novática
UPGRADE 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). ***Novática*** edita asimismo **UP**GRADE, revista digital de **CEPIS** (*Council of European Professional Informatics Societies*), en lengua inglesa, y es miembro fundador de **UP**ENET (**UP**GRADE ***E**uropean **NET**work*).

<http://www.ati.es/novatica/>
<http://www.ati.es/reicis/>
<http://www.upgrade-cepis.org/>

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** 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./fax 963330392 <secreval@ati.es>
Administración y Redacción ATI Cataluña
Ciudad de Granada 131, 08018 Barcelona
Tlfn.934125235; fax 934127713 <secregen@ati.es>
Redacción ATI Andalucía
Isaac Newton, s/n, Ed. Sadiel,
Isla Cartuja 41092 Sevilla. Tlfn./fax 954460779 <secreand@ati.es>
Redacción ATI Aragón
Lagasca 9, 3-B, 50006 Zaragoza.
Tlfn./fax 976235181 <secreara@ati.es>
Redacción ATI Asturias-Cantabria <gp-astucant@ati.es>
Redacción ATI Castilla-La Mancha <gp-clmancha@ati.es>
Suscripción y Ventas
<http://www.ati.es/novatica/interes.html>, o en ATI Cataluña o ATI Madrid
Publicidad
Padilla 66, 3º, dcha., 28006 Madrid
Tlnf.914029391; fax.913093685 <novatica.publicidad@ati.es>
Imprenta
Derra S.A., Juan de Austria 66, 08005 Barcelona.
Depósito legal: B 15.154-1975 – ISSN: 0211-2124; CODEN NOVAEC
Portada: "Pioneros TIC" (Ada Byron), RFCalvo / (C) Rafael Fernández Calvo 2006
Diseño: Fernando Agresta / © ATI 2006

Nº 182, julio-agosto 2006, año XXXII

sumario

Monografía del próximo número: "*Servicios web*"

Subexpresiones

Ángel Herranz Nieva, Manuel Carro Liñares
Facultad de Informática, Universidad Politécnica de Madrid

<{aherranz, mcarro}@fi.upm.es

El enunciado de este problema apareció en el número 181 de ***Novática*** (mayo-junio 2006, p. 74). Es el problema H de los planteados en la fase final del III Concurso Universitario de la Comunidad Autónoma de Madrid (CUPCAM 2005), del que ATI fue entidad colaboradora.

Recordemos que el problema consistía en determinar el número de veces que una expresión es subexpresión de otra bajo el siguiente lenguaje:

$$E ::= f/0 \mid f/m\ e_1 \ldots e_m \qquad (0 < m < 10)$$

donde f/n es un símbolo de función, f una letra de la a a la z y n un número natural menor que 10.

La ventaja de ser el autor de un enunciado es que, al comentar la solución, uno sabe en lo que pensaba el autor que ideó el problema. En este caso se pretendía proponer un problema sencillo de resolver pero forzando una implementación basada en árboles. Sin embargo, la gramática con la que se ha definido el lenguaje es no ambigua lo que convierte el problema en un problema extremadamente sencillo: número de apariciones de una cadena dentro de otra cadena.

Subcadena

Una buena estrategia en un concurso de programación es intentar una solución *naive* a una problema más general, en este caso el problema más general es el de encaje de cadenas. Sin embargo, la solución genérica al problema de encaje de cadenas puede adaptarse a nuestro problema si se aprecian las siguientes propiedades:

- Los símbolos del alfabeto ocupan tres caracteres.
- Si la cadena s de longitud $|s|$ que representa una expresión del lenguaje es encontrada a partir de la posición p en la cadena e que representa otra expresión, entonces la siguiente aparición de s en e no se producirá antes de la posición $p+|s|$.

No consideramos necesario comentar el siguiente programa C que resuelve el problema.

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

#define N 3000

/*
 * Devuelve le número de apariciones
 * de s en e
 */
int n_ocurrencias(char *s,
                  char *e);
int main()
{
  char s[N], e[N];

  while (gets(s))
    {
      gets(e);
      printf("%i\n",
             n_ocurrencias(s,e));
    }
  return 0;
}

int es_subexpr(char *s, char *e)
{
  while (*s != '\0' && *e != '\0'
         && *s == *e) {
    s++;
    e++;
  }
  if (*s == '\0') return 0;
  if (*e == '\0') return -1;
  return 1;
}

int n_ocurrencias(char *s, char *e)
{
  int l = strlen(s);
  int n = 0;
  /* Cuidado si |e| < |s| */
  while (*e != '\0') {
    switch (es_subexpr(s,e)) {
    case -1:
      /* |e| < |s| */
      return 0;
      break;
    case 0:
      /* s ES PREFIJO de e */
      n++;
      e += l;
      break;
    case 1:
      /* s NO ES PREFIJO DE e */
      e += 3;
      break;
    }
  }
  return n;
}
```

¿Y con árboles?

El complejidad de la solución anterior es lineal con respecto al producto de las longitudes de las cadenas. Si n es el número de veces que la expresión s es subexpresión de e y $|e|$ y $|s|$ son las longitudes de las frases que representan e y s respectivamente, entonces la complejidad de la solución es[1]

$$O(n|s| + |e||s| - n|s|^2)$$

Las propiedades mencionadas anteriormente nos llevan a que $n < |e|/|s|$ siendo el peor caso $n = 0$:

$$O(|e||s|)$$

Dicha complejidad es del mismo orden que la complejidad en el peor caso de la solución *naive* al encaje de cadenas.

Una duda que nos surge es si, en el caso de representar las expresiones directamente como árboles, la complejidad de una solución *naive* sobre árboles puede llegar a ser inferior a la del programa presentado. No vamos a realizar una demostración rigurosa de que la complejidad no es menor pero vamos a intentar ofrecer al lector argumentos que le permitan confiar en la anterior afirmación.

Si lo que se busca es comprobar el número de veces que la expresión s es subexpresión de e se hace necesario comprobar si el árbol s es idéntico a cada subárbol de e. El número de subárboles de e es exactamente su número de nodos, que es a su vez un tercio de la longitud de la cadena que representa la expresión. La complejidad de la igualdad entre árboles es del orden del número de nodos menor de los dos árboles a comparar. La complejidad en el peor caso es $O(|e||s|)$.

Mejor solución

En [1] se analiza con detalle el problema de encontrar el número de apariciones de una cadena en otra. Desde el punto de vista de la complejidad, las mejores soluciones son aquéllas que construyen un autómata finito a partir de la expresión patrón por buscar. Dicho autómata es alimentado con la cadena sobre la que se hace la búsqueda y en tiempo lineal con respecto a la longitud de dicha cadena el autómata decide sobre la aparición del patrón. La construcción del autómata se hace en tiempo lineal con respecto a la cadena, por lo que la complejidad total es $O(|e|+|s|)$.

Referencias

[1] Thomas H. Cormen, Charles E. Leiserson y Ronald L. Rivest. *Introduction to Algorithms*. The MIT Electrical Engineering and Computer Science Series. The MIT Press, 1990.

Nota

[1] Dejamos al lector la comprobación de que dicha complejidad es correcta.