**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-14 Шляхтун Денис Михайлович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Соколовський Владислав Володимирович*

Київ 2022

Зміст

[1 Мета лабораторної роботи 3](#_Toc69772242)

[2 ЗаВдання 4](#_Toc69772243)

[3 Виконання 5](#_Toc69772244)

[3.1 Аналіз алгоритму на відповідність властивостям 5](#_Toc69772245)

[3.2 Псевдокод алгоритму 5](#_Toc69772246)

[3.3 Аналіз часової складності 6](#_Toc69772247)

[3.4 Програмна реалізація алгоритму 8](#_Toc69772248)

[3.4.1 Вихідний код 8](#_Toc69772249)

[3.4.2 Приклад роботи 9](#_Toc69772250)

[3.5 Тестування алгоритму 12](#_Toc69772251)

[3.5.1 Часові характеристики оцінювання 12](#_Toc69772252)

[3.5.2 Графіки залежності часових характеристик оцінювання від розмірності масиву 15](#_Toc69772253)

[Висновок 17](#_Toc69772254)

[Критерії оцінювання 18](#_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 | + | - |
| «Природність» поведінки (Adaptability) | - | + |
| Базуються на порівняннях | + | + |
| Необхідність в додатковій пам'яті (об'єм) | O(1), не залежить від кількості елементів | O(1) , не залежить від кількості елементів |
| Необхідність в знаннях про структури даних | - | - |

## Псевдокод алгоритму

bubbleSort(A)

1 **for** i ← 0 to length(A) - 1

2 **do for** j ← 0 to length(A) - i - 2

3 **do if** A[j] > A[j+1]

4 swap(A[j], A[j+1])

bubbleSortOptim(A)

1 i ← 0

2 flag ← True

3 **while** flag

4 **do** swapped ← False

5 **for** j ← 0 to length(A) - i - 2

6 **do if** A[j] > A[j+1]

7 swap(A[j], A[j+1])

8 swapped ← True

9 i = i + 1

10 **if** i == length or not swapped

11 flag ← False

combSort(A)

1 gap ← length ← length(A)

2 swapped ← True

3 **while** gap > 1 or swapped

4 **do** gap ← max(1, roundFloor(gap / 1.3))

5 **if** gap > 8 and gap < 11

6 gap ← 11

7 swapped ← False

8 **for** i ← 0 to length - gap - 1

9 **do** **if** A[i] > A[i + gap]

10 swap(A[i], A[i + gap])

11 swapped ← True

## Аналіз часової складності

bubbleSort(A) //найгірший випадок

1 n

2 (n^2 - n)/2

3 (n^2 - n)/2

4 (n^2 - n)/2

О(n^2)

bubbleSort(A) //найкращий випадок

1 n

2 (n^2 - n)/2

3 (n^2 - n)/2

4 0

О(n^2)

bubbleSortOptim(A) //найкращий випадок

1 1

2 1

3 2

4 1

5 n-1

6 n-1

7 0

8 0

9 1

10 1

11 1

О(n)

combSort(A) //найгірший випадок

1 1

2 1

3 n + log n + 1

4 n + log n

5 n + log n

6 1

7 n + log n

8 n + log n

9 n^2 + n log n

10 n^2 + n log n

11 n^2 + n log n

O(n^2)

combSort(A) //найкращий випадок

1 1

2 1

3 log n + 1

4 log n

5 log n

6 1

7 log n

8 n \* log n

9 n \* log n

10 0

11 0

О(n \* log n)

## Програмна реалізація алгоритму

### Вихідний код

Зображення, що містить текст

Автоматично згенерований опис

Зображення, що містить текст

Автоматично згенерований опис

Зображення, що містить текст

Автоматично згенерований опис

### Приклад роботи

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

Зображення, що містить текст

Автоматично згенерований опис

Рисунок 3.1 – Сортування масиву на 100 елементів

Зображення, що містить текст

Автоматично згенерований опис

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 та її оптимізованої форми для упорядкованої послідовності елементів у масиві

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45/9 | 0 |
| 100 | 4950/99 | 0 |
| 1000 | 499500/999 | 0 |
| 5000 | 12497500/4999 | 0 |
| 10000 | 49995000/9999 | 0 |
| 20000 | 199990000/19999 | 0 |
| 50000 | 1249975000/49999 | 0 |

В таблиці 3.3 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.3 – Характеристики оцінювання алгоритму сортування гребінцем для упорядкованої послідовності елементів у масиві

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 32 | 0 |
| 100 | 1096 | 0 |
| 1000 | 19706 | 0 |
| 5000 | 128379 | 0 |
| 10000 | 286729 | 0 |
| 20000 | 613402 | 0 |
| 50000 | 1733405 | 0 |

В таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

Таблиця 3.4 – Характеристики оцінювання алгоритму сортування бульбашки для зворотно упорядкованої послідовності елементів у масиві.

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

В таблиці 3.5 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

Таблиця 3.5 – Характеристики оцінювання алгоритму сортування гребінцем для зворотно упорядкованої послідовності елементів у масиві.

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 41 | 7 |
| 100 | 1195 | 116 |
| 1000 | 20705 | 1536 |
| 5000 | 133378 | 9308 |
| 10000 | 296728 | 20094 |
| 20000 | 633401 | 42634 |
| 50000 | 1783404 | 115816 |

У таблиці 3.6 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування бульбашки для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.6 – Характеристика оцінювання алгоритму сортування бульбашки для випадкової послідовності елементів у масиві.

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 45/44 | 18 |
| 100 | 4950/4949 | 2627 |
| 1000 | 499500/498720 | 256160 |
| 5000 | 12497500/12492550 | 6292215 |
| 10000 | 49995000/49991514 | 24932685 |
| 20000 | 199990000/199854019 | 99511593 |
| 50000 | 1249975000/1249968784 | 626073215 |

У таблиці 3.7 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму сортування гребінцем для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.7 – Характеристика оцінювання алгоритму сортування гребінцем для випадкової послідовності елементів у масиві.

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 41 | 8 |
| 100 | 1195 | 267 |
| 1000 | 21704 | 4374 |
| 5000 | 143376 | 28521 |
| 10000 | 306727 | 62179 |
| 20000 | 673399 | 126573 |
| 50000 | 1883402 | 379991 |

### Графіки залежності часових характеристик оцінювання від розмірності масиву

На рисунку 3.3 показані графіки залежності часових характеристик оцінювання від розмірності масиву для сортування бульбашкою. Дані для

Рисунок 3.3 – Графіки залежності часових характеристик оцінювання

На рисунку 3.4 показані графіки залежності часових характеристик оцінювання від розмірності масиву для сортування гребінцем.

Рисунок 3.4 – Графіки залежності часових характеристик оцінювання

Висновок

При виконанні даної лабораторної роботи було досліджено два алгоритми сортування – бульбашкою та гребінцем. Було досліджено властивості алгоритмів – стійкість, природність, базування на порівняннях, необхідність в додатковій пам’яті та структур. На основі псевдокоду було математично проаналізовано часову складність алгоритмів та визначено асимптотичну складність. Потім алгоритми сортування були реалізовані програмно та було проведено тестування для різної кількості елементів, упорядкованих по зростанню, по спаданню, та неупорядкованих. На основі тестування та асимптотичної складності було побудовано графіки.

В результаті роботи було порівняно два алгоритми сортування. Як наслідок, можна констатувати, що сортування гребінцем має значні переваги над сортуванням бульбашкою, зокрема природність поведінки та нижча асимптотична складність. Також було практично доведено, що кількість операцій, що здійснюється для упорядкування сортуванням гребінцем, значно нижча у порівнянні з бульбашкою.

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%.