МIНIСТЕРСТВО ОСВIТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. Ігоря Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 2

з дисципліни “Комп’ютерна дискретна математика”

тема “Знаходження екстремальних шляхів у графі”

|  |  |  |
| --- | --- | --- |
| Виконав  студент I курсу  групи КП-62  Лук’янець Михайло Олександрович |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 2017 р.  викладач  Жабіна Валентина Валеріївна |

Київ 2017

**Завдання на лабораторну роботу**

1. Для даних викладачем графів знайти матрицю суміжності розмірності *n*, елементами якої є ваги відповідних ребер.
2. Написати програму, яка для першого графа на основі матриці суміжності знаходить найкоротші відстані від заданої вершини до решти вершин графа.
3. Написати програму, яка для другого графа на основі матриці суміжності знаходить максимальні шляхи від заданої вершини до решти вершин графа (вважати, що матриця має верхньо-трикутний вигляд).

**Вказівки щодо виконання завдання**

Програма може бути розроблена студентом на будь-якій мові програмування високого рівня: C++, C#, Java та інш. При написанні програми студент має використовувати відомі йому алгоритми, при цьому вони мають виділятися в програмі в окремі підпрограми або функції. Програма має працювати для матриці довільної розмірності.

**Опис роботи програми.**

Програма, написана мовою Java, містить наступні методи, описані у абстрактному класі:

1. *private static void printMatrix(int[][] matrix, char zero)*

Виводить матрицю

1. *private static int[] getMinDistanceVector(int aMatrix[][], int start)*

Повертає масив мінімальних відстаней від вершини start за методом Дейкстри

1. *private static void printVector(int[] arr)*

Виводить список відстаней

1. *private static void printVector(int[] arr, int[] rebuiltSequence)*

Виводить список відстаней, повернувшись до початкового порядку вершин

1. *private static int[] getNewSequence(int[][] matrix)*

Повертає нову послідовність вершин для трикутної матриці

1. *private static int[][] getTriangleMatrix(int[][] matrix, int[] newSequence)*

Повертає трикутну матрицю

1. *private static int [] getMaxDistanceVector(int[][] matrix, int start)*

Повертає масив максимальних відстаней від вершини start за методом Фалкерсона

**Лістинг коду програми**

|  |
| --- |
| Main.java |
| package com.company;  public abstract class Main  {  private static final int INF = (int)(Integer.MAX\_VALUE / 2);  private static void printMatrix(int[][] matrix, char zero) {  //вывести матрицу на экран  for (int[] aMatrix : matrix) {  for (int anAMatrix : aMatrix)  if (anAMatrix <= -1 || anAMatrix == INF) System.out.print(" ∞ ");  else if (anAMatrix == 0) System.out.format(" %c ", zero);  else System.out.format("%2d ", anAMatrix);  System.out.println("");  }  System.out.println("");  }  private static int[] getMinDistanceVector(int aMatrix[][], int start) {  //рассчитать минимальные расстояния от точки start алгоритмом Дейкстры  int size = aMatrix.length;  int[] res = new int[size];  int[] visited = new int[size];  int visitedAmount = 0;  for (int i = 0; i < size; i++)  res[i] = (i == start) ? 0 : INF;  while (visitedAmount != size) {  for (int j = 0; j < size; j++)  if (visited[start] == 0 && aMatrix[start][j] > 0)  if (res[start] + aMatrix[start][j] < res[j])  res[j] = res[start] + aMatrix[start][j];  visited[start] = 1;  int minIndex = -1;  for (int i = 0; i < size; i++)  if (visited[i] == 0 && res[i] < INF)  if ( minIndex == -1 || res[minIndex] > res [i] )  minIndex = i;  if (minIndex == -1) break;  start = minIndex;  visitedAmount++;  }  return res;  }  private static void printVector(int[] arr) {  //вывести минимальные расстояния списком  for (int i = 0; i < arr.length; i++)  System.out.format("M%d: %s\n", i + 1, (arr[i] <= -1) ? "∞ " : (arr[i] + " "));  }  private static void printVector(int[] arr, int[] rebuiltSequence) {  //вывести максимальные расстояния списком  int size = arr.length;  int[] oldSequenceArr = new int[arr.length];  for (int i = 0; i < size; i++)  oldSequenceArr[rebuiltSequence[i]] = arr[i];  for (int i = 0; i < arr.length; i++)  System.out.format("M%d: %s\n", i + 1, (oldSequenceArr[i] <= -1) ? "∞ " : (oldSequenceArr[i] + " "));  }  private static int[] getNewSequence(int[][] matrix) {  //получить новую последовательность вершин в треугольной матрице  int size = matrix.length;  int[] newSequence = new int[size];  int newSequenceIterator = 0;  boolean[] crossed = new boolean[size];  while (newSequenceIterator != size)  for (int j = 0; j < size; j++) {  if (!crossed[j]) {  boolean toCross = true;  for (int i = 0; i < size; i++)  if (matrix[i][j] > 0 && !crossed[i] && !crossed[j]) toCross = false;  if (toCross) {  crossed[j] = true;  newSequence[newSequenceIterator++] = j;  }  }  }  return newSequence;  }  private static int[][] getTriangleMatrix(int[][] matrix, int[] newSequence) {  //получить треугольную матрицу  int size = matrix.length;  int[][] rebuilt = new int[size][size];  for (int i = 0; i < size; i++)  for (int j = 0; j < size; j++)  rebuilt[i][j] = matrix[newSequence[i]][newSequence[j]];  return rebuilt;  }  private static int[] getMaxDistanceVector(int[][] matrix, int start) {  //рассчитать максимальные расстояния от точки start методом Фалкерсона  int size = matrix.length;  int res[] = new int[size];  res[start] = 0;  for (int j = 0; j < size; j++) {  if (j != start) {  int max = -INF;  for (int i = 0; i < size; i++) {  if (matrix[i][j] > 0) {  max = (res[i] + matrix[i][j] > max) ? (res[i] + matrix[i][j]) : max;  }  }  res[j] = max;  }  }  return res;  }  public static void main(String[] args) {  int[][] matrixForMinWays = {  {0,6,11,5,-1,-1},  {-1,0,-1,-1,7,2},  {-1,5,0,-1,6,-1},  {-1,-1,-1,0,4,5},  {-1,-1,-1,-1,0,7},  {-1,-1,-1,-1,-1,0}  };  System.out.println("Input matrix:");  printMatrix(matrixForMinWays, '-');  int startPointOfMinDistance = 1;  System.out.format("Minimal distances from point №%d:\n", startPointOfMinDistance);  printVector(getMinDistanceVector(matrixForMinWays, startPointOfMinDistance - 1));  int[][] matrixForMaxWays = {  {0,6,11,5,-1,-1},  {-1,0,-1,-1,7,2},  {-1,5,0,-1,6,-1},  {-1,-1,-1,0,4,5},  {-1,-1,-1,-1,0,7},  {-1,-1,-1,-1,-1,0}  };  int[] newSequence = getNewSequence(matrixForMaxWays);  int[][] triangleMatrix = getTriangleMatrix(matrixForMaxWays, newSequence);  System.out.println("Triangle matrix:");  printMatrix(triangleMatrix, '-');  int startPointOfMaxDistance = 1;  System.out.format("Maximal distances from point №%d:\n", startPointOfMaxDistance);  printVector(getMaxDistanceVector(triangleMatrix, startPointOfMaxDistance - 1), newSequence);  }  } |

**Висновки**

Було опрацьовано методи знаходження екстремальних шляхів між вершинами: алгоритм Дейскстри для найкоротшого шляху та алгоритм Фалкерсона для знаходження найдовшого шляху.