**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 Головня Олександр Ростиславович*

**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** |
| 7 | Збалансоване багатошляхове злиття |

# Виконання

## Псевдокод алгоритму

Розбиття початкового файлу на задану кількість інших файлів

**Smash\_file(f, n\_of\_files):**

**повторити для** file **у діапазоні**(0, n\_of\_files):

file\_lst.append(open(str(file) + ".txt", "w"))

with open(f, "r") as file:

**повторити для** line **у** file:

**якщо** current\_index != 0 **та** tmp\_lst[current\_index - 1] >= int(line):

write\_to\_file(tmp\_lst, file\_lst)

current\_file += 1

current\_index = 0

current\_index += 1

write\_to\_file(tmp\_lst, file\_lst)

Злиття менших серій у більші, їх запис у нові файли(нові файли <= попередні). Створюється масив з перших чисел у кожному файлі, потім обирається найменше з них та записується у вихідний файл, на його місце у масиві стає наступне число у поточному файлі. Процедура відбувається поки хоча б в одному з початкових фалів є не оброблені числа з цієї серії. Далі процедура повторюється для наступної серії поки всі серії не будуть оброблені

**merge\_files(file\_names):**

**повторити для** i, name **у** previous\_names

symbol = file\_lst[i].readline()

**якщо** symbol != "\n" **та** symbol != "":

curr\_elements.append(int(symbol))

**інакше**:

sizes\_of\_files[i] = 0

**поки** any(x > 1 for x in size\_of\_files):

**поки** real\_length > 0:

min\_element = min(curr\_elements)

write\_to\_file(min\_element, new\_file\_lst[])

**якщо** symbol != "\n" **та** symbol != "":

curr\_elements.insert(min\_index, int(symbol))

**інакше**:

real\_length -= 1

current\_new\_file += 1

**повторити для** i **у діапазоні** (num\_of\_files):

symbol = file\_lst [i].readline()

**якщо** symbol != "\n" **та** symbol != "":

curr\_elements.append(int(symbol))

else:

sizes\_of\_files[i] = 0

**повторити для** i, file **у** enumerate(file\_lst):

**якщо** current\_new\_file - 1 >= i:

new\_f.append(str(int(file.name[:-4]) + num\_of\_files))

**повернути** new\_f

## Програмна реалізація алгоритму

### Вихідний код

#Злиття менших серій у більші

def merge\_files(file\_names):

num\_of\_files = len(file\_names)

curr\_elements = []

new\_file\_lst = []

file\_lst = []

sizes\_of\_files = [0] \* num\_of\_files

new\_f = []

current\_new\_file = 0

real\_length = num\_of\_files

#----------відкриття та обчислення розмірів файлів для відстеження їх кінця----------------

for i, name in enumerate(file\_names):

sizes\_of\_files[i] = os.path.getsize(file\_names[i] + ".txt")

new\_file\_lst.append(open(str(int(name) + num\_of\_files) + ".txt", "w"))

file\_lst.append(open(name + ".txt", "r"))

symbol = file\_lst[i].readline()

sizes\_of\_files[i] -= len(symbol) + 1

#----------читання перших символів кожного файлу та додавання їх до списку-----------------

if symbol != "\n" and symbol != "":

curr\_elements.append(int(symbol))

else:

#----------якщо файл пустий, потрібно зменшити загальну кількість--------------------------

sizes\_of\_files[i] = 0

curr\_elements.append(float('inf'))

real\_length -= 1

while any(x > 1 for x in sizes\_of\_files): #Поки не будуть оброблені всі числа у всіх файлах

while real\_length > 0: #Поки хоча б один файл містить числа, пов'язані з поточною серією

min\_element = min(curr\_elements)

min\_index = curr\_elements.index(min\_element)

#-----------запис найменшого числа зі списку перших чисел у кожному файлі у вихідний файл-----

write\_to\_file(min\_element, new\_file\_lst[current\_new\_file % num\_of\_files])

curr\_elements.pop(min\_index)

#-----------заміна цього числа на наступне з того ж файлу--------------------------------------

symbol = file\_lst[min\_index].readline()

sizes\_of\_files[min\_index] -= len(symbol) + 1

if symbol != "\n" and symbol != "":

curr\_elements.insert(min\_index, int(symbol))

else:

curr\_elements.insert(min\_index, float('inf'))

real\_length -= 1

write\_to\_file(None, new\_file\_lst[current\_new\_file % num\_of\_files], True)

current\_new\_file += 1

curr\_elements.clear()

real\_length = num\_of\_files

#-----------повторення для наступної серії та файлу-------------------------------------------

for i in range(num\_of\_files):

symbol = file\_lst[i].readline()

sizes\_of\_files[i] -= len(symbol) + 1

if symbol != "\n" and symbol != "":

curr\_elements.append(int(symbol))

else:

sizes\_of\_files[i] = 0

curr\_elements.append(float('inf'))

real\_length -= 1

for file in new\_file\_lst:

file.close()

for i, file in enumerate(file\_lst):

if current\_new\_file - 1 >= i:

new\_f.append(str(int(file.name[:-4]) + num\_of\_files))

file.close()

os.remove(file.name)

if len(new\_f) < num\_of\_files:

for i in range(num\_of\_files - len(new\_f)):

os.remove(str(int(new\_f[-1]) + i + 1) + ".txt")

return new\_f

#Запис числа/серії/пропуска у файл

def write\_to\_file(series, file\_lst, space=0):

str\_= ""

if space:

file\_lst.write("\n")

return

if type(series) == list:

for i in series:

str\_ += str(i) + "\n"

file\_lst.write(str\_ + "\n")

else:

file\_lst.write(str(series) + "\n")

#Pозділення вихідного файлу на серії та запис їх у нові файли

def smash\_file(f, n\_files=5):

tmp\_lst = []

file\_lst = []

current\_i = 0

current\_f = 0

for file in range(n\_files):

file\_lst.append(open(str(file) + ".txt", "w"))

with open(f, "r") as file:

for line in file:

if current\_i != 0 and tmp\_lst[current\_i - 1] >= int(line):

write\_to\_file(tmp\_lst, file\_lst[current\_f % n\_files])####

current\_f += 1

current\_i = 0

tmp\_lst.clear()

current\_i += 1

tmp\_lst.append(int(line))

write\_to\_file(tmp\_lst, file\_lst[current\_f % n\_files])

for file in file\_lst:

file.close()

print("Start!")

f = "rand\_write1.txt"

n\_files = 8 + int(math.log2(os.path.getsize(f) / 1000000))#К-сть допоміжних файлів

#print(n\_files) #8(4-х шляхове злиття)

smash\_file(f, n\_files)

a = time.time()#Початок часу

file\_names = merge\_files([str(i) for i in range(n\_files)])

while len(file\_names) > 1:

file\_names = merge\_files(file\_names)#Злиття усіх файлів

os.rename(str(file\_names[0]) + ".txt", "sorted.txt")#Останній файл

b = time.time()#Кінець часу

print("Finish!")

print("Time : ",str(round((b-a),2)),"seconds / ",str(round(((b-a)/60),1))," minutes")

Модифікація:  
Змінимо функцію виділення серій та його поділ, яка при цьому буде внутрішньо сортувати ці елементи. Функцію злиття: приберемо обробку усіх серій, тепер викликатиметься лише один раз, а не декілька, оскільки усі проміжні файли вже будуть відсортовані, то вистачить лише одного злиття.

Висновок

При виконанні даної лабораторної роботи було розглянуто випадки, у яких неможливо здійснити внутрішнє сортування даних, коли їх обсяг значно перевищує обсяги оперативної пам’яті. Було розроблено алгоритм збалансованого багатошляхового злиття та здійснена його модифікація, яка поєднала у собі внутрішнє та зовнішнє сортування, для зменшення часу виконання.

ТЕСТУВАННЯ

|  |  |  |
| --- | --- | --- |
| Розмір файлу | Не модифікований | Модифікований |
| 10мб | 27 с | 5 с |
| 100мб | 5 хв | 1 хв |
| 1гб | 47 хв | 14 хв |

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%.