НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №3

з дисципліни **«**Комп’ютерне моделювання**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІО-31

Долинний Олександр

Перевірив:

Марковський О.П.

Київ – 2015 р.

**Лістинг програми**

**Model.java**

**package km.lab3;**

**import org.apache.commons.math3.linear.\*;**

**import java.io.FileInputStream;**

**import java.io.FileNotFoundException;**

**import java.io.FileOutputStream;**

**import java.io.IOException;**

**import java.util.LinkedList;**

**import java.util.Set;**

**/\*\***

**\* Created by alex\_\_000 on 25.11.2015.**

**\*/**

**public class Model {**

**LinkedList<State> states;**

**LinkedList<int[]> crossMatr;**

**double[][] crossing;**

**double[][] matrixCrossing = {**

**{0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},// ЦП**

**{0.3, 0.3, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},// Кеш**

**{0.0, 0.1, 0.1, 0.5, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},// ПнМ**

**{0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},// ОП**

**{0.0, 0.0, 0.3, 0.3, 0.3, 0.4, 0.5, 0.65, 0.8, 1.0, 1.0, 1.0},// ПдМ**

**{1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},// АУ**

**{0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},// КДП**

**{0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},// СА**

**{0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},// КОП**

**{0.0, 0.0, 0.0, 0.0, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 1.0},// PSI**

**{1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0},// LPT**

**{0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0},// USB**

**// ЦП, Кеш, ПнМ, ОП, ПдМ, АУ, КДП, СА, КОП, PSI, LPT, USB**

**};**

**double[] arrayIntens = {1, 2, 1, 0.8, 0.8, 0.8, 0.8, 0.5, 0.5, 0.816, 0.3, 0.5};**

**int[] startState = {5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};**

**// int[] startState = {5, 0, 0};**

**// double[][] matrixCrossing = {**

**// {0.1, 1.0, 1.0},**

**// {0.0, 0.0, 1.0},**

**// {1.0, 1.0, 1.0}**

**//**

**// };**

**Model() {**

**states = new LinkedList<State>();**

**//buildTree();**

**readTree();**

**getGraph();**

**}**

**void buildTree() {**

**String strResult = "";**

**String crossResult = "";**

**int LENGTH = startState.length;**

**for (int l = 0; l < LENGTH - 1; l++) {**

**strResult = strResult + startState[l] + " ";**

**}**

**strResult += startState[LENGTH - 1];**

**strResult += "\n";**

**states.add(new State(startState));**

**int count = 0;**

**int stat = 1;**

**System.out.println(1 + " " + states.get(0).toString());**

**for (; (!states.get(states.size() - 1).isCheckFlag()); count++) {**

**for (int i = 0; i < LENGTH; i++) {**

**if (states.get(count).getDevices()[i] > 0) {**

**if (matrixCrossing[i][0] > 0) {**

**int[] temp = new int[LENGTH];**

**for (int k = 0; k < LENGTH; k++)**

**temp[k] = states.get(count).getDevices()[k];**

**temp[i] = temp[i] - 1;**

**temp[0] = temp[0] + 1;**

**State st = new State(temp);**

**boolean flag = true;**

**for (int l = 0; flag && (l < states.size()); l++) {**

**if (st.equal(states.get(l))) {**

**flag = false;**

**crossResult = crossResult + " " + l;**

**}**

**}**

**if (flag) {**

**for (int l = 0; l < LENGTH - 1; l++) {**

**strResult = strResult + temp[l] + " ";**

**}**

**strResult += temp[LENGTH - 1];**

**strResult += "\n";**

**states.add(st);**

**crossResult = crossResult + " " + stat;**

**stat++;**

**System.out.println(stat + " " + st.toString());**

**}**

**}**

**for (int j = 1; j < LENGTH; j++) {**

**if ((matrixCrossing[i][j] - matrixCrossing[i][j - 1]) > 0) {**

**int[] temp = new int[LENGTH];**

**for (int k = 0; k < LENGTH; k++)**

**temp[k] = states.get(count).getDevices()[k];**

**temp[i] = temp[i] - 1;**

**temp[j] = temp[j] + 1;**

**State st = new State(temp);**

**boolean flag = true;**

**for (int l = 0; (l < states.size()) && flag; l++) {**

**if (st.equal(states.get(l))) {**

**flag = false;**

**crossResult = crossResult + " " + l;**

**}**

**}**

**if (flag) {**

**for (int l = 0; l < LENGTH - 1; l++) {**

**strResult = strResult + temp[l] + " ";**

**}**

**strResult += temp[LENGTH - 1];**

**strResult += "\n";**

**states.add(st);**

**crossResult = crossResult + " " + stat;**

**stat++;**

**System.out.println(stat + " " + st.toString());**

**}**

**}**

**}**

**}**

**}**

**crossResult = crossResult + "\n";**

**states.get(count).setCheckFlag(true);**

**}**

**FileOutputStream f2 = null;**

**FileOutputStream f3 = null;**

**try {**

**f3 = new FileOutputStream("D:/crossing.txt");**

**f2 = new FileOutputStream("D:/sol.txt");**

**byte[] bCross = crossResult.getBytes();**

**byte[] ch = strResult.getBytes();**

**f2.write(ch);**

**f3.write(bCross);**

**} catch (FileNotFoundException e) {**

**e.printStackTrace();**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**}**

**void readTree() {**

**int LENGTH = 12;**

**String result = "";**

**FileInputStream f1 = null;**

**FileInputStream f4 = null;**

**String crossResult = "";**

**try {**

**f1 = new FileInputStream("D:/sol.txt");**

**int size = f1.available();**

**for (int i = 0; i < size; i++) {**

**result += (char) f1.read();**

**}**

**f4 = new FileInputStream("D:/crossing.txt");**

**int sizeCross = f4.available();**

**for (int i = 0; i < sizeCross; i++) {**

**crossResult += (char) f4.read();**

**}**

**} catch (FileNotFoundException e) {**

**e.printStackTrace();**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**System.out.println(result);**

**LinkedList<State> states0 = new LinkedList<State>();**

**String[] strArray = result.split("\n");**

**for (int i = 0; i < strArray.length; i++) {**

**int[] temp = new int[LENGTH];**

**String[] bufArray = strArray[i].split(" ");**

**for (int k = 0; k < temp.length; k++) {**

**temp[k] = Integer.parseInt(bufArray[k]);**

**}**

**states0.add(new State(temp));**

**}**

**for (int k = 0; k < states0.size(); k++)**

**System.out.println(k + " " + states0.get(k).toString());**

**crossMatr = new LinkedList<int[]>();**

**String[] crossArray = crossResult.split("\n");**

**for (int i = 0; i < crossArray.length; i++) {**

**int[] temp;**

**if (crossArray[i] != "") {**

**temp = new int[1];**

**temp[0] = -1;**

**String res = "";**

**crossArray[i] = crossArray[i].trim();**

**String[] bufArray = crossArray[i].split(" ");**

**temp = new int[bufArray.length];**

**for (int k = 0; k < bufArray.length; k++) {**

**temp[k] = Integer.parseInt(bufArray[k]);**

**}**

**} else {**

**temp = new int[1];**

**temp[0] = -1;**

**}**

**crossMatr.add(temp);**

**}**

**for (int k = 0; k < crossMatr.size(); k++) {**

**String res = "";**

**for (int j = 0; j < crossMatr.get(k).length; j++) {**

**res = res + " " + crossMatr.get(k)[j];**

**}**

**System.out.println(res);**

**}**

**}**

**void getGraph() {**

**crossing = new double[crossMatr.size()][crossMatr.size()];**

**for (int i = 0; i < crossMatr.size(); i++) {**

**//Set<Integer > tempSet = null;**

**for (int j = 0; j < crossMatr.get(i).length; j++) {**

**int a = states.get(i).compareA(states.get(crossMatr.get(i)[j]));**

**int b = states.get(i).compareA(states.get(crossMatr.get(i)[j]));**

**crossing[i][crossMatr.get(i)[j]] = arrayIntens[a]\*matrixCrossing[a][b];**

**// tempSet.add(states.get(i).compareA(states.get(crossMatr.get(i)[j])));**

**}**

**}**

**double[][] systArray = new double[crossMatr.size()][crossMatr.size()];**

**for(int i=0; i<systArray.length; i++)**

**systArray[0][i]=1;**

**for(int i=1; i<systArray.length; i++) {**

**for(int j=0;j<systArray.length; j++){**

**systArray[i][j]=crossing[i][j]-crossing[j][i];**

**}**

**}**

**double[] vectArray = new double[crossMatr.size()];**

**vectArray[0]=1;**

**RealMatrix coef = new Array2DRowRealMatrix(systArray);**

**DecompositionSolver solver= new LUDecomposition(coef).getSolver();**

**RealVector right = new ArrayRealVector(vectArray);**

**RealVector x = solver.solve(right);**

**double[] ans = x.toArray();**

**double[] solution = new double[12];**

**for(int i=0; i<12; i++){**

**for (int j=0; j<crossMatr.size(); j++) {**

**if(states.get(j).getDevices()[i]>0)**

**solution[i]+=ans[j];**

**}**

**}**

**String result = "";**

**String[] nameArray = {"ЦП", "Кеш", "ПнМіст", "ОП", "ПдМіст", "АУ", "КДП", "СА", "КОП", "PSI", "LPT", "USB"};**

**for(int i=0; i<solution.length; i++){**

**result = result + " " + nameArray[i] + ": " + solution[i] + "\n";**

**}**

**System.out.println(result);**

**}**

**}**