**Баніт Вадим Віталійович**

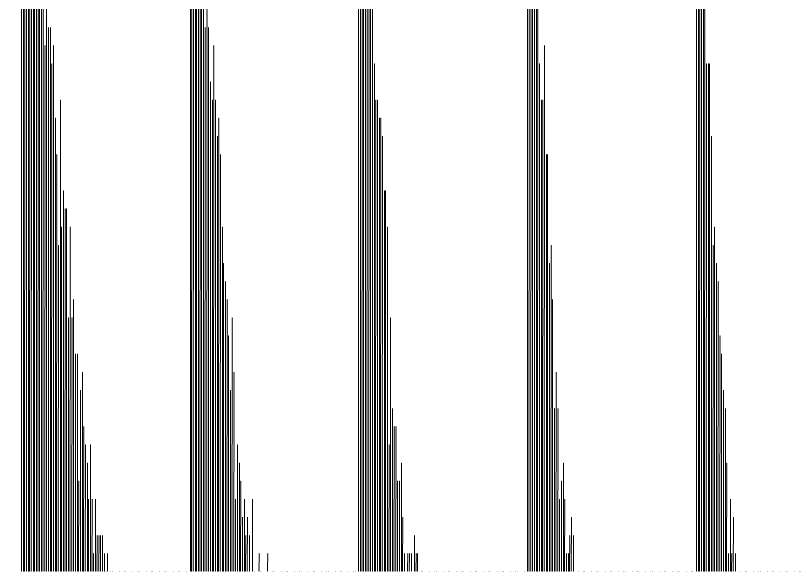
**№заліковки=202;**

(02)mod 13 + 1=3;

3. Визначити вірогідність появи конфліктних призначень при зміні заповненості матриці "1" зв'язності від 1% до 100% для заданої розмірності від 10 до 30 з кроком 5.

ГРАФІК:

Ймовірність появи конфліктних призначеннь(від шагу і заповненності):



Лістинг:

package

{

import air.update.logging.Level;

import flash.display.Sprite;

import flash.text.\*;

import flash.display.Sprite;

import flash.display.DisplayObject;

import flash.display.Shape;

/\*\*

\* ...

\* @author Vadim

\*/

public class Main extends Sprite

{

public var MC:Array = [];

public var step:int = 5;

public var percent:Number = 50;

public var countConflicts:int = 0;

public var P:Array = [];

public function Main():void

{

var format:TextFormat = new TextFormat();

format.font = "Verdana";

format.color = 0xFF0000;

format.size = 6;

var scoreTextBox:TextField=new TextField();

addChild(scoreTextBox);

scoreTextBox.autoSize = TextFieldAutoSize.LEFT;

scoreTextBox.defaultTextFormat=format;

scoreTextBox.x=110;

var scoreText:TextField=new TextField();

scoreText.defaultTextFormat=format;

scoreText.text="Матрица: ";

addChild(scoreText);

scoreText.x = 10;

var i:int = 0;

for (step = 10; step <= 30; step += 5) {

for (percent = 1; percent <= 100; percent++) {

for (var count:int = 0; count <= 30;count++){

MC = generateMatrix(step,percent);

sortMatrixGlobal(MC);

if (findConflictAppoinment(MC) == true) {

countConflicts++;

}

}

P[i] = countConflicts / count;

i++;

countConflicts = 0;

}

}

/\*for (var n:int = 50; n < P.length; n++ ) {

scoreTextBox.text += P[n].toString();

scoreTextBox.text += '\n';

}\*/

/\*for (var i:int = 0; i < MC.length; i++ ) {

scoreTextBox.text += '\n';

for (var j:int = 0; j < MC.length; j++ ) {

scoreTextBox.text += MC[i][j].toString();

}

}\*/

/\*scoreTextBox.text += '\n';

for (var i:int = 0; i < MC.length; i++ ) {

scoreTextBox.text += '\n';

for (var j:int = 0; j < MC.length; j++ ) {

scoreTextBox.text += MC[i][j].toString();

}

}

sortMatrixGlobal(MC);

//sortMatrix(MC, 1);

scoreTextBox.text += '\n';

for (var i:int = 0; i < MC.length; i++ ) {

scoreTextBox.text += '\n';

for (var j:int = 0; j < MC.length; j++ ) {

scoreTextBox.text += MC[i][j].toString();

}

}

scoreTextBox.text += '\n';

scoreTextBox.text+=findConflictAppoinment(MC).toString();\*/

DrawGraphic(P);

}

public function DrawGraphic(P:Array):void {

var j:Number = 0;

for (var i:int = 0; i < P.length;i++ ){

j+=1.5;

var shape:Shape = new Shape();

shape.graphics.lineStyle(1,0,1);

shape.graphics.moveTo(100+j,500 );

shape.graphics.lineTo(100+j+0.1,(1-P[i])\*500);

addChild(shape);

}

}

public function generateMatrix(N:int,percent:Number):Array {

var MC:Array = [];

for (var i:int = 0; i < N; i++ ) {

MC[i] = [];

for (var j:int = 0; j < N; j++) {

if(Math.random()<=(percent/100)){

MC[i][j] = 1;

}else {

MC[i][j] = 0;

}

}

/\*MC [0][0] = 0;

MC [0][1] = 0;

MC [0][2] = 0;

MC [0][3] = 1;\*/

}

return MC;

}

public function zeroRowColumn(MC:Array):Array {

//обнуляем строки и столбцы с одной единицей

var one:int = 0;

var \_j:int = 0;

//cтроки

for (var i:int = 0; i < MC.length; i++ ) {

one = 0;

for (var j:int = 0; j < MC[i].length; j++ ) {

if (MC[i][j] == 1) {

one++;

\_j = j;

if (one > 1) {

break;

}

}

}

if (one == 1) {

MC[i][\_j] = 0;

}

}

//столбцы

for (var i:int = 0; i < MC.length; i++ ) {

one = 0;

for (var j:int = 0; j < MC[i].length; j++ ) {

if (MC[j][i] == 1) {

one++;

\_j = j;

if (one > 1) {

break;

}

}

}

if (one == 1) {

MC[\_j][i] = 0;

}

}

return MC;

}

public function sortMatrix(MC:Array,rank:int):Array {

var one:Array = [];

//находим строку с минимальной суммой единиц

for (var i:int=0+rank; i < MC.length; i++ ) {

one[i] = 0;

for (var j:int=0+rank; j < MC[i].length; j++ ) {

if (MC[i][j] == 1) {

one[i]++;

}

}

}

//строка с минимальной сумммой единиц

var tempIndex:int = 0+rank;

for (i= 1+rank; i < one.length; i++ ) {

if (one[i - 1] > one[i]) {

tempIndex = i;

}

}

//заменяем с первой

var temp:int = 0;

if (tempIndex != 0+rank) {

for (j = 0 + rank; j < MC.length; j++ ) {

temp = MC[0 + rank][j];

MC[0 + rank][j] = MC[tempIndex][j];

MC[tempIndex][j]= temp;

}

}

//поиск столбца с максимальной суммой 1

for (var i:int = 0+rank; i < MC.length; i++ ) {

one[i] = 0;

for (var j:int = 0+rank; j < MC[i].length; j++ ) {

if(MC[0+rank][i]==1){

if (MC[j][i] == 1) {

one[i]++;

}

}else break;

}

}

//столбец с максимальной сумммой единиц

var tempIndex:int = 0+rank;

for (i= 1+rank; i < one.length; i++ ) {

if (one[i - 1] < one[i]) {

tempIndex = i;

}

}

//заменяем с первым

var temp:int = 0;

if (tempIndex != 0+rank) {

for (j = 0 + rank; j < MC.length; j++ ) {

temp = MC[j][0 + rank];

MC[j][0 + rank] = MC[j][tempIndex];

MC[j][tempIndex] = temp;

}

}

return MC;

}

public function sortMatrixGlobal(MC:Array):Array {

zeroRowColumn(MC);

for (var i:int = 0; i < MC.length-1;i++ ){

sortMatrix(MC, i);

}

return MC;

}

public function findConflictAppoinment(MC:Array):Boolean {

var conflict:Boolean = false;

for (var k:int = 1; k < MC.length - 1; k++ ) {

for (var i:int = 0; i < k; i++ ) {

for (var j:int = 0 + k; j < MC.length; j++ ) {

if (MC[i][j] == 0) {

conflict = true; }

else { conflict = false; break;}

}

if (conflict == false) {

break;

}

}

}

return conflict;

}

}

}