//He añadido más código para justificar la percolación de un vector pozo en función de una //matriz de caminos según las filas de la matriz.

//Esto tiene la característica de que se puede reprogramar los caminos para verificar la //percolación.

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package percolacion;

/\*\*

\*

\* @author Antonio Luis Caballero Sosa

\* Técnico Superior de Desarrollo Multiplataforma

\*/

import java.lang.Math;

public class Percolacion {

/\*\*

\* @param args the command line arguments

\*/

//valor inicial para el cálculo del tamaño del vector árbol

int n = 7;

//valor del tamaño del vector de suma de ceros o huecos

int m = 8;

//cálculo del tamaño del vector que va a representar el árbol

int calculo = (2 \* n) + 1;

int[] vector\_pozo = new int[calculo];

int[] vector = new int[m];

//definir correctamente esta matriz

//matriz de caminos para detectar la percolaciòn

int[][] matriz\_caminos = {{0, 1, 3, 7}, {0, 1, 4, 9}, {0, 1, 3, 8}, {0, 1, 4, 10}, {0, 2, 6, 14}, {0, 2, 7, 11}, {0, 2, 5, 12}, {0, 2, 6, 13}};

//vector plantilla de vuelco de los datos de vector\_pozo

int[][] vector\_suma = {{1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}, {1, 1, 1, 1}};

Percolacion() {

}

//rellenar vector con ceros

void paso\_primero() {

for (int c = 0; c < vector\_pozo.length; c++) {

vector\_pozo[c] = 0;

}

for (int c = 0; c < vector\_pozo.length; c++) {

System.out.print(" Vector\_pozo\_inicial: " + vector\_pozo[c]);

}

}

// rellenar vector con ceros, unos y does

void paso\_segundo() {

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

for (int c = 0; c < vector\_pozo.length; c++) {

vector\_pozo[c] = (int) (Math.random() \* 3 + 0);

}

for (int c = 0; c < vector\_pozo.length; c++) {

System.out.print(" vector\_pozo\_aleatorio: " + vector\_pozo[c]);

}

}

// abrir si elemento es 1 y ponerlo a 0

void paso\_tercero() {

for (int c = 0; c < vector\_pozo.length; c++) {

if (isopen(c)) {

open(c);

}

}

}

//leer los caminos del árbol que están a ceros

void paso\_cuarto() {

//int dato=0;

for (int f = 0; f < matriz\_caminos.length; f++) {

for (int c = 0; c < matriz\_caminos[f].length; c++) {

System.out.println("fila: " + f + " columna: " + c);

//int d=matriz\_caminos[f][c];

//System.out.println("d: "+d);

System.out.println("completo vector pozo: " + vector\_pozo[matriz\_caminos[f][c]]);

if ((vector\_pozo[matriz\_caminos[f][c]]) == 0) {

vector\_suma[f][c] = vector\_pozo[matriz\_caminos[f][c]];

//vector\_suma[f]=dato;

}

//System.out.println("dato: "+dato);

//vector\_suma[f]=dato;

}

}

}

//leer el vector suma según se ha dado los caminos

void paso\_quinto() {

for (int f = 0; f < vector\_suma.length; f++) {

for (int c = 0; c < vector\_suma[f].length; c++) {

System.out.println("fila: " + f + " Columna: " + c + " Resultados: " + vector\_suma[f][c]);

}

}

}

//poner en vector los datos sumados para verificar que hay percolación

void paso\_sexto() {

int dat = 0;

for (int f = 0; f < vector\_suma.length; f++) {

for (int c = 0; c < vector\_suma[f].length; c++) {

System.out.println("fila: " + f + " Columna: " + c + " Resultados2: " + vector\_suma[f][c]);

dat += vector\_suma[f][c];

}

vector[f] = dat;

dat = 0;

}

}

//visualizar los datos del vector para ver los valores sumados a cero

void paso\_septimo() {

for (int f = 0; f < vector.length; f++) {

System.out.println("casilla: " + f + " información: " + vector[f]);

}

}

void paso\_octavo(){

//int dat = 0;

System.out.println("Detalle de la matriz de percolación");

for (int f = 0; f < vector\_suma.length; f++) {

for (int c = 0; c < vector\_suma[f].length; c++) {

System.out.print(vector\_suma[f][c]);

//dat += vector\_suma[f][c];

}

System.out.println("");

//vector[f] = dat;

//dat = 0;

}

}

void paso\_noveno(){

System.out.println("Detalle vector pozo según matriz de caminos");

for (int f = 0; f < matriz\_caminos.length; f++) {

for (int c = 0; c < matriz\_caminos[f].length; c++) {

//System.out.println("fila: " + f + " columna: " + c);

//int d=matriz\_caminos[f][c];

//System.out.println("d: "+d);

System.out.print(vector\_pozo[matriz\_caminos[f][c]]);

//if ((vector\_pozo[matriz\_caminos[f][c]]) == 0) {

// vector\_suma[f][c] = vector\_pozo[matriz\_caminos[f][c]];

//vector\_suma[f]=dato;

//}

//System.out.println("dato: "+dato);

//vector\_suma[f]=dato;

}

System.out.println("");

}

}

void paso\_decimo() {

for (int f = 0; f < vector.length; f++) {

if(vector[f]==0){

System.out.println("El Sistema percola...");

}else

{

System.out.println("El Sistema no percola...");

}

}

}

void paso\_ultimo(){

int cont=0;

System.out.println("Recuento de ceros...");

for (int f = 0; f < vector\_suma.length; f++) {

for (int c = 0; c < vector\_suma[f].length; c++) {

if(vector\_suma[f][c]==0){cont++;}

//dat += vector\_suma[f][c];

}

// System.out.println("");

//vector[f] = dat;

//dat = 0;

}

System.out.println("Ceros contados: "+cont);

}

public static void main(String[] args) {

// TODO code application logic here

Percolacion percolacion = new Percolacion();

percolacion.paso\_primero();

percolacion.paso\_segundo();

percolacion.paso\_tercero();

percolacion.paso\_cuarto();

percolacion.paso\_quinto();

percolacion.paso\_sexto();

percolacion.paso\_septimo();

percolacion.paso\_octavo();

percolacion.paso\_noveno();

percolacion.paso\_decimo();

percolacion.paso\_ultimo();

}

boolean isopen(int c) {

if (vector\_pozo[c] == 1) {

return true;

} else {

return false;

}

}

void open(int c) {

vector\_pozo[c] = 0;

}

}