**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-13 Замковий Д. В.*

**Перевірив**

(прізвище, ім'я, по батькові)

*Сопов О. О.*

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 9](#_Toc109342188)

[3.2.1 Вихідний код 9](#_Toc109342189)

[Висновок 16](#_Toc109342190)

[Критерії оцінювання 17](#_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

Function join:

оpen file A as file\_0

оpen file B as file\_1

оpen file C as file\_2

e1 = false;

e2 = false;

r1 = true;

r2 = true;

num1 = i32::MIN;

num2 = i32::MIN;

l1 = false;

l2 = false;

seek = false;

end1 = true;

end2 = true;

loop:

if not e1 and r1:

if ReadNum::Error to read\_num\_space(file\_2, buffer):

return Err

else if ReadNum::EndFile to read\_num\_space(file\_2, buffer):

e1 = true

else if ReadNum::EndLine(n) to read\_num\_space(file\_2, buffer):

num1 = n

l1 = true

end1 = false

else if ReadNum::Number(n) to read\_num\_space(file\_2, buffer):

num1 = n

l1 = false

end1 = false

end if

end if

not e2 and r2

if ReadNum::Error to read\_num\_space(file\_2, buffer):

return Err

else if ReadNum::EndFile to read\_num\_space(file\_2, buffer):

e2 = true

else if ReadNum::EndLine(n) to read\_num\_space(file\_2, buffer):

num2 = n

l2 = true

end1 = false

else if ReadNum::Number(n) to read\_num\_space(file\_2, buffer):

num1 = n

l2 = false

end2 = false

end if

end if

if e1 and e2:

break

end if

if seek:

if Err to file\_0.write([NL]):

return Err

end if

else:

seek = true

end if

if not e1 and e2:

if Err to file\_0.write(num1.to\_string().as\_bytes()):

return Err

end if

r1 = true

r2 = false

else if e1 and not e2:

if Err to file\_0.write(num2.to\_string().as\_bytes()):

return Err

end if

r1 = false

r2 = true

else if num1 < num2:

if Err to file\_0.write(num1.to\_string().as\_bytes()):

return Err

end if

r1 = true

r2 = false

else:

if Err to file\_0.write(num2.to\_string().as\_bytes()):

return Err

end if

r1 = false

r2 = true

end if

if r1 and l1 and not e2:

num1 = i32::MAX

r1 = false

else if r2 and l2 and not e1:

num2 = i32::MAX

r2 = false

if not r1 and not r2 and l1 and l2 and num1 == i32::MAX and num2 == i32::MAX:

r1 = true

r2 = true

end if

end loop

if (end1 and not end2) or (not end1 and en2):

return true

end if

return false

end function

function split:

оpen file A as file\_0

оpen file B as file\_1

оpen file C as file\_2

last: i32 = i32::MIN;

first = true;

nl\_1 = false;

nl\_2 = false;

seek\_1 = false;

seek\_2 = false;

loop:

i = read\_num\_br(&mut file\_0, buffer)

if last > i:

if first:

seek\_1 = false

else:

seek\_2 = false

end if

first = not first

end if

if first:

if nl\_1 and not seek\_1:

if Err to file\_1.write([NL]):

return Err

end if

else:

nl\_1 = true

end if

if seek\_1:

if Err to file\_1.write([SPACE]):

return Err

end if

else:

seek\_1 = true

end if

if Err to file\_1.write(i.to\_string().as\_bytes()):

return Err

end if

else:

if nl\_2 and not seek\_2:

if Err to file\_2.write([NL]):

return Err

end if

else:

nl\_2 = true

end if

if seek\_2:

if Err to file\_2.write([SPACE]):

return Err

end if

else:

seek\_2 = true

end if

if Err to file\_2.write(i.to\_string().as\_bytes()):

return Err

end if

end if

end loop

return

end function

## Програмна реалізація алгоритму

### Вихідний код на мові програмування Rust lang

use std::{fs::File, io::{Read, Write}, str::from\_utf8};

const NL: u8 = b'\n';

const SPACE: u8 = b' ';

const BUF\_LEN: usize = 32;

enum ReadNum {

    Error,

    EndFile,

    EndLine(i32),

    Number(i32),

}

fn main() {

    let type\_0 = "type\_0.txt";

    let type\_1 = "type\_1.txt";

    let type\_2 = "type\_2.txt";

    let mut buffer: [u8; BUF\_LEN] = [0; BUF\_LEN];

    loop {

        if let Err(e) = split(&mut buffer, type\_0, type\_1, type\_2) {

            eprintln!("{}", e);

            return;

        }

        match join(&mut buffer, type\_0, type\_1, type\_2) {

            Ok(res) => if res {

                break;

            },

            Err(e) => {

                eprintln!("{}", e);

                return;

            },

        };

    }

    println!("Завершено");

}

fn join(buffer: &mut [u8], type\_0: &str, type\_1: &str, type\_2: &str) -> Result<bool, String> {

    let mut file\_0 = match File::create(type\_0) {

        Ok(f) => f,

        Err(e) => return Err(format!("При відкритті файлу <{}> виникла помилка: {}", type\_0, e.to\_string())),

    };

    let mut file\_1 = match File::open(type\_1) {

        Ok(f) => f,

        Err(e) => return Err(format!("При відкритті файлу <{}> виникла помилка: {}", type\_1, e.to\_string())),

    };

    let mut file\_2 = match File::open(type\_2) {

        Ok(f) => f,

        Err(e) => return Err(format!("При відкритті файлу <{}> виникла помилка: {}", type\_2, e.to\_string())),

    };

    let mut e1 = false;

    let mut e2 = false;

    let mut r1 = true;

    let mut r2 = true;

    let mut num1 = i32::MIN;

    let mut num2 = i32::MIN;

    let mut l1 = false;

    let mut l2 = false;

    let mut seek = false;

    let mut end1 = true;

    let mut end2 = true;

    loop {

        if !e1 && r1 {

            match read\_num\_space(&mut file\_1, buffer) {

                ReadNum::Error => return Err(format!("При читанні з файлу <{}> виникла помилка", type\_1)),

                ReadNum::EndFile => e1 = true,

                ReadNum::EndLine(n) => {

                    num1 = n;

                    l1 = true;

                    end1 = false;

                },

                ReadNum::Number(n) => {

                    num1 = n;

                    l1 = false;

                    end1 = false;

                },

            };

        }

        if !e2 && r2 {

            match read\_num\_space(&mut file\_2, buffer) {

                ReadNum::Error => return Err(format!("При читанні з файлу <{}> виникла помилка", type\_2)),

                ReadNum::EndFile => e2 = true,

                ReadNum::EndLine(n) => {

                    num2 = n;

                    l2 = true;

                    end2 = false;

                },

                ReadNum::Number(n) => {

                    num2 = n;

                    l2 = false;

                    end2 = false;

                },

            };

        }

        if e1 && e2 {

            break;

        }

        if seek {

            if let Err(e) = file\_0.write(&[NL]) {

                return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_0, e.to\_string()));

            };

        } else {

            seek = true;

        }

        if !e1 && e2 { // Закінчився 2-й файл

            if let Err(e) = file\_0.write(num1.to\_string().as\_bytes()) {

                return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_0, e.to\_string()));

            };

            r1 = true;

            r2 = false;

        } else if e1 && !e2 { // Закінчився 1-й файл

            if let Err(e) = file\_0.write(num2.to\_string().as\_bytes()) {

                return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_0, e.to\_string()));

            };

            r1 = false;

            r2 = true;

        } else if num1 < num2 {

            if let Err(e) = file\_0.write(num1.to\_string().as\_bytes()) {

                return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_0, e.to\_string()));

            };

            r1 = true;

            r2 = false;

        } else {

            if let Err(e) = file\_0.write(num2.to\_string().as\_bytes()) {

                return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_0, e.to\_string()));

            };

            r1 = false;

            r2 = true;

        }

        if r1 && l1 && !e2{

            num1 = i32::MAX;

            r1 = false;

        } else if r2 && l2 && !e1 {

            num2 = i32::MAX;

            r2 = false;

        }

        if !r1 && !r2 && l1 && l2 && num1 == i32::MAX && num2 == i32::MAX {

            r1 = true;

            r2 = true;

        }

    }

    if (end1 && !end2) || (!end1 && end2) {

        return Ok(true);

    }

    Ok(false)

}

fn split(buffer: &mut [u8], type\_0: &str, type\_1: &str, type\_2: &str) -> Result<(), String> {

    let mut file\_0 = match File::open(type\_0) {

        Ok(f) => f,

        Err(e) => return Err(format!("При відкритті файлу <{}> виникла помилка: {}", type\_0, e.to\_string())),

    };

    let mut file\_1 = match File::create(type\_1) {

        Ok(f) => f,

        Err(e) => return Err(format!("При відкритті файлу <{}> виникла помилка: {}", type\_1, e.to\_string())),

    };

    let mut file\_2 = match File::create(type\_2) {

        Ok(f) => f,

        Err(e) => return Err(format!("При відкритті файлу <{}> виникла помилка: {}", type\_2, e.to\_string())),

    };

    let mut last: i32 = i32::MIN;

    let mut first = true;

    let mut nl\_1 = false;

    let mut nl\_2 = false;

    let mut seek\_1 = false;

    let mut seek\_2 = false;

    loop {

        let i = match read\_num\_br(&mut file\_0, buffer) {

            Some(i) => i,

            None => break,

        };

        if last > i {

            if first {

                seek\_1 = false;

            } else {

                seek\_2 = false;

            }

            first = !first;

        }

        if first {

            if nl\_1 && !seek\_1 {

                if let Err(e) = file\_1.write(&[NL]) {

                    return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_1, e.to\_string()));

                };

            } else {

                nl\_1 = true;

            }

            if seek\_1 {

                if let Err(e) = file\_1.write(&[SPACE]) {

                    return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_1, e.to\_string()));

                };

            } else {

                seek\_1 = true;

            }

            if let Err(e) = file\_1.write(i.to\_string().as\_bytes()) {

                return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_1, e.to\_string()));

            };

        } else {

            if nl\_2 && !seek\_2 {

                if let Err(e) = file\_2.write(&[NL]) {

                    return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_2, e.to\_string()));

                };

            } else {

                nl\_2 = true;

            }

            if seek\_2 {

                if let Err(e) = file\_2.write(&[SPACE]) {

                    return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_2, e.to\_string()));

                };

            } else {

                seek\_2 = true;

            }

            if let Err(e) = file\_2.write(i.to\_string().as\_bytes()) {

                return Err(format!("При запису файлу <{}> виникла помилка: {}", type\_2, e.to\_string()));

            };

        }

        last = i;

    }

    Ok(())

}

fn read\_num\_br(file: &mut File, buffer: &mut [u8]) -> Option<i32> {

    for i in 0..BUF\_LEN {

        let count = match file.read(&mut buffer[i..i+1]) {

            Ok(count) => count,

            Err(\_) => return None,

        };

        if buffer[i] == NL || count == 0 {

            if i == 0 {

                return None;

            }

            return match from\_utf8(&buffer[0..i]) {

                Ok(s) => {

                    match s.parse::<i32>() {

                        Ok(int) => Some(int),

                        Err(\_) => None,

                    }

                },

                Err(\_) => None,

            }

        }

    }

    None

}

fn read\_num\_space(file: &mut File, buffer: &mut [u8]) -> ReadNum {

    for i in 0..BUF\_LEN {

        let count = match file.read(&mut buffer[i..i+1]) {

            Ok(count) => count,

            Err(\_) => return ReadNum::Error,

        };

        if buffer[i] == SPACE || count == 0 {

            if i == 0 {

                return ReadNum::EndFile;

            }

            return match from\_utf8(&buffer[0..i]) {

                Ok(s) => {

                    match s.parse::<i32>() {

                        Ok(int) => {

                            if count == 0 {

                                ReadNum::EndLine(int)

                            } else {

                                ReadNum::Number(int)

                            }

                        },

                        Err(\_) => ReadNum::Error,

                    }

                },

                Err(\_) => ReadNum::Error,

            }

        } else if buffer[i] == NL {

            if i == 0 {

                return ReadNum::Error;

            }

            return match from\_utf8(&buffer[0..i]) {

                Ok(s) => {

                    match s.parse::<i32>() {

                        Ok(int) => ReadNum::EndLine(int),

                        Err(\_) => ReadNum::Error,

                    }

                },

                Err(\_) => ReadNum::Error,

            }

        }

    }

    ReadNum::EndFile

}

Висновок

При виконанні даної лабораторної роботи ми вивчили основні алгоритми зовнішнього сортування та способи їх модифікації, оцінити поріг їх ефективності. Також реалізував даний алгоритм на мові програмування Rust lang.

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%.