**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**”

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

**Виконав(ла)**

(шифр, прізвище, ім'я, по батькові)

*ІП-12 Дулов Денис Валерійвочи*

Київ 2022

Зміст

[1 Мета лабораторної роботи 3](file:///C:\Users\Denys\Downloads\Telegram%20Desktop\lr1_2022.ukr%20(2).docx#_Toc109342184)

[2 ЗаВдання 4](file:///C:\Users\Denys\Downloads\Telegram%20Desktop\lr1_2022.ukr%20(2).docx#_Toc109342185)

[3 Виконання 6](file:///C:\Users\Denys\Downloads\Telegram%20Desktop\lr1_2022.ukr%20(2).docx#_Toc109342186)

[3.1 Псевдокод алгоритму 6](file:///C:\Users\Denys\Downloads\Telegram%20Desktop\lr1_2022.ukr%20(2).docx#_Toc109342187)

[3.2 Програмна реалізація алгоритму 6](file:///C:\Users\Denys\Downloads\Telegram%20Desktop\lr1_2022.ukr%20(2).docx#_Toc109342188)

[3.2.1 Вихідний код 6](file:///C:\Users\Denys\Downloads\Telegram%20Desktop\lr1_2022.ukr%20(2).docx#_Toc109342189)

[Висновок 7](file:///C:\Users\Denys\Downloads\Telegram%20Desktop\lr1_2022.ukr%20(2).docx#_Toc109342190)

[Критерії оцінювання 8](file:///C:\Users\Denys\Downloads\Telegram%20Desktop\lr1_2022.ukr%20(2).docx#_Toc109342191)

# Мета лабораторної роботи

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

# 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 |

# Виконання

## Псевдокод алгоритму

counter = 0

buffLine = null

***adaptiveSort(path)***

**start**

**while** isSorted(path) == false **do**

A\_reader = new StreamReader(path)

File.WriteAllText(“B.txt”, String.Empty)

File.WriteAllText(“C.txt”, String.Empty)

**while** A\_reader.EndOfStream == false **do**

splitInTwoFiles(path, A\_reader)

**end while**

series\_B = findSeries("B.txt")

series\_C = findSeries("C.txt")

mergeSeries(series\_B, series\_C, path)

**end while**

**end**

***splitInTwoFiles(path, A\_reader)***

**start**

series = []

line = buffLine

**if** line == null **do**

line = A\_reader.ReadLine()

**end if**

series.Add(line)

**while** true **do**

prevLine = line

line = A\_reader.ReadLine()

**if** prevLine <= line **do**

series.Add(line)

**else do**

buffLine = line

break

**end if**

**end while**

writeToFile(series)

**end**

***merge(pathA, pathB, pathC)***

**start**

reader\_B, reader\_C, writer\_A

numB = reader\_B.ReadLine()

numC = reader\_C.ReadLine()

**while** reader.EndOfStream == false

**if** numB > numC **do**

writer\_A.WriteLine(numC)

prevNumC = numC

numC = reader\_C.ReadLine()

**if** numC < prevNumC **do**

**do**

writer\_A.WriteLine(numB)

prevNumB = numB

numB = reader\_B.ReadLine()

**while** numB >= prevNumB

**end** **if**

**else do**

writer\_A.WriteLine(numB)

prevNumB = numB

numB = reader\_B.ReadLine()

**if** numB < prevNumB **do**

**do**

writer\_A.WriteLine(numC)

prevNumC = numC

numC = reader\_C.ReadLine()

**while** numC >= prevNumC

**end** **if**

**end if**

**end while**

writer\_A.WriteLine(min(numB, numC))

writer\_A.WriteLine(max(numB, numC))

**if** reader\_B.EndOfStream == true **do**

**while** reader\_C.EndOfStream == false

writer\_A.WriteLine(reader\_C.ReadLine())

**end** **while**

**else** **do**

**while** reader\_B.EndOfStream == false

writer\_A.WriteLine(reader\_B.ReadLine())

**end while**

**end if**

**end**

***isSorted(path)***

**start**

reader = new StreamReader(path)

num1 = reader.ReadLine()

num2 = reader.ReadLine()

**while** reader.EndOfStream == false **do**

**if** num1 > num2 **do**

**return** false

**end if**

num1 = num2

num2 = reader.ReadLine()

**end while**

**return** true

**end**

## Програмна реалізація алгоритму

### Вихідний код

using System;

using System.Collections.Generic;

using System.IO;

namespace lab1

{

class Program

{

static int counter = 1;

static string buffLine = null;

static void Main(string[] args)

{

string path = "A.txt";

DateTime time = DateTime.Now;

adaptiveSort(path);

DateTime time2 = DateTime.Now;

Console.WriteLine(time2.Subtract(time).Minutes);

}

static void adaptiveSort(string path)

{

while (!isSorted(path))

{

StreamReader A\_reader = new StreamReader(path);

File.WriteAllText("B.txt", String.Empty);

File.WriteAllText("C.txt", String.Empty);

while (!A\_reader.EndOfStream)

{

splitInTwoFiles(path, A\_reader);

}

A\_reader.Close();

var series\_B = findSeries("B.txt");

var series\_C = findSeries("C.txt");

mergeSeries(series\_B, series\_C, path);

}

}

static void splitInTwoFiles(string path, StreamReader A\_reader)

{

List<string> series = new List<string>();

string line = buffLine;

if(line == null) { line = A\_reader.ReadLine(); }

series.Add(line);

while (true)

{

string prevLine = line;

line = A\_reader.ReadLine();

if(Convert.ToInt32(prevLine) <= Convert.ToInt32(line))

{

series.Add(line);

}

else

{

buffLine = line;

break;

}

}

writeToFile(series);

}

static void writeToFile(List<string> series)

{

if(counter % 2 == 1)

{

StreamWriter writer = new StreamWriter("B.txt", true);

foreach(string line in series)

{

writer.WriteLine(line);

}

writer.Flush();

writer.Close();

}

else

{

StreamWriter writer = new StreamWriter("C.txt", true);

foreach (string line in series)

{

writer.WriteLine(line);

}

writer.Flush();

writer.Close();

}

counter++;

}

static List<List<int>> findSeries(string path)

{

var series = new List<List<int>>();

var reader = new StreamReader(path);

int i = 0;

string line = reader.ReadLine();

series.Add(new List<int>());

series[0].Add(Convert.ToInt32(line));

string prevLine = line;

while ((line = reader.ReadLine()) != null)

{

if (Convert.ToInt32(prevLine) > Convert.ToInt32(line))

{

series.Add(new List<int>());

i++;

}

series[i].Add(Convert.ToInt32(line));

prevLine = line;

}

reader.Close();

return series;

}

static void mergeSeries(List<List<int>> series\_B, List<List<int>> series\_C, string path)

{

StreamWriter writer = new StreamWriter(path);

for(int i = 0;i < Math.Min(series\_B.Count, series\_C.Count); i++)

{

int j = 0, k = 0;

while(j < series\_C[i].Count && k < series\_B[i].Count)

{

if(series\_C[i][j] > series\_B[i][k])

{

writer.WriteLine(series\_B[i][k]);

k++;

}

else

{

writer.WriteLine(series\_C[i][j]);

j++;

}

}

if (k == series\_B[i].Count)

{

while (j < series\_C[i].Count)

{

writer.WriteLine(series\_C[i][j]);

j++;

}

}

else

{

while (k < series\_B[i].Count)

{

writer.WriteLine(series\_B[i][k]);

k++;

}

}

}

if(series\_B.Count > series\_C.Count)

{

for(int i = series\_C.Count;i < series\_B.Count; i++)

{

for(int j = 0;j < series\_B[i].Count; j++)

{

writer.WriteLine(series\_B[i][j]);

}

}

}else if (series\_B.Count < series\_C.Count)

{

for (int i = series\_B.Count; i < series\_C.Count; i++)

{

for (int j = 0; j < series\_C[i].Count; j++)

{

writer.WriteLine(series\_C[i][j]);

}

}

}

writer.Flush();

writer.Close();

}

static bool isSorted(string path)

{

StreamReader reader = new StreamReader(path);

int num1 = Convert.ToInt32(reader.ReadLine());

int num2 = Convert.ToInt32(reader.ReadLine());

while(!reader.EndOfStream)

{

if(num1 > num2)

{

reader.Close();

return false;

}

num1 = num2;

num2 = Convert.ToInt32(reader.ReadLine());

}

reader.Close();

return true;

}

}

}

Висновок

При виконанні даної лабораторної роботи було досліджено випадки, коли обсяг даних, які потрібно відсортувати наскільки великий, що класичні алгоритми сортування не в змозі з цим впоратись. Було досліджено деякі алгоритми зовнішнього сортування, зокрема алгоритм природного (адаптивного) сортування, що був описаний псевдокодом та реалізований мовою програмування. Під час написання програми, що реалізує вищеназваний алгоритм було подолано велику кількість труднощів й покращено знання у сферах проектування алгоритмів, роботи пам’яті комп’ютера, а також логічне мислення

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%.