«Cortando cadenas»: solución

Álvaro Martínez Echevarría

<alvaro@mail.utexas.edu>

El enunciado de este problema apareció en el número 147 de *Novática* (septiembre-octubre 2000, p. 76). Es el programa C de los planteados en el 24º Concurso Internacional de Programación de la ACM de 2000

```
/*
Se ha optado por un acercamiento de fuerza bruta, probando todas las posibilidades, puesto que el número máximo de eslabones es muy reducido (15, lo que supone únicamente 2^15=32768 combinaciones). Primero se representan las cadenas de entrada como grafos (en forma de matrices de adyacencia). Luego se recorren todas las posibilidades mediante un bitmap cuyos bits indican si un eslabón debe quedar abierto o cerrado. Para cada combinación se cuentan los eslabones que quedan abiertos, y se construye una nueva matriz de adyacencia en la que se han eliminado éstos. Después de comprobar si el grafo que queda está formado por fragmentos lineales de cadena, basta con averiguar si todos esos fragmentos pueden enlazarse de nuevo entre sí de forma lineal utilizando los eslabones que quedaron abiertos.

Para ello se cuentan los extremos que hay en el grafo y se comprueba si:

 eslabones_abiertos*2 >= extremos-2

contando los eslabones aislados como dos extremos. Por ejemplo, en estos cuatro fragmentos lineales de cadena hay 8 extremos:

   oooooo   ooo   ooooo   o
  ^      ^ ^   ^ ^     ^ ^ ^
  1      2 3   4 5     6 7 8

Bastaría con 3 eslabones abiertos para convertirlos en una única cadena lineal, uniendo 2-3, 4-5 y 6-7.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXESLABONES 15
char  matriz[MAXESLABONES][MAXESLABONES];
char nuevamatriz[MAXESLABONES][MAXESLABONES];
int  eslabones;

/*
Recorre el grafo nuevamatriz[][] empezando por «hacia» (viniendo desde «desde») marcando los nodos visitados en «visitado». Devuelve 0 si pasa por algún nodo con más de dos enlaces o en el que haya estado previamente; ó 1 en caso contrario.
*/

int
recorre(int desde, int hacia, char *visitado)
{
  int i;
  int enlaces;

  if (visitado[hacia]) {
    return 0;
  }
  visitado[hacia]=1;

  enlaces=0;
  for (i=0;i<eslabones;i++) {
    if (nuevamatriz[hacia][i]) {
      enlaces++;
      if (enlaces>2) {
        return 0;
      }
      if (i!=desde) {
        if (!recorre(hacia,i,visitado)) {
          return 0;
        }
      }
    }
  }

  return 1;
}

/*
Devuelve 0 si el grafo nuevamatriz[][] es un conjunto de fragmentos lineales de cadena.
*/

int
viable(void) {
  int i;
  char visitado[MAXESLABONES];

  memset(visitado,0,sizeof(visitado));
  for (i=0;i<eslabones;i++) {
    if (visitado[i]) {
      continue;
    }
    if (!recorre(i,i,visitado)) {
      return 0;
    }
  }
  return 1;
}

/*
Devuelve el número de bits no nulos del parámetro n.
*/
int
bitcount(int n) {
  int b=0;

  while (n) {
    b++;
    n&=(n-1);
  }
  return b;
}
```

```
/*
 * Calcula el mínimo pedido en el conjunto de eslabones representado en el grafo matriz[][].
 */

int
minimo(void) {
  int bitmap,maxbitmap;
  char abrir[MAXESLABONES];
  int minabiertos;
  int abiertos,enlaces,extremos;
  int i,j;

  minabiertos=eslabones+1;
  maxbitmap=1<<eslabones;
 for (bitmap=0; bitmap<maxbitmap; bitmap++)
{

    /*
     * Pequeña optimización...
     */

   abiertos=bitcount(bitmap);
   if (abiertos>=minabiertos) {
     continue;
   }

   /*
     *  Generamos el nuevo grafo resultante de abrir los eslabones.
     */

 memcpy(nuevamatriz,matriz,sizeof(nuevamatriz));
   for (i=0;i<eslabones;i++) {
     /*
      * ¿Hace falta abrir el eslabón «i»?
      */
     abrir[i]=(bitmap>>i)&1;
     if (abrir[i]) {
       for (j=0;j<eslabones;j++) {
     nuevamatriz[i][j]=nuevamatriz[j][i]=0;
       }
     }
   }

   /*
    * ¿Es viable este esquema? (es decir, ¿son fragmentos de cadena lineales?)
    */

   if (!viable()) {
     continue;
   }

   /*
    * Analizamos los trozos.
    */
   extremos=0;
   for (i=0;i<eslabones;i++) {
     enlaces=0;
     for (j=0;j<eslabones;j++) {
       if (nuevamatriz[i][j]) {
         enlaces++;
       }
     }
     if (enlaces>2) {
       abort();
     } else if (enlaces==1) {
       extremos++;
     } else if (enlaces==0 && !abrir[i]) {
       extremos+=2;
     }
   }

   /*
    * ¿Es posible reconstruir la cadena?
    */
   if (extremos-2<=2*abiertos) {
     if (abiertos<minabiertos) {
       minabiertos=abiertos;
     }
   }
 }

 return minabiertos;
}


int
main(void) {
 FILE *fdin;
 int a,b;
 int casodeprueba=1;

 fdin=fopen(«chains.in»,»r»);
 if (fdin==NULL) {
   abort();
 }

 while (1) {
   fscanf(fdin,»%d»,&eslabones);
   if (eslabones==0) {
     break;
   }
   memset(matriz,0,sizeof(matriz));

   while (1) {
     fscanf(fdin,»%d %d»,&a,&b);
     if (a==-1 && b==-1) {
       break;
     }
     matriz[a-1][b-1]=matriz[b-1][a-1]=1;
   }

   printf(«Set %d: Minimum links to open is %d\n»,casodeprueba++,minimo());
 }

 exit(0);

}
```