Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Лабораторна робота №4

з системного програмного забезпечення

Виконав:

студент ІІІ курсу

ФІОТ, групи ІО-93

Паламарчук Валерій

ЗК: 9313

Київ 2012

**Дослідження принципів проектування та роботи динамічного та статичного планування**

**Варіант:**

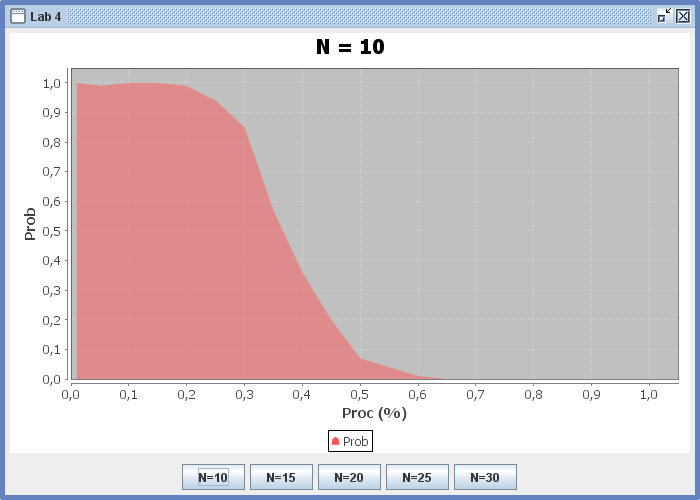
13 mod 13 + 1 = 1 – визначити вірогідність появи обов'язкових призначень при зміні заповненості матриці "1" зв'язності від 1% до 100% для заданої розмірності від 10 до 30 з кроком 5.

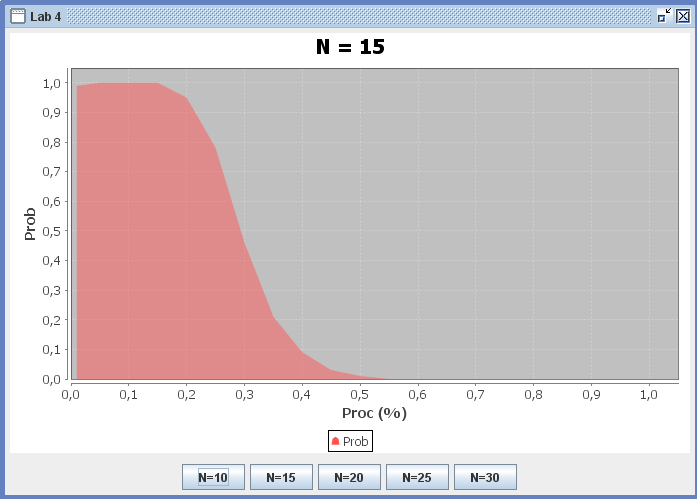
**Опис алгоритму:**

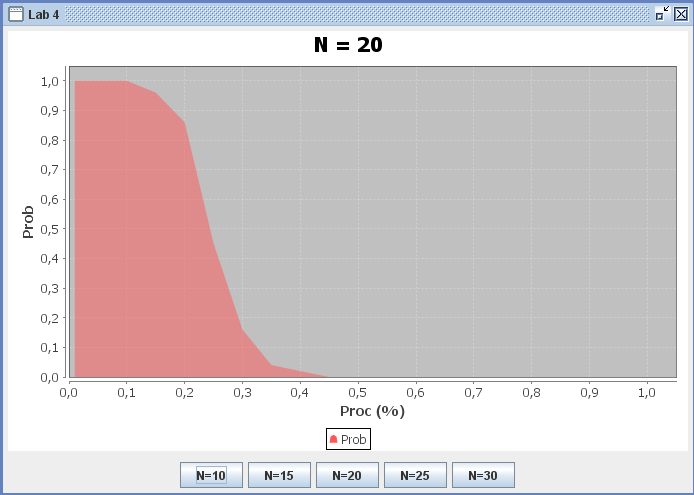
Якщо для незваженого дводольного графа існує довершене паросполучення й в графі є вершина із ступенем один, то ребро, інцидентне цій вершині, і вершини, інцидентні цьому ребру обов'язково входять в довершене паро сполучення. Алгоритм перевіряє цю умову шляхом підсумування елементів матриці, які знаходяться в одному рядку або в одному стовпцю.

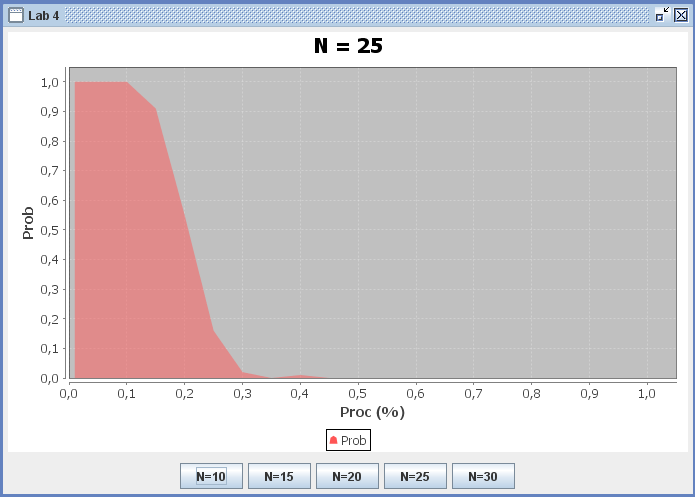
Форма графіків правильна, так як при малому відсотку наповненості матриці ймовірність появи вершини із ступенем 1 дуже велика, при збільшенні відсотку вона зменшується і при достатньо великому відсотку приблизно 65% дорівнює нулю.

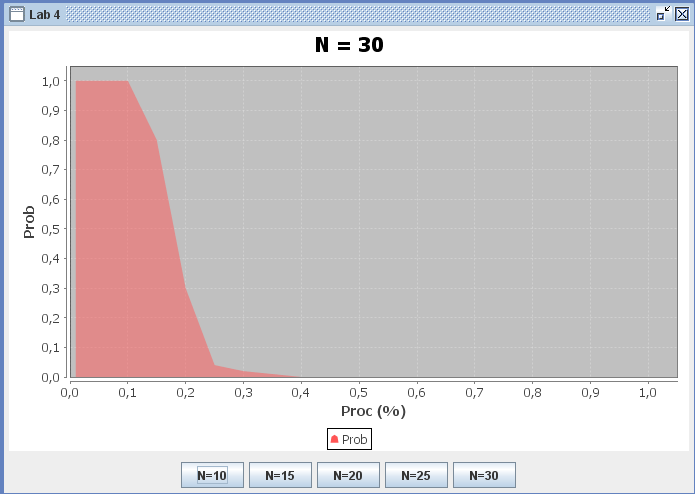
**Отримані графіки:**

****

******







**Листинг:**

***package*** *lab4;*

**import** java.util.ArrayList;

**public** **class** start {

**static** mFrame *f*;

**public** **static** **void** **main**(String[] args) {

*run*(10);

}

**start**(**int** index) {

*run*(index);

}

**public** **static** **void** **run**(**int** size) {

*f* = **new** mFrame();

matrBuilder m = **new** matrBuilder(size);

m.run();

ArrayList<Double> percent = m.getPercent();

ArrayList<Double> probability = m.getProbability();

*f*.addChart(percent, probability, "N = " + size, "Proc (%)", "Prob");

*f*.setVisible(**true**);

}

}

**package lab4;**

**import java.\*;**

**import javax.swing.\*;**

**import org.jfree.\*;**

**public class mFrame extends JFrame {**

**private JPanel panel;**

**private JPanel butPan;**

**private JButton but10 = new JButton("N=10");**

**private JButton but15 = new JButton("N=15");**

**private JButton but20 = new JButton("N=20");**

**private JButton but25 = new JButton("N=25");**

**private JButton but30 = new JButton("N=30");**

**start t;**

**public mFrame() {**

**this.setDefaultCloseOperation(EXIT\_ON\_CLOSE);**

**this.setTitle("Lab 4");**

**this.setUndecorated(true);**

**this.getRootPane().setWindowDecorationStyle(JRootPane.FRAME);**

**this.setBounds(300, 200, 700, 500);**

**this.setResizable(false);**

**this.setLayout(new BorderLayout());**

**panel = new JPanel();**

**add(panel, BorderLayout.NORTH);**

**butPan = new JPanel();**

**add(butPan, BorderLayout.SOUTH);**

**butPan.setLayout(new FlowLayout());**

**butPan.add(but10);**

**but10.addActionListener(a);**

**butPan.add(but15);**

**but15.addActionListener(a);**

**butPan.add(but20);**

**but20.addActionListener(a);**

**butPan.add(but25);**

**but25.addActionListener(a);**

**butPan.add(but30);**

**but30.addActionListener(a);**

**}**

**public void addChart(ArrayList<Double> x, ArrayList<Double> y, String name,**

**String namex, String namey) {**

**XYSeries series = new XYSeries(namey);**

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

**series.add(x.get(i), y.get(i));**

**}**

**XYDataset xyDataset = new XYSeriesCollection(series);**

**JFreeChart chart = ChartFactory.createXYAreaChart(name, namex, namey,**

**xyDataset, PlotOrientation.VERTICAL, true, true, true);**

**panel.add(new ChartPanel(chart));**

**}**

**ActionListener a = new ActionListener() {**

**@Override**

**public void actionPerformed(ActionEvent e) {**

**String but\_Text = ((JButton) e.getSource()).getText();**

**int index = Integer.parseInt(but\_Text.substring(2));**

**t = new start(index);**

**dispose();**

**}**

**};**

**}**

**package lab4;**

**import java.util.ArrayList;**

**import java.util.Random;**

**public class matrBuilder {**

**private int[][] matrix;**

**private int size;**

**private ArrayList<Double> percent = new ArrayList<Double>();**

**private ArrayList<Double> probability = new ArrayList<Double>();**

**public matrBuilder(int size) {**

**this.size = size;**

**}**

**public void generateMatrix(double perc) {**

**matrix = new int[size][size];**

**if (perc < 1) {**

**double part = size \* size \* perc;**

**int count = (int) part;**

**if ((part - count) >= 0.5)**

**count++;**

**Random r = new Random();**

**int row;**

**int column;**

**while (count > 0) {**

**row = r.nextInt(size);**

**column = r.nextInt(size);**

**if (matrix[row][column] == 0) {**

**matrix[row][column] = 1;**

**count--;**

**}**

**}**

**} else {**

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

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

**matrix[i][j] = 1;**

**}**

**}**

**public int checkMatrix() {**

**int sum;**

**double rand = Math.random();**

**if (rand < 0.5)**

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

**sum = 0;**

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

**sum += matrix[i][j];**

**}**

**if (sum == 1)**

**return 1;**

**}**

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

**sum = 0;**

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

**sum += matrix[j][i];**

**}**

**if (sum == 1)**

**return 1;**

**}**

**return 0;**

**}**

**public void run() {**

**double perc = 0.01;**

**int count = 100;**

**double sum;**

**while (perc <= 1.0001) {**

**sum = 0;**

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

**generateMatrix(perc);**

**sum += checkMatrix();**

**}**

**percent.add(perc);**

**probability.add(sum / count);**

**if (perc == 0.01)**

**perc = 0.05;**

**else**

**perc += 0.05;**

**}**

**}**

**public ArrayList<Double> getPercent() {**

**return percent;**

**}**

**public ArrayList<Double> getProbability() {**

**return probability;**

**}**

**}**