**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)**

**Перевірив**

(прізвище, ім'я, по батькові)

Головченко М.М.

Київ 2022

Зміст

[1 Мета лабораторної роботи 3](#_Toc109342184)

[2 ЗаВдання 4](#_Toc109342185)

[3 Виконання 6](#_Toc109342186)

[3.1 Псевдокод алгоритму 6](#_Toc109342187)

[3.2 Програмна реалізація алгоритму 6](#_Toc109342188)

[3.2.1 Вихідний код 6](#_Toc109342189)

[Висновок 7](#_Toc109342190)

[Критерії оцінювання 8](#_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

CountIntInFIle = 1024\*1024/2

servingSize = 1

**While** servingSize < countIntInFIle / 2

***SrlipFile***();  
 ***JoinFile***();  
 *servingSize* \*= 2;

**EndWhile**

**SplitFile:**

**For** i:i<countIntInFile / 2: 1

Зчитати рядок з файла "input.txt"

Записати рядок в файл **"**first.txt**”**

**EndFor**

**For** i:i<countIntInFile / 2: 1

Зчитати рядок з файла "input.txt"

Записати рядок в файл "second.txt"

**EndFor**

**End**

***JoinFile:***

len : countIntInFIle / (2 \* servingSize)

Array: first, second

**For** i:i<len:1

**For** j:j<servingSize: j

Зчитати число з файла "first.txt" і записати в масив first

**EndFor**

**For** j:j<servingSize: j

Зчитати число з файла "second.txt" і записати в масив first

**EndFor**

**SortServe(first, second, servingSize, writer)**

**EndFor**

**End**

**SortServe:**

bool: flagA, flagB

int: posA, posB

int: num1, num2

**While: true**

**If** posA = len **OR** posB = len

***Write*(second OR first, posB, writer);**

**End**

**If** flag1 = true

num1 = first[posA]

I**f** flagB = true

num2 = second[posB]

**If** num1 > num2

Записати num2 в файл (writer)

posB++;

flag1 = false

flag2 = true

**Else if** num1 < num2

Записати num1 в файл (writer)

posA++;

flag1 = true

flag2 = false

**Else**

Записати num1 і num2 в файл (writer)

posA++;

posb++;

flag1 = true

flag2 = true

**EndWhile**

**End**

**Write:**

**For** i = pos: i< arr.Length:1

Записати число arr[i] в файл (writer)

**EndFor**

**End**

## Програмна реалізація алгоритму

### Вихідний код

public static void *StartSort*() //початок сортування  
{  
 while (*servingSize* < *countIntInFIle* / 2) //сортуємо доки в двох файлах не буде відсортованої послідовності для подальшого злиття  
 {  
 *SplitFile*();  
 *JoinFile*();  
 *servingSize* \*= 2;  
 }  
}  
  
private static void *SplitFile*() //записуємо по половині вхідного файлу у вихідні  
{  
 using var reader = new StreamReader("input.txt", Encoding.UTF32);  
 using (var writer = new StreamWriter("first.txt", false, Encoding.UTF32))  
 {  
 for (int i = 0; i < *countIntInFIle* / 2; i++) //записуємо половину вхідного файлу у вихідний  
 {  
 string a = reader.ReadLine();  
 var num = int.*Parse*(a);  
 writer.WriteLine(num);  
 }   
 }  
 using (var writer = new StreamWriter("second.txt", false, Encoding.UTF32))  
 {  
 for (int i = 0; i < *countIntInFIle* / 2; i++)  
 {  
 string a = reader.ReadLine();  
 var num = int.*Parse*(a);  
 writer.WriteLine(num);  
 }   
 }  
}  
  
private static void *JoinFile*() //зливаємо два файли в один  
{  
 using var writer = new StreamWriter("input.txt", false, Encoding.UTF32);  
 using var reader1 = new StreamReader("first.txt", Encoding.UTF32);  
 using var reader2 = new StreamReader("second.txt", Encoding.UTF32);  
 var len = *countIntInFIle* / (2 \* *servingSize*);  
 for (var i = 0; i < len; i++) //ділимо файл на порції та зливаємо їх  
 {  
 var first = new int[*servingSize*];  
 var second = new int[*servingSize*];  
 string l;  
 for (var j = 0; j < *servingSize*; j++) //зчитаємо по порції данних з першого і другого файлу в масив  
 {  
 l = reader1.ReadLine();  
 first[j] = int.*Parse*(l);  
 }  
 for (var j = 0; j < *servingSize*; j++)  
 {  
 l = reader2.ReadLine();  
 second[j] = int.*Parse*(l);  
 }  
 *SortServ*(first, second, *servingSize*, writer);  
 }  
}  
  
private static void *SortServ*(int[] first, int[] second, int len, StreamWriter writer) //зливає дві порції в файл  
{  
 bool flag1 = true, flag2 = true;  
 int posA = 0, posB = 0;  
 int num1 = first[posA], num2 = second[posB];  
 while (true)  
 {  
 if (posA == len)  
 {  
 *Write*(second, posB, writer);  
 return;  
 }  
 if (posB == len)  
 {  
 *Write*(first, posA, writer);  
 return;  
 }  
 if (flag1)  
 {  
 num1 = first[posA];  
 }  
 if (flag2)  
 {  
 num2 = second[posB];  
 }  
 if (num1 > num2)  
 {  
 writer.WriteLine(num2);  
 posB++;  
 flag1 = false;  
 flag2 = true;  
 }  
 else if (num1 < num2)  
 {  
 writer.WriteLine(num1);  
 posA++;  
 flag2 = false;  
 flag1 = true;  
 }  
 else   
 {  
 writer.WriteLine(num1);  
 writer.WriteLine(num2);  
 posA++;  
 posB++;  
 flag1 = true;  
 flag2 = true;  
 }  
 }  
}  
private static void *Write*(int [] arr, int pos, StreamWriter writer) //дозаписує частину масиву в файл (якщо другий масив уже записаний)  
**{**  
for (var i = pos; i < arr.Length; i++)  
 {  
 writer.WriteLine(arr[i]);  
 }  
}

Програмна реалізація модифікованого алгоритму:

public static class Sort  
**{**  
public static long *sizeInputFile* = 1024 \* 1024 \* 1024; //розмір вхідного файла в байтах  
 public static int *countServ* = 4;  
 public static long *servSize* = *sizeInputFile* / *countServ*; //розмір куска для сортування (128 мб)  
 public static long *countIntInServ* = *servSize* / 32; //кількість чисел в порції для сортування  
 //"12345678" = 32 bytes;  
 public static long *countIntInFIle* = *sizeInputFile* / 32; //кількість чисел у всьому файлі  
 public static string *inputPath* = "input.txt";  
 public static void CreateFile() //метод для створення файлу  
 {  
 var random = new Random();  
 using var writer = new StreamWriter(*inputPath*, false, Encoding.UTF32);  
 for (long i = 0; i < *countIntInFIle*; i++)  
 {  
 writer.WriteLine(random.Next(10000000, 99999999).ToString());  
 }  
 }  
  
 public static string *StartSort*() //початок сорутвання  
 {  
 var paths = *CreateServFile*(); //шляхи до створених файлів  
 while (paths.Count !=1) //поки не залишиться один (відсортований) файл  
 {  
 var p = new List<string>();  
 for (int i = 0; i < paths.Count; i++)  
 {  
 p.Add(paths[i] + paths[i+1]); //два файли зливаються і утворюють ім'я яке є сумою їх назв (наприкоад: input.txt0input.txt1)  
 *SplitTwoFilesInOne*(paths[i], paths[i+1]); //злиття файлу в один  
 i++;  
 }  
  
 foreach (var path in paths) //видалення непотрібних файлів  
 {  
 File.*Delete*(path);  
 }  
 paths = p;  
 }  
 return paths.First();  
 }  
  
 public static void *SplitTwoFilesInOne*(string path1, string path2)  
 {  
 using var reader1 = new StreamReader(path1, Encoding.UTF32);  
 using var reader2 = new StreamReader(path2, Encoding.UTF32);  
 string resPath = path1 + path2; //назва вихідного файлу  
 var writer = new StreamWriter(resPath, true, Encoding.UTF32);  
 bool flag1 = true, flag2 = true; //вони потрібні щоб числа з одного файлу записувались в вихідний, доки вони менші за число з другого файлу   
 string num1 = String.*Empty*, num2 = String.*Empty*;  
 while (true)  
 {  
 if (flag1)  
 {  
 num1 = reader1.ReadLine();  
 }  
 if (flag2)  
 {  
 num2 = reader2.ReadLine();  
 }  
 if (num1 == null) //якщо один з зчитаних рядків порожній, то ми досягли кінця файу  
 {  
 *Write*(reader2, writer);  
 return;  
 }  
 if (num2 == null)  
 {  
 *Write*(reader1, writer);  
 return;  
 }  
 int a = int.*Parse*(num1);  
 int b = int.*Parse*(num2);  
 if (a > b) //порівняння двох файлів і запис меншого в файл  
 {  
 writer.WriteLine(b);  
 flag1 = false; //завдяки цьому нове число не буде зчитане з файлу, а буде порівнюватися з числами з іншого файлу, доки не буде записане   
 flag2 = true;  
 }  
 else if (a < b)  
 {  
 writer.WriteLine(a);  
 flag1 = true;  
 flag2 = false;  
 }  
 else   
 {  
 writer.WriteLine(a);  
 writer.WriteLine(b);  
 flag1 = true;  
 flag2 = true;  
 }  
 }  
 }  
 //якщо дані з одного файла були записані в рузультуючий, а в іншому залишились (це відсортовані числа які більші ніж останнє число файлу який закінчився),  
 //то вони записуються в кінець результуючого файлу  
 public static void *Write*(StreamReader reader, StreamWriter writer)   
 {  
 string line;  
 while ((line = reader.ReadLine()) != null)  
 {  
 writer.WriteLine(line);  
 }  
 }  
 public static List<string> *CreateServFile*() //метод для розділення файлу на 4 частини  
 {  
 using var reader = new StreamReader(*inputPath*, Encoding.UTF32);  
 var paths = new string[*countServ*];  
 for (int i = 0; i < *countServ*; i++)  
 {  
 paths[i] = *inputPath* + i; //створення імені нового файлу (наприклад: input.txt0)  
 using var writer = new StreamWriter(paths[i], false, Encoding.UTF32);  
 var arr = new int[*countIntInServ*];  
 for (int j = 0; j < *countIntInServ*; j++) //запис частини файлу в масив  
 {  
 arr[j] = int.*Parse*(reader.ReadLine());  
 }  
 ///////  
 Array.*Sort*(arr); //сортування масиву  
 //це операція використовує найбільше оперативної пам'яті в програмі (розмір файлу (1.2 гб) / countServ), в даному випадку це 300 мегабайт  
 //при збільшенні кількості частин (countServ = 16) оперативної пам'яті буде використовуватися менше 100 мегабайт, але час сортування збільшиться в 2 рази  
 for (int j = 0; j < *countIntInServ*; j++) //запис відсортованого масиву в файл  
 {  
 writer.WriteLine(arr[j]);  
 }  
 }  
  
 return paths.ToList();  
 }  
 //так як вихідний файл неможливо відкрити за допомогою текстового редактора, створений метод, який  
 //зчитує невелику постідовність даних для демонстрації коректності роботи програми  
 public static void *CheckRes*(string path)   
 {   
 using var reader = new StreamReader(path, Encoding.UTF32);  
 for (int i = 0; i < 40000; i++)  
 {  
 reader.ReadLine();  
 }  
 for (int i = 0; i < 190; i++)  
 {  
 Console.*WriteLine*(int.*Parse*(reader.ReadLine()));  
 }  
 }  
}

Висновок

При виконанні даної лабораторної роботи я навчився реалізовувати алгоритми зовнішнього сортування. Оцінив ефективність того чи іншого алгоритму при роботі з різним набором даних. Модифікував алгоритм відповідно до вимог, дослідив наскільки об’єм оперативної пам’яті впливає на швидкість роботи алгоритму. Після модифікації вдалося досягти вискокої швидкості сортування завдяки збільшенню об’єму оперативної пам’яті який використовується програмою.

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%.