novática
Revista de la Asociación de Técnicos de Informática

Novática, revista fundada en 1975 y decana de la prensa informática española, es el órgano oficial de expresión y formación continua de **ATI** (Asociación de Técnicos de Informática), organización que edita también la revista **REICIS** (Revista Española de Innovación, Calidad e Ingeniería del Software).

<http://www.ati.es/novatica/>
<http://www.ati.es/reicis/>

ATI es miembro fundador de **CEPIS** (*Council of European Professional Informatics Societies*) y es representante de España en **IFIP** (*International Federation for Information Processing*); tiene un acuerdo de colaboración con **ACM** (*Association for Computing Machinery*), así como acuerdos de vinculación o colaboración con **AdaSpain**, **AI2**, **ASTIC**, **RITSI** e **Hispalinux**, junto a la que participa en **ProInnova**.

Coordinación Editorial, Redacción Central y Redacción ATI Madrid
Padilla 66, 3º, dcha., 28006 Madrid
Tlfn.914029391; fax.913093685 <novatica@ati.es>
Composición, Edición y Redacción ATI Valencia
Av. del Reino de Valencia 23, 46005 Valencia
Tlfn. 963740173 <novatica_prod@ati.es>
Administración y Redacción ATI Cataluña
Via Laietana 46, ppal. 1ª, 08003 Barcelona
Tlfn.934125235; fax 934127713 <secregen@ati.es>
Redacción ATI Aragón
Lagasca 9, 3-B, 50006 Zaragoza.
Tlfn./fax 976235181 <secreara@ati.es>
Redacción ATI Andalucía <secreand@ati.es>
Redacción ATI Galicia<secregal@ati.es>
Suscripción y Ventas <novatica.subscripciones@atinet.es>
Publicidad Padilla 66, 3º, dcha, 28006 Madrid.
Tlnf.914029391; fax.913093685 <novatica@ati.es>
Imprenta: Derra S.A., Juan de Austria 66, 08005 Barcelona.
Depósito legal: B 15.154-1975 -- ISSN: 0211-2124; CODEN NOVAEC
Portada: Lenguaje primario - Concha Arias Pérez / © ATI
Diseño: Fernando Agresta / © ATI 2003

sumario

Nº 222, marzo-abril 2013, año XXXIX

Tema del próximo número: "Minería de procesos"

El problema del Buscaminas Cuadrado en 3D

Julio Javier Castillo, Diego Javier Serrano, Marina Elizabeth Cárdenas

Laboratorio de Investigación de Software MsLabs, Dpto. Ing. en Sistemas de Información, Facultad Regional Córdoba - Universidad Tecnológica Nacional (Argentina)

<jotacastillo@gmail.com>, <diegojserrano@gmail.com>, <ing.marinacardenas@gmail.com>

El enunciado de este problema apareció en el número 220 de ***Novática*** (noviembre-diciembre 2012, p.78)

El problema planteado consiste en determinar la cantidad de bombas (o minas) adyacentes a un punto dado en un campo minado en 3 dimensiones.

Para resolver este problema es conveniente pensar en el problema en 3 dimensiones en donde se pueden ubicar las minas, de manera tal que el campo minado quede representado por los ejes X e Y, y el campo de eje Z represente el campo de minas contiguo en el espacio.

Adicionalmente, el problema presenta otra complicación, que consiste en informar con un «-» como salida en el caso en el que haya una mina en la misma posición de alguna otra dimensión adyacente.

Para modelar este problema utilizamos matrices 3-dimensionales, para representar el espacio X,Y,Z. Así, el campo minado podría representarse como un arreglo de caracteres conteniendo '*' y '.', pero en vez de ello, la solución utiliza un arreglo de enteros principalmente por dos razones. La primera es para evitar el empleo de otra estructura de datos auxiliar para realizar el cálculo de la cantidad de minas, y la segunda es para ahorrar espacio de memoria y espacio en la codificación de la solución.

Se podrá observar que la solución emplea un arreglo tridimensional de tamaño t+2 en todas sus dimensiones. Se ha elegido t+2 para contar con una fila, y columna ficticia, a los efectos de simplificar las validaciones necesarias en el chequeo de los límites válidos del arreglo. De esta manera es posible trabajar con menos validaciones pagando un mínimo costo de memoria.

Convenientemente se ha elegido representar en la matriz entera al '*' con el número 1, y el '.' con el número 0, esto es para poder efectuar el cómputo de la sumatoria de las minas en cada dimensión simplemente al indizar la matriz.

En cuanto al diseño de la solución, la misma hace uso solamente de una clase denominada Buscaminas3D, la cual en su método principal se encarga de la entrada de datos, y en base a ellos, de la creación del campo minado en 3 dimensiones.

El método bombas(int [][][]c, int i, int j, int k) recibe el campo minado y una posición en el espacio dentro de ese campo, y como resultado retorna la cantidad de bombas adyacentes. El resultado de este método solamente es utilizado al informar las salidas en el método mostrarBuscaminas(). Nótese que la cantidad de bombas a retornar será b con 0 <= b <= 26, pues son todas las posiciones adyacentes a un punto dado.

El método mostrarBuscaminas() recorre el buscaminas en 3D y muestra por consola las minas (*) y en lugar de puntos ('.') la cantidad de bombas adyacentes pero teniendo en cuenta la restricción de que se debe informar con '-' en el caso que haya una mina en la misma posición en una dimensión adyacente. Además, este método muestra convenientemente los caracteres «###» después de procesar una dimensión, y los caracteres «DDD» al terminar de procesar cada buscamina, tal como lo requiere el planteo del problema.

A continuación se expone el código de la solución del problema en el lenguaje de programación Java:

```
package buscaminas3d;

import java.util.Scanner;

/**
 *
 * @author Castillo-Serrano-Cardenas
 */
public class Buscaminas3D {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int c=Integer.parseInt(sc.nextLine()); //buscaminas a analizar
         for(int i=0; i < c; i++)
         {
            int t=Integer.parseInt(sc.nextLine());    //profundidad del buscaminas
            int [][][]buscamina=new int[t+2][t+2][t+2];
             String cad=new String();
             for(int dt=1; dt <= t; dt++){
                 for(int d=1; d <= t; d++){
                     cad=sc.nextLine();
                     for(int s=0; s < cad.length(); s++){
                         if(cad.charAt(s)=='*')
                             buscamina[d][s+1][dt]=1; //si es mina
                          else
```

```
                            buscamina[d][s+1][dt]=0; //si es punto
                        }
                    }
                    if(dt!=t)
                        cad=sc.nextLine();         //separadores
                }
                mostrarBuscaminas(buscamina,t);
            }
        }

    public static void mostrarBuscaminas(int [][][]campo_o, int dim)
        {
            for(int k=1;k <= dim; k++)
            {
                for(int i=1;i <= dim; i++)
                {
                    for(int j=1; j <= dim; j++){
                        if(campo_o[i][j][k]==1) {
                            System.out.print('*');
                        }
                    //si hay una mina en la posición simétrica de otro campo adyacente
                            else if (k==1 && campo_o[i][j][k]==0 && campo_o[i][j][k+1]==1)
                 System.out.print('-');
                            else if (k==(dim) && campo_o[i][j][k]==0 && campo_o[i][j][k-1]==1)
                                System.out.print('-');
                                else if (k>1 && k < dim && campo_o[i][j][k]==0 && (campo_o[i][j][k-1]==1
||campo_o[i][j][k+1]==1))
                                System.out.print('-');

                            else if (campo_o[i][j][k]==0)
                            {
                               System.out.print(""+bombas(campo_o,i,j,k));
                            }
                    }
                    System.out.println("");
                }
                if((k+1) <= dim)            //formato salida
                    System.out.println("###");
            }
            System.out.println("DDD");
        }

    public static int bombas(int [][][]c,int i,int j, int k)
        {
            int bombas=0;
           bombas+=c[i-1][j-1][k-1]+c[i-1][j][k-1]+c[i-1][j+1][k-1];
           bombas+=c[i][j-1][k-1]+c[i][j][k-1]+c[i][j+1][k-1];
           bombas+=c[i+1][j-1][k-1]+c[i+1][j][k-1]+c[i+1][j+1][k-1];

           bombas+=c[i-1][j-1][k]+c[i-1][j][k]+c[i-1][j+1][k];
           //(i,j,k)es el punto actual, y siempre tendrá 0.
           bombas+=c[i][j-1][k]+c[i][j][k]+c[i][j+1][k];
           bombas+=c[i+1][j-1][k]+c[i+1][j][k]+c[i+1][j+1][k];

           bombas+=c[i-1][j-1][k+1]+c[i-1][j][k+1]+c[i-1][j+1][k+1];
           bombas+=c[i][j-1][k+1]+c[i][j][k+1]+c[i][j+1][k+1];
           bombas+=c[i+1][j-1][k+1]+c[i+1][j][k+1]+c[i+1][j+1][k+1];

            return bombas;
        }
}
```