**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 Скорик Родіон Олегович*

**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** |
| 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 |

**Варіант 22**

Завдання - природне (адаптивне) злиття

# Виконання

## Псевдокод алгоритму

**procedure** natural\_marge\_sort (file\_name)  
 is\_sorted = False  
 **while (not** is\_sorted) **do**  
 num = split\_file(A, B, C)  
 if (num!=1)  
 merge\_file(A, B, C)  
 else  
 **copyfile** B to A  
 is\_sorted = True

**end if**

**end while**

**procedure** merge\_file(A, B, C)  
 **open** A **for writing,** **open** B for reading **open** C for reading  
 read num\_b from B  
 read num\_c from C  
  
 **while** num\_b or num\_c have been read **do**  
  
 **if** num\_b **and not** num\_c  
 **while** num\_b **do**  
 **write** num\_b to A  
 **read** num\_b from B  
 **end while**  
 **else if** num\_c **and not** num\_b  
 **while** num\_c **do**  
 **write** num\_c **to** A  
 **read** num\_c **from** C  
 **end while**  
 **else if** next\_num(b) >= num\_b and next\_num(c) >= num\_c  
 **if** num\_b<num\_c  
 **write** num\_b **to** A  
 **read** num\_b **from** B  
 **else**  
 **write** num\_c **to** A  
 **read** num\_c **from** C  
 **end if**  
 **else if** next\_num(b) >= num\_b **and** next\_num(c) < num\_c  
 **while** next\_num(b) >= num\_b **do**  
 **if** (num\_c>num\_b)  
 **write** num\_b **to** A  
 **read** num\_b **from** B  
 **else**  
 **write** num\_c **to** A  
 **read** num\_c **from** C  
 **while** next\_num(b) >= num\_b **do**  
 **write** num\_b **to** A  
 *read* num\_b **from** B

**end while**  
  **end if**

**end while**  
 **else if** next\_num(c) >= num\_c **and** next\_num(b) < num\_b  
 **while** next\_num(c) >= num\_c **do**  
 **if**(num\_b>num\_c)  
 **write** num\_c **to** A  
 **read** num\_c **from** C  
 **else**  
 **write** num\_b **to** A

**read** num\_b **from** B  
 **while** next\_num(c) >= num\_c **do**  
 **write** num\_c **to** A

**read** num\_c from c

**end if  
 end while  
 else**  
 **if** num\_b<num\_c  
 **write** num\_b **to** A

**read** num\_b **from** B  
 **write** num\_c **to** A

**read** num\_c **from** C  
 **else**  
 **write** num\_c **to** A

**read** num\_c **from** C  
 **write** num\_b **to** A

**read** num\_b **from** B

**end if**

**end if**

**end while**

**close** A, B, C

**function** split\_file(A, B, C):  
 number\_of\_seq=1  
 **open** A **for** reading**,** **open** B for writing **open** C for writing  
 **read** current **from** A  
 **write** current **to** B  
 **while** current **do**  
 **if** next\_num(a)<current  
 number\_of\_seq+=1

**end if**

**read** current **from** A  
 **if** number\_of\_seq%2 == 1  
 **write** current **to** B  
  **else**  
 write current to C

**end if**

**end while**

**close** A, B, C  
 **return** (number\_of\_seq - 1)

**function** next\_num(file)  
 pos = file.tell()

**read** num **from** file  
 file.seek(pos)  
 **return** num

## Програмна реалізація алгоритму

### Вихідний код

Базовий алгоритм природного сортування:

Файл main.py:

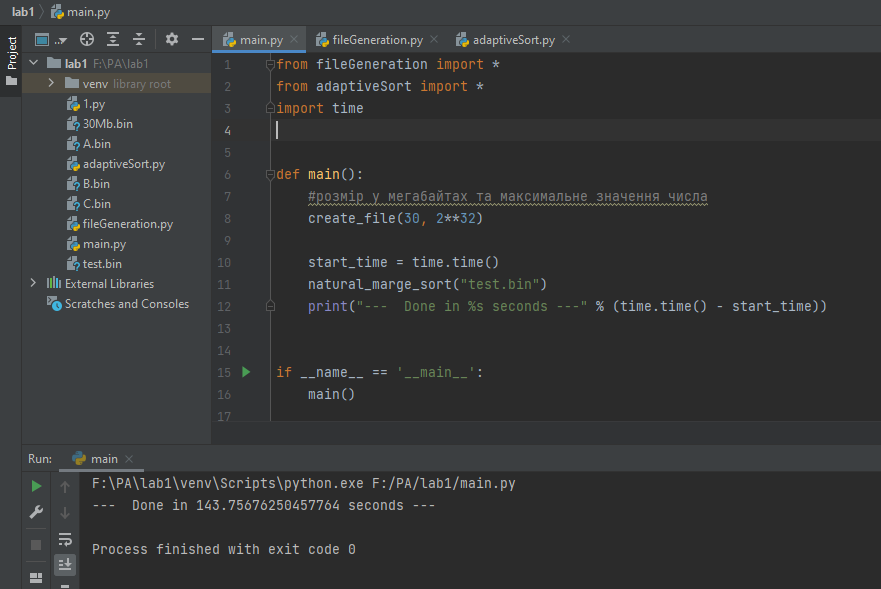
from fileGeneration import \*  
from adaptiveSort import \*  
import time  
  
  
def main():  
 #розмір у мегабайтах та максимальне значення числа  
 create\_file(30, 2\*\*32)  
  
 start\_time = time.time()  
 natural\_marge\_sort("test.bin")  
 print("--- Done in %s seconds ---" % (time.time() - start\_time))  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Файл fileGeneration.py

import random  
  
  
def create\_file(n, max):  
 with open("test.bin", "wb") as file:  
 for i in range(0, int((n\*1024\*1024)/32)):  
 file.write(random.randint(0, max+1).to\_bytes(32, byteorder='big'))  
  
  
def display\_file(name):  
 with open(name, "rb") as file:  
 num = file.read(32)  
 while num:  
 print(int.from\_bytes(num, "big"))  
 num = file.read(32)

Файл adaptiveSort.py

import shutil  
  
  
def next\_num(file):  
 pos = file.tell()  
 num = file.read(32)  
 file.seek(pos)  
 return num  
  
  
def split\_file(name\_a, name\_b, name\_c):  
 number\_of\_seq=1  
 with open(name\_a, "rb") as a, open(name\_b, "wb") as b, open(name\_c, "wb") as c:  
 current = a.read(32)  
 b.write(current)  
 while current:  
 if next\_num(a)<current:  
 number\_of\_seq+=1  
 current = a.read(32)  
 if number\_of\_seq%2 == 1:  
 b.write(current)  
 else:  
 c.write(current)  
 return number\_of\_seq-1  
  
  
def merge\_file(name\_a, name\_b, name\_c):  
 with open(name\_a, "wb") as a, open(name\_b, "rb") as b, open(name\_c, "rb") as c:  
 num\_b =b.read(32)  
 num\_c =c.read(32)  
  
 while num\_b or num\_c:  
  
 if num\_b and not num\_c:  
 while num\_b:  
 a.write(num\_b)  
 num\_b = b.read(32)  
  
 elif num\_c and not num\_b:  
 while num\_c:  
 a.write(num\_c)  
 num\_c = c.read(32)  
  
 elif next\_num(b) >= num\_b and next\_num(c) >= num\_c:  
 if num\_b<num\_c:  
 a.write(num\_b)  
 num\_b = b.read(32)  
 else:  
 a.write(num\_c)  
 num\_c =c.read(32)  
  
 elif next\_num(b) >= num\_b and next\_num(c) < num\_c:  
 while next\_num(b) >= num\_b:  
 if (num\_c>num\_b):  
 a.write(num\_b)  
 num\_b = b.read(32)  
 else:  
 a.write(num\_c)  
 num\_c=c.read(32)  
 while next\_num(b) >= num\_b:  
 a.write(num\_b)  
 num\_b = b.read(32)  
  
 elif next\_num(c) >= num\_c and next\_num(b) < num\_b:  
 while next\_num(c) >= num\_c:  
 if(num\_b>num\_c):  
 a.write(num\_c)  
 num\_c = c.read(32)  
 else:  
 a.write(num\_b)  
 num\_b = b.read(32)  
 while next\_num(c) >= num\_c:  
 a.write(num\_c)  
 num\_c = c.read(32)  
  
 else:  
 if num\_b<num\_c:  
 a.write(num\_b)  
 num\_b = b.read(32)  
 a.write(num\_c)  
 num\_c = c.read(32)  
 else:  
 a.write(num\_c)  
 num\_c =c.read(32)  
 a.write(num\_b)  
 num\_b = b.read(32)  
  
  
def natural\_marge\_sort(file\_name):  
 is\_sorted = False  
 while not is\_sorted:  
 num = split\_file(file\_name, "B.bin", "C.bin")  
 if(num!=1):  
 merge\_file(file\_name, "B.bin", "C.bin")  
 else:  
 shutil.copyfile("B.bin", file\_name)  
 is\_sorted = True



Висновок

При виконанні даної лабораторної роботи було досліджено алгоритми зовнішнього сортування, зокрема природнє або адаптивне сортування злиттям, який було реалізовано.

Основні підзадачі алгоритму – пошук серій, їх розподіл по двом допоміжних файлах та злиття їх у поатковий, з об’єднанням серій. Перевагою алгоритму є те, що для сортування файлу довільного розміру необхідно тримати в оперативній пам’яті одночасно не більше 2 ключів, однак значним недоліком є швидкість його роботи. Для покращення швидкодії можна модифікувати алгоритм, наприклад, попередньо частково відсортувавши файл частинами. Якщо без такої модифікації час на обробку файлу розміром 30Мб був 140 секунд, то з нею за той же час було відсортовано 100Мб

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%.