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-22 Іщенко Кіра Віталіївна*
(шифр, прізвище, ім'я, по батькові)

Перевірив *Ахаладзе І.Е.*
(прізвище, ім'я, по батькові)

0

Київ 2023

ЗМІСТ

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. Достатньо штучно обмежити доступну ОП, для уникнення багатогодинних сортувань (наприклад використовуючи віртуальну машину).

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

Алгоритм generateFile(degree):

Відкрити файл A.bin та start.bin для запису
num = 0
створити вектор
sizeGroup = 2^degree
Для кожного і від 0 до 9:
n = 0
Очистити вміст вектору
Поки n < sizeGroup:
Згенерувати випадкове число num від 0 до 199999
Додати num до вектору
Збільшити n на 1
Відсортувати вектор
Для кожного nums в групі:
Записати nums в файли A.bin та start.bin
Вивести повідомлення "File A was created"
Закрити файли

Алгоритм сортування:

поточне_значення_2_в_степені=0
цикл, поки (numberCount / 2 >= поточне_значення_2_в_степені)
поточне_значення_2_в_степені = 2^степінь
виклик_функції writeInBC(поточне_значення_2_в_степені)
виклик_функції mergeInA(поточне_значення_2_в_степені)
степінь++

Алгоритм writeInBC(sizeOfGroup):

Відкрити файл A.bin для читання в бінарному режимі і файл B.bin та C.bin для запису в бінарному режимі
Поки не досягнуто кінця файлу A.bin:
Прочитати sizeOfGroup цілих чисел і записати їх в файл B.bin
Прочитати ще sizeOfGroup цілих чисел і записати їх в файл C.bin
Закрити всі файли

Алгоритм mergeInA(sizeOfGroup):

Відкрити файл A.bin для запису в бінарному режимі і файли B.bin та C.bin для читання в бінарному режимі

b = 0;

c = 0;

Прочитати число numberB з файлу B.bin

Прочитати число numberC з файлу C.bin

Поки файли B.bin та C.bin не закінчились:

Поки b < sizeOfGroup і c < sizeOfGroup і файл C.bin не закінчився:

Якщо numberB < numberC:

Записати numberB в файл A.bin

Прочитати наступне число numberB з файлу B.bin

Збільшити b на 1

Якщо numberB > numberC:

Записати numberC в файл A.bin

Прочитати наступне число numberC з файлу C.bin

Збільшити c на 1

Якщо numberB == numberC:

Записати numberB та numberC в файл A.bin

Прочитати наступні числа numberB та numberC з файлів B.bin та C.bin

Збільшити b та c на 1

Поки b не досягнув sizeOfGroup і файл C.bin не закінчився:

Записати numberB в файл A.bin

Прочитати наступне число numberB з файлу B.bin

Збільшити b на 1

Поки c не досягнув sizeOfGroup і файл B.bin не закінчився:

Записати numberC в файл A.bin

Прочитати наступне число numberC з файлу C.bin

Збільшити c на 1

Скинути значення b та c на 0

Поки файл B.bin не закінчився:

Записати numberB в файл A.bin

Прочитати наступне число numberB з файлу B.bin

Закрити всі файли

3.2 Програмна реалізація алгоритму

3.2.1 Вихідний код

```
#include <iostream>
#include <iterator>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;
void generateFile(int degree) {
    ofstream aWrite("A.bin");
    ofstream sWrite("start.bin");
    int num = 0;
    int sizeGroup = pow(2, degree);


    for (int i = 0; i < 10; i++) {
        int n = 0;
        group.clear();
        while (n < sizeGroup) {
            num = rand() % 200000;
            group.push_back(num);
            n++;
        }
        sort(group.begin(), group.end());
        for (int nums : group) {
            aWrite << nums << " ";
            sWrite << nums << " ";
        }
    }
    cout << "File A was create" << endl;
    aWrite.close();
    sWrite.close();}


void  writeInBC(int sizeOfGroup) {
    ifstream aRead("A.bin", ios::binary);
    ofstream bWrite(firstHelpFile, ios::binary), cWrite(secondHelpFile, ios::binary);
    int t;


    if (sizeOfGroup < 0) {
```

```
        throw invalid_argument("sizeOfGroup can`t be less then 0");
    }
    int b = 0;
    int c = 0;
    while (aRead.peek() != EOF) {
        for (int i = 0; i != sizeOfGroup && (aRead >> t); i++) {
            bWrite << t << " ";
        }
        for (int i = 0; i != sizeOfGroup && (aRead >> t); i++) {
            cWrite << t << " ";
        }
    }
    aRead.close();
    bWrite.close();
    cWrite.close();
}
void  mergeInA(int sizeOfGroup) {
    ofstream aWrite("A.bin", ios::binary);
    ifstream bRead(firstHelpFile, ios::binary), cRead(secondHelpFile, ios::binary);
    int numberC, numberB;
    if (sizeOfGroup < 0) {
        throw invalid_argument("sizeOfGroup can`t be less then 0");
    }
    bRead >> numberB;
    cRead >> numberC;
    while (!bRead.eof() && !cRead.eof()) {
        while (b < sizeOfGroup && c < sizeOfGroup && !cRead.eof()) {
            if (numberB < numberC) {
                aWrite << numberB << " ";
                bRead >> numberB;
                b++;
            }
            else if (numberB > numberC) {
                aWrite << numberC << " ";
                cRead >> numberC;
                c++;
            }
            else {
```

```
                aWrite << numberB << " " << numberC << " ";
                bRead >> numberB;
                cRead >> numberC;
                b++;
                c++;
            }
        }
        while (b != sizeOfGroup && !cRead.eof()) {
            aWrite << numberB << " ";
            bRead >> numberB;
            b++;
        }
        while (c != sizeOfGroup && !cRead.eof()) {
            aWrite << numberC << " ";
            cRead >> numberC;
            c++;
        }
        c = b = 0;
    }
    while (!bRead.eof()) {
        aWrite << numberB << " ";
        bRead >> numberB;
    }

  aWrite.close();
  bRead.close();
  cRead.close();
}
```

ВИСНОВОК

При виконанні даної лабораторної роботи я ознайомилась з алгоритмом зовнішнього сортування - пряме злиття. При базовій реалізації розмір груп на які ми розбивали стартував з 1, тобто перебирав кожну цифру. Файл B містив числа з непарним порядковим номером, а C з парним. Це значно сповільнювало роботу алгоритму для великих файлів.

Моя модифікація полягала у тому, аби при генерації розбити файл на 10 частин і відсортувати всередині кожну з них, використовуючи вектор. Це дозволило починати злиття з цих груп і уникати перших н кроків. В результаті чого робота алгоритму з 30 хв для 1 гігабайту скоротилась до 2-3 хв.

КРИТЕРІЇ ОЦІНЮВАННЯ

У випадку здачі лабораторної роботи до 08.10.2022 включно максимальний бал дорівнює – 5. Після 08.10.2022 максимальний бал дорівнює – 4,5.

Критерії оцінювання у відсотках від максимального балу:

- псевдокод алгоритму – 15%;
- програмна реалізація алгоритму – 20%;
- програмна реалізація модифікацій – 20%;
- робота з git – 40%;
- висновок – 5%.