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

злабораторної роботи № 1з дисципліни
«Проектування алгоритмів»

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

Виконав(ла) *ІП-15 Лазюта Олексій Сергійович* _____
(шифр, прізвище, ім'я, по батькові)

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

Київ 2022

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ **3**

2 ЗАВДАННЯ **4**

3 ВИКОНАННЯ **6**

3.1 ПСЕВДОКОД АЛГОРИТМУ 6

3.2 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ 6

3.2.1 Вихідний код *6*

ВИСНОВОК **7**

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 способуза 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 10Мб, 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 модифікованупрограми. У висновку деталізувати, які саме модифікації було виконано і який ефект вони дали.

15	Збалансоване багатошляхове злиття

3 ВИКОНАННЯ

3.1 Псевдокод алгоритму

```
Procedure FillFile()
    Write to “AFile” string.Empty
    Write to “B1File” string.Empty
    Write to “B2File” string.Empty
    Write to “C1File” string.Empty
    Write to “C2File” string.Empty
    create rand from Random()
    open Filestream fileStream of "AFile" for Writing
        open StreamWriter streamWriter
            while fs.Length is less AFile.Length do
                streamWriter.WriteToFile(rand.Next(from 0 to int.MaxValue))
            end while
        close streamWriter
    close fileStream
end FillFile
```

```
Procedure SplitToOtherFiles()
    open Filestream fileStream of "AFile" for Reading
        open StreamReader streamReader
            declare string line
            declare string series = ""
            declare bool writeToB1 = true
            declare int prevNum = 0
            while(line = streamReader.ReadLine is Not null) do
                if (int)line is less prevNum then
                    if writeToB1 == then
                        Write to "B1File" series
                    else
                        Write to "B2File" series
                    writeToB1 = !writeToB1
                    series = string.Empty
                series =series + line + " "
                prevNum = (int)line;
            endwhile
        close streamReader
    close fileStream
end SplitToOtherFiles
```

```
Procedure Algorithm(string pathToWrite1, string pathToWrite2, string pathToRead1, string pathToRead2)
    Write to pathToWrite1 string.Empty
    Write to pathToWrite2 string.Empty
    int length = pathToRead1.CountOfLines
    int length1 = pathToRead2.CountOfLines
    declare line;
    declare int[length][] file1Series
    declare int[length1][] file2Series
    open Filestream fileStream of pathToRead1 for Reading
        open StreamReader streamReader
            declare int i = 0;
            while (line = streamReader.ReadLine is Not null) do
                file1Series[i] = (int)line.ToArray();
                i++;
            endwhile
        close streamReader
    close fileStream
    open Filestream fileStream of pathToRead2 for Reading
        open StreamReader streamReader
            declare int i = 0;
            while (line = streamReader.ReadLine is Not null) do
                file2Series[i] = (int)line.ToArray()
                i++
            endwhile
          close streamReader
```

```
    declare int k = 0
    declare int[] helpArr
    if length1 is higher length1 then
        foreach serie in file1Series do
            if k is less length1 then
                helpArr = Join(serie, file2Series[k]).ToArray()
                helpArr.Sort()
            else
                helpArr = serie
            if k % 2 == 0 then
                pathToWrite1.WriteToFile(helpArr)
            else
                pathToWrite2.WriteToFile(helpArr)
            k++
        end foreach
    else
        foreach serie in file2Series do
            if k is less length then
                helpArr = Join(serie, file1Series[k]).ToArray()
                helpArr.Sort()
            else
                helpArr = serie
            if k % 2 == 0 then
                pathToWrite1.WriteToFile(helpArr)
            else
                pathToWrite2.WriteToFile(helpArr)
            k++
        end foreach
end Algorithm
```

```
Procedure MergeResults(string pathResult1, string pathResult2)
    Write to “AFile” string.Empty
    declare string line
    declare int[] file1Series
    declare int[] file2Series
    open Filestream fileStream of pathResult1 for Reading
        open StreamReader streamReader
            file1Series = (int)streamReader.ReadLine().ToArray()
        close streamReader
    close fileStream
    open Filestream fileStream of pathResult2 for Reading
        open StreamReader streamReader
            file2Series = (int)streamReader.ReadLine().ToArray()
        close streamReader
    close fileStream
    declare int k = 0
    declare int[] helpArr = Join(file1Series, file2Series)
    Array.Sort(helpArr)
    Write to “AFile” helpArr
end MergeResults
```

```
Procedure SelectAndMergeSeries()
    declare int i = 0;
    while true do
        if pathB1File.CountOfLines == 1 And pathB2File.CountOfLines == 1 then
            MergeResults(pathB1File, pathB2File)
            Exit

        if pathC1File.CountOfLines == 1 And pathC2File.CountOfLines == 1 then
            MergeResults(pathC1File, pathC2File)
            Exit

        if i % 2 == 0 then
            Algorithm(pathC1File, pathC2File, pathB1File, pathB2File)
        else
            Algorithm(pathB1File, pathB2File, pathC1File, pathC2File)
        i++
    endwhile
end SelectAndMergeSeries

Procedure Main()
    FillFile()
    SplitToOtherFiles()
    SelectAndMergeSeries()
end Main
```

3.2 Програмна реалізація алгоритму

3.2.1 Вихідний код

```
string pathAFile = @"D:\Algorithms\algorithms\AFile.txt";
string pathB1File = @"D:\Algorithms\algorithms\B1File.txt";
string pathB2File = @"D:\Algorithms\algorithms\B2File.txt";
string pathC1File = @"D:\Algorithms\algorithms\C1File.txt";
string pathC2File = @"D:\Algorithms\algorithms\C2File.txt";

int AFileIngth = 100 * 1024 * 1024;

void FillFile()                          // заповнюємо рандомними числами початковий файл "AFile"
{
   File.WriteAllText(pathAFile, string.Empty);
   File.WriteAllText(pathB1File, string.Empty);   //очищаємо файли
   File.WriteAllText(pathB2File, string.Empty);
   File.WriteAllText(pathC1File, string.Empty);
   File.WriteAllText(pathC2File, string.Empty);
   Random rnd = new Random();
   int randNum;
   using (FileStream fs = new FileStream(pathAFile, FileMode.Append, FileAccess.Write))
   using (StreamWriter sw = new StreamWriter(fs))
   {
      while (fs.Length < AFileIngth)
      {
         randNum = rnd.Next(0, Int32.MaxValue);
         sw.WriteLine($"{randNum}");
      }
      Console.WriteLine(fs.Length);
   }
}

void SplitToOtherFiles()       // перепис з файлу AFile у допоміжні файли B1File, B2File
{
   using (FileStream fs = new FileStream(pathAFile, FileMode.Open, FileAccess.Read))
   using (StreamReader sr = new StreamReader(fs))
   {
      string line;
      string series = "";
      bool writeToB1 = true;
      int prevNum = 0;
      while((line = sr.ReadLine()) != null)      // розбиває всі числа на серії і записує в файли
      {
         if (int.Parse(line) < prevNum)
         {
            if(writeToB1)
               WriteToFile(pathB1File, series); // складаємо серії
            else
               WriteToFile(pathB2File, series);
            writeToB1 = !writeToB1;
            series = "";
         }

         series += line + " ";
         prevNum = int.Parse(line);
      }
   }
}

void Algorithm(string pathToWrite1, string pathToWrite2, string pathToRead1, string pathToRead2)
{
   File.WriteAllText(pathToWrite1, string.Empty);                  // очищуємо файли
   File.WriteAllText(pathToWrite2, string.Empty);
   int length = File.ReadLines(pathToRead1).Count();
   int length1 = File.ReadLines(pathToRead2).Count();
   string line;
   int[][] file1Series = new int[length][];
   int[][] file2Series = new int[length1][];
   using (FileStream fs = new FileStream(pathToRead1, FileMode.Open, FileAccess.Read))
```

```
    using (StreamReader sr = new StreamReader(fs))
    {
        int i = 0;
        while ((line = sr.ReadLine()) != null)
        {
            file1Series[i] = line.TrimEnd().Split().Select(s => int.Parse(s)).ToArray();  // представляємо кожен рядок як массив
            i++;
        }
    }
    using (FileStream fs = new FileStream(pathToRead2, FileMode.Open, FileAccess.Read))
    using (StreamReader sr = new StreamReader(fs))
    {
        int i = 0;
        while ((line = sr.ReadLine()) != null)
        {
            file2Series[i] = line.TrimEnd().Split().Select(s => int.Parse(s)).ToArray();
            i++;
        }
    }
    int k = 0;
    int[] helpArr;
    if (length > length1)
    {
        foreach (var serie in file1Series)
        {
            if (k < length1)
            {
                helpArr = serie.Concat(file2Series[k]).ToArray();          // сортуємо та зливаємо серії
                Array.Sort(helpArr);
            }
            else
                helpArr = serie;
            if (k % 2 == 0)
                WriteToFile(pathToWrite1, string.Join(" ", helpArr));
            else
                WriteToFile(pathToWrite2, string.Join(" ", helpArr));
            k++;
        }
    }
    else
    {
        foreach (var serie in file2Series)
        {
            if (k < length)
            {
                helpArr = serie.Concat(file1Series[k]).ToArray();
                Array.Sort(helpArr);
            }
            else
                helpArr = serie;
            if (k % 2 == 0)
                WriteToFile(pathToWrite1, string.Join(" ", helpArr));
            else
                WriteToFile(pathToWrite2, string.Join(" ", helpArr));
            k++;
        }
    }
}

void MergeResults(string pathResult1, string pathResult2)  // Зливаємо два фінальні допоміжні відсортовані файли в початкоий "AFile.txt"
{
    File.WriteAllText(pathAFile, string.Empty);
    string line;
    int[] file1Series;
```

```
    int[] file2Series;
    using (FileStream fs = new FileStream(pathResult1, FileMode.Open, FileAccess.Read))
    using (StreamReader sr = new StreamReader(fs))
    {
        file1Series = sr.ReadLine().TrimEnd().Split().Select(s => int.Parse(s)).ToArray();  // розділяємо всі числа рядка на массив чисел
    }
    using (FileStream fs = new FileStream(pathResult2, FileMode.Open, FileAccess.Read))
    using (StreamReader sr = new StreamReader(fs))
    {
        int i = 0;
        file2Series = sr.ReadLine().TrimEnd().Split().Select(s => int.Parse(s)).ToArray();
    }
    int k = 0;
    int[] helpArr;
    helpArr = file1Series.Concat(file2Series).ToArray();     // зєднуємо та сортуємо кінцевий массив
    Array.Sort(helpArr);
    WriteToFile(pathAFile, string.Join(" ", helpArr));       // записуємо його у файл

}

void SelectAndMergeSeries()  // чергує файли для зчитування та запису
{
    int i = 0;
    while (true)
    {
        if (File.ReadLines(pathB1File).Count() == 1 && File.ReadLines(pathB2File).Count() == 1)
        {
            MergeResults(pathB1File, pathB2File);
            return;
        }
        if (File.ReadLines(pathC1File).Count() == 1 && File.ReadLines(pathC2File).Count() == 1)
        {
            MergeResults(pathC1File, pathC2File);
            return;
        }
        if (i % 2 == 0)
            Algorithm(pathC1File, pathC2File, pathB1File, pathB2File);
        else
            Algorithm(pathB1File, pathB2File, pathC1File, pathC2File);
        i++;
    }
}

void WriteToFile(string path, string number)        // заповнює допоміжний файл серіями
{
    using (FileStream fs = new FileStream(path, FileMode.Append, FileAccess.Write))
    using (StreamWriter sw = new StreamWriter(fs))
    {
        sw.WriteLine(number);
    }
}



Main();

void Main()
{
    FillFile();
    SplitToOtherFiles();
    SelectAndMergeSeries();
}
```

ВИСНОВОК

При виконанні даної лабораторної роботи я навчився реалізовувати зовнішній алгоритм для сортування великих об’ємів даних, а саме метод збалансованого багатошляхового злиття. Сортування 10 mb даних відбулось за 3хв і 7 сек (187 сек), а сортування 100 mb даних відбулось за 40хв і 40 сек. Таким чином середня швидкість алгоритму становить приблизно 21,55 сек на 1 mb даних. Часова складність: O(n).