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 № 7 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

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

Виконав(ла) *ІП-02 Гущін Д.О.* _____
(шифр, прізвище, ім'я, по батькові)

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

Київ 2021

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 методу за рахунок використання оперативної пам’яті і відсортувати випадковим чином згенерований масив цілих чисел, що зберігається у файлі розміром не менше 1Гб. Необхідно попередньо впорядкувати серії елементів довжиною, що займає не менше 100Мб оперативної пам’яті.

Зробити узагальнений висновок з лабораторної роботи, у якому порівняти основний та модифікований методи.

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

```
 for (int i = 0; i < n; i++)
    if (0 != new FileInfo(Directory.GetCurrentDirectory() + nameForReader + i +
".txt").Length)
    end if
    streamReaders[i] = new StreamReader(Directory.GetCurrentDirectory() + nameForReader + i
+ ".txt")
            StreamWriter[] streamWriters = new StreamWriter[n]
            for (int i = 0; i < n; i++)
                streamWriters[i] = new StreamWriter(Directory.GetCurrentDirectory() +
nameForWriter + i + ".txt")
                streamWriters[i].Write("")
            end for
            int[] series = new int[n]
            bool[] toRead = new bool[n]
            bool flag = true
            for (int i = 0; i < n; i++)
                toRead[i] = true;
                int firstNumber = ReadingNums(streamReaders[i])
                if (firstNumber != -1)
                    series[i] = firstNumber;
                end if
                else
                    toRead[i] = false;
               end else
            End for
            while (flag)
                for (int i = 0; i < n; i++)
                    while (true)
                        int index = Minimal(series, toRead);
                        if (index == -1)
                            break;
                         end if
                        int minElem = series[index];
                        streamWriters[i].Write(minElem + " ")
                        if (toRead[index])
                            int nextNumber = ReadingNums(streamReaders[index])
                            if (nextNumber < series[index])
                                toRead[index] = false;
                            End if
                            if (nextNumber != -1)
                                series[index] = nextNumber
                            end if
                        end if
                     End while
                End for
                    for (int j = 0; j < n; j++)
                        if (streamReaders[j] != null && !streamReaders[j].EndOfStream)
                            toRead[j] = true;
                        end if
                    end for
                flag = false;
     for (int i = 0; i < n; i++)
         if (streamReaders[i] != null && !streamReaders[i].EndOfStream)
                        flag = true
                        break
         end if
    End for
End while
 for (int i = 0; i < n; i++)
```

```
         if (new FileInfo(Directory.GetCurrentDirectory() + nameForWriter + i +
".txt").Length != 0)
                streamWriters[i].Write(" ")
                streamWriters[i].Close()
                if (streamReaders[i] != null)
                    streamReaders[i].Close()
                end if
        End if
End for
```

3.2 Програмна реалізація алгоритму

3.2.1 Вихідний код

```
using System;
using System.IO;

namespace Lab_7
{
    class Program
    {
        static void Main(string[] args)
        {
            FilesProcessing.SegmentsSorting();
            FilesProcessing.Parse(Directory.GetCurrentDirectory(), @"\res");
            while (true)
            {
                MultyWayMerge.FilesMerging(3, @"\res", @"\result");
                if (new FileInfo(Directory.GetCurrentDirectory() + @"\res" + "1" +
".txt").Length == 0 && new FileInfo(Directory.GetCurrentDirectory() + @"\result" + "2" +
".txt").Length == 0) break;
                MultyWayMerge.FilesMerging(3, @"\result", @"\res");
                if (new FileInfo(Directory.GetCurrentDirectory() + @"\res" + "1" +
".txt").Length == 0 && new FileInfo(Directory.GetCurrentDirectory() + @"\res" + "2" +
".txt").Length == 0) break;
            }
        }
    }
        }
namespace Lab_7
{
    public class FilesProcessing
    {
        public static void SegmentsSorting()
        {
            int length = 204800;
            List<int> chunk = new List<int>();
            int totalLength = 0;
            int index1 = 0;
            int index2 = 0;
            using StreamWriter writer = new StreamWriter(Directory.GetCurrentDirectory() +
@"\res" + ".txt");
            using StreamReader reader = new StreamReader(Directory.GetCurrentDirectory() +
@"\512Mb" + ".txt");
            string numberString;
            List<int> numberOfValue = new List<int>();
            while ((numberString = reader.ReadLine()) != null)
            {
                List<int> numbers = numberString.Split(' ').Where(x => x != "").Select(x =>
int.Parse(x)).ToList();
                numberOfValue.Add(numbers.Count);
                foreach (int number in numbers)
                    chunk.Add(number);
```

```
                totalLength++;
                if (totalLength % length == 0)
                {
                    chunk.Sort();
                    index1 = 0;
                    index2 = 0;
                    for (int i = 0; i < numberOfValue.Count; i++)
                    {
                        index2 += numberOfValue[i];
                        for (int j = index1; j < index2; j++)
                            writer.Write(chunk[j].ToString() + " ");
                        index1 += numberOfValue[i];
                        writer.WriteLine();
                    }
                    chunk.Clear();
                    numberOfValue = new List<int>();
                }
            }
            chunk.Sort();
            index1 = 0;
            index2 = 0;
            for (int i = 0; i < numberOfValue.Count; i++)
            {
                index2 += numberOfValue[i];
                for (int j = index1; j < index2; j++)
                    writer.Write(chunk[j].ToString() + " ");
                index1 += numberOfValue[i];
                writer.WriteLine();
            }
        }
        public static void Parse(string path, string name)
        {
            using StreamReader reader = new StreamReader(path + name + ".txt");
            using StreamWriter writer0 = new StreamWriter(path + name + "0" + ".txt");
            using StreamWriter writer1 = new StreamWriter(path + name + "1" + ".txt");
            using StreamWriter writer2 = new StreamWriter(path + name + "2" + ".txt");
            while (!reader.EndOfStream)
            {
                string a = reader.ReadLine();
                int temp = 0;
                int index = 0;
                int[] numbers = a.Split(' ').Where(x => x != "").Select(x =>
int.Parse(x)).ToArray();
                while (index < numbers.Length - 1)
                {
                    writer0.Write(numbers[index] + " ");
                    for (int i = index + 1; i < numbers.Length; i++)
                    {
                        if (numbers[i - 1] <= numbers[i])
                        {
                            writer0.Write(numbers[i] + " ");
                            if (i == numbers.Length - 1)
                            {
                                index = i;
                            }
                        }
                        else
                        {
                            temp = numbers[i];
                            index = i;
                            break;
                        }
                    }
                    writer1.Write(temp + " ");
```

```
                    for (int i = index + 1; i < numbers.Length; i++)
                    {
                        if (numbers[i - 1] <= numbers[i])
                        {
                            writer1.Write(numbers[i] + " ");
                            if (i == numbers.Length - 1)
                            {
                                index = i;
                            }
                        }
                        else
                        {
                            temp = numbers[i];
                            index = i;
                            break;
                        }
                    }
                    writer2.Write(temp + " ");
                    for (int i = index + 1; i < numbers.Length; i++)
                    {
                        if (numbers[i - 1] <= numbers[i])
                        {
                            writer2.Write(numbers[i] + " ");
                            if (i == numbers.Length - 1)
                            {
                                index = i;
                            }
                        }
                        else
                        {
                            temp = numbers[i];
                            index = i;
                            break;
                        }
                    }
                }
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace Lab_7
{
    public class MultyWayMerge
    {
        public static void FilesMerging(int n, string nameForReader, string nameForWriter)
        {
            StreamReader[] streamReaders = new StreamReader[n];
            for (int i = 0; i < n; i++)
            {
                if (0 != new FileInfo(Directory.GetCurrentDirectory() + nameForReader + i +
".txt").Length)
                    streamReaders[i] = new StreamReader(Directory.GetCurrentDirectory() +
nameForReader + i + ".txt");
            }
            StreamWriter[] streamWriters = new StreamWriter[n];
            for (int i = 0; i < n; i++)
            {
```

```
                streamWriters[i] = new StreamWriter(Directory.GetCurrentDirectory() +
nameForWriter + i + ".txt");
                streamWriters[i].Write("");
            }
            int[] series = new int[n];
            bool[] toRead = new bool[n];
            bool flag = true; ;
            for (int i = 0; i < n; i++)
            {
                toRead[i] = true;
                int firstNumber = ReadingNums(streamReaders[i]);
                if (firstNumber != -1)
                {
                    series[i] = firstNumber;
                }
                else
                {
                    toRead[i] = false;
                }
            }
            while (flag)
            {
                for (int i = 0; i < n; i++)
                {
                    while (true)
                    {
                        int index = Minimal(series, toRead);
                        if (index == -1) break;
                        int minElem = series[index];
                        streamWriters[i].Write(minElem + " ");
                        if (toRead[index])
                        {
                            int nextNumber = ReadingNums(streamReaders[index]);
                            if (nextNumber < series[index])
                            {
                                toRead[index] = false;
                            }
                            if (nextNumber != -1)
                            {
                                series[index] = nextNumber;
                            }
                        }
                    }
                    for (int j = 0; j < n; j++)
                    {
                        if (streamReaders[j] != null && !streamReaders[j].EndOfStream)
                        {
                            toRead[j] = true;
                        }
                    }
                }
                flag = false;
                for (int i = 0; i < n; i++)
                {
                    if (streamReaders[i] != null && !streamReaders[i].EndOfStream)
                    {
                        flag = true;
                        break;
                    }
                }
            }
            for (int i = 0; i < n; i++)
            {
                if (new FileInfo(Directory.GetCurrentDirectory() + nameForWriter + i +
".txt").Length != 0)
```

```
                    streamWriters[i].Write(" ");
                streamWriters[i].Close();
                if (streamReaders[i] != null)
                    streamReaders[i].Close();
            }
        }
        static int ReadingNums(StreamReader reader)
        {
            if (reader == null) return -1;
            string number = "";
            while (!reader.EndOfStream)
            {
                char numberChar = (char)reader.Read();
                if (numberChar == ' ') break;
                number += numberChar;
            }
            if (number == "") return -1;
            return int.Parse(number);
        }
        static int Minimal(int[] elems, bool[] toRead)
        {
            int min = Int32.MaxValue;
            int ind = -1;
            for (int i = 0; i < elems.Length; i++)
            {
                if (min > elems[i] && toRead[i])
                {
                    min = elems[i];
                    ind = i;
                }
            }
            return ind;
        }
    }
}
```

ВИСНОВОК

При виконанні даної лабораторної роботи ми вивчили основні алгоритми зовнішнього сортування та способи їх модифікації, написали псевдокод, програмну реалізацію алгоритму за варіантом та оцінили поріг ефективності алгоритму на файлах розмірів 10Mb (до 5-10 сек), 100Mb (до 30-40 сек).

КРИТЕРІЇ ОЦІНЮВАННЯ

У випадку здачі лабораторної роботи до 25.05.2020 включно максимальний бал дорівнює – 5. Після 25.05.2020 максимальний бал дорівнює – 1.

Критерії оцінювання у відсотках від максимального балу:

- псевдокод алгоритму – 20%;
- програмна реалізація алгоритму – 25%;
- програмна реалізація модифікованого алгоритму – 50%;
- висновок – 5%.