**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)

*Бабашев Олексій Дмитрович ІП-13*

**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 |

# Виконання

## Псевдокод алгоритму

**ПОЧАТОК** NaturalSort()

**ПОКИ** isSorted()!=True:

divide()

merge()

Bfile = open(self.Bpath, "wb")

Cfile = open(self.Cpath, "wb")

Bfile.close()

Cfile.close()

**КІНЕЦЬ** NaturalSort()

**ПОЧАТОК** divide()

Afile = open(self.Apath, "rb")

Bfile = open(self.Bpath, "wb")

Cfile = open(self.Cpath, "wb")

flag = True

current = Afile.read(4)

next = Afile.read(4)

**ПОКИ** current:

**ЯКЩО** flag == True:

Bfile.write(current)

**ІНАКШЕ**:

Cfile.write(current)

**ЯКЩО** current > next:

**ЯКЩО** flag == True:

flag = False

**ІНАКШЕ**:

flag = True

current = next

next = Afile.read(4)

Afile.close()

Bfile.close()

Cfile.close()

**КІНЕЦЬ** divide()

**ПОЧАТОК** merge()

Afile = open(self.Apath, "wb")

Bfile = open(self.Bpath, "rb")

Cfile = open(self.Cpath, "rb")

Bcurrent = Bfile.read(4)

Ccurrent = Cfile.read(4)

Bnext = Bfile.read(4)

Cnext = Cfile.read(4)

**ПОКИ** Ccurrent and Bcurrent:

**ЯКЩО** Bcurrent <= Bnext and Ccurrent <= Cnext:

**ЯКЩО** Bcurrent <= Ccurrent:

Afile.write(Bcurrent)

Bcurrent = Bnext

Bnext = Bfile.read(4)

**ІНАКШЕ**:

Afile.write(Ccurrent)

Ccurrent = Cnext

Cnext = Cfile.read(4)

**ІНАКШЕ** **ЯКЩО** Bcurrent >= Bnext and Ccurrent <= Cnext:

**ПОКИ** Ccurrent <= Cnext:

**ЯКЩО** Bcurrent <= Ccurrent:

Afile.write(Bcurrent)

Bcurrent = Bnext

Bnext = Bfile.read(4)

**ПОКИ** Ccurrent <= Cnext:

Afile.write(Ccurrent)

Ccurrent = Cnext

Cnext = Cfile.read(4)

Afile.write(Ccurrent)

Ccurrent = Cnext

Cnext = Cfile.read(4)

**ПЕРЕРВАТИ**

**ІНАКШЕ**:

Afile.write(Ccurrent)

Ccurrent = Cnext

Cnext = Cfile.read(4)

**ІНАКШЕ** **ЯКЩО** Ccurrent >= Cnext and Bcurrent <= Bnext:

**ПОКИ** Bcurrent <= Bnext:

**ЯКЩО** Ccurrent <= Bcurrent:

Afile.write(Ccurrent)

Ccurrent = Cnext

Cnext = Cfile.read(4)

**ПОКИ** Bcurrent <= Bnext:

Afile.write(Bcurrent)

Bcurrent = Bnext

Bnext = Bfile.read(4)

Afile.write(Bcurrent)

Bcurrent = Bnext

Bnext = Bfile.read(4)

**ПЕРЕРВАТИ**

**ІНАКШЕ**:

Afile.write(Bcurrent)

Bcurrent = Bnext

Bnext = Bfile.read(4)

**ІНАКШЕ**:

**ЯКЩО** Ccurrent <= Bcurrent:

Afile.write(Ccurrent)

Afile.write(Bcurrent)

**ІНАКШЕ**:

Afile.write(Bcurrent)

Afile.write(Ccurrent)

Ccurrent = Cnext

Cnext = Cfile.read(4)

Bcurrent = Bnext

Bnext = Bfile.read(4)

**ЯКЩО** not Bcurrent(пустий) and Ccurrent(заповнений):

**ПОКИ** Ccurrent:

Afile.write(Ccurrent)

Ccurrent = Cnext

Cnext = Cfile.read(4)

**ІНАКШЕ** **ЯКЩО** not Ccurrent(пустий)and Bcurrent(заповнений):

**ПОКИ** Bcurrent:

Afile.write(Bcurrent)

Bcurrent = Bnext

Bnext = Bfile.read(4)

Afile.close()

Bfile.close()

Cfile.close()

**КІНЕЦЬ** merge()

**ПОЧАТОК** isSorted()

Afile = open(self.Apath,"rb")

current = Afile.read(4)

next = Afile.read(4)

while next:

if current > next:

Afile.close()

return False

current = next

next = Afile.read(4)

Afile.close()

**ПОВЕРНУТИ** True

**КІНЕЦЬ** isSorted()

## Програмна реалізація алгоритму

### Вихідний код

main.py

*from* NaturalSort *import* NaturalSorting  
  
*#from random import randint  
# def CreateFile(path: str, numOfElements: int, maxElement: int):  
# with open(path, "wb") as file:  
# for i in range(numOfElements):  
# file.write(randint(0, maxElement).to\_bytes(4, byteorder="big"))  
  
  
  
def* main():  
  
 sorting = NaturalSorting("A.bin", "B.bin", "C.bin")  
 sorting.CopyFile("10Mb.bin")  
  
 sorting.ShowFirstElementsOfArray("A.bin")  
 *print*("====================================================\n")  
 sorting.NaturalSort()  
 sorting.ShowFirstElementsOfArray("A.bin")  
  
 *if* sorting.isSorted():  
 *print*("array is succesfully sorted!\n")  
  
*if* \_\_name\_\_ == "\_\_main\_\_":  
 main()

NaturalSort.py

*import* shutil  
  
*class* NaturalSorting:  
 *def \_\_init\_\_*(*self*, Apath: *str*, Bpath: *str*, Cpath: *str*):  
 *self*.Apath = Apath  
 *self*.Bpath = Bpath  
 *self*.Cpath = Cpath  
  
 *def* CopyFile(*self*, path: *str*):  
 shutil.copy(path, *self*.Apath)  
  
 *def* NaturalSort(*self*):  
 *while self*.isSorted()!=*True*:  
 *self*.divide()  
 *self*.merge()  
  
 Bfile = *open*(*self*.Bpath, "wb")  
 Cfile = *open*(*self*.Cpath, "wb")  
 Bfile.close()  
 Cfile.close()  
  
 *def* divide(*self*):  
 Afile = *open*(*self*.Apath, "rb")  
 Bfile = *open*(*self*.Bpath, "wb")  
 Cfile = *open*(*self*.Cpath, "wb")  
  
 flag = *True* current = Afile.read(4)  
 next = Afile.read(4)  
 *while* current:  
 *if* flag:  
 Bfile.write(current)  
 *else*:  
 Cfile.write(current)  
 *if* current > next:  
 flag = *not* flag  
 *# if flag:  
 # flag = False  
 # else:  
 # flag = True* current = next  
 next = Afile.read(4)  
 Afile.close()  
 Bfile.close()  
 Cfile.close()  
  
 *def* merge(*self*):  
 Afile = *open*(*self*.Apath, "wb")  
 Bfile = *open*(*self*.Bpath, "rb")  
 Cfile = *open*(*self*.Cpath, "rb")  
 Bcurrent = Bfile.read(4)  
 Ccurrent = Cfile.read(4)  
 Bnext = Bfile.read(4)  
 Cnext = Cfile.read(4)  
 *while* Ccurrent *and* Bcurrent:  
 *if* Bcurrent <= Bnext *and* Ccurrent <= Cnext:  
 *if* Bcurrent <= Ccurrent:  
 Afile.write(Bcurrent)  
 Bcurrent = Bnext  
 Bnext = Bfile.read(4)  
 *else*:  
 Afile.write(Ccurrent)  
 Ccurrent = Cnext  
 Cnext = Cfile.read(4)  
 *elif* Bcurrent >= Bnext *and* Ccurrent <= Cnext:  
 *while* Ccurrent <= Cnext:  
 *if* Bcurrent <= Ccurrent:  
 Afile.write(Bcurrent)  
 Bcurrent = Bnext  
 Bnext = Bfile.read(4)  
 *while* Ccurrent <= Cnext:  
 Afile.write(Ccurrent)  
 Ccurrent = Cnext  
 Cnext = Cfile.read(4)  
 Afile.write(Ccurrent)  
 Ccurrent = Cnext  
 Cnext = Cfile.read(4)  
 *break  
 else*:  
 Afile.write(Ccurrent)  
 Ccurrent = Cnext  
 Cnext = Cfile.read(4)  
 *elif* Ccurrent >= Cnext *and* Bcurrent <= Bnext:  
 *while* Bcurrent <= Bnext:  
 *if* Ccurrent <= Bcurrent:  
 Afile.write(Ccurrent)  
 Ccurrent = Cnext  
 Cnext = Cfile.read(4)  
 *while* Bcurrent <= Bnext:  
 Afile.write(Bcurrent)  
 Bcurrent = Bnext  
 Bnext = Bfile.read(4)  
 Afile.write(Bcurrent)  
 Bcurrent = Bnext  
 Bnext = Bfile.read(4)  
 *break  
 else*:  
 Afile.write(Bcurrent)  
 Bcurrent = Bnext  
 Bnext = Bfile.read(4)  
 *else*:  
 *if* Ccurrent <= Bcurrent:  
 Afile.write(Ccurrent)  
 Afile.write(Bcurrent)  
 *else*:  
 Afile.write(Bcurrent)  
 Afile.write(Ccurrent)  
 Ccurrent = Cnext  
 Cnext = Cfile.read(4)  
 Bcurrent = Bnext  
 Bnext = Bfile.read(4)  
 *if not* Bcurrent *and* Ccurrent:  
 *while* Ccurrent:  
 Afile.write(Ccurrent)  
 Ccurrent = Cnext  
 Cnext = Cfile.read(4)  
 *elif not* Ccurrent *and* Bcurrent:  
 *while* Bcurrent:  
 Afile.write(Bcurrent)  
 Bcurrent = Bnext  
 Bnext = Bfile.read(4)  
 Afile.close()  
 Bfile.close()  
 Cfile.close()  
  
 *def* ShowFirstElementsOfArray(*self*, path: *str*):  
 *with open*(path, "rb") *as* file:  
 s = ""  
 *for* i *in range*(1000):  
 s = s + *str*(*int*.from\_bytes(file.read(4), byteorder="big")) + " "  
 *print*(s)  
  
 *def* isSorted(*self*):  
 Afile = *open*(*self*.Apath,"rb")  
 current = Afile.read(4)  
 next = Afile.read(4)  
 *while* next:  
 *if* current > next:  
 Afile.close()  
 *return False* current = next  
 next = Afile.read(4)  
 Afile.close()  
 *return True*

Висновок

При виконанні даної лабораторної роботи вивчив та дослідив на практиці основні алгоритми зовнішнього сортування, ознайомився з можливостями їх модифікацій.

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%.