**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-13 Бабашев Олексій Дмитрович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Сопов Олексій Олександрович*

Київ 2022

Зміст

[1 Мета лабораторної роботи 3](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772242)

[2 ЗаВдання 4](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772243)

[3 Виконання 7](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772244)

[3.1 Аналіз алгоритму на відповідність властивостям 7](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772245)

[3.2 Псевдокод алгоритму 7](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772246)

[3.3 Аналіз часової складності 7](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772247)

[3.4 Програмна реалізація алгоритму 7](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772248)

[3.4.1 Вихідний код 7](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772249)

[3.4.2 Приклад роботи 8](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772250)

[3.5 Тестування алгоритму 9](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772251)

[3.5.1 Часові характеристики оцінювання 9](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772252)

[3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву 11](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772253)

[Висновок 12](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_Toc69772254)

[Критерії оцінювання 13](file:///C:\Users\alexb\Downloads\Telegram%20Desktop\ASD_lr1_1курс.ukr%20(2).docx#_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 та сортування гребінцем на відповідність властивостям наведено в таблиці 3.1.

Таблиця 3.1 – Аналіз алгоритму на відповідність властивостям

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

|  |  |
| --- | --- |
| **Властивість** | **Сортування гребінцем** |
| Стійкість | так |
| «Природність» поведінки (Adaptability) | ні |
| Базуються на порівняннях | так |
| Необхідність в додатковій пам'яті (об'єм) | ні |
| Необхідність в знаннях про структури даних | так |

## Псевдокод алгоритму

Бульбашка:

size = len(arr)

**для** i **від** 0 **до** size -1 **з кроком** i++

**для** j **від** 0 **до** size - i - 1 **з кроком** j++

**якщо** arr[j] > arr[j + 1]

temp = arr[j]

arr[j] = arr[j+1]

arr[j+1] = temp

Гребінець:

step = int(len(arr) / 1.247)

swap = 1

**поки** True:

swap = 0

i = 0

**поки** i + step < len(arr):

**якщо** arr[i] > arr[i + step]

temp = arr[j]

arr[j] = arr[j+1]

arr[j+1] = temp

swap ++

i ++

**якщо** step == 1 && swap == 0

break

step = int(step / 1.247)

**якщо** step < 1

step = 1

## Аналіз часової складності

Бульбашка:

Кількість кроків у даному сортуванні залежить від кількості ітерації двох даних циклах(один вкладений в інший). Кількість ітерацій першого:

size-1, другого: size -1-i, (size-1)\*(size -1-i), звідси маємо часову складність O(n^2)

Гребінець:

Аналогічно має два вкладених цикли кількість ітерацій в яких напряму залежить від складності алгоритму, звідси мажмо часову складність O(n^2).

## Програмна реалізація алгоритму

### Вихідний код

main.py

*from* module *import*\*  
size = *int*(*input*("size = "))  
arr = initArr(size)  
  
*print*(arr)  
sortedArr = sortArray(arr)  
*print*(sortedArr)

module.py

*from* random *import* randint  
  
*def* BUBLE(arr):  
 size = *len*(arr)  
 swap = 0  
 comp = 0  
 *for* i *in range*(size - 1):  
 *for* j *in range*(size - i - 1):  
 comp+=1  
 *if* arr[j] > arr[j + 1]:  
 arr[j],arr[j+1] = arr[j+1] , arr[j]  
 swap += 1  
 *print*("comp = ", comp)  
 *print*("swap = ", swap)  
  
*def* GREBINEC(arr):  
 step = *int*(*len*(arr) / 1.247)  
 swap = 1  
 swaps = 0  
 comp = 0  
 *while True*:  
 swap = 0  
 i = 0  
 *while* (i + step < *len*(arr)):  
 comp+=1  
 *if* (arr[i] > arr[i + step]):  
 arr[i], arr[i + step] = arr[i + step], arr[i]  
 swap += 1  
 swaps += 1  
 i = i + 1  
 *if* (step == 1 *and* swap == 0):  
 *break* step = *int*(step / 1.247)  
 *if* (step < 1):  
 step = 1  
 *print*("comp = ", comp)  
 *print*("swap = ", swaps)  
  
*def* sortArray(arr):  
 typeOfSort = *input*('BULB or GREBINEC: ')  
 *while*(typeOfSort != 'GREBINEC' *and* typeOfSort != 'BULB'):  
 typeOfSort = *input*('enter againe: ')  
 *if*(typeOfSort == 'BULB'):  
 BUBLE(arr)  
 *elif*(typeOfSort == 'GREBINEC'):  
 GREBINEC(arr)  
 *return* arr  
  
*def* initArr(size):  
  
 typeOfArr = *input*('BEST WORST RANDOM: ')  
 *while*(typeOfArr != 'BEST' *and* typeOfArr != 'WORST' *and* typeOfArr != 'RANDOM'):  
 typeOfArr = *input*('enter againe: ')  
  
 *if*(typeOfArr == "RANDOM"):  
 arr = [randint(0, size) *for* i *in range*(size)]  
 *elif*(typeOfArr == "BEST"):  
 arr = [i *for* i *in range*(size)]  
 *elif* (typeOfArr == "WORST"):  
 arr = [size - 1 - i *for* i *in range*(size)]  
 *return* arr

### 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

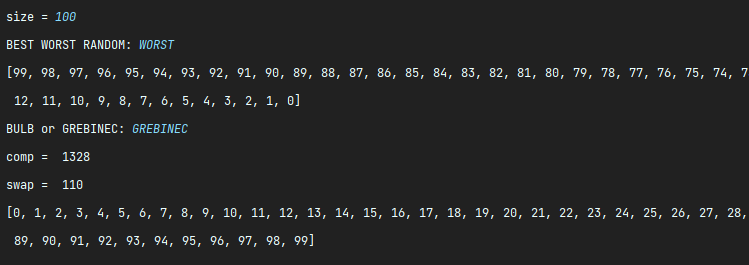


Рисунок 3.2 – Сортування масиву на 1000 елементів

## Тестування алгоритму

### Часові характеристики оцінювання

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

Таблиця 3.2 – Характеристики оцінювання алгоритму сортування бульбашки та гребінця відповідно 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 | 36 | 0 |
| 100 | 1229 | 0 |
| 1000 | 22022 | 0 |
| 5000 | 144832 | 0 |
| 10000 | 329598 | 0 |
| 20000 | 719136 | 0 |
| 50000 | 1997680 | 0 |

В таблиці 3.3 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки та гребінця для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів. Таблиця 3.3 – Характеристики оцінювання алгоритму сортування бульбашки та гребінця 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 | 45 | 9 |
| 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 та гребінця для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4 – Характеристика оцінювання алгоритму сортування бульбашки та гребінця для випадкової послідовності елементів у масиві.

Бульбашка:

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45 | 23 |
| 100 | 4950 | 2425 |
| 1000 | 499500 | 248048 |
| 5000 | 12497500 | 6235997 |
| 10000 | 49995000 | 24979795 |
| 20000 | 199990000 | 100033327 |
| 50000 | 1249975000 | 625701029 |

Гребінець:

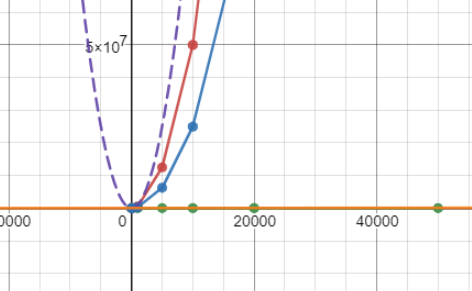
|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 36 | 5 |
| 100 | 1328 | 223 |
| 1000 | 23021 | 3948 |
| 5000 | 154830 | 25789 |
| 10000 | 349596 | 56715 |
| 20000 | 759134 | 124246 |
| 50000 | 2097678 | 351105 |

### Графіки залежності часових характеристик оцінювання від розмірності масиву

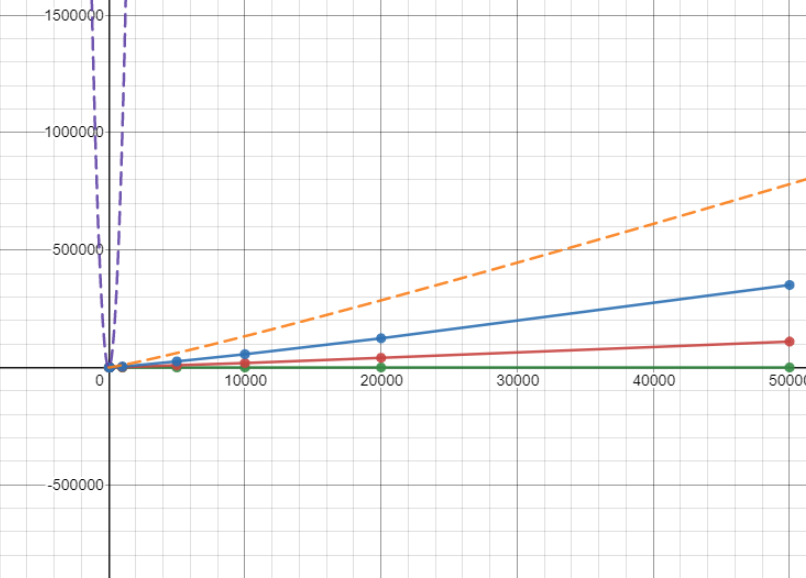
На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для випадків, коли масиви містять упорядковану послідовність елементів (зелений графік), коли масиви містять зворотно упорядковану послідовність елементів (червоний графік), коли масиви містять випадкову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

Бульбашка:



Гребінець:



Висновок

Вивчив основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування і оцінити поріг їх ефективності.

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%.