Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Катедра ІПІ

Звіт

з лабораторної роботи №2 з дисципліни
«Алгоритми та структури даних 2. Структури даних»

„Метод декомпозиції. Пошук інверсій”

Виконав *ІП-11 Лесів Владислав Ігорович* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Халус Олена Андріївна* _____
(прізвище, ім'я, по батькові)

Київ 2022

Лабораторна робота 2

Метод декомпозиції. Пошук інверсій

Мета – вивчити метод декомпозиції і пошук інверсій в масивах даних.

Завдання.

Існує веб сервіс, який надає своїм користувачам можливість перегляду фільмів онлайн. Періодично система надає нові рекомендації користувачам — які фільми, що їх користувач ще не дивився, можливо будуть йому або їй цікаві.

В основі рекомендаційного алгоритму лежить ідея, що користувачі, які подивились однакові фільми та також оцінили їх схожим чином, мають схожі смаки. Наприклад, нехай є два користувача: Аліса та Богдан. Обидва вони переглянули наступні фільми: “Зоряні війни”, “Гравітація”, “Пірати карибського моря”, “Володар перснів”, “Матриця”.

Спочатку система просить користувачів оцінити ці фільми і розташувати їх у порядку вподобання, іншими словами — створити власний хіт-парад. Так Аліса розташувала вказані фільми у порядку від найбільш до найменш вподобаного: “Пірати карибського моря”, “Володар перснів”, “Матриця”, “Гравітація”, “Зоряні війни”. Хіт-парад Богдана: “Зоряні війни”, “Володар перснів”, “Гравітація”, “Матриця”, “Пірати карибського моря”.

Після цього система може надати кількісну оцінку наскільки схожими є смаки двох користувачів. Для цього використовується алгоритм підрахунку інверсій поміж двома масивами.

Нехай A[1..n] — масив з n чисел. Якщо i<j та A[i]>A[j], то пара (i, j) — інверсія в A.

Щоб звести задачу порівняння двох хіт-парадів до задачі підрахунку інверсій у нашому прикладі, побудуємо два масиви A та B. Масив A = [1, 2, 3, 4, 5]. Масив B будується наступним чином: елементом B[j] є число, яке відповідає позиції фільму в хіт-параді Богдана, який в хіт-параді Аліси посідав місце j. Наприклад, j = 1 у хіт-параді Аліси відповідає фільму “Пірати карибського моря”. Цей фільм в списку Богдана стоїть на позиції 5, тому B[1] = 5. Загалом отримуємо масив B = [5, 2, 4, 3, 1].

Масив B = [5, 2, 4, 3, 1] має наступні інверсії (вказуються індекси елементів, а не їх значення): (1,2), (1,3), (1,4), (1,5), (2,5), (3,4), (3,5), (4,5). Загалом 8 інверсій. І це число вказує наскільки сильно відрізняється список вподобань Аліси від списку вподобань Богдана. Ми порахували віддаленість списку Аліси від списку Богдана. Якщо порахувати цю відстань в іншому напрямку, то чи буде вона такою самою? Тобто визначити кількість інверсій в списку Аліси по відношенню до списку Богдана.

Сервіс перегляду фільмів онлайн має базу даних D вподобань користувачів. Ця база є матрицею.

Рядки цієї матриці відповідають користувачам, а стовпці — фільмам. Її розмірність u*m, де u — це кількість користувачів, m — кількість фільмів. Кожний елемент матриці D[i, j] вказує на позицію фільму j в списку вподобань користувача i. Для спрощення припускаємо, що всі користувачі переглянули всі фільми.

Тепер щоб визначити наскільки подібні смаки деякого користувача x до смаків інших користувачів, система попарно порівнює списки вподобань x та всіх інших користувачів *i* не дорівнює *x*: за вказаним вище принципом підраховується кількість інверсій у масиві D[x] відносно масиву D[i].

Визначене число інверсій буде кількісною оцінкою наскільки смаки x є близькими до смаків кожного i — чим менше значення цього числа, тим більш подібними є смаки двох користувачів.

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).

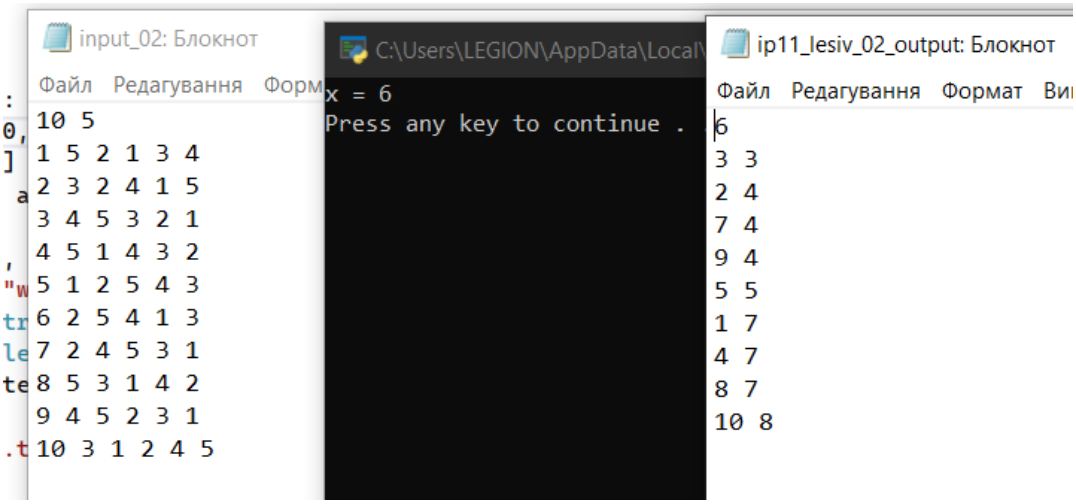
Виконання мовою Python

```
def reading(filename):
    with open(filename, "r") as inFile:
        u, m=map(int, inFile.readline().split())
        d=[]
        for i in range(u):
            d.append([int(j) for j in inFile.readline().split()])
    return u,m,d

def allDataInv(u,m,x,d):
    b, bRev=[0 for i in range(m)], [0 for i in range(m)]    #списки віддалення уподобань глядача 1 від глядача 2 і навпаки
    invArr=[]
    for i in range(u):
        if d[i][0]==x:
            ix=i
    for i in range(u):
        if d[i][0]!=x:
            for j in range(1,m+1):
                b[d[ix][j]-1]=d[i][j]
                bRev[d[i][j]-1]=d[ix][j]
            sArr,c=countInv(b)
            sArrRev,cRev=countInv(bRev)
            if c==cRev:                          #переконуємося, що кількість інверсій в обох випадках незмінна
                invArr.append([d[i][0],c])
    return invArr

def countInv(a):
    if len(a)==1:
        return a, 0
    else:
        l, x=countInv(a[:len(a)//2])
        r, y=countInv(a[len(a)//2:])
        a, z=countSplitInv(a, l, r)
        return a,x+y+z

def countSplitInv(a,l,r):
    l.append(max(a)*2)
    r.append(max(a)*2)
    i,j,c=0,0,0
    for k in range(len(a)):
        if l[i]<=r[j]:
            a[k]=l[i]
            i+=1
        else:
            a[k]=r[j]
            j+=1
            c+=(len(l)-i-1)
    return a,c

def bubbleSort(arr):
    n = len(arr)
    for i in range(n-1):
        for j in range(0, n-i-1):
            if arr[j][1] > arr[j + 1][1] :
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
```

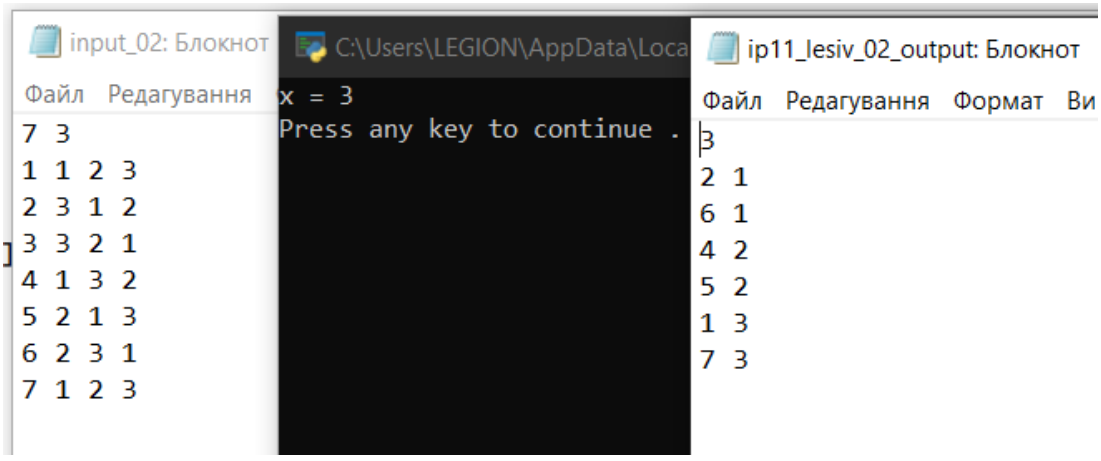
```
def writing(x, filename, invArr):
    with open(filename,"w") as outFile:
        outFile.write(str(x)+"\n")
        for i in range(len(invArr)):
            outFile.write(str(invArr[i][0])+" "+str(invArr[i][1])+"\n")

u,m,d=reading("input_02.txt")
x=int(input("x = "))
invArr=allDataInv(u,m,x,d)
bubbleSort(invArr)
writing(x, "ip11_lesiv_02_output.txt", invArr)
```

Випробування алгоритму.

1)



2)



Висновок.

При виконанні даної лабораторної роботи я вивчив метод декомпозиції і пошук інверсій. У результаті лабораторної роботи були проаналізовані та програмно реалізовані алгоритми мовою Python, які виконують задачу відповідно до постановки, використовуючи метод декомпозиції. Використовуючи написану програму та порівнюючи результати аналітично

отримуємо такий результат: кількість інверсій не залежить від того, у якому напрямку (у нашій задачі – відносно якого глядача) рахувати умовну відстань між уподобаннями.