**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-12 Шоман Данило Володимирович*

**Перевірив**

(прізвище, ім'я, по батькові)

*Сопов Олексій Олександрович*

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** |
| 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

n = розмірність масиву

x = 0

**ПОКИ** (2 \*\* x <= n) **ВИКОНАТИ:**

**ВІДКРИТИ** ‘Початковий\_файл.bin’ **у режимі зчитування**

**ВІДКРИТИ** ‘Додатковий\_файл\_1.bin**’ у режимі запису**

**ВІДКРИТИ** ‘Додатковий\_файл\_2.bin’ **у режимі запису**

i = 0

**ПОКИ** (i < n) **ВИКОНАТИ:**

k1 = 0

**ПОКИ** (k1 != 2 \*\* x) **ВИКОНАТИ:**

**ЗАПИСАТИ ЕЛЕМЕНТ В** ‘Додатковий\_файл\_1.bin’ **З** ‘Початковий\_файл.bin’

i += 1

k1 += 1

**КІНЕЦЬ ПОКИ**

k2 = 0

**ПОКИ** (k2 != 2 \*\* x) **ВИКОНАТИ:**

**ЗАПИСАТИ ЕЛЕМЕНТ В** ‘C.bin’ **З** ‘Початковий\_файл.bin’

i += 1

k2 += 1

**КІНЕЦЬ ПОКИ**

**КІНЕЦЬ ПОКИ**

**ЗАКРИТИ** ‘Початковий\_файл.bin’

**ЗАКРИТИ** ‘Додатковий\_файл\_1.bin’

**ЗАКРИТИ** ‘C.bin’

**ВІДКРИТИ** ‘Початковий\_файл.bin’ **у режимі запису**

**ВІДКРИТИ** ‘Додатковий\_файл\_1.bin**’ у режимі зчитування**

**ВІДКРИТИ** ‘Додатковий\_файл\_2.bin’ **у режимі зчитування**

i = 0

el1 = **ЗЧИТАТИ ЕЛЕМЕНТ З** ‘Додатковий\_файл\_1.bin’

el2 = **ЗЧИТАТИ ЕЛЕМЕНТ З** ‘C.bin’

**ПОКИ** el1 та el2 існують **ВИКОНАТИ:**

k1 = 1

k2 = 1

**ПОКИ**(k1 != 2 \*\* x + 1 and k2 != 2 \*\* x + 1)**ВИКОНАТИ:**

**ЯКЩО** (el1 <= el2) **ВИКОНАТИ:**

**ЗАПИСАТИ ЕЛЕМЕНТ** el1 **В** ‘Початковий\_файл.bin’

el1 = **ЗЧИТАТИ ЕЛЕМЕНТ З** ‘Додатковий\_файл\_1.bin’

k1 += 1

i += 1

**ІНАКШЕ ВИКОНАТИ:**

**ЗАПИСАТИ ЕЛЕМЕНТ** el2 **В** ‘Початковий\_файл.bin’

el2 = **ЗЧИТАТИ ЕЛЕМЕНТ З** ‘Додатковий\_файл\_2.bin’

k2 += 1

i += 1

**ПОКИ** (k1 != 2 \*\* x + 1) **ВИКОНАТИ:**

**ЗАПИСАТИ ЕЛЕМЕНТ** el1 **В** ‘Початковий\_файл.bin’

k1 += 1

i += 1

el1 = **ЗЧИТАТИ ЕЛЕМЕНТ З** ‘Додатковий\_файл\_1.bin’

**КІНЕЦЬ ПОКИ**

**ПОКИ** (k2 != 2 \*\* x + 1) **ВИКОНАТИ:**

**ЗАПИСАТИ ЕЛЕМЕНТ** el2 **В** ‘Початковий\_файл.bin’

k2 += 1

i += 1

el2 = **ЗЧИТАТИ ЕЛЕМЕНТ З** ‘Додатковий\_файл\_2.bin’

**КІНЕЦЬ ПОКИ**

**КІНЕЦЬ ПОКИ**

**ПОВТОРИТИ** n-1 разів

**ЗАПИСАТИ ЕЛЕМЕНТ** el1 **В** ‘Початковий\_файл.bin’

el1 = **ЗЧИТАТИ ЕЛЕМЕНТ З** ‘Додатковий\_файл\_1.bin’

і += 1

**КІНЕЦЬ ПОВТОРИТИ**

**ЗАКРИТИ** ‘Початковий\_файл.bin’

**ЗАКРИТИ** ‘Додатковий\_файл\_1.bin’

**ЗАКРИТИ** ‘C.bin’

**КІНЕЦЬ ПОКИ**

## Програмна реалізація алгоритму

### Вихідний код

main:

from sorting import \*  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 n = 10000000  
 i = 0  
 default\_alg(n, i)  
 modified\_alg(n, i)

sorting:

from random import randrange as rand  
import psutil  
import math  
import time  
from shutil import copyfile as copyf  
  
def default\_alg(n, i):  
 randomize\_file(n)  
 time\_start = time.time()  
 # print("####before - ")  
 # print\_all()  
 # print("####alg - ")  
  
 while (2 \*\* i <= n):  
 # print("@@@ Перед входом в разделение i = ", i, ", n = ", n)  
 split\_to\_files(n, i)  
 # print("###Вміст файлів###")  
 # print\_all()  
 # print("###ENDВміст файлів###")  
 # print("@@@ Перед входом в слияние i = ", i, ", n = ", n)  
 merge(n, i)  
 i += 1  
 # print("i = ", i)  
  
 time\_end = time.time()  
 # print("####after - \n")  
 # print\_all()  
  
 if (is\_sorted('A.bin')):  
 print("Sorted correctly")  
 else:  
 print("error")  
  
 print("Default algorting sorting time: " + str(time\_end - time\_start) + "\n")  
 clear\_file("A.bin")  
 clear\_file("B.bin")  
 clear\_file("C.bin")  
 return  
  
def modified\_alg(n, i):  
 randomize\_file(n)  
 time\_start = time.time()  
 i = pre\_sort('A.bin')  
 # print("i = ", i)  
 copyf('preSortedFile.bin', 'A.bin')  
 # print("####before - ")  
  
 while(2 \*\* i <= n):  
 split\_to\_files(n, i)  
 merge(n, i)  
 i += 1  
  
 time\_end = time.time()  
 # print("####after - ")  
 # print\_all()  
  
 if(is\_sorted('A.bin')):  
 print("Sorted correctly")  
 else:  
 print("error")  
  
 print("Modified algoritm sorting time: " + str(time\_end - time\_start) + "\n")  
 clear\_file("A.bin")  
 clear\_file("B.bin")  
 clear\_file("C.bin")  
 return  
  
def randomize\_file(n):  
 fa = open("A.bin", 'wb')  
 for i in range(n):  
 fa.write(rand(0, 100).to\_bytes(32,'big'))  
 fa.close()  
 return  
  
def print\_file(path):  
 with open(path, "rb") as file:  
 num = file.read(32)  
 print("---", path, "---")  
 while num:  
 num = int.from\_bytes(num, 'big')  
 print(num, end = ' ')  
 num = file.read(32)  
 return  
  
def print\_all():  
 print\_file("A.bin")  
 print()  
 print\_file("B.bin")  
 print()  
 print\_file("C.bin")  
 print()  
 return  
  
def clear\_file(path):  
 with open(path, "wb") as file:  
 file.write(b'')  
  
  
def split\_to\_files(n, iteration):  
 fa = open('A.bin', 'br')  
 fb = open('B.bin', 'wb')  
 fc = open('C.bin', 'wb')  
  
 i = 0  
 while(i < n):  
 k1 = 0  
 while(k1 != 2 \*\* iteration):  
 fb.write(fa.read(32))  
 # print("Записано у файл б - \n", print\_all())  
 i += 1  
 k1 += 1  
  
 k2 = 0  
 while (k2 != 2 \*\* iteration):  
 fc.write(fa.read(32))  
 # print("Записано у файл с - \n", print\_all())  
 i += 1  
 k2 += 1  
  
 fa.close()  
 fb.close()  
 fc.close()  
 return  
  
  
def merge(n, iteration):  
 fa = open('A.bin', 'wb')  
 fb = open('B.bin', 'rb')  
 fc = open('C.bin', 'rb')  
  
 i = 0  
  
 bel1 = fb.read(32)  
 el1 = int.from\_bytes(bel1, 'big')  
  
 bel2 = fc.read(32)  
 el2 = int.from\_bytes(bel2, 'big')  
  
 while bel1 and bel2:  
 k1 = 1  
 k2 = 1  
 while (k1 != 2 \*\* iteration + 1 and k2 != 2 \*\* iteration + 1):  
 if (el1 <= el2):  
 # print("Записано у файл значення el1 - ", el1)  
 fa.write(bel1)  
 bel1 = fb.read(32)  
 el1 = int.from\_bytes(bel1, 'big')  
 k1 += 1  
 i += 1  
 # print("Тепер el1 - ", el1, ", k1 = ", k1, ", i = ", i)  
 else:  
 # print("Записано у файл значення el2 - ", el2)  
 fa.write(bel2)  
 bel2 = fc.read(32)  
 el2 = int.from\_bytes(bel2, 'big')  
 k2 += 1  
 i += 1  
 # print("Тепер el2 - ", el2, ", k2 = ", k2, ", i = ", i)  
 while (k1 != 2 \*\* iteration + 1):  
 # print("У дод циклі записано у файл значення el1 - ", el1)  
 fa.write(bel1)  
 bel1 = fb.read(32)  
 k1 += 1  
 i += 1  
 el1 = int.from\_bytes(bel1, 'big')  
 # print("Тепер el1 - ", el1, ", k1 = ", k1, ", i = ", i)  
  
 while (k2 != 2 \*\* iteration + 1):  
 # print("У дод циклі записано у файл значення el2 - ", el2)  
 fa.write(bel2)  
 bel2 = fc.read(32)  
 k2 += 1  
 i += 1  
 el2 = int.from\_bytes(bel2, 'big')  
 # print("Тепер el2 - ", el2, ", k2 = ", k2, ", i = ", i)  
 # print("-$----end group----$- i = ", i, ", k1 = ", k1, ", k2 = ", k2)  
 for i in range(n-i):  
 # print("Надлишково додано елемент - ", el1)  
 fa.write(bel1)  
 bel1 = fb.read(32)  
 i += 1  
 el1 = int.from\_bytes(bel1, 'big')  
  
 fa.close()  
 fb.close()  
 fc.close()  
 return  
  
def is\_sorted(path):  
 with open(path, "rb") as file:  
 prev = file.read(32)  
 num = file.read(32)  
 while num:  
 if (int.from\_bytes(prev, 'big') > int.from\_bytes(num, 'big')):  
 # print("NO ")  
 return False  
 prev = num  
 num = file.read(32)  
 # print("Yes ")  
 return True  
  
  
def pre\_sort(path):  
 available\_ints = int(psutil.virtual\_memory()[1]/32)  
 clear\_file("preSortedFile.bin")  
  
 with open(path, "rb") as file:  
 int\_list = []  
 while True:  
 num = file.read(32)  
 if num == b'':  
 break  
  
 for i in range(available\_ints):  
 int\_list.append(int.from\_bytes(num, 'big'))  
 num = file.read(32)  
 if num == b'':  
 break  
  
 int\_list.sort()  
  
 with open('preSortedFile.bin', "ab") as preFile:  
 for i in int\_list:  
 preFile.write(i.to\_bytes(32, 'big'))  
  
 return math.log2(available\_ints)

Висновок

При виконанні даної лабораторної роботи досліджував алгоритм зовнішнього сортування (Пряме злиття) та намагався його модифікувати. Модифікація представляла собою розбиття початкового файлу на послідовності чисел ті їх сортування, і тільки вже потім застосування алгоритму прямого злиття. Це допомогло значно пришвидшити сортування файлу.

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%.