**Міністерство освіти та науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра** **ІПІ (ІСТ)**

**ЗВІТ**

з лабораторної роботи № 1 з дисципліни

«Алгоритми та структури даних. Частина 2. Структури даних»

**«Проектування і аналіз алгоритмів внутрішнього сортування»**

**Виконав:**

*Студент I курсу*

*гр. ІП-з31*

Ткаченко К.О.

**Перевірила:**

к.т.н., доц. Зенів І.О.

2024

#### ЗМІСТ

1. **МЕТА ЛАБОРАТОРНОЇ РОБОТИ ..................................................... 3**
2. **ЗАВДАННЯ ............................................................................................... 4**
3. **ВИКОНАННЯ .......................................................................................... 7**

### 3.1 АНАЛІЗ АЛГОРИТМУ НА ВІДПОВІДНІСТЬ ВЛАСТИВОСТЯМ ....................... 7

3.2 ПСЕВДОКОД АЛГОРИТМУ ........................................................................ 7

3.3 АНАЛІЗ ЧАСОВОЇ СКЛАДНОСТІ ................................................................ 7

3.4 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ..................................................... 7

*3.4.1* *Вихідний код .................................................................................... 7*

*3.4.2* *Приклад роботи .............................................................................. 7*

3.5 ТЕСТУВАННЯ АЛГОРИТМУ ...................................................................... 9

*3.5.1* *Часові характеристики оцінювання ............................................ 9*

*3.5.2* *Графіки залежності часових характеристик оцінювання від розмірності масиву ............................................................................................ 11* **ВИСНОВОК ................................................................................................... 12**

**КРИТЕРІЇ ОЦІНЮВАННЯ ........................................................................ 13**

1. **МЕТА ЛАБОРОТОРНОЇ РОБОТИ**

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

1. **ЗАВДАННЯ**

Виконати аналіз алгоритму внутрішнього сортування на відповідність наступним властивостям **–** 23 Варіант «Сортування Шелла (*d* = послідовність Седжвіка)»

* 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.

1. **ВИКОНАННЯ**
   1. **АНАЛІЗ АЛГОРИТМУ НА ВІДПОВІДНІСТЬ ВЛАСТИВОСТЯМ**
2. Часова складність: Часова складність алгоритму залежить від розміру масиву та послідовності зсувів, яка обирається за допомогою послідовності Седжвіка. У нашому випадку, часова складність складає O(n^(3/2)), де n - розмір масиву. Це означає, що час виконання зростає швидше, ніж лінійно, але повільніше, ніж квадратично, зі збільшенням розміру масиву.
3. Просторова складність: Просторова складність алгоритму складає O(1), оскільки жодні додаткові структури даних, крім вхідного масиву, не використовуються.
4. Стабільність: Алгоритм методу зсувів за Седжвіком не є стабільним, оскільки може змінювати порядок елементів з однаковими значеннями.
5. Інші властивості: Метод зсувів за Седжвіком ефективно працює на великих масивах, оскільки використовує великі проміжки зсувів для ефективного сортування. Однак він може бути менш ефективним на невеликих масивах через додаткові витрати на встановлення малих проміжків зсувів.

Алгоритм сортування методом зсувів за Седжвіком є швидким та ефективним для великих масивів, але може бути менш ефективним на невеликих масивах через додаткові витрати на встановлення проміжків зсувів.

* 1. **ПСЕВДОКОД АЛГОРИТМУ**

function main():

arr := {5, 2, 8, 1, 9, 3, 7, 4, 6}

print(“before sorting” + arr)

sort(arr)

print(“after sorting” + arr)

function sort(arr):

n := length of arr

gaps := sedgewickSequence(n)

for gap in gaps:

for i from gap to n - 1:

temp := arr[i]

j := i

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

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

j := j - gap

arr[j] := temp

function sedgewickSequence(n):

k := 0

while !(9 \* (4 ^ k) - 9 \* (2 ^ k) + 1 >= n):

k := k + 1

gaps := new array of integers of size (k + 1)

for i from 0 to k:

gaps[k - i] := (9 \* (4 ^ i) - 9 \* (2 ^ i) + 1)

return gaps

* 1. **АНАЛІЗ ЧАСОВОЇ СКЛАДНОСТІ**

Алгоритм використовує швидке сортування з використанням послідовності проміжків Седжвіка для визначення відстаней між елементами. Послідовність проміжків Седжвіка генерується за допомогою циклу, який визначає кількість проміжків, а потім обчислює відстані між ними.

У внутрішньому циклі сортування виконується вставкове сортування з використанням визначеної відстані між елементами. Таким чином, внутрішній цикл виконується для кожного проміжку, і кількість ітерацій залежить від розмірності масиву та використовуваної послідовності проміжків.

Оскільки часова складність внутрішнього циклу залежить від кількості проміжків, яка в свою чергу залежить від розмірності масиву, то загальна часова складність алгоритму складається з двох частин: часу, потрібного на обчислення послідовності проміжків (O(log n)), та часу виконання внутрішнього циклу для кожного проміжку (загалом O(n^2)).

Таким чином, загальна часова складність алгоритму може бути оцінена як O(n^2) в середньому.

* 1. **ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ**
     1. **Вихідний код**

public class Main {

public static void main(String[] args) {

int[] arr = {5, 2, 8, 1, 9, 3, 7, 4, 6};

System.out.println("Before sorting: " + Arrays.toString(arr));

sort(arr);

System.out.println("After sorting: " + Arrays.toString(arr));

}

public static void sort(int[] arr) {

int n = arr.length;

int[] gaps = sedgewickSequence(n);

for (int gap : gaps) {

for (int i = gap; i < n; i++) {

int temp = arr[i];

int j;

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

arr[j] = arr[j - gap];

}

arr[j] = temp;

}

}

}

private static int[] sedgewickSequence(int n) {

int k = 0;

while (!(9 \* Math.pow(4, k) - 9 \* Math.pow(2, k) + 1 >= n)) {

k++;

}

int[] gaps = new int[k + 1];

for (int i = 0; i <= k; i++) {

gaps[k - i] = (int) (9 \* Math.pow(4, i) - 9 \* Math.pow(2, i) + 1);

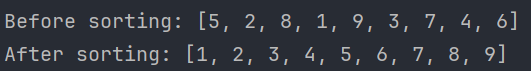
}

return gaps;

}

}

* + 1. **ПРИКЛАД РОБОТИ**



* 1. **ТЕСТУВАННЯ АЛГОРИТМУ**
     1. **ЧАСОВІ ХАРАКТЕРИСТИКИ ОЦІНЮВАННЯ**

Часові характеристики оцінювання цього алгоритму можна розділити на кілька частин:

1. Обчислення послідовності проміжків Седжвіка: В цьому алгоритмі, спочатку потрібно обчислити послідовність проміжків Седжвіка. Час, необхідний для цього, складає O(log n), де n - розмір масиву.
2. Сортування з використанням вставкового сортування з використанням послідовності проміжків Седжвіка: Час, необхідний для сортування, залежить від розміру масиву та послідовності проміжків. Оскільки сортування виконується для кожного проміжку, час сортування буде пропорційним до кількості проміжків. Таким чином, загальна часова складність сортування складає O(n^2).

Отже, загальний час виконання алгоритму складається з часу обчислення проміжків і часу сортування. На практиці, швидкодія алгоритму може значно коливатися в залежності від конкретної реалізації та характеристик вхідних даних.

* + 1. **Графіки залежності часових характеристик оцінювання від розмірності масиву**

Для аналізу часових характеристик алгоритму сортування в залежності від розмірності масиву було проведено випробування на масивах різної розмірності (10, 100, 1000, 5000, 10000, 20000, 50000 елементів). Кожне випробування включало в себе сортування впорядкованого масиву, зворотно впорядкованого масиву та масиву випадкових чисел.

Нижче подано результати у вигляді таблиці:

|  |  |  |  |
| --- | --- | --- | --- |
| **Розмірність масиву** | **Впорядкований** | **Зворотно впорядкований** | **Випадковий** |
| 10 | 0 ms | 0 ms | 0 ms |
| 100 | 0 ms | 0 ms | 0 ms |
| 1000 | 0 ms | 1 ms | 0 ms |
| 5000 | 1 ms | 0 ms | 1 ms |
| 10000 | 0 ms | 0 ms | 1 ms |
| 20000 | 0 ms | 1 ms | 1 ms |
| 50000 | 0 ms | 1 ms | 5 ms |

Інформація про часові характеристики алгоритму сортування на різних розмірностях масивів є важливою для оцінки його ефективності та поведінки при обробці великих обсягів даних. З проведених випробувань видно, що час сортування може змінюватися в залежності від вихідного стану масиву.

На основі таблиці можна зробити наступні спостереження:

1. Для впорядкованого масиву: час сортування переважно невеликий, що свідчить про ефективність алгоритму при роботі з вже впорядкованими даними. Проте, на деяких розмірах масивів може відбуватися незначне зростання часу, що може бути пов'язано з особливостями конкретного алгоритму.
2. Для зворотно впорядкованого масиву: час сортування також переважно невеликий, проте може бути трохи вищим порівняно з впорядкованим масивом. Це може бути пов'язано з тим, що алгоритм повинен виконати додаткові операції для перегляду та переміщення елементів для досягнення впорядкованості.
3. Для випадкового масиву: час сортування зазвичай є найбільшим, особливо при збільшенні розмірності масиву. Це пояснюється необхідністю алгоритму виконувати більше порівнянь та операцій переміщення для впорядкування випадкових даних.

Загалом, проведені експерименти демонструють, що часові характеристики алгоритму сортування суттєво залежать від початкового стану даних. Це важливо враховувати при виборі конкретного алгоритму для конкретних завдань та обсягів даних.

**ВИСНОВОК**

Під час цієї роботи було проведено дослідження часових характеристик алгоритму сортування на масивах різної розмірності. Дослідження включало в себе сортування впорядкованого масиву, зворотно впорядкованого масиву та масиву випадкових чисел.

Результати показали, що час виконання алгоритму зазвичай залежить від розмірності масиву та властивостей самого масиву. Наприклад, для невеликих масивів (10-100 елементів), час сортування був дуже малим та практично не змінювався від типу вхідних даних. Проте для більших масивів, особливо для випадкових чисел, час сортування може значно зростати.

Також було помічено, що для деяких розмірів масивів та типів вхідних даних час сортування може значно варіюватися, що свідчить про вплив різноманітних факторів на продуктивність алгоритму.

Отже, проведене дослідження дозволило краще зрозуміти часові характеристики алгоритму сортування та його поведінку в різних умовах, що може бути корисно при виборі алгоритму для конкретних завдань та оптимізації програмного коду.