Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра Автоматизованих Систем Обробки Інформації та Управління

Лабораторна робота № 3

з дисципліни «Теорія алгоритмів»

на тему:

**"Метод декомпозиції. Пошук інверсій"**

Виконала:

студентка гр. ІС-02

Павлущенко Ольга

Викладач:

Новікова П. А.

Київ – 2020

**Завдання**

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).

**Програмний код**

def main():

input\_file = input("Enter the file name: ") #введення назви файла для відкриття

f = open(input\_file) #читання файлу

user\_X = int(input("Enter the user number: ")) #користувач для порівняння

arr = f.readlines()

del arr[0]

arr = [[int(n) for n in x.split()]for x in arr] #зчитаний масив

A = arr[user\_X - 1][1:] #вподобання обраного користувача

n1 = len(arr)

comparison\_rating=[[0 for i in range(2)]for j in range(n1)] #рейтинг порівнянь вподобань

for i in range(0,n1):

if arr[i][1:] != A:

B = arr[i][1:]

cout = MergeSortInvCount(user\_preferences(A,B)) #підрахунок інверсій

comparison\_rating[i][1] = arr[i][0]

comparison\_rating[i][0] = cout

MergeSort(comparison\_rating,0,len(comparison\_rating)-1) #сортування за кількістю інверсій

file=open("is02\_OlhaPavlushchenko\_03\_output.txt", "w") #виведення в файл

file.write(str(user\_X) + '\n')

for i in range(len(comparison\_rating)):

if(comparison\_rating[i][0] != 0 and comparison\_rating[i][1] != 0):

file.write(str(comparison\_rating[i][1]) + ' ' + str(comparison\_rating[i][0]) + '\n')

file.write(str(user\_X) + '\n')

file.close()

def MergeSortInvCount(arr): #MergeSort з підрахунком інверсій

inv = 0 #інверсії

if len(arr) > 1:

q = len(arr)//2; #середина масиву

L = arr[:q] #перша половина масиву

R = arr[q:] #друга половина масиву

inv += MergeSortInvCount(L) #рекурсивний виклик MergeSort для сортування першої половини

inv += MergeSortInvCount(R) #рекурсивний виклик MergeSort для сортування другої половини

i = k = 0

j = 0

while i < len(L) and j < len(R):

if L[i] <= R[j]:

arr[k] = L[i]

i=i+1

k=k+1

else:

arr[k] = R[j]

j=j+1

inv = inv + (len(L) - i)

k=k+1

while i < len(L):

arr[k] = L[i]

i += 1

k += 1

while j < len(R):

arr[k] = R[j]

j += 1

k += 1

return inv

def MergeSort(arr, start, end): #MergeSort

if len(arr) > 1:

q=len(arr)//2; #середина масиву

L=arr[:q] #перша половина масиву

R=arr[q:] #друга половина масиву

MergeSort(L, start, q)

MergeSort(R, q+1, end)

i = k = 0

j = 0

while i < len(L) and j < len(R):

if L[i] <= R[j]:

arr[k]=L[i]

i=i+1

k=k+1

else:

arr[k] = R[j]

j=j+1

k=k+1

while i < len(L):

arr[k] = L[i]

i += 1

k += 1

while j < len(R):

arr[k] = R[j]

j += 1

k += 1

return arr

def user\_preferences(User\_X, another\_User):

arr3=[0 for i in range(len(User\_X))]

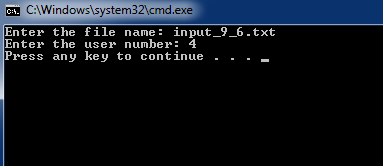
for i in range(len(User\_X)):

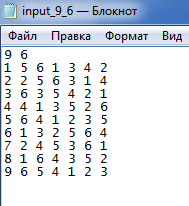
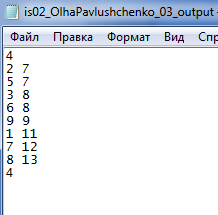
arr3[User\_X[i]-1]=another\_User[i]

return arr3

main()

**Результати**

**Власний файл**



**4** **4 1 3 5 2 6**

*1* 5 6 1 3 4 2

*6*  1 3 2 5 6 4

**К-сть інверсій: 13**

**К-сть інверсій: 12**

**К-сть інверсій: 7**

**К-сть інверсій: 8**

**К-сть інверсій: 9**

5 2 4 6 1 3

6 5 4 1 3 2

4 6 5 2 3 1

3 6 2 1 5 4

*9*  6 5 4 1 2 3

*8*  1 6 4 3 5 2

*7*  2 4 5 3 6 1

*5*  6 4 1 2 3 5

4 3 1 6 2 5

**К-сть інверсій: *8***

3 2 5 6 4 1

*3* 6 3 5 4 2 1

**К-сть інверсій: *7***

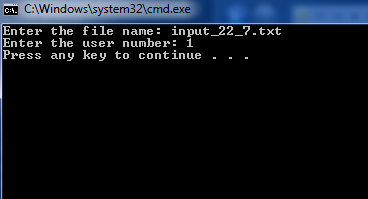
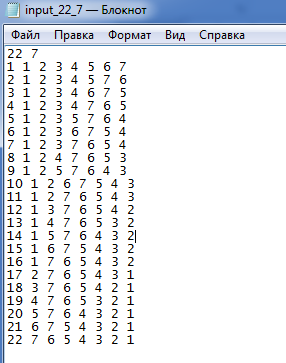
5 1 6 2 3 4

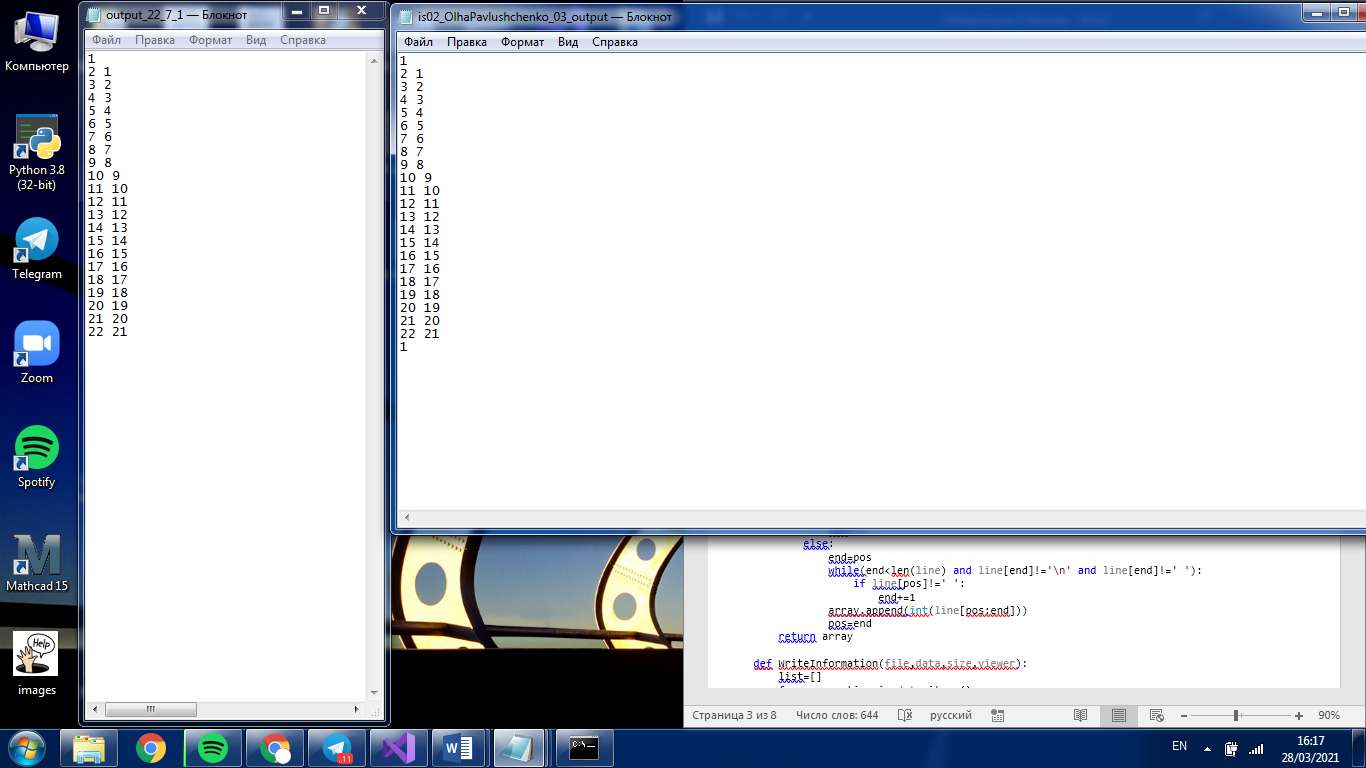
*2* 2 5 6 3 1 4

6 4 1 5 3 2

**К-сть інверсій: *11***

(1,2), (1,3), (1,4), (1,5), (1,6), (2,3), (2,5), (2,6), (4,5), (4,6), (5,6)

**Мудл файл**



**Висновок**

У ході лабораторної роботи було реалізовано алгоритм Merge Sort для підрахунку інверсій (пошуку степені схожості вподобань обраного користувача та інших). В основні алгоритму лежить метод декомпозиції. Метод можна описати наступним чином: розділення задачі на підзадачі, їх рекрсивний розв’язок, тобто сортування методом злиття, та злиття відсортованих послідовностей.

Отриманий список вподобань було відсортовано за зростанням, використовуючи Merge Sort. Результати виведено в файл. Достовірність програми первірено на власних даних та даних з платформи moodle.