**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-21 Скрипець Ольга Олександрівна*

**Перевірив**

(прізвище, ім'я, по батькові)

*Головченко М.М.*

Київ 2023

**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. Достатньо штучно обмежити доступну ОП, для уникнення багатогодинних сортувань (наприклад використовуючи віртуальну машину). 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.

# Виконання

## Псевдокод алгоритму

Розділення вхідного файлу на серії та запис їх у нові файли

**Split\_input\_file(input\_file, n\_files):**

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

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

current\_i = 0

current\_f = 0

series\_list = []  
file\_list = []

with open(input\_file, "r") **як** file:

**повторити для** line у file:

**якщо** current\_i != 0 **та** series\_list[current\_i - 1] >= int(line):

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

current\_f += 1

current\_i = 0

series\_list.clear()

current\_i += 1

series\_list.append(int(line))

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

**повторити для** file у file\_lst:

file.close()

Тут виконується процес розділення вхідного файлу на менші серії чисел та запис цих серій у нові файли. Основна ідея полягає в тому, щоб розділити великий файл на менші частини, що спрощує подальший процес обробки та сортування даних. Програма розпочинається зі створення та відкриття нових файлів для запису менших серій. Кількість таких файлів визначається параметром n\_files. Далі розглядається вихідний файл input\_file, з якого будуть читатися дані. Під час читання вихідного файлу, програма визначає серії чисел, розділяючи їх, коли зустрічає число, яке менше або рівне попередньому числу у серії. Кожна окрема серія записується у відповідний файл із створених на початку. Якщо поточна серія закінчилася, вона записується у відповідний файл, а програма переходить до наступної серії. Цей процес триває, доки не будуть оброблені всі числа з вихідного файлу та всі серії будуть розділені та записані у відповідні файли. Усі файли закриваються, завершуючи операцію розділення.

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

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

current\_series = []

new\_output\_files = []

file\_list = []

file\_sizes = [0] \* num\_of\_files

new\_file\_indices = []

current\_new\_file\_index = 0

remaining\_files = num\_of\_files

**повторити** **для** i, name **в переліку** file\_names:

file\_sizes[i] = **отримати розмір файлу**(file\_names[i] + ".txt")

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

file\_list.append(**відкрити файл**(name + ".txt", "r"))

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

file\_sizes[i] -= **отримати довжину**(symbol) + 1

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

current\_series.append(**перетворити у ціле**(symbol))

**інакше**:

file\_sizes[i] = 0

current\_series.append(нескінченність)

remaining\_files -= 1

**поки є хоча б один елемент більший за** 1 в file\_sizes:

**поки** remaining\_files > 0:

min\_element = **знайти мінімальний елемент**(current\_series)

min\_index = **знайти індекс елементу**(current\_series, min\_element)

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

**видалити елемент**(current\_series, min\_index)

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

file\_sizes[min\_index] -= **отримати довжину**(symbol) + 1

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

**вставити елемент**(current\_series, min\_index, перетворити\_у\_ціле(symbol))

**інакше:**

**вставити елемент**(current\_series, min\_index, нескінченність)

remaining\_files -= 1

write\_to\_file(нуль, new\_output\_files[current\_new\_file\_index % num\_of\_files], Істина)

current\_new\_file\_index += 1

**очистити список**(current\_series)

remaining\_files = num\_of\_files

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

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

file\_sizes[i] -= **отримати довжину**(symbol) + 1

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

**вставити елемент**(current\_series, **перетворити у ціле**(symbol))

**інакше:**

file\_sizes[i] = 0

**вставити елемент**(current\_series, нескінченність)

remaining\_files -= 1

**повторити для** file **в** new\_output\_files:

file.close()

**повторити для** i, file **в** enumerate(file\_list):

**якщо** current\_new\_file\_index - 1 >= i:

**додати елемент**(new\_file\_indices**, перетворити у рядок**(**перетворити у ціле**(file.name[:-4]) + num\_of\_files))

file.close()

**видалити файл**(file.name)

**якщо розмір**(new\_file\_indices) < num\_of\_files:

**повторити** **для** i **у діапазоні**(num\_of\_files - розмір(new\_file\_indices)):

**видалити файл**(str(**перетворити у ціле**(**отримати останній елемент**(new\_file\_indices)) + i + 1) + ".txt")

**повернути** new\_file\_indices

Цей алгоритм виконує процес злиття менших серій чисел у більші, які були розділені до цьог. Основна мета - об'єднати ці частини знову в один файл з відсортованими числами. Спочатку визначається загальна кількість файлів, які потрібно злити (параметр `file\_names`). Далі створюються та відкриваються файли для запису об'єднаних серій. Кількість цих файлів визначається також на підставі загальної кількості файлів. Потім обчислюються розміри кожного файлу, щоб відстежити кінець кожної серії та забезпечити правильне злиття. Створюється список `current\_series`, який містить перші числа з кожного файлу (перший елемент кожної серії). Далі виконується зовнішній цикл, поки хоча б один файл містить числа для злиття (за умови, що розмір файлів більше 1). У внутрішньому циклі вибирається найменше число зі списку `current\_series` та записується у вихідний файл. Потім це число замінюється на наступне число з того ж файлу. Після злиття однієї серії відповідний файл закривається, і процес повторюється для наступної серії. Після об'єднання всіх серій в один файл, програма закриває усі відкриті файли та видаляє тимчасові файли, залишивши тільки об'єднаний файл. Повертається список індексів нових файлів, які стали результатом об'єднання. Ця функція дозволяє об'єднувати та сортувати великі обсяги даних, розділені на менші серії, щоб забезпечити ефективну роботу з великими файлами.

## Програмна реалізація алгоритму

### Вихідний код

import math  
import time  
import os  
  
  
def combine\_files(file\_names): #зливає менші серії у більші  
 num\_of\_files = len(file\_names) #загальна кількість файлів яку потрібно злити  
 current\_series = [] # Поточна серія чисел  
 new\_output\_files = []  
 file\_list = []  
 file\_sizes = [0] \* num\_of\_files  
 new\_file\_indices = []  
 current\_new\_file\_index = 0  
 remaining\_files = num\_of\_files # Кількість файлів, які ще містять числа  
  
 for i, name in enumerate(file\_names): #відкриває та обчислює розміри файлів для відстеження їх кінця  
 file\_sizes[i] = os.path.getsize(file\_names[i] + ".txt")  
 new\_output\_files.append(open(str(int(name) + num\_of\_files) + ".txt", "w"))  
 file\_list.append(open(name + ".txt", "r"))  
 symbol = file\_list[i].readline()  
 file\_sizes[i] -= len(symbol) + 1  
 if symbol != "" and symbol != "\n": #читаються перші символи кожного файлу та додаються до списку  
 current\_series.append(int(symbol))  
 else: #якщо файл пустий, потрібно зменшити загальну кількість  
 file\_sizes[i] = 0  
 current\_series.append(float('inf'))  
 remaining\_files -= 1  
  
  
 while any(x > 1 for x in file\_sizes): #Поки не будуть оброблені всі числа у всіх файлах  
 while remaining\_files > 0: #Поки хоча б один файл містить числа, пов'язані з поточною серією  
 min\_element = min(current\_series)  
 min\_index = current\_series.index(min\_element)  
  
 write\_to\_file(min\_element, new\_output\_files[current\_new\_file\_index % num\_of\_files]) #записується найменше число зі списку перших чисел у кожному файлі у вихідний файл  
 current\_series.pop(min\_index)  
 symbol = file\_list[min\_index].readline() #замінюється це число на наступне з того ж файлу  
 file\_sizes[min\_index] -= len(symbol) + 1  
 if symbol != "\n" and symbol != "":  
 current\_series.insert(min\_index, int(symbol))  
 else:  
 current\_series.insert(min\_index, float('inf'))  
 remaining\_files -= 1  
 write\_to\_file(None, new\_output\_files[current\_new\_file\_index % num\_of\_files], True)  
 current\_new\_file\_index += 1  
 current\_series.clear()  
 remaining\_files = num\_of\_files  
  
 for i in range(num\_of\_files): #повторення для наступної серії та файлу  
 symbol = file\_list[i].readline()  
 file\_sizes[i] -= len(symbol) + 1  
 if symbol != "\n" and symbol != "":  
 current\_series.append(int(symbol))  
 else:  
 file\_sizes[i] = 0  
 current\_series.append(float('inf'))  
 remaining\_files -= 1  
 for file in new\_output\_files:  
 file.close()  
  
 for i, file in enumerate(file\_list):  
 if current\_new\_file\_index - 1 >= i:  
 new\_file\_indices.append(str(int(file.name[:-4]) + num\_of\_files))  
 file.close()  
 os.remove(file.name)  
 if len(new\_file\_indices) < num\_of\_files:  
 for i in range(num\_of\_files - len(new\_file\_indices)):  
 os.remove(str(int(new\_file\_indices[-1]) + i + 1) + ".txt")  
 return new\_file\_indices  
  
  
  
def write\_to\_file(series\_list, file\_list, space=0):  
 line= ""  
 if space:  
 file\_list.write("\n")  
 return  
 if type(series\_list) == list:  
 for i in series\_list:  
 line += str(i) + "\n"  
 file\_list.write(line + "\n")  
 else:  
 file\_list.write(str(series\_list) + "\n")  
  
  
  
  
#Pозділення вихідного файлу на серії та запис їх у нові файли  
def split\_input\_file(input\_file, n\_files):  
 series\_list = []  
 file\_list = []  
  
 for file in range(n\_files):  
 file\_list.append(open(str(file) + ".txt", "w")) #створюються та відкриваються файли для запис  
  
 current\_i = 0  
 current\_f = 0  
 with open(input\_file, "r") as file:  
 for line in file:  
 if current\_i != 0 and series\_list[current\_i - 1] >= int(line): #порівнюється попередній елемент серії з наступним числом, яке зчитане з файла  
 write\_to\_file(series\_list, file\_list[current\_f % n\_files]) #записує поточну серію з series\_list у відповідний файл зі списку file\_list.  
 current\_f += 1  
 current\_i = 0  
 series\_list.clear()  
 current\_i += 1  
 series\_list.append(int(line))  
 write\_to\_file(series\_list, file\_list[current\_f % n\_files])  
 for file in file\_list:  
 file.close()  
  
  
  
  
  
  
input\_file = "input.txt"  
  
n\_files = 8 + int(math.log2(os.path.getsize(input\_file) / 1000000))#визначення розміру файлу в байтах, переторення на мб К-сть допоміжних файлів  
   
  
  
split\_input\_file(input\_file, n\_files)  
  
print("Початок збалансованого багатошляхового злиття")  
t1 = time.time()  
  
file\_names = combine\_files([str(i) for i in range(n\_files)]) #генератор списку, який перетворює кожне число в послідовності у рядок  
  
while len(file\_names) > 1:  
 file\_names = combine\_files(file\_names) #Злиття усіх файлів  
   
os.rename(str(file\_names[0]) + ".txt", "output.txt")  
  
t2 = time.time()  
print("Час в секундах : ",str(round((t2-t1),2)))  
print("Час в хвилинах : ", str(round(((t2-t1)/60),1)))

**Модифікація**: У другій версії програми була додана модифікація для покращення ефективності роботи програми. Тепер функція розділяє вихідний файл на серії за заданим розміром size, а не на фіксовану кількість файлів n\_files. Це дозволяє краще використовувати ресурси та оптимізовує роботу програми в разі, коли вихідний файл містить дуже багато даних. За допомогою цієї модифікації, програма читає size рядків з вихідного файлу, сортує їх та записує у відповідний файл. Після цього вона переходить до наступних size рядків, повторюючи цей процес, доки не дочитає до кінця вихідного файлу. Це дозволяє програмі працювати з файлами більшого розміру в оперативній пам'яті, а не зберігати всі дані в пам'яті одночасно.

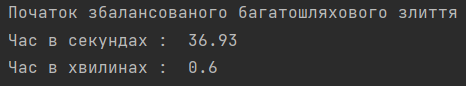
Також у функції combine\_files замість того, щоб одразу записувати всі числа у вихідний файл, як у першій версії, у другій версії програма використовує додатковий файл для вихідних даних output\_file. Таким чином, числа не записуються безпосередньо у вихідний файл при їх знаходженні, а зберігаються в списку current\_series. Ця модифікація була додана для зменшення кількості операцій запису в файл. Замість запису числа при кожному виявленні мінімального елементу, програма виконує запис тільки раз, коли весь набір чисел в current\_series відсортований. Крім того, дана модифікація дозволяє краще контролювати роботу з файлами, так як всі операції запису в файл відбуваються після обробки всіх чисел, а не кожного окремого числа.

### Вихідний код з модифікацією

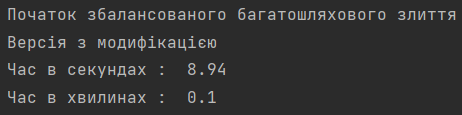
import math  
import time  
import os  
  
  
  
#Злиття менших серій у більші  
def combine\_files(file\_names):  
 num\_of\_files = len(file\_names) #загальна кількість файлів яку потрібно злити  
 current\_series = []  
 file\_handler = []  
 remaining\_files = num\_of\_files # Кількість файлів, які ще містять числа  
  
 output\_file = open("output\_modif.txt", "w")  
 for i, name in enumerate(file\_names):  
 file\_handler.append(open(name + ".txt", "r")) #відкривається файл та додається до списку file\_handler  
 symbol = file\_handler[i].readline()  
 if symbol != "\n" and symbol != "": #читаються перші символи кожного файлу та додаються до списку current\_series  
 current\_series.append(int(symbol))  
 else: #якщо файл пустий, потрібно зменшити загальну кількість  
 current\_series.append(float('inf')) #----------  
 remaining\_files -= 1  
 while remaining\_files > 0: # поки хоча б один файл містить числа, пов'язані з поточною серією  
 min\_element = min(current\_series)  
 min\_index = current\_series.index(min\_element)  
 write\_to\_file(min\_element, output\_file) #запис найменшого числа зі списку перших чисел у кожному файлі у вихідний файл  
 current\_series.pop(min\_index) #видалення елементу зі списку  
 symbol = file\_handler[min\_index].readline() #заміна цього числа на наступне з того ж файлу  
 if symbol != "\n" and symbol != "":  
 current\_series.insert(min\_index, int(symbol))  
 else:  
 current\_series.insert(min\_index, float('inf'))  
 remaining\_files -= 1  
 write\_to\_file(None, output\_file, True)  
 current\_series.clear()  
 remaining\_files = num\_of\_files  
  
 for i in range(num\_of\_files): #повторення для наступної серії та файлу  
 symbol = file\_handler[i].readline()  
 if symbol != "\n" and symbol != "":  
 current\_series.append(int(symbol))  
 else:  
 current\_series.append(float('inf'))  
 remaining\_files -= 1  
 output\_file.close()  
 for file in file\_handler:  
 file.close()  
 os.remove(file.name)  
  
  
  
def write\_to\_file(series\_list, file\_number, space=0):  
 line= ""  
 if space:  
 file\_number.write("\n")  
 return  
 if type(series\_list) == list:  
 for i in series\_list:  
 line += str(i) + "\n"  
 file\_number.write(line + "\n")  
 else:  
 file\_number.write(str(series\_list) + "\n")  
  
  
  
#Pозділення вихідного файлу на серії та запис їх у нові файли  
def split\_input\_file(input\_file, size):  
 with open(input\_file, "r") as file:  
 num\_line = 0  
 num\_file = 0  
 series\_list = []  
  
 line = file.readline()  
 while line != "":  
 series\_list.append(int(line)) #додається ціле число з line до кінця списку  
 num\_line += 1  
 line = file.readline()  
 if num\_line == size:  
 with open(f"{num\_file}.txt", "w") as file\_number:  
 series\_list.sort()  
 write\_to\_file(series\_list, file\_number)  
 num\_file += 1  
 series\_list.clear()  
 num\_line = 0  
 return [f"{i}" for i in range(num\_file)]  
  
  
  
size = 265000  
  
# файл10 ліній 1590000 ділю на 6=265000  
# файл100 ліній 14516267 ділю на 21=691251 (від 19 до 25 один ефект)  
#  
  
print("Початок збалансованого багатошляхового злиття")  
print("Версія з модифікацією")  
t1 = time.time()  
  
file\_names = split\_input\_file("input.txt", size)  
combine\_files(file\_names)  
  
  
t2 = time.time()  
print("Час в секундах : ",str(round((t2-t1),2)))  
print("Час в хвилинах : ", str(round(((t2-t1)/60),1)))

ТЕСТУВАННЯ

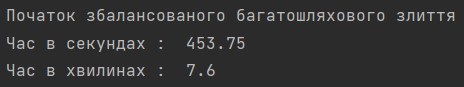
Файл розміром 10 мб



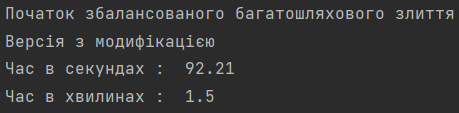
Файл розміром 10 мб + модифікація



Файл розміром 100 мб

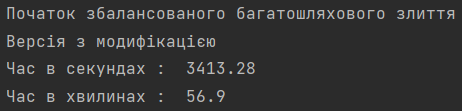


Файл розміром 100 мб + модифікація



Файл розміром 1000 мб

Файл розміром 1000 мб + модифікація



ВИСНОВОК

При виконанні цієї лабораторної роботи був розглянутий алгоритм злиття менших серій чисел в більші та його модифікована версія для оптимізації роботи з великими обсягами даних. В результаті дослідження було з'ясовано, що обидва підходи є ефективними для об'єднання та сортування великих файлів, розділених на менші серії.

У першій версії алгоритму файли розділялися на фіксовану кількість частин, що може бути корисно для деяких завдань. Проте у другій версії була введена модифікація, яка розділяє файли на серії за заданим розміром. Ця модифікація поліпшує оптимізацію ресурсів та дозволяє працювати з файлами навіть в разі, коли їхні обсяги дуже великі.

Додатково, у другій версії було введено оптимізацію, яка дозволяє виконувати запис в вихідний файл тільки після відсортування всього набору чисел, зменшуючи кількість операцій запису. Ця модифікація підвищує ефективність програми та полегшує контроль над роботою з файлами.

Отже, використання цих алгоритмів та їх модифікацій дозволяє ефективно об'єднувати та сортувати великі обсяги даних.

Критерії оцінювання

У випадку здачі лабораторної роботи до 08.10.2022 включно максимальний бал дорівнює – 5. Після 08.10.2022 максимальний бал дорівнює – 4,5.

Критерії оцінювання у відсотках від максимального балу:

− псевдокод алгоритму

– 15%;

− програмна реалізація алгоритму

– 20%;

− програмна реалізація модифікацій – 20%;

− робота з git – 40%;

− висновок – 5%