Лабораторна робота №1

Виконав: студент 1-го курсу ФІОТ групи ІП-92 Медведєв Михайло Євгенович

**Завдання:**

Реалізувати програмне застосування (програму), яке виконує наступні функції:

**Перша частина:**

1. Зчитування графу з вхідного файлу. На вхід подається текстовий файл наступного

вигляду:

n m

....

Тут n – кількість вершин графу (ціле число, більше нуля), m – кількість ребер графу

(ціле число, більше нуля), v i та u i – початкова та кінцева вершина ребра i (1≤ ≤n,

1≤ ≤n, цілі числа). Індексація вершин у файлі ведеться з 1. Вважається, що граф є

неорієнтованим.

Таким чином можна сказати, що граф задається у файлі списком ребер.

1. Вивід матриць інцидентності та суміжності. За вимогою користувача програма

повинна виводити матриці інцидентності та суміжності (окремі функції) на екран

та/або у текстовий файл, який вказує користувач.

1. Визначити степінь вершин графу. За запитом користувача програма на екран та/або

у файл виводить степінь усіх вершин графу. Визначити, чи граф є однорідним та

якщо так, то вказати степінь однорідності графу.

1. Визначити всі висячі та ізольовані вершини. За запитом користувача програма на

екран виводить перелік усіх висячих та ізольованих вершин графу.

**Друга частина**: на вхід програми подається вхідний файл з описом орієнтованого графу,

зі структурою, яка вказана вище.

1. Вивід матриць інцидентності та суміжності. За вимогою користувача програма

повинна виводити матриці інцидентності та суміжності (окремі функції) на екран

та/або у текстовий файл, який вказує користувач.

1. Визначити напівстепені заходу та виходу вершин графу. За запитом користувача

програма на екран та/або у файл виводить степінь усіх вершин графу. Визначити,

чи граф є однорідним та якщо так, то вказати степінь однорідності графу

Код:

**package** com.company;  
  
**import** java.io.File;  
**import** java.io.FileNotFoundException;  
**import** java.io.FileWriter;  
**import** java.io.IOException;  
**import** java.util.ArrayList;  
**import** java.util.Scanner;  
  
**public class** DS\_IP92\_LR1\_MedvedievM {  
  
 **public static void** main(String[] args) **throws** IOException {  
  
 Scanner consoleScanner = **new** Scanner(System.*in*);  
 **while** (**true**) {  
 **int** firstChoice = *getFirstChoice*(consoleScanner);  
 **if** (firstChoice == -1)  
 **break**;  
  
 File inputFile = *getFile*(firstChoice);  
 **if** (inputFile == **null**)  
 **break**;  
  
 Graph mainGraph = *getGraph*(firstChoice, inputFile);  
 *startMenu*(firstChoice, mainGraph, consoleScanner);  
 }  
 consoleScanner.close();  
 }

**private static void** startMenu(**int** choice, Graph graph, Scanner scanner) **throws** IOException {  
 **while** (**true**) {  
 System.*out*.print("Введите цифру: \n1 - вывести матрицы инцидентности и смежности\n2 - вывести степени вершин графа"  
 + ((choice == 1) ? "\n3 - вывести все висячие и изолированые вершини графа" : "")  
 + "\n -1 - выход в меню выбора графа\nВвод: ");  
  
 **int** currentChoice = scanner.nextInt();  
 String output = **null**;  
 **if** (currentChoice == -1)  
 **break**;  
 **else if** (currentChoice == 1)  
 output = graph.getStringOfIncidenceAndAdjacencyMatrixs();  
  
 **else if** (currentChoice == 2)  
 output = graph.getStringOfDegreesOfGraphVertices();  
  
 **else if** (choice == 1 && currentChoice == 3)  
 output = ((UndirectedGraph) graph).getStringOfPendentAndDisconnectedVertices();  
 **else** System.*out*.println("Неправильный ввод, попробуйте снова");  
  
 **if** (output != **null**)  
 *writeResult*(output, scanner);  
 }  
 }  
  
 **private static void** writeResult(String result, Scanner scanner) **throws** IOException {  
 System.*out*.print("Вывести в консоль (1) или в файл (2)? (введите цифру): ");  
 **int** choice = scanner.nextInt();  
 **if** (choice == 1)  
 System.*out*.println(result);  
 **else if** (choice == 2) {  
 System.*out*.print("Введите имя файла, в который хотите сохранить данные: ");  
 *writeToFile*(result, scanner.next());  
 }  
 }  
  
 **private static void** writeToFile(String text, String path) **throws** IOException {  
 FileWriter writer = **new** FileWriter("outputs/" + path);  
 writer.append(text);  
 writer.close();  
 }  
  
  
 **private static** Graph getGraph(**int** choice, File file) **throws** FileNotFoundException {  
 **if** (choice == 1)  
 **return new** UndirectedGraph(file);  
 **else  
 return new** DirectedGraph(file);  
 }  
  
  
 **private static int** getFirstChoice(Scanner scanner) {  
 System.*out*.print("Вы хотите ввести неориенториваный (1) или ориентированый (2) граф или вы хотите выйти (-1)? (введите цифру):");  
 **return** scanner.nextInt();  
 }  
  
 **private static** File getFile(**int** choise) {  
 String path;  
 **if** (choise == 1)  
 path = "inputs/neorient.txt";  
 **else if** (choise == 2)  
 path = "inputs/orient.txt";  
 **else** {  
 System.*out*.println("Wrong input");  
 **return null**;  
 }  
 **return new** File(path);  
 }  
  
}  
  
**abstract class** Graph {  
 **protected int**[][] verges;  
 **protected int** numberOfNodes, numberOfVerges;// n вершин, m ребер  
 **protected int**[][] incidenceMatrix, adjacencyMatrix;  
  
 **protected** Graph(File file) **throws** FileNotFoundException {  
 parseFile(file);  
 preSetAdjacencyMatrix();  
 preSetIncidenceMatrix();  
 }  
  
 **private void** parseFile(File file) **throws** FileNotFoundException {  
 Scanner fileScanner = **new** Scanner(file);  
 **this**.numberOfNodes = fileScanner.nextInt();  
 **this**.numberOfVerges = fileScanner.nextInt();  
 **this**.verges = **new int**[**this**.numberOfVerges][2];  
 **for** (**int** i = 0; i < **this**.numberOfVerges; i++) {  
 verges[i][0] = fileScanner.nextInt();  
 verges[i][1] = fileScanner.nextInt();  
 }  
 }  
  
 **protected void** preSetIncidenceMatrix() {  
 **this**.incidenceMatrix = **new int**[**this**.numberOfNodes][**this**.numberOfVerges];  
 }  
  
 **protected void** preSetAdjacencyMatrix() {  
 **this**.adjacencyMatrix = **new int**[**this**.numberOfNodes][**this**.numberOfNodes];  
 }  
  
 **public int**[][] getIncidenceMatrix() {  
 **return** incidenceMatrix;  
 }  
  
 **public int**[][] getAdjacencyMatrix() {  
 **return** adjacencyMatrix;  
 }  
  
  
  
 **private** String matrixToString(**int**[][] matrix, String extraText){  
 String outputText = extraText + "\n";  
  
 **for** (**int** i = 0; i < matrix.length; i++) {  
 **for** (**int** j = 0; j < matrix[0].length; j++)  
 outputText += ((matrix[i][j] >= 0) ? " " : "") + matrix[i][j] + " ";  
  
 outputText += "\n";  
 }  
 **return** outputText;  
 }  
  
 **public** String getStringOfIncidenceAndAdjacencyMatrixs() {  
 String outputText = matrixToString(**this**.getIncidenceMatrix(), "Матрица инцидентности: ") +  
 matrixToString(**this**.getAdjacencyMatrix(), "Матрица смежности: ");  
 **return** outputText;  
  
 }  
  
 **abstract** String getStringOfDegreesOfGraphVertices();  
  
  
  
}  
  
**class** UndirectedGraph **extends** Graph {  
 **protected** UndirectedGraph(File file) **throws** FileNotFoundException {  
 **super**(file);  
 }  
  
 @Override  
 **protected void** preSetIncidenceMatrix() {  
 **super**.preSetIncidenceMatrix();  
 **for** (**int** i = 0; i < **this**.numberOfNodes; i++) {  
 **for** (**int** j = 0; j < **this**.numberOfVerges; j++) {  
 **if** (**this**.verges[j][0] == i + 1 || **this**.verges[j][1] == i + 1)  
 **this**.incidenceMatrix[i][j] = 1;  
  
 **else this**.incidenceMatrix[i][j] = 0;  
 }  
 }  
 }  
  
 @Override  
 **protected void** preSetAdjacencyMatrix() {  
 **super**.preSetAdjacencyMatrix();  
 **for** (**int** i = 0; i < **this**.numberOfVerges; i++) {  
 **this**.adjacencyMatrix[**this**.verges[i][0] - 1][**this**.verges[i][1] - 1] = 1;  
 **this**.adjacencyMatrix[**this**.verges[i][1] - 1][**this**.verges[i][0] - 1] = 1;  
 }  
 }  
  
 **public** String getStringOfDegreesOfGraphVertices() {  
 String outputText = "Степени всех вершин графа: \n";  
 **int** lastDegree = -1;  
 **boolean** isUniform = **true**;  
  
 **for** (**int** i = 0; i < **this**.adjacencyMatrix.length; i++) {  
 **int** degree = 0;  
  
 **for** (**int** j = 0; j < adjacencyMatrix[0].length; j++)  
 degree += adjacencyMatrix[i][j];  
  
 **if** (i == 0)  
 lastDegree = degree;  
 **else** {  
 **if** (isUniform && lastDegree != degree)  
 isUniform = **false**;  
 }  
 outputText += "Вершина " + (i + 1) + " имеет степень " + degree + "\n";  
 }  
  
 **if** (isUniform)  
 outputText += "Граф однородный, степень однородности: " + lastDegree + "\n";  
 **return** outputText;  
 }  
  
 **public** String getStringOfPendentAndDisconnectedVertices() {  
 ArrayList<Integer> indexsOfPendentVertices = **new** ArrayList<>();  
 ArrayList<Integer> indexsOfDisconnectedVertices = **new** ArrayList<>();  
 String outputText = "";  
  
 **for** (**int** i = 0; i < **this**.adjacencyMatrix.length; i++) {  
 **int** degree = 0;  
  
 **for** (**int** j = 0; j < **this**.adjacencyMatrix[0].length; j++)  
 degree += adjacencyMatrix[i][j];  
  
 **if** (degree == 0)  
 indexsOfDisconnectedVertices.add(i + 1);  
 **else if** (degree == 1)  
 indexsOfPendentVertices.add(i + 1);  
 }  
  
  
 outputText += "Висячие указатели: ";  
 **for** (Integer indexOfPendentVertex : indexsOfPendentVertices)  
 outputText += indexOfPendentVertex + " ";  
  
 outputText += "\nИзолированые указатели: ";  
 **for** (Integer indexOfDisconnectedVertex : indexsOfDisconnectedVertices)  
 outputText += indexOfDisconnectedVertex + " ";  
 outputText += "\n";  
  
 **return** outputText;  
 }  
}  
  
**class** DirectedGraph **extends** Graph {  
 **protected** DirectedGraph(File file) **throws** FileNotFoundException {  
 **super**(file);  
 }  
  
 @Override  
 **protected void** preSetIncidenceMatrix() {  
 **super**.preSetIncidenceMatrix();  
 **for** (**int** i = 0; i < **this**.numberOfVerges; i++) {  
 **if** (**this**.verges[i][0] != **this**.verges[i][1]) {  
 **this**.incidenceMatrix[**this**.verges[i][0] - 1][i] = -1;  
 **this**.incidenceMatrix[**this**.verges[i][1] - 1][i] = 1;  
 } **else this**.incidenceMatrix[**this**.verges[i][0] - 1][i] = 2;  
 }  
 }  
  
 @Override  
 **protected void** preSetAdjacencyMatrix() {  
 **super**.preSetAdjacencyMatrix();  
 **for** (**int** i = 0; i < **this**.numberOfVerges; i++)  
 **this**.adjacencyMatrix[**this**.verges[i][0] - 1][**this**.verges[i][1] - 1] = 1;  
 }  
  
 **public** String getStringOfDegreesOfGraphVertices() {  
 String outputText = "Полустепени захода и исхода вершин графа: \n";  
  
 **for** (**int** i = 0; i < **this**.incidenceMatrix.length; i++) {  
 **int** indegree = 0, outdegree = 0;  
  
 **for** (**int** j = 0; j < **this**.incidenceMatrix[0].length; j++) {  
 **if** (**this**.incidenceMatrix[i][j] == 1 || **this**.incidenceMatrix[i][j] == 2)  
 indegree++;  
  
 **if** (**this**.incidenceMatrix[i][j] == -1 || **this**.incidenceMatrix[i][j] == 2)  
 outdegree++;  
 }  
  
 outputText += "Вершина " + (i + 1) + " имеет степень захода " + indegree + " и степень исхода " + outdegree + "\n";  
 }  
 **return** outputText;  
  
 }  
}

Результати роботи програми: 

