**Міністерство освіти і науки України**

**Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"**

**Факультет інформатики та обчислювальної техніки Кафедра ІПІ**

Звіт

з лабораторної роботи № 2 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

„ **Метод декомпозиції. Пошук інверсій**”

**Виконав(ла)**

Скрипець Ольга Олександрівна

(шифр, прізвище, ім'я, по батькові)

**Перевірив**

(прізвище, ім'я, по батькові)

Київ 2022

**Формальна постановка задачі**

За допомогою методу декомпозиції розробити алгоритм, який буде розв’язувати наступну задачу. Вхідні дані. Матриця D натуральних чисел розмірності u\*m, де u — ці кількість користувачів, m — кількість фільмів. Кожний елемент матриці D[i, j] вказує на позицію фільму j в списку вподобань користувача i. Іншим вхідним елементом є x — номер користувача, з яким будуть порівнюватись всі інші користувачі. Вихідні дані. Список з впорядкованих за зростанням другого елементу пар (i, c), де i — номер користувача, c — число, яке вказує на степінь схожості вподобань користувачів x та c (кількість інверсій).

**Формат вхідних/вихідних даних**

Розроблена програма повинна зчитувати вхідні дані з файлу заданого формату та записувати дані у файл заданого формату. У вхідному файлі зберігається матриця вподобань всіх користувачів D. Номер користувача Х, з яким відбувається порівняння всіх інших користувачів, передається аргументом виклику програми через командний рядок. Вхідний файл представляє собою текстовий файл із U+1 рядків. Перший рядок містить два числа: U та M, де U — кількість користувачів, M — кількість фільмів. Кожен наступний рядок представляє список вподобань (хіт-парад) фільмів відповідних користувачів і містить M+1 число, розділених пробілом. Перше число в рядку є номером користувача (від 1 до U). Решта M чисел є номерами фільмів 1 ,..., M у хіт-параді відповідного користувача. Вихідний файл представляє також текстовий файл із U рядків. Перший рядок містить одне число — номер користувача Х, з яким відбувалось порівняння всіх інших користувачів. Далі йде U-1 рядків, кожен з яких містить два числа через пробіл: номер користувача i та число c, яке визначає степінь подібності списків вподобань користувачів x та і. Рядки з парами i та c впорядковані за значенням елементу c.

**Реалізація коду**

import sys

def mergeSort(arr):

n = len(arr)

temp\_arr = [0]\*n

return \_mergeSort(arr, temp\_arr,

0, n - 1)

def \_mergeSort(arr, temp\_arr, left, right):

inv\_count = 0

if left < right:

mid = (left + right)//2

inv\_count += \_mergeSort(arr, temp\_arr,

left, mid)

inv\_count += \_mergeSort(arr, temp\_arr,

mid + 1, right)

inv\_count += merge(arr, temp\_arr,

left, mid, right)

return inv\_count

def merge(arr, temp\_arr, left, mid, right):

i = left

j = mid + 1

k = left

inv\_count = 0

while i <= mid and j <= right:

if arr[i] <= arr[j]:

temp\_arr[k] = arr[i]

k += 1

i += 1

else:

temp\_arr[k] = arr[j]

inv\_count += (mid-i + 1)

k += 1

j += 1

while i <= mid:

temp\_arr[k] = arr[i]

k += 1

i += 1

while j <= right:

temp\_arr[k] = arr[j]

k += 1

j += 1

for loop\_var in range(left, right + 1):

arr[loop\_var] = temp\_arr[loop\_var]

return inv\_count

def newarr(firstarr, secondarr):

newarr = [0 for i in range(len(firstarr))]

for i in range(len(firstarr)):

newarr[firstarr[i] - 1] = secondarr[i]

return mergeSort(newarr)

# Зчитуємо вхідні дані з файлу

file\_path = "./input\_1000\_5.txt"

with open(file\_path, "r") as input\_file:

u, m = map(int, input\_file.readline().split())

matrix = [[] for i in range(u)]

for i in range(u):

row = list(map(int, input\_file.readline().split()))

matrix[row[0]-1] = row[1:]

# Зчитуємо номер користувача x, з яким порівнюємо

x = 289

# Підраховуємо кількість інверсій для кожного користувача

similarities = []

for i in range(u):

if i+1 != x:

inversions = newarr(matrix[i],matrix[x-1])

similarities.append((i+1, inversions))

# Сортуємо користувачів за зростанням ступеня подібності

similarities.sort(key=lambda x: x[1])

# Записуємо результат у вихідний файл

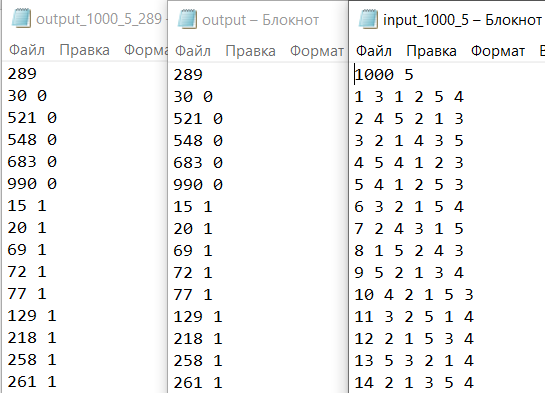
with open("./output.txt", "w") as output\_file:

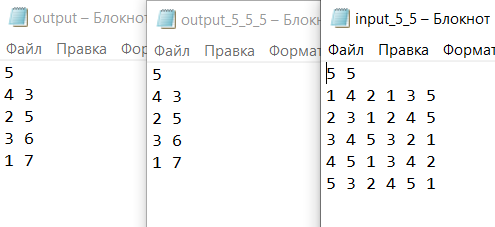
output\_file.write(str(x) + "\n")

for similarity in similarities:

output\_file.write(str(similarity[0]) + " " + str(similarity[1]) + "\n")

**Приклад вхідного/вихідного файлу**

****

****