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») |

# Виконання

## 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** | **Сортування бульбашкою** | **Сортування гребінцем** |
| Стійкість | Наявна | Не наявна |
| «Природність» поведінки (Adaptability) | Не природній | Наявна |
| Базуються на порівняннях | Наявна | Наявна |
| Необхідність в додатковій пам'яті (об'єм) | O(1) | O(1) |
| Необхідність в знаннях про структури даних | Наявна | Наявна |

## Псевдокод алгоритму

3.2.1     Сортування бульбашкою:

**Bubble\_sort(array):**

n := size(array)

**для** i **від** 0 **до** n – 1

**для** j **від** 0 **до** n – 1

**якщо** array[j] > array[j + 1]

**то**

temp := array[j]

array[j] := array[j + 1]

array[j + 1] := temp

**все якщо**

**все повторити**

**все повторити**

**return** array

3.2.2      Сортування гребінцем:

**Combo\_sort(array):**

swapped := false

n := size(array)

gap := n / 1.247

**поки** gap > 1 | | swapped == true

**повторити**

swapped := false

**для** і **від** 0 **до** n – gap

**якщо** array[i] > array[i + gap]

**то**

temp := array[i]

array[i] := array[i + gap]

array[i + gap] := temp

swapped := true

**все якщо**

**все повторити**

**якщо** gap > 1

**то**

gap := gap / 1.247

**все якщо**

**все повторити**

**return** array

## Аналіз часової складності

3.3.1     Сортування бульбашкою:

Найгірший випадок: O (n2)

Середній випадок: Ω (n2)

Найкращий випадок: Θ (n2)

3.3.2      Сортування гребінцем:

Найгірший випадок: O (n2)

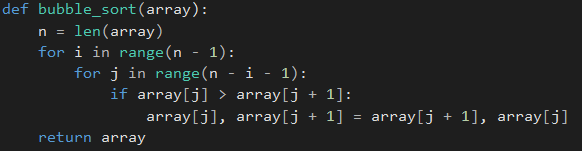
Середній випадок: Ω (n2)

Найкращий випадок: Θ (n \* log(n))

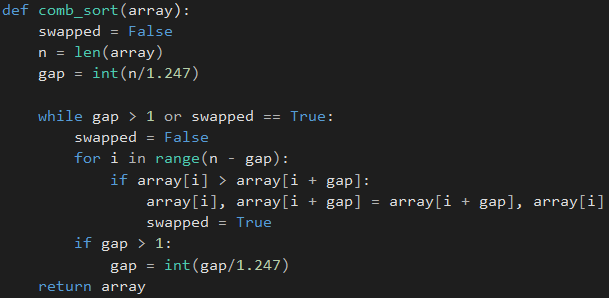
## Програмна реалізація алгоритму

### Вихідний код

**Бульбашка:**



**Гребінець:**



### Приклад роботи

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

**Бульбашка:**

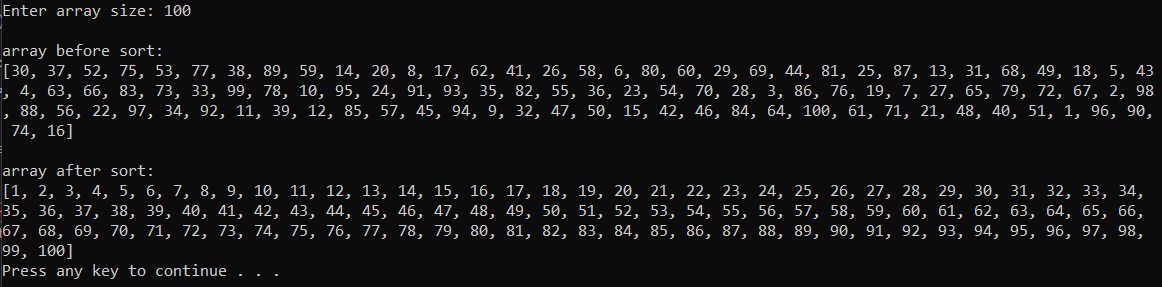


Рисунок 3.1 – Сортування масиву на 100 елементів

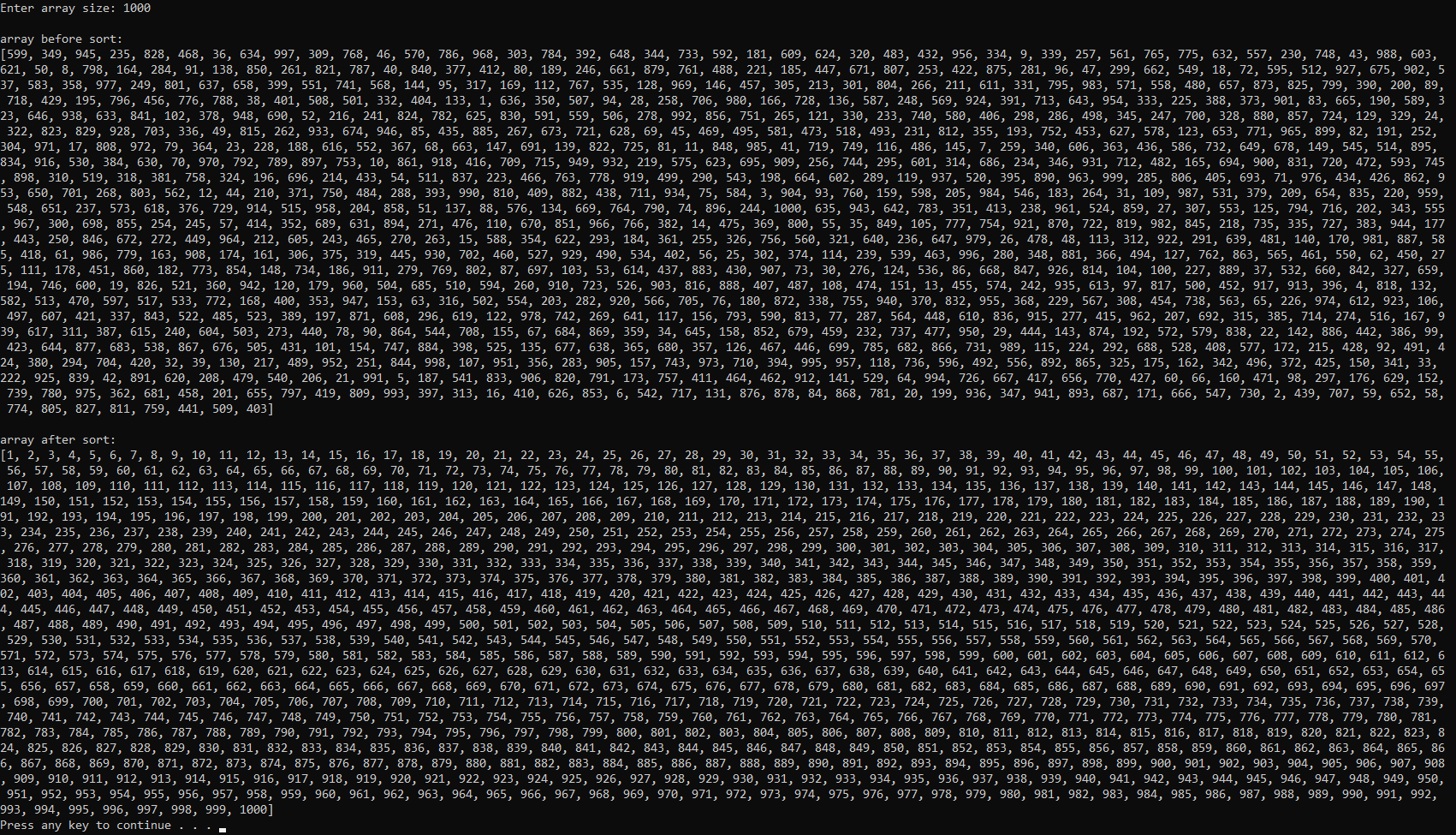


Рисунок 3.2 – Сортування масиву на 1000 елементів

**Гребінець:**

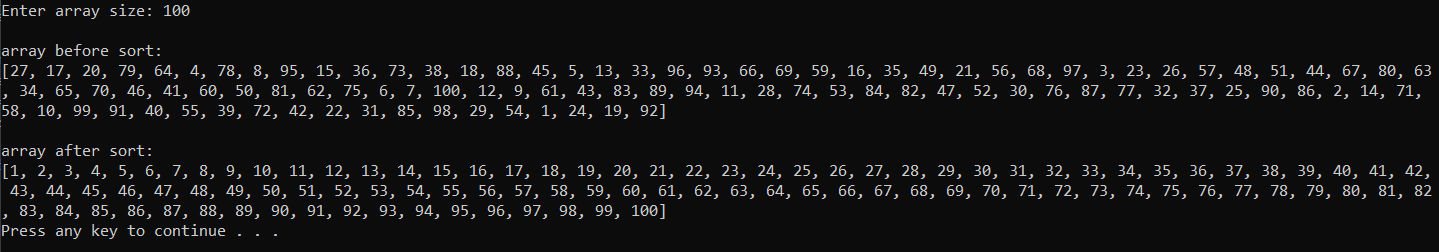


Рисунок 3.1 – Сортування масиву на 100 елементів

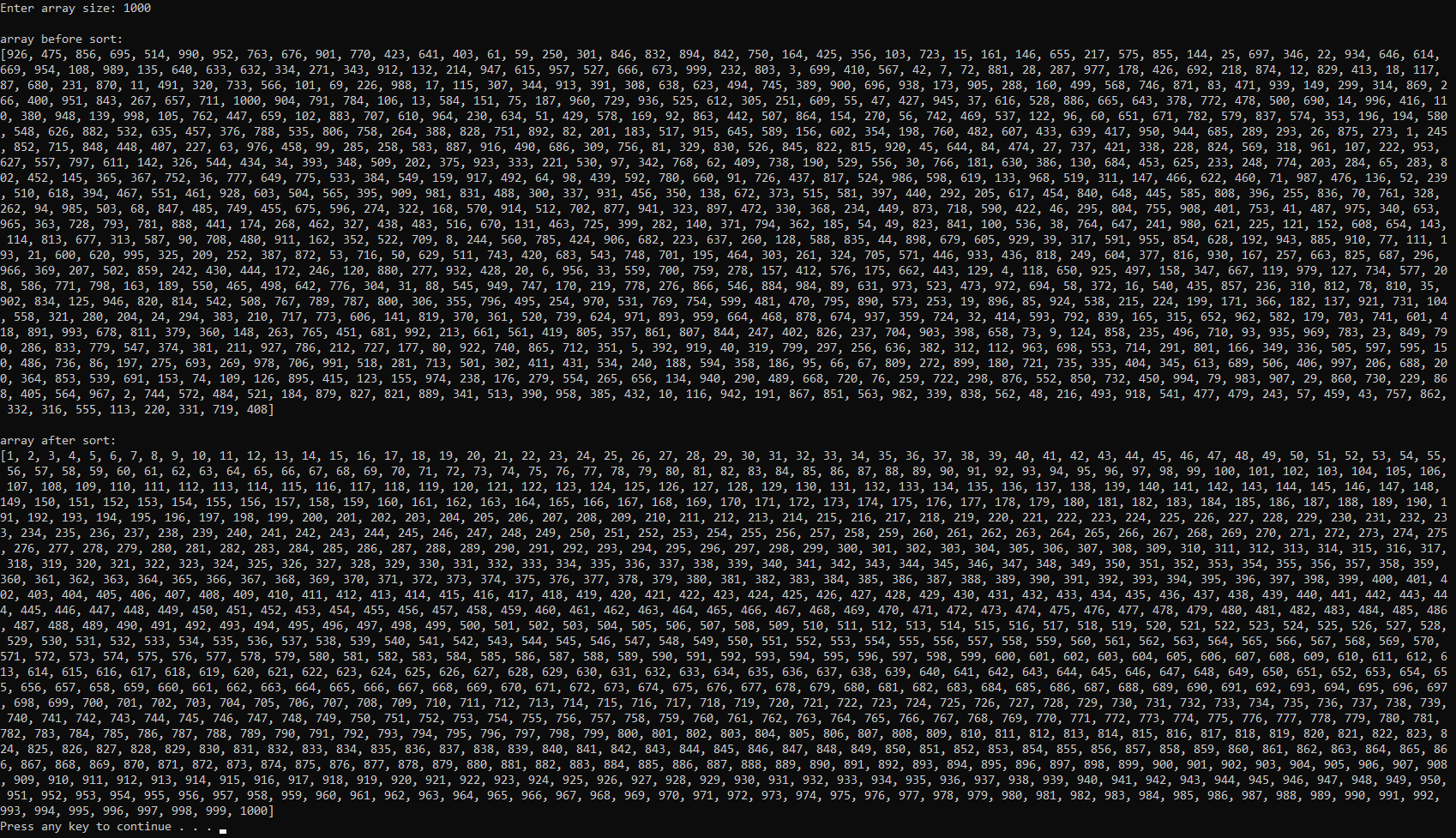


Рисунок 3.2 – Сортування масиву на 1000 елементів

## Тестування алгоритму

### 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

**Бульбашка:**

|  |  |  |
| --- | --- | --- |
| 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 |

**Гребінець:**

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 27 | 0 |
| 100 | 1130 | 0 |
| 1000 | 21023 | 0 |
| 5000 | 139833 | 0 |
| 10000 | 319599 | 0 |
| 20000 | 699137 | 0 |
| 50000 | 1947681 | 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.

**Бульбашка:**

|  |  |  |
| --- | --- | --- |
| 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 |

**Гребінець:**

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 27 | 4 |
| 100 | 1328 | 110 |
| 1000 | 23021 | 1512 |
| 5000 | 149831 | 9154 |
| 10000 | 339597 | 19018 |
| 20000 | 739135 | 40730 |
| 50000 | 2047679 | 110332 |

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.

**Бульбашка:**

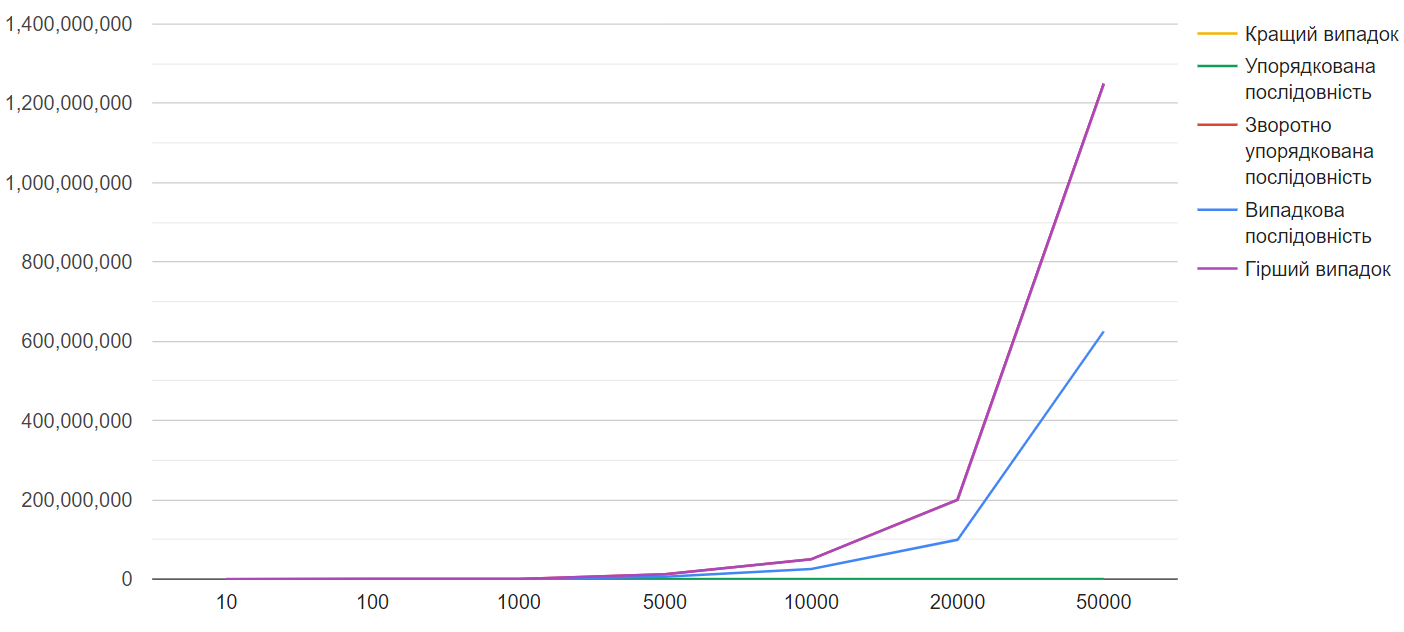
|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45 | 19 |
| 100 | 4950 | 2163 |
| 1000 | 499500 | 247381 |
| 5000 | 12497500 | 6274430 |
| 10000 | 49995000 | 25273598 |
| 20000 | 199990000 | 99070452 |
| 50000 | 1249975000 | 624969264 |

**Гребінець:**

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45 | 9 |
| 100 | 1328 | 221 |
| 1000 | 23021 | 4310 |
| 5000 | 154830 | 27257 |
| 10000 | 349596 | 59567 |
| 20000 | 759134 | 130997 |
| 50000 | 2097678 | 368272 |

### 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 для сортування бульбашкою

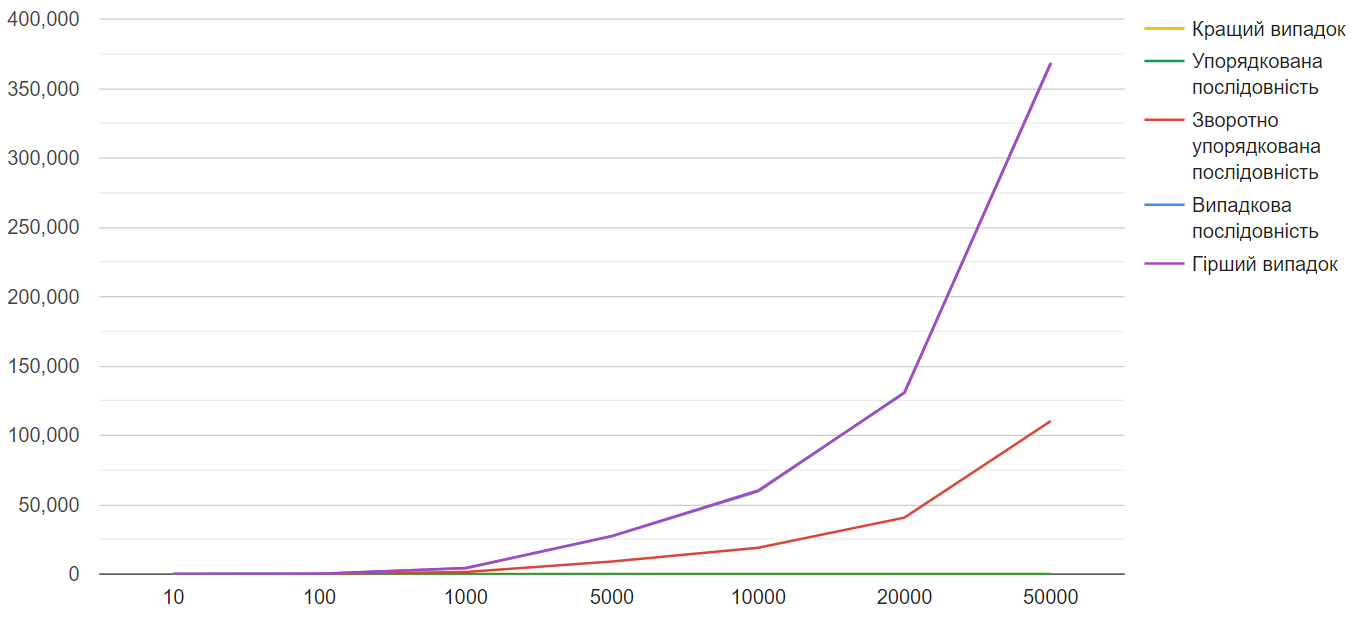


Рисунок 3.3 – Графіки залежності часових характеристик оцінювання для сортування гребінцем

Висновок

При виконанні даної лабораторної роботи було вивчено основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування(сортування бульбашкою та сортування гребінцем) і оцінено поріг їх ефективності.

Було проведено ряд випробувань алгоритмів 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 побудувано графіки залежності часових характеристик оцінювання від розмірності масиву.

Реультатом випробувань було те, шо combo sort більш ефективніший, особливо в роботі з масивами великої розмірності, що видно на графіках.