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-13 Бондаренко М.В.* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Сопов О.О.* _____
(прізвище, ім'я, по батькові)

Київ 2022

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.

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

1. Створити список, що зберігає назву вхідного файлу.
2. Створити списки, що зберігають назви допоміжних файлів.
3. Створити пусті файли B та C для подальшого використання.
4. Розподілити дані основного файлу по допоміжним.
5. Ініціалізувати змінну flag = 1 для перемикання злиття файлів.
6. ПОКИ нема відсортованого файлу
   6.1. ЯКЩО flag = 1
      6.1.1. Злити B файли в C.
   6.2. ІНАКШЕ
      6.2.1. Злити C файли в B.
   6.3. ВСЕ ЯКЩО
7. ВСЕ ПОКИ

Алгоритм злиття:

1. Ініціалізувати список sequence для збереження послідовностей.
2. Ініціалізувати списки потоків зчитування та запису файлів.
3. Ініціалізувати змінну j = 0 для переміщення між файлами для запису.
4. ПОКИ не дійшли до кінця всіх файлів з даними
   4.1. Ініціалізувати змінну min_number та min_index.
   4.2. ЦИКЛ i ВІД 0 ДО кількість файлів для зчитування
      4.2.1. Зчитати значення файлу (number).
      4.2.2. ЯКЩО number існує
         4.2.2.1. ЯКЩО sequence пуста АБО number >= останнє значення sequence
            4.2.2.1.1. ЯКЩО number < min_number
               4.2.2.1.1.1. min_number = number
               4.2.2.1.1.2. min_index = i

4.2.2.1.2. ВСЕ ЯКЩО

4.2.2.2. ВСЕ ЯКЩО

4.2.3. ВСЕ ЯКЩО

4.3. КІНЕЦЬ ЦИКЛУ

4.4. ЯКЩО min_index не змінився

4.4.1. ЦИКЛ number B sequence

4.4.1.1. Записати число в новий файл.

4.4.2. ВСЕ ЦИКЛ

4.4.3. Почистити sequence

4.4.4. j = (j + 1) % кількість файлів

4.5.ІНАКШЕ

4.5.1. Записати в sequence min_number

4.5.2. Зчитуємо наступні дані з заданого файлу для зчитування.

4.6.ВСЕ ЯКЩО

5. ВСЕ ПОКИ

3.2 Програмна реалізація алгоритму

3.2.1 Вихідний код

```
public static string BalancedKwayMerge(int files_amount)
{
   List<string> A_files_list = new List<string>();
   A_files_list.Add("A1.dat");
   List<string> B_files_list = CreateFilesLists(files_amount, 'B');
   List<string> C_files_list = CreateFilesLists(files_amount, 'C');
   GenerateFile();
   CreateFiles(files_amount);

   MultiwayMerge(A_files_list, B_files_list, files_amount);

   int flag = 1;
   while (!isSorted(A_files_list, B_files_list, C_files_list))
   {
      if (flag == 1)
         MultiwayMerge(B_files_list, C_files_list, files_amount);
      else
         MultiwayMerge(C_files_list, B_files_list, files_amount);
```

```
        flag = -flag;
    }

    var A_size = new FileInfo(A_files_list[0]).Length;
    var B_size = new FileInfo(B_files_list[0]).Length;
    if (A_size == B_size)
        return B_files_list[0];
    return C_files_list[0];
}

private static Boolean isSorted(List<string> A_files_list, List<string>
B_files_list, List<string> C_files_list)
{
    var A_size = new FileInfo(A_files_list[0]).Length;
    var B_size = new FileInfo(B_files_list[0]).Length;
    var C_size = new FileInfo(C_files_list[0]).Length;
    return A_size == B_size || A_size == C_size;
}

private static void MultiwayMerge(List<string> files_to_divide, List<string>
divided_files, int amount_of_files)
{
    List<int> sequence = new List<int>();

    List<Reader> readers_list = new List<Reader>();
    foreach (var file in files_to_divide)
    {
        var reader = new Reader(file);
        readers_list.Add(reader);
    }

    List<BinaryWriter> writers_list = new List<BinaryWriter>();
    foreach (var file in divided_files)
    {
        BinaryWriter writer = new BinaryWriter(File.Open(file,
FileMode.Create));
        writers_list.Add(writer);
    }

    int j = 0; // index to move between files for writing

    while (!isMerged(readers_list))
    {
        int min_number = 150000;
        int min_index = -1;
```

```
        for (int i = 0; i < readers_list.Count; i++)
        {
            var bin_num = readers_list[i].GetCurrent();
            if (bin_num.Length > 0)
            {
                var int_num = BitConverter.ToInt32(bin_num, 0);
                if (sequence.Count == 0 || int_num >= sequence.Last())
                {
                    if (int_num <= min_number)
                    {
                        min_number = int_num;
                        min_index = i;
                    }
                }
            }
        }
        if (min_index == -1)
        {
            foreach (var number in sequence)
                writers_list[j].Write(BitConverter.GetBytes(number));
            sequence.Clear();
            j = (j + 1) % amount_of_files;
        }
        else
        {
            sequence.Add(min_number);
            readers_list[min_index].GetNext();
        }
    }

    foreach (var number in sequence)
        writers_list[j].Write(BitConverter.GetBytes(number));

    foreach (var reader in readers_list)
        reader.Close();

    foreach (var writer in writers_list)
        writer.Close();
}
private static Boolean isMerged(List<Reader> readers_list)
{
    foreach (var reader in readers_list)
    {
        if (reader.GetCurrent().Length != 0)
```

```
        return false;
    }
    return true;
}
```

ВИСНОВОК

При виконанні даної лабораторної роботи дослідив та реалізував алгоритм збалансованого багатошляхового злиття. Це є алгоритм зовнішнього сортування, що передбачений для роботи з великими файлами, але для досягнення нормальної швидкості роботи його треба модифікувати. Моя реалізація алгоритму виконує задачу сортування файла розміром 10 Мб за 8 секунд. Для розгляду різних ситуацій провів тести з діленням на 3, 10 та 50 файлів. З трьома файлами впорався за 8 секунд, з десятьма за той же час, з п’ятдесятьма - за 15 секунд. Звісно, за такого обсягу даних не раціонально використовувати алгоритми зовнішнього сортування, бо вони значно повільніше, але вони сильно потрібні для роботи з файлами, що більші за обсяги ОЗУ серверів.

Алгоритми зовнішнього сортування мають місце в природі та потрібні у деяких ситуаціях, але вони менш ефективні за алгоритми внутрішнього сортування.

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%.