**Міністерство освіти і науки України**

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

**Факультет інформатики та обчислювальної техніки**

**Кафедра автоматизованих систем обробки інформації**

**і управління**

**Звіт**

з лабораторної роботи № 6 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

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

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

Київ 2021

Зміст

1. Мета лабораторної роботи 3
2. ЗаВдання 4
3. Виконання 7
   1. Аналіз алгоритму на відповідність властивостям 7
   2. Псевдокод алгоритму 7
   3. Аналіз часової складності 7
   4. Програмна реалізація алгоритму 8
      1. Вихідний код 8
      2. Приклад роботи 9
   5. Тестування алгоритму 10
      1. Часові характеристики оцінювання 10
      2. Графіки залежності часових характеристик оцінювання від розмірності масиву 12

Висновок 14

Критерії оцінювання 15

# Мета лабораторної роботи

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

# ЗаВдання

Згідно варіанту (таблиця 2.1), виконати аналіз алгоритму внутрішнього сортування на відповідність наступним властивостям:

* 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 узагальнений висновок з лабораторної роботи.

Таблиця 2.1 – Варіанти алгоритмів

|  |  |
| --- | --- |
| **№** | **Алгоритм сортування** |
| 1 | Сортування перемішуванням |
| 2 | Сортування гребінцем («розчіскою») |
| 3 | Сортування вибором |
| 4 | Сортування Шелла (класичне) |
| 5 | Сортування Шелла (*d* = послідовність Хіббарда) |
| 6 | Сортування Шелла (*d* = послідовність Пратта) |
| 7 | Сортування Шелла (*d* = послідовність Кнута) |
| 8 | Сортування Шелла (*d* = послідовність Седжвіка) |
| 9 | Сортування Шелла (*d* = послідовність Фібоначі) |
| 10 | Сортування Шелла (*d* = (3j−1)≤n, j ∈ N) |
| 11 | Сортування Шелла (*d* = послідовність Гоннета и Бєза-Єтс) |
| 12 | Сортування Шелла (*d* = послідовність Токуда) |
| 13 | Сортування Шелла (*d* = послідовність Марцина Циура) |
| 14 | Швидке сортування (розбиття Ломуто) |
| 15 | Швидке сортування (розбиття Хоара) |
| 16 | Швидке сортування (медіана із трьох) |
| 17 | Швидке сортування (елементи, що повторюються) |
| 18 | Пірамідальне сортування |
| 19 | Плавне сортування |
| 20 | Інтроспективне сортування |
| 21 | Stdsort |
| 22 | Сортування за допомогою двійкового дерева |
| 23 | Швидке сортування (медіана із трьох) |
| 24 | Сортування Шелла (*d* = послідовність Хіббарда) |
| 25 | Сортування Шелла (*d* = послідовність Пратта) |
| 26 | Сортування Шелла (*d* = послідовність Кнута) |
| 27 | Сортування Шелла (*d* = послідовність Седжвіка) |
| 28 | Сортування Шелла (*d* = послідовність Фібоначі) |
| 29 | Пірамідальне сортування |
| 30 | Плавне сортування |
| 31 | Інтроспективне сортування |
| 32 | Stdsort |
| 33 | Швидке сортування (розбиття Ломуто) |
| 34 | Швидке сортування (розбиття Хоара) |
| 35 | Швидке сортування (елементи, що повторюються) |

# Виконання

## Аналіз алгоритму на відповідність властивостям

Аналіз алгоритму Сортування Шелла (*d* = послідовність Кнута) на відповідність властивостям наведено в таблиці 3.1.

Таблиця 3.1 – Аналіз алгоритму на відповідність властивостям

|  |  |
| --- | --- |
| **Властивість** | Сортування Шелла (*d* = послідовність Кнута) |
| Стійкість | 5/5 |
| «Природність» поведінки (Adaptability) | 3/5 |
| Базуються на порівняннях | 5/5 |
| Необхідність в додатковій пам'яті (об'єм) | О(*n*) всего, O(*1*) вспомогательный |
| Необхідність в знаннях про структури даних | 5/5 |

## Псевдокод алгоритму

## Аналіз часової складності

## Програмна реалізація алгоритму

### Вихідний код

import java.util.\*;

public class ShellSort {

public static void main(String[] args) {

int[] arr = new int[50000];

for (int i = 0; i < arr.length; i++) {

arr[i] = i;

}

System.out.println("Input array- " + Arrays.toString(arr));

long a = System.currentTimeMillis();

int[] sortedArray = shellSort(arr);

long b = System.currentTimeMillis();

System.out.println(b - a);

System.out.println("Sorted array after shell sort- " + Arrays.toString(sortedArray));

}

private static int[] shellSort(int[] arr){

int interval = 1;

int temp;

int num\_por = 0;

int num\_per = 0;

while(interval <= arr.length/3){

interval = (interval \* 3) + 1;

}

while(interval > 0){

for(int i = interval; i < arr.length; i++){

num\_por++;

temp = arr[i];

int j;

for(j = i; j > interval - 1 && arr[j-interval] >= temp; j=j-interval){

arr[j] = arr[j - interval];

num\_per++;

}

arr[j] = temp;

}

interval = (interval - 1)/3;

}

System.out.println("Por : " + num\_per);

System.out.println("Shaf : " +num\_por);

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 Шелла (*d* = послідовність Кнута) для масивів різної розмірності, коли масив містить упорядковану послідовність елементів.

Таблиця 3.2 – Характеристики оцінювання Сортування Шелла (*d* = послідовність Кнута) для упорядкованої послідовності елементів у масиві

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 15 | 0 |
| 100 | 342 | 0 |
| 1000 | 5457 | 0 |
| 5000 | 35084 | 0 |
| 10000 | 75243 | 0 |
| 20000 | 165243 | 0 |
| 50000 | 455719 | 0 |

В таблиці 3.3 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму Сортування Шелла (*d* = послідовність Кнута) для масивів різної розмірності, коли масиви містять зворотно упорядковану послідовність елементів.

Таблиця 3.3 – Характеристики оцінювання Сортування Шелла (*d* = послідовність Кнута) для зворотно упорядкованої послідовності елементів у масиві.

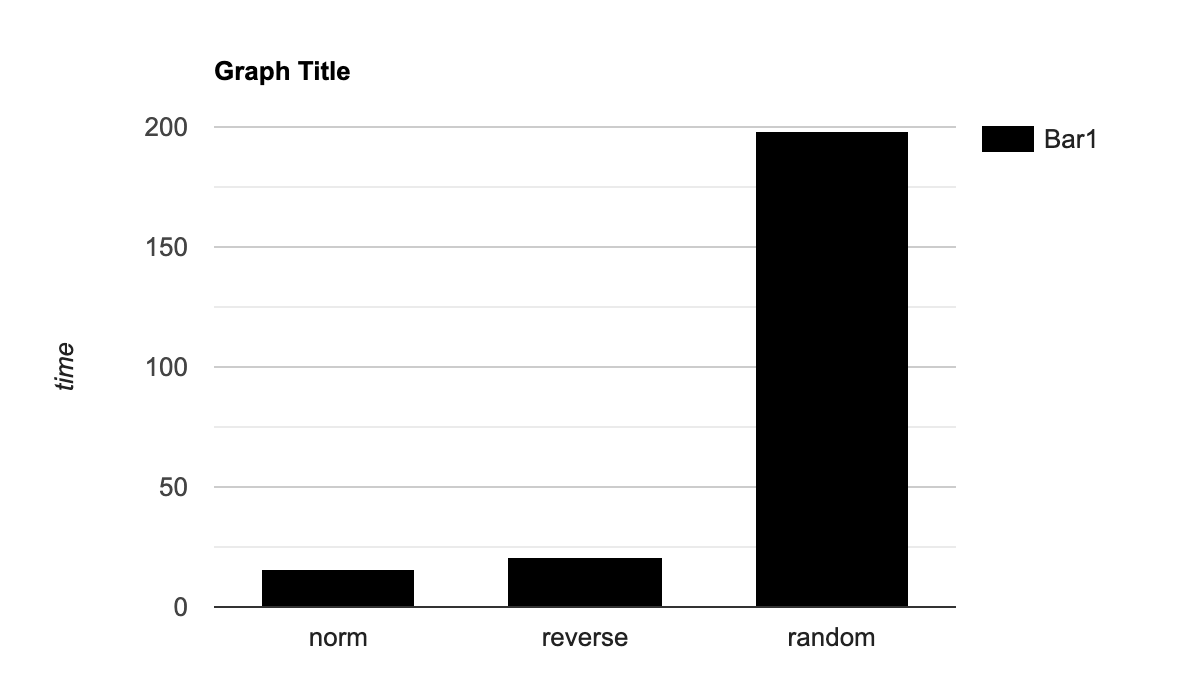
|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 15 | 13 |
| 100 | 342 | 230 |
| 1000 | 5457 | 3920 |
| 5000 | 35084 | 19212 |
| 10000 | 75243 | 53704 |
| 20000 | 165243 | 113812 |
| 50000 | 455719 | 270768 |

У таблиці 3.4 наведені характеристики оцінювання числа порівнянь і числа перестановок алгоритму Сортування Шелла (*d* = послідовність Кнута) для масивів різної розмірності, масиви містять випадкову послідовність елементів.

Таблиця 3.4 – Характеристика оцінювання Сортування Шелла (*d* = послідовність Кнута) для випадкової послідовності елементів у масиві.

|  |  |  |
| --- | --- | --- |
| Розмірність масиву | Число порівнянь | Число перестановок |
| 10 | 17 | 15 |
| 100 | 534 | 342 |
| 1000 | 22088 | 5457 |
| 5000 | 448430 | 35084 |
| 10000 | 1730934 | 75243 |
| 20000 | 6799287 | 165243 |
| 50000 | 41928209 | 455719 |

### Графіки залежності часових характеристик оцінювання від розмірності масиву

Рисунок 3.3 – Графіки залежності часових характеристик оцінювання

Висновок

При виконанні даної лабораторної роботи ми протестували алгоритм сортування шелла побудували таблицi часових залежностей та дослідили залежності перестановок та обмінів. А також вивчили основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування і оцінили поріг їх ефективності.

Критерії оцінювання

У випадку здачі лабораторної роботи до 18.05.2020 включно максимальний бал дорівнює – 5. Після 18.05.2020 максимальний бал дорівнює – 1.

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%.