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-13 Замковий Д. В.* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Сопов О. О.* _____
(прізвище, ім'я, по батькові)

Київ 2022

ЗМІСТ

1 МЕТА ЛАБОРАТОРНОЇ РОБОТИ

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

2 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

3 ВИКОНАННЯ

3.1 Псевдокод алгоритму

```
Function join:
      open file A as file_0
      open file B as file_1
      open 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:
            open file A as file_0
            open file B as file_1
            open 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
```

3.2 Програмна реалізація алгоритму

3.2.1 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%.