**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)**

(шифр, прізвище, ім'я, по батькові)

*ІП-11 Трикош І. В.*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

Київ 2022

Зміст

[1 Мета лабораторної роботи 3](#_Toc109342184)

[2 ЗаВдання 4](#_Toc109342185)

[3 Виконання 5](#_Toc109342186)

[3.1 Псевдокод алгоритму 5](#_Toc109342187)

[3.2 Програмна реалізація алгоритму 8](#_Toc109342188)

[3.2.1 Вихідний код 8](#_Toc109342189)

[Висновок 13](#_Toc109342190)

[Критерії оцінювання 14](#_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** |
| 31 | Збалансоване багатошляхове злиття |

# Виконання

## Псевдокод алгоритму

**function start():**

flag = true

write\_from\_A\_to\_B()

while (Розмір файлу A ≠ розмірам файлів B1 і C1):

if (flag):

write\_from\_files1\_to\_files2("B", "C")

else:

write\_from\_files1\_to\_files2("C", "B")

end if

flag = !flag

end while

end function

**function write\_from\_A\_to\_B():**

Відкрити файл A для читання і файли B (масив B[8]) для запису

first = 0

second = None

i = 1

while (Поки не кінець файлу A):

Зчитати число second з файлу A

if (first > second):

i = i + 1

end if

Записати число second у файл B[i % 8 + 1]

first = second

end while

Закрити файли A і B

end function

**function write\_from\_files1\_to\_files2(file1, file2):**

Відкрити файли files1 (назва file1) для читання і файли files2 (назва file2) для запису

n = 0

first[8] = { 0, 0, 0, 0, 0, 0, 0, 0 }

second[8] = {Максимальні значення поточного типу даних}

flags[8] = {None}

flags\_not\_empty[8] = {None}

for i = 0 to 7 do:

flags\_not\_empty[i] = Файл files1[i] не пустий

flags[i] = flags\_not\_empty[i]

end for

while (flags[0] or flags[1] or flags[2] or flags[3] or flags[4] or flags[5] or flags[6] or flags[7]):

flag = true

while (true):

for i = 0 to 7 do:

if (flags[i]):

Зчитуємо second[i] з файлу files1[i]

if (first[i] > second[i]):

Повертаємо покажчик зчитування на 1 число назад

flags[i] = false

second[i] = Максимальне значення поточного типу даних

end if

end if

end for

min = Максимальне значення поточного типу даних

index = -1

for i = 0 to 7 do:

if (min > second[i]):

min = second[i]

index = i

end if

end for

if (i ≠ -1):

index\_file2 = n % 8

Записуємо min у файл files2[index\_file2]

for i = 0 to 7 do:

if (i != index):

flags[i] = false

end if

end for

flags[index] = true

first[index] = second[index]

second[index] = Максимальне значення поточного типу даних

else:

flag = false

for i = 0 to 8 do:

first[i] = 0

if (!flags\_not\_empty[i] or Кінець файлу files1[i]):

flags[i] = false

else

flags[i] = true

end if

end for

end if

end while

end while

end function

## Програмна реалізація алгоритму

### Вихідний код

**Базовий алгоритм**

*Laba-C++.cpp*

#include "Func.h"

int main()

{

// The basic algorithm

start();

return 0;

}

*Func.h*

#pragma once

#include <iostream>

#include <fstream>

#include <filesystem>

#include <string>

using namespace std;

void createfile();

void start();

void write\_from\_A\_to\_B();

void write\_from\_files1\_to\_files2(string, string);

bool checkfile(ifstream&);

*Func.cpp*

#include "Func.h"

// Перевірка, чи не пустий файл

bool checkfile(ifstream& file)

{

bool flag = true;

if (file.peek() == ifstream::traits\_type::eof())

{

flag = false;

}

return flag;

}

// Початок програми

void start()

{

// Paths to the files

filesystem::path a("Files/txt/A.txt");

filesystem::path b("Files/txt/B1.txt");

filesystem::path c("Files/txt/C1.txt");

unsigned long long int size\_a = filesystem::file\_size(a); // The size of the file A.txt

bool flag = true; // To changing files

write\_from\_A\_to\_B(); // Write from file A to files B

// While file A != B1 and A != C

while (size\_a != filesystem::file\_size(b) && size\_a != filesystem::file\_size(c))

{

if (flag)

{

write\_from\_files1\_to\_files2("B", "C");

}

else

{

write\_from\_files1\_to\_files2("C", "B");

}

flag = !flag;

}

if (size\_a == filesystem::file\_size(b))

{

cout << "The answer has saved in the file B1.txt";

}

else

{

cout << "The answer has saved in the file C1.txt";

}

}

// Запис серій з файлу А у файли В1-В8

void write\_from\_A\_to\_B()

{

ifstream A("Files/txt/A.txt");

ofstream B[8] = { ofstream("Files/txt/B1.txt"), ofstream("Files/txt/B2.txt"), ofstream("Files/txt/B3.txt"), ofstream("Files/txt/B4.txt"), ofstream("Files/txt/B5.txt"), ofstream("Files/txt/B6.txt"),

ofstream("Files/txt/B7.txt"), ofstream("Files/txt/B8.txt") };

unsigned int first = 0, second; // The numbers

unsigned int i = 0; // The index of the file

// Read the numbers from the file A.txt and write these ones to the files B1-B8.txt

while (!A.eof())

{

A >> second;

if (first > second)

{

i++;

}

B[i % 8] << " " << second;

first = second;

}

A.close();

for (int i = 0; i < 8; i++)

{

B[i].close();

}

}

// Запис з файлів В у файли С (і навпаки)

void write\_from\_files1\_to\_files2(string file1, string file2)

{

ifstream files1[8] = { ifstream("Files/txt/" + file1 + "1.txt"), ifstream("Files/txt/" + file1 + "2.txt"), ifstream("Files/txt/" + file1 + "3.txt"),

ifstream("Files/txt/" + file1 + "4.txt"), ifstream("Files/txt/" + file1 + "5.txt"), ifstream("Files/txt/" + file1 + "6.txt"), ifstream("Files/txt/" + file1 + "7.txt"), ifstream("Files/txt/" + file1 + "8.txt") };

ofstream files2[8] = { ofstream("Files/txt/" + file2 + "1.txt"), ofstream("Files/txt/" + file2 + "2.txt"), ofstream("Files/txt/" + file2 + "3.txt"),

ofstream("Files/txt/" + file2 + "4.txt"), ofstream("Files/txt/" + file2 + "5.txt"), ofstream("Files/txt/" + file2 + "6.txt"), ofstream("Files/txt/" + file2 + "7.txt"), ofstream("Files/txt/" + file2 + "8.txt") };

int n = 0; // Індекс файлу files2, в який треба записати серію

unsigned int first[8] = { 0, 0, 0, 0, 0, 0, 0, 0 }, // Попереднє число у файлі files1

second[8] = { UINT32\_MAX, UINT32\_MAX, UINT32\_MAX, UINT32\_MAX, UINT32\_MAX, UINT32\_MAX, UINT32\_MAX, UINT32\_MAX }; // Наступне число у файлі files1

bool flags[8]; // Закінчення серії у файлі files1

bool flags\_not\_empty[8]; // Перевірка на пустий файл files1

// Перевірка, які файли files1 пусті

for (int i = 0; i < 8; i++)

{

flags\_not\_empty[i] = checkfile(files1[i]);

flags[i] = flags\_not\_empty[i];

}

// Прохід по файлах files1, поки хоча б в одному з них є числа

while (flags[0] || flags[1] || flags[2] || flags[3] || flags[4] || flags[5] || flags[6] || flags[7])

{

bool flag = true; // Для завершення наступного циклу

// Проходимо по файлах files1 та записуємо серію у файл files2[n % 8]

while (flag)

{

// Проходимо по файлах files1 та зчитуємо з них по одному числу(якщо потрібно)

for (int i = 0; i < 8; i++)

{

// Якщо потрібно зчитати число, то зчитуємо

if (flags[i])

{

int pointer = files1[i].tellg(); // Запам'ятовуємо позицію зчитування

files1[i] >> second[i];

// Якщо кінець серії, то ставимо початкові значення для змінних і не більше його не зчитуємо до закінчення внутрішнього циклу while

if (first[i] > second[i])

{

files1[i].seekg(pointer, ios::beg);

flags[i] = false;

second[i] = UINT32\_MAX;

}

}

}

// Знаходимо мінімальне число та запам'ятовуємо індекс файлу

unsigned int min = UINT32\_MAX;

int index = -1;

for (int i = 0; i < 8; i++)

{

if (min > second[i])

{

min = second[i];

index = i;

}

}

// Якщо знайшли мінімальне число, то записуємо його у файл files2, інакше виходимо з внутрішнього циклу while

if (index != -1)

{

int index\_file2 = n % 8; // Індекс файл, у який треба записати число

files2[index\_file2] << " " << min;

// Позначаємо файли files1, з яких не треба зчитувати

for (int i = 0; i < 8; i++)

{

if (i != index)

{

flags[i] = false;

}

}

// Позначаємо файл files1, з якого треба зчитати число на наступній ітерації

flags[index] = true;

first[index] = second[index];

second[index] = UINT32\_MAX;

}

else

{

// Виходимо з внутрішнього циклу while

flag = false;

// Знаходимо файли files1, які уже повністю зчитані або пусті і далі пропускаємо ці файли

for (int i = 0; i < 8; i++)

{

first[i] = 0;

if (!flags\_not\_empty[i] || files1[i].eof())

{

flags[i] = false;

}

else

{

flags[i] = true;

}

}

}

}

n++; // Збільшуємо індекс файлу

}

for (int i = 0; i < 8; i++)

{

files1[i].close();

files2[i].close();

}

}

**Модифікований алгоритм**

*Laba-C++.cpp*

#include "function.h"

int main()

{

// The modified algorithm

long long int n = 134217728; // 1 GB

Start(n);

return 0;

}

*function.h*

#pragma once

#include <iostream>

#include <fstream>

using namespace std;

void Start(long long int);

void Write\_from\_A\_to\_B(long long int);

void Write\_from\_B\_to\_C(long long int);

bool Checkfile(ifstream&);

void quickSort(unsigned long long int\*, long long int, long long int);

long long int partition(unsigned long long int\*, long long int, long long int);

void swap(unsigned long long int\*, unsigned long long int\*);

*function.cpp*

#include "function.h"

// Перевірка, чи не пустий файл

bool Checkfile(ifstream& file)

{

bool flag = true;

if (file.peek() == ifstream::traits\_type::eof())

{

flag = false;

}

return flag;

}

// Початок програми

void Start(long long int n)

{

time\_t t1 = time(NULL);

Write\_from\_A\_to\_B(n); // Write from file A to files B

time\_t t2 = time(NULL);

cout << "Write from A to B: " << t2 - t1 << "s" << endl;

t1 = time(NULL);

Write\_from\_B\_to\_C(n); // Write from files B to file C

t2 = time(NULL);

cout << "Write from B to C: " << t2 - t1 << "s" << endl;

cout << "The answer has saved in the file C1.txt";

}

// Перестановка двох елементів

void swap(unsigned long long int\* a, unsigned long long int\* b)

{

unsigned long long int t = \*a;

\*a = \*b;

\*b = t;

}

// Визначення індексу опорного елемента і порівняння елементів з опорним

long long int partition(unsigned long long int\* arr, long long int low, long long int high)

{

unsigned long long int number = arr[high]; // Опорний елемент

long long int i = low - 1; // Індекс найлівішого елемента

// Проходимо по заданому проміжку

for (long long int j = low; j <= high - 1; j++)

{

// Якщо поточний елемент менший за опорний

if (arr[j] < number)

{

// Збільшуємо індекс найлівішого елемента та міняємо його з поточним

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[high]); // Міняємо найлівіший елемент і опорний

return (i + 1); // Повернаємо індекс опорного елемента

}

// Швидке сортування

void quickSort(unsigned long long int\* arr, long long int low, long long int high)

{

// Якщо low < high, то виконуємо:

if (low < high) {

// Знаходимо індекс опорного елемента

long long int n = partition(arr, low, high);

// Сортуємо елементи до і після опорного

quickSort(arr, low, n - 1);

quickSort(arr, n + 1, high);

}

}

// Запис з файлу А у файли В

void Write\_from\_A\_to\_B(long long int n)

{

ifstream A("Files/bin/A.bin", ios::binary);

ofstream B[16] = { ofstream("Files/bin/B1.bin", ios::binary), ofstream("Files/bin/B2.bin", ios::binary), ofstream("Files/bin/B3.bin", ios::binary),

ofstream("Files/bin/B4.bin", ios::binary), ofstream("Files/bin/B5.bin", ios::binary), ofstream("Files/bin/B6.bin", ios::binary),

ofstream("Files/bin/B7.bin", ios::binary), ofstream("Files/bin/B8.bin", ios::binary), ofstream("Files/bin/B9.bin", ios::binary),

ofstream("Files/bin/B10.bin", ios::binary), ofstream("Files/bin/B11.bin", ios::binary), ofstream("Files/bin/B12.bin", ios::binary),

ofstream("Files/bin/B13.bin", ios::binary), ofstream("Files/bin/B14.bin", ios::binary),

ofstream("Files/bin/B15.bin", ios::binary), ofstream("Files/bin/B16.bin", ios::binary) };

unsigned long long int number; // Зчитуване число

long long int sizes[16]; // Кількість чисел у файлах

// Знаходимо кількість чисел, які будуть у файлах В

for (int i = 0; i < 16; i++)

{

sizes[i] = n / 16;

}

for (int i = 0; i < n % 16; i++)

{

sizes[i]++;

}

// Проходимо по файлах

for (int i = 0; i < 16; i++)

{

// Якщо кількість чисел більша за 0

if (sizes[i] > 0)

{

unsigned long long int\* arr = new unsigned long long int[sizes[i]]; // Масив чисел

A.read((char\*)arr, 8 \* sizes[i]); // Зчитуємо масив чисел

quickSort(arr, 0, sizes[i] - 1); // Сортуємо масив

B[i].write((char\*)arr, 8 \* sizes[i]); // Записуємо масив у файл B[i]

delete[] arr; // Очищуємо пам'ять

}

}

A.close();

for (int i = 0; i < 16; i++)

{

B[i].close();

}

}

// Запис з файлів В у файл С

void Write\_from\_B\_to\_C(long long int n)

{

ifstream B[16] = { ifstream("Files/bin/B1.bin", ios::binary), ifstream("Files/bin/B2.bin", ios::binary), ifstream("Files/bin/B3.bin", ios::binary),

ifstream("Files/bin/B4.bin", ios::binary), ifstream("Files/bin/B5.bin", ios::binary), ifstream("Files/bin/B6.bin", ios::binary),

ifstream("Files/bin/B7.bin", ios::binary), ifstream("Files/bin/B8.bin", ios::binary),

ifstream("Files/bin/B9.bin", ios::binary), ifstream("Files/bin/B10.bin", ios::binary), ifstream("Files/bin/B11.bin", ios::binary),

ifstream("Files/bin/B12.bin", ios::binary), ifstream("Files/bin/B13.bin", ios::binary), ifstream("Files/bin/B14.bin", ios::binary),

ifstream("Files/bin/B15.bin", ios::binary), ifstream("Files/bin/B16.bin", ios::binary) };

ofstream C("Files/bin/C1.bin", ios::binary);

bool flag[16]; // Для перевірки, чи треба зчитати число з файлу В

unsigned long long int number[16]; // Зчитані числа

// Встановлюємо початкові значення

for (int i = 0; i < 16; i++)

{

number[i] = ULLONG\_MAX;

}

// Перевірка, чи не пусті файли В

for (int i = 0; i < 16; i++)

{

flag[i] = Checkfile(B[i]);

}

// Проходимо по кількості чисел у файлах В

for (long long int i = 0; i < n; i++)

{

// Проходимо по файлах В

for (int j = 0; j < 16; j++)

{

// Якщо треба зчитати число з файлу В[j], то зчитуємо його

if (flag[j])

{

// Якщо число не можна зчитати, то встановлюємо файл В[j] як прочитаний

if (!B[j].read((char\*)&number[j], 8))

{

flag[j] = false;

}

}

}

// Знаходимо мінімальне число

unsigned long long int min = ULLONG\_MAX;

int index = -1;

for (int k = 0; k < 16; k++)

{

if (min > number[k])

{

min = number[k];

index = k;

}

}

C.write((char\*)&min, 8); // Записуємо мінімальне число у файл С

// Проходимо по файлах та встанолюємо на них flag = false

for (int k = 0; k < 16; k++)

{

if (k != index)

{

flag[k] = false;

}

}

// Позначаємо файл, як той, з якого потрібно зчитати число і присвоюємо числу початкове значення

flag[index] = true;

number[index] = ULLONG\_MAX;

}

for (int i = 0; i < 16; i++)

{

B[i].close();

}

C.close();

}

Висновок

При виконанні даної лабораторної роботи я вивчив основні алгоритми зовнішнього сортування та способи їх модифікації, оцінив поріг їх ефективності.

Я розробив алгоритм збалансованого багатошляхового злиття на мові C++. Алгоритм полягає у розбитті початкового файлу (файл А) на n файлів (файли В), в яких записані серії з файлу А. Далі n серій зливаються в одну і почергово записуються у файли С. Потім з файлів С беруть по n серій і зливають їх у файли В. Алгоритм повторюється, поки у файлі В1 або С1 не вийде одна серія.

Час роботи *базового алгоритму* для файлу розміром 10,2 МБ становить 1 хв 15 секунд.

Для модифікації алгоритму я виконав такі дії: подвоїв кількість файлів В (з 8 до 16); з початкового файлу зчитав n/16 (n - кількість чисел у файлі) чисел як масив і відсортував їх за допомогою швидкого сортування, записуючи їх як одну серію (відсортований масив) у файл , і потім ці серії злив в одну у файл С1; використав бінарні файли замість текстових.

Час роботи *модифікованого алгоритму* для файлу результати такі:

|  |  |
| --- | --- |
| Розмір файлу | Час роботи |
| 1 ГБ | 2 хв 14 с |
| 2 ГБ | 4 хв 30 с |
| 4 ГБ | 9 хв 23 с |
| 8 ГБ | 18 хв 35 с |

Як бачимо з графіка, складність алгоритму .

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%.