**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки**

**Кафедра ІСТ**

Звіт

з лабораторної роботи № 1 з дисципліни

«Теорія алгоритмів»

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

**Виконав(ла)**

ІА-34 Сухоручкін Гліб Андрійович   
ІА-34 Янович Марія Олександрівна  
(шифр, прізвище, ім'я, по батькові)

**Перевірив**

Халус Олена Андріївна

(прізвище, ім'я, по батькові)

Київ 2024

# ЗМІСТ

1. [**МЕТА ЛАБОРАТОРНОЇ РОБОТИ** **3**](#_gjdgxs)
2. [**ЗАВДАННЯ** **4**](#_30j0zll)
3. [**ВИКОНАННЯ** **7**](#_1fob9te)
   1. [Аналіз алгоритму на відповідність властивостям 7](#_3znysh7)
   2. [Псевдокод алгоритму 7](#_2et92p0)
   3. [Аналіз часової складності 7](#_26in1rg)
   4. [Програмна реалізація алгоритму 7](#_lnxbz9)
      1. [*Вихідний код* *7*](#_tyjcwt)
      2. [*Приклад роботи* *7*](#_3dy6vkm)
   5. [Тестування алгоритму 9](#_1t3h5sf)
      1. [*Часові характеристики оцінювання* *9*](#_4d34og8)
      2. [*Графіки залежності часових характеристик оцінювання від*](#_2s8eyo1)[*розмірності масиву* *11*](#_2s8eyo1)

[**ВИСНОВОК** **12**](#_17dp8vu)

[**КРИТЕРІЇ ОЦІНЮВАННЯ** **13**](#_3rdcrjn)

# 1 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 трьох алгоритмів Сортування бульбашкою, Модифікований алгоритм бульбашки та ВАШ АЛГОРИТМ ЗГІДНО ВАРІАНТУ.

Зробити узагальнений висновок з лабораторної роботи. Таблиця 2.1 – Варіанти алгоритмів

| **№** | **Алгоритм сортування** |
| --- | --- |
| **1.** | **Insertion Sort** |

| 2. | Сортування перемішуванням |
| --- | --- |
| 3. | Сортування гребінцем («розчіскою») |
| 4. | Сортування вибором |
| 5. | Сортування Шелла (класичне) |
| 6. | Сортування Шелла (*d* = послідовність Хіббарда) |
| 7. | Сортування Шелла (*d* = послідовність Пратта) |
| 8. | Сортування Шелла (*d* = послідовність Кнута) |
| 9. | Сортування Шелла (*d* = послідовність Седжвіка) |
| 10. | Сортування Шелла (*d* = послідовність Фібоначі) |
| 11. | Сортування Шелла (*d* = (3j−1)≤n, j ∈ N) |
| 12. | Сортування Шелла (*d* = послідовність Гоннета и Бєза-Єтс) |
| 13. | Сортування Шелла (*d* = послідовність Токуда) |
| 14. | Сортування Шелла (*d* = послідовність Марцина Циура) |
| 15. | **Insertion Sort** |
| 16. | Сортування перемішуванням |
| 17. | Сортування гребінцем («розчіскою») |
| 18. | Сортування вибором |
| 19. | Сортування Шелла (класичне) |
| 20. | Сортування Шелла (*d* = послідовність Хіббарда) |
| 21. | Сортування Шелла (*d* = послідовність Пратта) |
| 22. | Сортування Шелла (*d* = послідовність Кнута) |
| 23. | Сортування Шелла (*d* = послідовність Седжвіка) |
| 24. | Сортування Шелла (*d* = послідовність Фібоначі) |
| 25. | Сортування Шелла (*d* = (3j−1)≤n, j ∈ N) |
| 26. | Сортування Шелла (*d* = послідовність Гоннета и Бєза-Єтс) |
| 27. | Сортування Шелла (*d* = послідовність Токуда) |
| 28. | Сортування Шелла (*d* = послідовність Марцина Циура) |
| 29. | **Insertion Sort** |
| 30. | Сортування перемішуванням |

| 31. | Сортування гребінцем («розчіскою») |
| --- | --- |
| 32. | Сортування вибором |
| 33. | Сортування Шелла (класичне) |
| 34. | Сортування Шелла (*d* = послідовність Хіббарда) |
| 35. | Сортування Шелла (*d* = послідовність Пратта) |
| 36. | Сортування Шелла (*d* = послідовність Кнута) |
| 37. | Сортування Шелла (*d* = послідовність Седжвіка) |
| 38. | Сортування Шелла (*d* = послідовність Фібоначі) |
| 39. | Сортування Шелла (*d* = (3j−1)≤n, j ∈ N) |
| 40. | Сортування Шелла (*d* = послідовність Гоннета и Бєза-Єтс) |
| 41. | Сортування Шелла (*d* = послідовність Токуда) |
| 42. | Сортування Шелла (*d* = послідовність Марцина Циура) |

# ВИКОНАННЯ

* 1. Аналіз алгоритму на відповідність властивостям

Аналіз алгоритмів сортування бульбашкою, модифікованого сортування бульбашкою та сортування Шелла на відповідність властивостям наведено в таблиці 3.1.

Таблиця 3.1 – Аналіз алгоритму на відповідність властивостям

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

* 1. Псевдокод алгоритму сортування Шелла

procedure shellSort(arr: Array)

n := length(arr)

gap := n / 2

while gap > 0 do

for i := gap; i < n; i += 1 do

temp := arr[i]

j := i

while j >= gap and arr[j - gap] > temp do

arr[j] := arr[j - gap]

j -= gap

end while

arr[j] := temp

end for

gap := gap / 2

end while

end procedure

* 1. Аналіз часової складності сортування Шелла

Найкращий випадок: O(n log n)

Найгірший випадок: O(n ^ (3/2))

* 1. Програмна реалізація алгоритму
     1. Вихідний код

def shell\_sort\_knuth(arr):

jump = 0

l = len(arr)

o = 0

p = 1

while jump < l:

jump = jump \* 3 + 1

p += 1

jump = round(jump)

p += 1

while jump > 0:

for i in range(jump, l):

v = arr[i]

ii = i

p += 2

while ii > jump - 1 and arr[ii - jump] >= v:

arr[ii] = arr[ii - jump]

o += 1

ii = ii - jump

p += 2

arr[ii] = v

o += 1

jump = round((jump - 1)/3)

p += 1

return o, p

* + 1. Приклад роботи

На рисунках 3.1, 3.2 і 3.3 показані приклади роботи програми сортування трьох алгоритмів для масивів на 10, 100 і 1000 елементів відповідно.

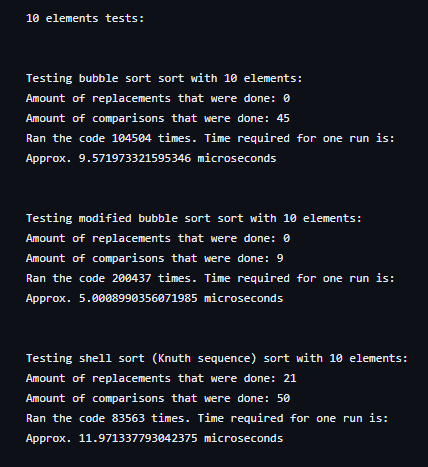


Рисунок 3.1 – Сортування масиву на 10 елементів

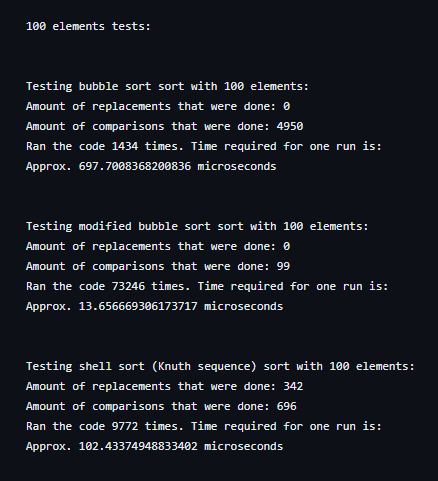


Рисунок 3.2 – Сортування масиву на 100 елементів

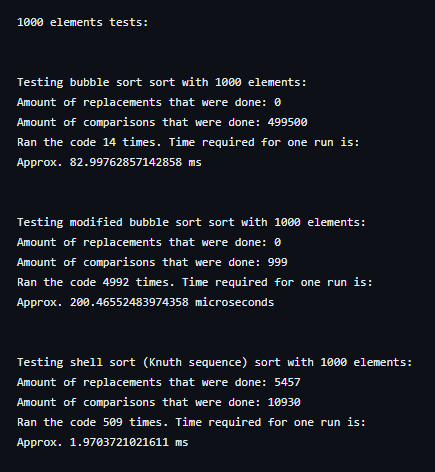


Рисунок 3.3 – Сортування масиву на 1000 елементів

* 1. Тестування алгоритму
     1. Часові характеристики оцінювання

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

Таблиця 3.2 – Характеристики оцінювання алгоритму сортування бульбашки, модифікованого алгоритму сортування бульбашки та сортування Шелла (з послідовністю Кнута) для упорядкованої послідовності елементів у масиві

| Розмірність масиву | Число порівнянь | Число перестановок |
| --- | --- | --- |
| 10 | **Сортування бульбашки:** 45  **Сортування бульбашки (модифіковано):** 9  **Сортування Шелла:** 50 | **Сортування бульбашки:** 0  **Сортування бульбашки (модифіковано):** 0  **Сортування Шелла:** 21 |
| 100 | **Сортування бульбашки:** 4950  **Сортування бульбашки (модифіковано):** 99  **Сортування Шелла:** 696 | **Сортування бульбашки:** 0  **Сортування бульбашки (модифіковано):** 0  **Сортування Шелла:** 342 |
| 1000 | **Сортування бульбашки:** 499500  **Сортування бульбашки (модифіковано):** 999  **Сортування Шелла:** 10930 | **Сортування бульбашки:** 0  **Сортування бульбашки (модифіковано):** 0  **Сортування Шелла:** 5457 |
| 5000 | **Сортування бульбашки:** 12497500  **Сортування бульбашки (модифіковано):** 4999  **Сортування Шелла:** 70192 | **Сортування бульбашки:** 0  **Сортування бульбашки (модифіковано):** 0  **Сортування Шелла:** 35086 |
| 10000 | **Сортування бульбашки:** 49995000  **Сортування бульбашки (модифіковано):** 9999  **Сортування Шелла:** 150508 | **Сортування бульбашки:** 0  **Сортування бульбашки (модифіковано):** 0  **Сортування Шелла:** 75243 |

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

Таблиця 3.3 – Характеристики оцінювання алгоритму сортування бульбашки, модифікованого алгоритму сортування бульбашки та сортування Шелла (з послідовністю Кнута) для зворотно упорядкованої послідовності елементів у масиві.

| Розмірність масиву | Число порівнянь | Число перестановок |
| --- | --- | --- |
| 10 | **Сортування бульбашки:** 45  **Сортування бульбашки (модифіковано):** 54  **Сортування Шелла:** 64 | **Сортування бульбашки:** 44  **Сортування бульбашки (модифіковано):** 45  **Сортування Шелла:** 28 |
| 100 | **Сортування бульбашки:** 4950  **Сортування бульбашки (модифіковано):** 5049  **Сортування Шелла:** 1156 | **Сортування бульбашки:** 4948  **Сортування бульбашки (модифіковано):** 4950  **Сортування Шелла:** 572 |
| 1000 | **Сортування бульбашки:** 499500  **Сортування бульбашки (модифіковано):** 500499  **Сортування Шелла:** 18770 | **Сортування бульбашки:** 499496  **Сортування бульбашки (модифіковано):** 499498  **Сортування Шелла:** 9377 |
| 5000 | **Сортування бульбашки:** 12497500  **Сортування бульбашки (модифіковано):** 12502499  **Сортування Шелла:** 108614 | **Сортування бульбашки:** 12497499  **Сортування бульбашки (модифіковано):** 12497499  **Сортування Шелла:** 54297 |
| 10000 | **Сортування бульбашки:** 49995000  **Сортування бульбашки (модифіковано):** 50004999  **Сортування Шелла:** 257916 | **Сортування бульбашки:** 49995000  **Сортування бульбашки (модифіковано):** 49994998  **Сортування Шелла:** 128947 |

У таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки, модифікованого алгоритму та сортування Шелла для масивів різної розмірності, коли масиви містять випадкову послідовність елементів.

Таблиця 3.4 – Характеристика оцінювання алгоритму сортування бульбашки, модифікованого алгоритму сортування бульбашки та сортування Шелла (з послідовністю Кнута) для випадкової послідовності елементів у масиві.

| Розмірність масиву | Число порівнянь | Число перестановок |
| --- | --- | --- |
| 10 | **Сортування бульбашки:** 45  **Сортування бульбашки (модифіковано):** 54  **Сортування Шелла:** 64 | **Сортування бульбашки:** 17  **Сортування бульбашки (модифіковано):** 25  **Сортування Шелла:** 28 |
| 100 | **Сортування бульбашки:** 4950  **Сортування бульбашки (модифіковано):** 4944  **Сортування Шелла:** 1512 | **Сортування бульбашки:** 2526  **Сортування бульбашки (модифіковано):** 2441  **Сортування Шелла:** 750 |
| 1000 | **Сортування бульбашки:** 499500  **Сортування бульбашки (модифіковано):** 500394  **Сортування Шелла:** 28090 | **Сортування бульбашки:** 249822  **Сортування бульбашки (модифіковано):** 251438  **Сортування Шелла:** 14037 |
| 5000 | **Сортування бульбашки:** 12497500  **Сортування бульбашки (модифіковано):** 12501904  **Сортування Шелла:** 209890 | **Сортування бульбашки:** 6243499  **Сортування бульбашки (модифіковано):** 6192392  **Сортування Шелла:** 104935 |
| 10000 | **Сортування бульбашки:** 49995000  **Сортування бульбашки (модифіковано):** 50002788  **Сортування Шелла:** 464676 | **Сортування бульбашки:** 24972683  **Сортування бульбашки (модифіковано):** 25318270  **Сортування Шелла:** 232327 |

* + 1. 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.

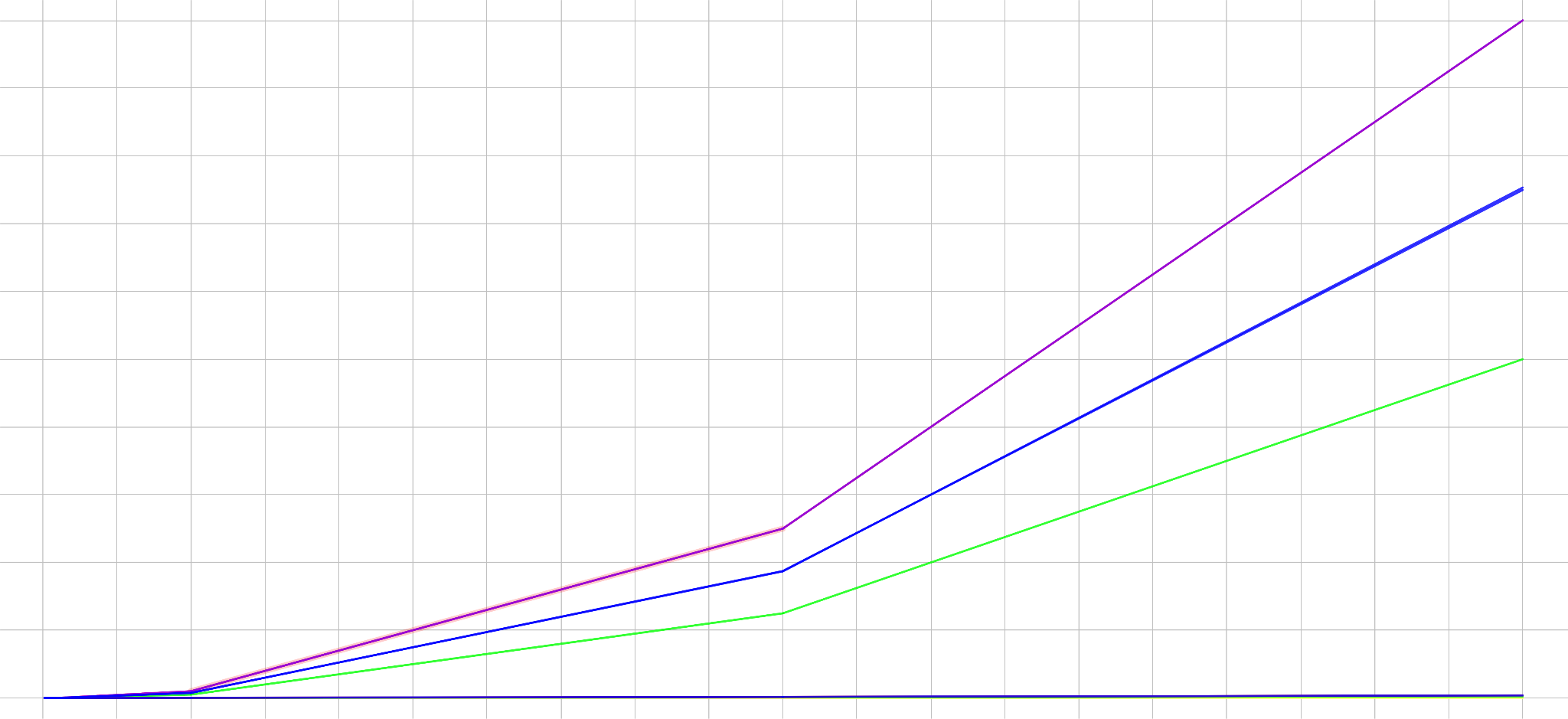


Рисунок 3.3 – Графіки залежності часових характеристик оцінювання

# ВИСНОВОК

При виконанні даної лабораторної роботи ми проаналізували запропоновані нам алгоритми. У ході аналізу було написано код, використовуючи мову програмування Python, для реалізації можливостей певних алгоритмів.

Після роботи з такими алгоритмами як: сортування бульбашкою, модифіковане сортування бульбашкою та сортування Шелла (з використанням послідовності Кнута), можемо зробити деякі висновки.

По-перше, алгоритм сортування бульбашкою є найменш ефективним серед протестованих нами.

По-друге, сортування Шелла – найбільш ефективний алгоритм: як у роботі з великою кількістю даних, так і у роботі з незначними її обсягами.

Отже, результати лабораторної роботи говорять нам про те, що в майбутньому, варто використовувати алгоритм Шелла. А також, аналізуючи те, що ми отримали в кінці, можна сказати, що алгоритм сортування бульбашкою також можна використовувати, але для маленької кількості вхідних даних, тому що його часова складність зростає квадратично. У той же час, дивлячись на псевдокод, можна зробити висновок про те, що це досить простий код, який не складно написати нашвидкуруч.

# КРИТЕРІЇ ОЦІНЮВАННЯ

У випадку здачі лабораторної роботи із захистом максимальний бал дорівнює – 6.

Критерії від максимального балу:

* аналіз алгоритму на відповідність властивостям ;
* псевдокод алгоритму ;
* аналіз часової складності ;
* програмна реалізація алгоритму ;
* тестування алгоритму ;
* висновок;
* захист роботи – 2 бали