Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни
«Проектування алгоритмів»

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

Виконав(ла) ІП-15, Лазьов Кирило Владиславович
(шифр, прізвище, ім'я, по батькові)

Перевірив *Головченко М.М.*
(прізвище, ім'я, по батькові)

Київ 2022

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

Мета роботи – вивчити основні алгоритми зовнішнього сортування та способи їх модифікації, оцінити поріг їх ефективності.

2 ЗАВДАННЯ

Згідно варіанту (таблиця 2.1), розробити та записати алгоритм зовнішнього сортування за допомогою псевдокоду (чи іншого способу за вибором).

Виконати програмну реалізацію алгоритму на будь-якій мові програмування та відсортувати випадковим чином згенерований масив цілих чисел, що зберігається у файлі (розмір файлу має бути не менше 10 Мб, можна значно більше).

Здійснити модифікацію програми і відсортувати випадковим чином згенерований масив цілих чисел, що зберігається у файлі розміром не менше ніж двократний обсяг ОП вашого ПК. Досягти швидкості сортування з розрахунку 1Гб на 3хв. або менше.

Рекомендується попередньо впорядкувати серії елементів довжиною, що займає не менше 100Мб або використати інші підходи для пришвидшення процесу сортування.

Зробити узагальнений висновок з лабораторної роботи, у якому порівняти базову та модифіковану програми. У висновку деталізувати, які саме модифікації було виконано і який ефект вони дали.

Таблиця 2.1 – Варіанти алгоритмів

№	**Алгоритм сортування**
1	Пряме злиття
2	Природне (адаптивне) злиття
3	Збалансоване багатошляхове злиття
4	Багатофазне сортування
5	Пряме злиття
6	Природне (адаптивне) злиття
7	Збалансоване багатошляхове злиття
8	Багатофазне сортування
9	Пряме злиття

10	Природне (адаптивне) злиття

11	Збалансоване багатошляхове злиття
12	Багатофазне сортування
13	Пряме злиття
14	Природне (адаптивне) злиття
15	Збалансоване багатошляхове злиття
16	Багатофазне сортування
17	Пряме злиття
18	Природне (адаптивне) злиття
19	Збалансоване багатошляхове злиття
20	Багатофазне сортування
21	Пряме злиття
22	Природне (адаптивне) злиття
23	Збалансоване багатошляхове злиття
24	Багатофазне сортування
25	Пряме злиття
26	Природне (адаптивне) злиття
27	Збалансоване багатошляхове злиття
28	Багатофазне сортування
29	Пряме злиття
30	Природне (адаптивне) злиття
31	Збалансоване багатошляхове злиття
32	Багатофазне сортування

33	Пряме злиття
34	Природне (адаптивне) злиття
35	Збалансоване багатошляхове злиття

3 ВИКОНАННЯ

3.1 Псевдокод алгоритму

Функція find_series:

Якщо len(number) == 1:

return [number]

Все якщо

all_series = []

this_series = []

Повторити для i від 1 до len(numbers):

Якщо number[i - 1] <= number[i]:

this_series.append(number[i - 1])

Інакше:

this_series.append(number[i - 1])

all_series.append(this_series)

this_series = []

Якщо i == len(number) - 1:

this_series.append(number[i])

all_series.append(this_series)

this_series = []

Все якщо

Все повторити

Функція series_to_files:

file_b = open("files/file_b.txt", "w")

file_c = open("files/file_c.txt", "w")

Повторити для i від 0 до len(series):

string_to_write = ""

Повторити для j від 0 до len(series[i]):

#Уникнення від пробілів у кінці файлу

#Якщо i та j - останні індекси

#або i-передостанній, а j останній індекси

Якщо (i == len(series)-1 та j == len(series[i]) - 1) або (i == len(series) - 2 та j == len(series[i]) - 1):

string_to_write += f"{series[i][j]}"

Інакше:

string_to_write += f"{series[i][j]} "

Все якщо

Все повторити

Якщо i % 2 == 0:

file_b.write(string_to_write)

Інакше:

file_c.write(string_to_write)

Все якщо

Все повторити

Функція merge_files():

numbers_b = convert_to_int(file_b)

numbers_c = convert_to_int(file_c)

series_b = find_series(numbers_b)

series_c = find_series(numbers_c)

clear_file()

Якщо(len(series_b) >= len(series_c)):

longest_series = series_b

Інакше:

longest_series = series_c

Все якщо

Якщо (len(series_b) < len(series_c)):

shortest_series = series_b

Інакше:

shortest_series = series_c

Все якщо

Повторити для і від 0 до len(longest series):

longest_series_elements = longest_series[i]

#Якщо і менше за довжину кортших серій

Якщо i < len(shortest_series):

shortest_series_elements = shortest_series[i]

Інакше:

shortest_series_elements = []

Все якщо

joined_list = longest_series_elements + shortest_series_elements

joined_list.sort()

with open("files/file_a.txt", "a") as file_a:

#Запис числе у файл

file_a.write(" ".join([str(i) **для і в** joined_list]))

#Якщо i!=len(longest_series) - 1: ставимо пробіл

file_a.write(" " **Якщо** i != len(longest_series) - 1 **Інакше** "")

Все повторити

3.2 Програмна реалізація алгоритму

3.2.1 Вихідний код

main.py

```
import functions

functions.generate_files()

counter = 1

while not functions.file_is_sorted():
    with open("files/file_a.txt") as file_a:
        data = file_a.read()

    print(f"Ітерація #{counter}...")

    numbers = functions.convert to int(data)

    series = functions.find_series(numbers)

    functions.series_to_files(series)

    functions.merge_files()

    counter += 1

print("Файл відсотован")
```

functions.py

```
def generate_files():
    import random
    with open("files/file_a.txt", "w") as file_a:
        nums = []
        for i in range(1_800_000):
            num = random.randint(0, 2_000_000)
            nums.append(num)
        file_a.write(" ".join([str(i) for i in nums]))

        print("Файл згенерован")

def convert_to_int(data) -> list[int]:
    numbers = [int(num) for num in data.split(' ')]

    return numbers

def find_series(numbers: list[int]) -> list:
    if len(numbers) == 1:
        return [numbers]

    #Усі серії
    all_series = []
    #Поточна
    this_series = []

    for i in range(1, len(numbers)):
        #Попоредне число менше
        if numbers[i - 1] <= numbers[i]:
            this_series.append(numbers[i - 1])
        else:
            #Додаємо число у поточну серію
            this_series.append(numbers[i - 1])
            #Поточну серію додоаємо до усіх
            all_series.append(this_series)
            #Обнуляємо поточну
            this_series = []

        #Якщо ітерована остання серія
        if i == len(numbers) - 1:
            this_series.append(numbers[i])
            all_series.append(this_series)
            this_series = []

    return all_series

def series_to_files(series: list):
    file_b = open("files/file_b.txt", "w")
    file_c = open("files/file_c.txt", "w")

    for i in range(len(series)):
        string_to_write = ""

        for j in range(len(series[i])):
            #Уникнення від пробілів у кінці файлу
            #Якщо i та j - останні індекси
            #або i-передостанній, а j останній індекси
            if (i == len(series)-1 and j == len(series[i]) - 1) or (i ==
len(series) - 2 and j == len(series[i]) - 1):
                string_to_write += f"{series[i][j]}"
            else:
                string_to_write += f"{series[i][j]} "
```

```
        if i % 2 == 0:
            file_b.write(string_to_write)
        else:
            file_c.write(string_to_write)

    file_b.close()
    file_c.close()

def clear_file():
    file_a = open("files/file_a.txt", "w")
    file_a.write("")
    file_a.close()

def merge_files():
    with open("files/file_b.txt") as b:
        file_b = b.read()

    with open("files/file_c.txt") as c:
        file_c = c.read()

    numbers_b = convert_to_int(file_b)
    numbers_c = convert_to_int(file_c)

    series_b = find_series(numbers_b)
    series_c = find_series(numbers_c)

    clear_file()

    if(len(series_b) >= len(series_c)):
        longest_series = series_b
    else:
        longest_series = series_c

    if (len(series_b) < len(series_c)):
        shortest_series = series_b
    else:
        shortest_series = series_c

    for i in range(len(longest_series)):
        longest_series_elements = longest_series[i]

        #Якщо і менше за довжину кортших серій
        if i < len(shortest_series):
            shortest_series_elements = shortest_series[i]
        else:
            shortest_series_elements = []

        joined_list = longest_series_elements + shortest_series_elements
        joined_list.sort()

        with open("files/file_a.txt", "a") as file_a:
            #Запис числе у файл
            file_a.write(" ".join([str(i) for i in joined_list]))
            #Якщо i!=len(longest_series) - 1: ставимо пробіл
            file_a.write(" " if i != len(longest_series) - 1 else "")

def file_is_sorted() -> bool:
    with open("files/file_a.txt") as file_a:
        data = file_a.read()

    numbers = convert_to_int(data)

    for i in range(1, len(numbers)):
```

```
        if numbers[i - 1]>numbers[i]:
            return False

    return True
```

ВИСНОВОК

При виконанні даної лабораторної роботи я отримав практичні навички роботи з алгоритмами сортування. Виконав написання псевдокоду і програмну реалізацію свого варіанта – природне адаптивне злиття.

Природне адаптивне злиття потребує від двох файлів і має часову складність O(n). Під час тестувань файл 10 мб був відсортован за 4 хвилини.

КРИТЕРІЇ ОЦІНЮВАННЯ

У випадку здачі лабораторної роботи до 09.10.2022 включно максимальний бал дорівнює – 5. Після 09.10.2022 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

- псевдокод алгоритму – 15%;
- програмна реалізація алгоритму – 40%; програмна реалізація модифікацій – 40%;
- висновок – 5%.