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 № 2 з дисципліни

Алгоритми та структури даних 2. Структури даних»

«Метод декомпозиції. Пошук інверсій»

Варіант **3**

**Виконав студент:** ІП-з21 Гавриленко Даяна Юріївна

**Перевірив:** Халус Олена Андріївна

Київ 2023

**2.1 Мета роботи**

Метою роботи є вивчення та практичне використання метода декомпозиції для пошуку інверсій у списку чисел.

**2.2 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).

**2.3 Псевдокод**

функція підрахунку\_інверсій(список):

якщо довжина(список) <= 1:

повернути 0

середина = довжина(список) / 2

лівий\_список = підрахунок\_інверсій(підсписок зі списку від 0 до середина - 1)

правий\_список = підрахунок\_інверсій(підсписок зі списку від середина до кінця)

злиті\_інверсії = злити\_списки\_та\_порахувати\_інверсії(список, лівий\_список, правий\_список)

повернути лівий\_список + правий\_список + злиті\_інверсії

функція злиття\_списки\_та\_порахувати\_інверсії(список, лівий\_список, правий\_список):

новий\_список = порожній\_список

інверсії = 0

індекс\_лівого\_списку = 0

індекс\_правого\_списку = 0

доки індекс\_лівого\_списку < довжина(лівий\_список) і індекс\_правого\_списку < довжина(правий\_список):

якщо лівий\_список[індекс\_лівого\_списку] <= правий\_список[індекс\_правого\_списку]:

додати\_елемент(новий\_список, лівий\_список[індекс\_лівого\_списку])

індекс\_лівого\_списку += 1

інакше:

додати\_елемент(новий\_список, правий\_список[індекс\_правого\_списку])

індекс\_правого\_списку += 1

інверсії += довжина(лівий\_список) - індекс\_лівого\_списку

додати\_елементи(новий\_список, лівий\_список, починаючи з індексу індекс\_лівого\_списку)

додати\_елементи(новий\_список, правий\_список, починаючи з індексу індекс\_правого\_списку)

повернути інверсії

функція додати\_елемент(список, елемент):

додати елемент у список

функція додати\_елементи(список\_приймач, список\_додати, починаючи\_з\_індексу):

для кожного елемента від починаючи\_з\_індексу до довжина(список\_додати):

додати\_елемент(список\_приймач, список\_додати[елемент])

основна\_функція:

зчитати вхідні дані

список\_чисел = перетворити рядок\_в\_список(вхідний\_рядок)

кількість\_інверсій = підрахунок\_інверсій(список\_чисел)

вивести кількість\_інверсій **2.4 Програмна реалізація алгоритму**

**2.4.1 Вихідний код на мові C#**

namespace Lab1;  
  
class Program  
{  
 static void Main(string[] args)  
 {  
 if (args.Length < 2)  
 {  
 Console.WriteLine("Потрібно вказати ім'я вхідного файлу та номер користувача X як аргументи командного рядка.");  
 return;  
 }  
  
 string inputFileName = args[0];  
 int userX = int.Parse(args[1]);  
 string outputFileName = GetOutputFileName(inputFileName);  
  
 UserMatrixAnalyzer analyzer = new UserMatrixAnalyzer();  
 MatrixReader matrixReader = new MatrixReader();  
 MatrixWriter matrixWriter = new MatrixWriter();  
  
 int[,] matrix = matrixReader.ReadMatrixFromFile(inputFileName);  
 List<Tuple<int, int>> similarities = analyzer.CalculateSimilarities(matrix, userX);  
 matrixWriter.WriteResultsToFile(outputFileName, userX, similarities);  
  
 Console.WriteLine("Результати були записані у вихідний файл.");  
 }  
  
 static string GetOutputFileName(string inputFileName)  
 {  
 string directory = Path.GetDirectoryName(inputFileName);  
 string fileName = Path.GetFileNameWithoutExtension(inputFileName);  
 string outputFileName = Path.Combine(directory, $"ip-s21\_Gavrylenko\_01\_output.txt");  
 return outputFileName;  
 }  
}  
  
class MatrixReader  
{  
 public int[,] ReadMatrixFromFile(string fileName)  
 {  
 string[] lines = File.ReadAllLines(fileName);  
 string[] dimensions = lines[0].Split(' ');  
 int users = int.Parse(dimensions[0]);  
 int movies = int.Parse(dimensions[1]);  
 int[,] matrix = new int[users, movies];  
 for (int i = 0; i < users; i++)  
 {  
 string[] values = lines[i + 1].Split(' ');  
 for (int j = 0; j < movies; j++)  
 {  
 matrix[i, j] = int.Parse(values[j]);  
 }  
 }  
 return matrix;  
 }  
}  
  
class UserMatrixAnalyzer  
{  
 public List<Tuple<int, int>> CalculateSimilarities(int[,] matrix, int userX)  
 {  
 int users = matrix.GetLength(0);  
 int movies = matrix.GetLength(1);  
 List<Tuple<int, int>> similarities = new List<Tuple<int, int>>();  
 for (int i = 0; i < users; i++)  
 {  
 if (i + 1 == userX) continue;  
 int[] userA = GetPreferences(matrix, userX - 1);  
 int[] userB = GetPreferences(matrix, i);  
 int similarity = CalculateSimilarity(userA, userB);  
 similarities.Add(new Tuple<int, int>(i + 1, similarity));  
 }  
 similarities.Sort((x, y) => x.Item2.CompareTo(y.Item2));  
 return similarities;  
 }  
  
 private int[] GetPreferences(int[,] matrix, int user)  
 {  
 int movies = matrix.GetLength(1);  
 int[] preferences = new int[movies];  
 for (int i = 0; i < movies; i++)  
 {  
 preferences[i] = matrix[user, i];  
 }  
 return preferences;  
 }  
  
 private int CalculateSimilarity(int[] arr1, int[] arr2)  
 {  
 int inversions = 0;  
 for (int i = 0; i < arr1.Length - 1; i++)  
 {  
 for (int j = i + 1; j < arr1.Length; j++)  
 {  
 if ((arr1[i] > arr1[j] && arr2[i] < arr2[j]) || (arr1[i] < arr1[j] && arr2[i] > arr2[j]))  
 {  
 inversions++;  
 }  
 }  
 }  
 return inversions;  
 }  
}  
  
class MatrixWriter  
{  
 public void WriteResultsToFile(string fileName, int userX, List<Tuple<int, int>> similarities)  
 {  
 using (StreamWriter writer = new StreamWriter(fileName))  
 {  
 writer.WriteLine(userX);  
 writer.WriteLine(similarities.Count);  
 for (int i = 0; i < similarities.Count; i++)  
 {  
 writer.WriteLine($"{similarities[i].Item1} {similarities[i].Item2}");  
 }  
 }  
 }  
}

**2.4.2 Приклад роботи**

На рисунках 2.1 та 2.2 наведено приклад вхідного та вихідного файлу для U = 5 та M = 6 і користувача X = 2.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рис.2.1 Приклад вхідного файлу

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Рис. 2.2 Приклад вихідного файлу

**2.5 Висновок** За допомогою цієї лабораторної робот я навчилася використовувати метод декомпозиції, що дозволило мені розділити задачу на менші підзадачі і спростити її розв'язання та забезпечити більшу зрозумілість та модульність коду.