**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**

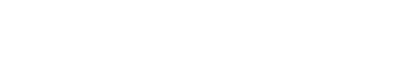
**Звіт**

з лабораторної роботи № 1 з дисципліни

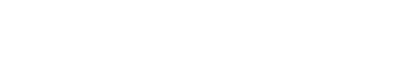
«Проектування алгоритмів»

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

**Виконав(ла)** *ІП-12 Авчаров Григорій*



(шифр, прізвище, ім'я, по батькові)



**Перевірив**



(

прізвище, ім'я, по батькові

)



*Сопов О.*

Київ 2022

ЗМІСТ

1. **МЕТА ЛАБОРАТОРНОЇ РОБОТИ ..................................................... 3**
2. **ЗАВДАННЯ ............................................................................................... 4**
3. **ВИКОНАННЯ ........................................................................................... 6**

### 3.1 ПСЕВДОКОД АЛГОРИТМУ ......................................................................... 6 3.2 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ..................................................... 8

#### 3.2.1Вихідний код..................................................................................... 8

**ВИСНОВОК ................................................................................................... 16**

**КРИТЕРІЇ ОЦІНЮВАННЯ ........................................................................ 17**

# 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

3.1 Псевдокод алгоритму

CreateFile(fileName, 1000000); - створення файла для сортування

StraightMergeSort sortObject = new StraightMergeSort(fileName); - об'єкт сортування

sortObject.Sort(); - сортування

OutputFile(fileName); - вивід даних

--------------------------------------------

OutputFile(string file) - вивід даних

{

using (reader = (File.OpenRead(file)))

{

length = reader.BaseStream.Length;

position = 0;

for (int i = 0; i < 1000; i++)

{

if (position == length)

{

break;

}

else

{

position += 4;

}

}

}

}

CreateFile(string fileName, int count) - створення файла

{

using generator = (File.Create(fileName)))

{

Random rnd = new Random();

for (int i = 0; i < count; i++)

{

generator.Write(rnd.Next(-500, 500));

}

}

}

private void Split() - розділення на два файли

{

cortaige = 1;

using br = (File.OpenRead(FileInput)))

using writerA = (File.Create("b.bin")))

using writerB = (File.Create("c.bin")))

{

counter = 0;

flag = true;

length = br.BaseStream.Length;

position = 0;

while (position != length)

{

if (counter == iterations)

{

flag = !flag;

counter = 0;

cortaige++;

}

currentElement = br.ReadInt32();

position += 4;

if (flag)

{

writerA.Write(currentElement);

}

else

{

writerB.Write(currentElement);

}

counter++;

}

}

}

private void Merge() - злиття в початковий файл

{

using readerA = (File.OpenRead("b.bin")))

using readerB = (File.OpenRead("c.bin")))

using bw = (File.Create(FileInput)))

{

counterA = iterations, counterB = iterations;

elementA = 0, elementB = 0;

pickedA = false, pickedB = false, endA = false, endB = false;

lengthA = readerA.BaseStream.Length;

lengthB = readerB.BaseStream.Length;

positionA = 0;

positionB = 0;

while (!endA || !endB)

{

if (counterA == 0 && counterB == 0)

{

counterA = iterations;

counterB = iterations;

}

if (positionA != lengthA)

{

if (counterA > 0 && !pickedA)

{

elementA = readerA.ReadInt32();

positionA += 4;

pickedA = true;

}

}

else

{

endA = true;

}

if (positionB != lengthB)

{

if (counterB > 0 && !pickedB)

{

elementB = readerB.ReadInt32();

positionB += 4;

pickedB = true;

}

}

else

{

endB = true;

}

if (pickedA)

{

if (pickedB)

{

if (elementA < elementB)

{

bw.Write(elementA);

counterA--;

pickedA = false;

}

else

{

bw.Write(elementB);

counterB--;

pickedB = false;

}

}

else

{

bw.Write(elementA);

counterA--;

pickedA = false;

}

}

else if (pickedB)

{

bw.Write(elementB);

counterB--;

pickedB = false;

}

}

iterations \*= 2;

}

}

public void Sort() - загальний метод класу сортування

{

while (true)

{

Split();

if (cortaige == 1)

{

break;

}

Merge();

}

Console.WriteLine();

}

3.2 Програмна реалізація алгоритму

## 3.2.1 Вихідний код

using System.Diagnostics;

internal class Program

{

public static void Main(string[] args)

{

Stopwatch timer = new Stopwatch();

string fileName = "a.bin";

timer.Start();

CreateFile(fileName, 1000000);

Console.WriteLine("Файл перед сортуванням: ");

OutputFile(fileName);

StraightMergeSort sortObject = new StraightMergeSort(fileName);

sortObject.Sort();

Console.WriteLine("Файл після сортування: ");

OutputFile(fileName);

timer.Stop();

Console.WriteLine($"Час виконання сортування: {(double)timer.ElapsedMilliseconds / 1000} секунд");

Console.ReadKey();

}

public static void OutputFile(string file)

{

using (BinaryReader reader = new BinaryReader(File.OpenRead(file)))

{

long length = reader.BaseStream.Length;

long position = 0;

for (int i = 0; i < 1000; i++)

{

if (position == length)

{

break;

}

else

{

Console.Write($"{reader.ReadInt32()} ");

position += 4;

}

}

Console.WriteLine();

}

}

public static void CreateFile(string fileName, int count)

{

using (BinaryWriter generator = new BinaryWriter(File.Create(fileName)))

{

Random rnd = new Random();

for (int i = 0; i < count; i++)

{

generator.Write(rnd.Next(-500, 500));

}

}

}

public class StraightMergeSort

{

public string FileInput { get; set; }

private long iterations, cortaige;

public StraightMergeSort(string input)

{

FileInput = input;

iterations = 1;

}

public void Sort()

{

while (true)

{

Split();

if (cortaige == 1)

{

break;

}

Merge();

}

Console.WriteLine();

}

private void Merge()

{

using (BinaryReader readerA = new BinaryReader(File.OpenRead("b.bin")))

using (BinaryReader readerB = new BinaryReader(File.OpenRead("c.bin")))

using (BinaryWriter bw = new BinaryWriter(File.Create(FileInput)))

{

long counterA = iterations, counterB = iterations;

int elementA = 0, elementB = 0;

bool pickedA = false, pickedB = false, endA = false, endB = false;

long lengthA = readerA.BaseStream.Length;

long lengthB = readerB.BaseStream.Length;

long positionA = 0;

long positionB = 0;

while (!endA || !endB)

{

if (counterA == 0 && counterB == 0)

{

counterA = iterations;

counterB = iterations;

}

if (positionA != lengthA)

{

if (counterA > 0 && !pickedA)

{

elementA = readerA.ReadInt32();

positionA += 4;

pickedA = true;

}

}

else

{

endA = true;

}

if (positionB != lengthB)

{

if (counterB > 0 && !pickedB)

{

elementB = readerB.ReadInt32();

positionB += 4;

pickedB = true;

}

}

else

{

endB = true;

}

if (pickedA)

{

if (pickedB)

{

if (elementA < elementB)

{

bw.Write(elementA);

counterA--;

pickedA = false;

}

else

{

bw.Write(elementB);

counterB--;

pickedB = false;

}

}

else

{

bw.Write(elementA);

counterA--;

pickedA = false;

}

}

else if (pickedB)

{

bw.Write(elementB);

counterB--;

pickedB = false;

}

}

iterations \*= 2;

}

}

private void Split()

{

cortaige = 1;

using (BinaryReader br = new BinaryReader(File.OpenRead(FileInput)))

using (BinaryWriter writerA = new BinaryWriter(File.Create("b.bin")))

using (BinaryWriter writerB = new BinaryWriter(File.Create("c.bin")))

{

long counter = 0;

bool flag = true;

long length = br.BaseStream.Length;

long position = 0;

while (position != length)

{

if (counter == iterations)

{

flag = !flag;

counter = 0;

cortaige++;

}

int currentElement = br.ReadInt32();

position += 4;

if (flag)

{

writerA.Write(currentElement);

}

else

{

writerB.Write(currentElement);

}

counter++;

}

}

}

}

}

# ВИСНОВОК

У ході виконання лабораторної роботи було досліджено роботу алгоритму зовнішнього сортування «Пряме злиття»

В результаті було створено алгоритм.

Алгоритм сортування складається з наступних кроків: розподіл вхідних даних на файли, поріняння серій та їх зливання в інші файли і повторення цих дій до закінчення сортування. Ознака закінчення сортування – один непорожній файл, що й міститиме у собі відсортовані дані.

Алгоритм сортує 1ГБ цілочисельних даних в середньому за 12 хвилин. Зважаючи на системні можливості ПК, можна вважати такі результати задовільними.

# 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%.