

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). Novática edita también Upgrade, revista digital de CEPIS (Council of European Professional Informatics Societies), en lengua inglesa.

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

ATI es miembro fundador de CEPIS (Council of European Professional Informatics Societies) y tiene un acuerdo de colaboración con ACM (Association for Computing Machinery). Tiene asimismo acuerdos de vinculación o colaboración con AdaSpain, AIZ y ASTIC.

CONSEJO EDITORIAL

Antoni Carbonell Nogueras, Francisco López Crespo, Julián Margelo Cocho, Celestino Martín Alonso, Josep Molas i Bertran, Roberto Moya Quiles, César Pérez Chirinos, Mario Piattini Velthuis, Fernando Píera Gómez (Presidente del Consejo), Miquel Sarríes Grifó, Asier Urbe Herranz

Coordinación Editorial
Rafael Fernández Calvo <r/calvo@ati.es>
Composición y autoedición

Jorge Llácer
Traducciones
Grupo de Lengua e Informática de ATI <<http://www.ati.es/g/lengua-informatica/>>
Administración
Tomás Brunete, María José Fernández, Enric Camarero, Felicidad López

SECCIONES TÉCNICAS: COORDINADORES

Administración Pública electrónica
Gumersindo García Arribas, Francisco López Crespo (MAP)
<gumersindo.garcia@map.es>, <flc@ati.es>
Arquitecturas
Jordi Tubella (DAC-UPC) <jortitub@ac.upc.es>
Victor Vitales Yofra (Univ. de Zaragoza) <vyofra@unizar.es>
Auditoría STIC
Marina Touriño, Manuel Palao (ASIA)
<marinatourino@marinatourino.com>, <manuel@palao.com>
Bases de datos
Coral Calero Muñoz, Mario G. Piattini Velthuis (Escuela Superior de Informática, UCLM)
<Coral.Calero@uclm.es>, <mpiatini@inf-cr.uclm.es>
Derecho y tecnologías
Isabel Hernando Collazos (Fac. Derecho de Donostia, UPV) <ihernando@legaltek.net>
Isabel Davara Fernández de Marcos (Davara & Davara) <isdavara@davara.com>
Enseñanza Universitaria de la Informática
Joaquín Ezpeleta Mateo (CPS-UZAR) <ezpeleta@posta.unizar.es>
Cristóbal Pareja Flores (DSIP-UCM) <cpajef@dsip.ucm.es>
Gestión del Conocimiento
Juan Baiget Solé (Cap Gemini Ernst & Young) <jbaiget@uoc.edu>
Informática y Filosofía
Josep Corco (UIC) <jcorco@unica.edu>
Esperanza Marcos (ESCET-URJC) <cuca@escet.urjc.es>
Informática Gráfica
Miguel Chover Sellés (Universitat Jaume I de Castellón) <mchover@lsi.uji.es>
Roberto Vivero (Eurographics, sección española) <r.vivero@dsic.upv.es>
Ingeniería del Software
Javier Dolado Cosin (DLSI-UPV) <dolado@si.ehu.es>
Luis Fernández (PRIS-EI-UEM) <lufern@pris.esi.uem.es>
Inteligencia Artificial
Federico Barber Vicente Botti (DSIC-UPV)
<fbvotti@barber.com>
Interacción Persona-Computador
Julio Abascal González (FI-UPV) <julio@si.ehu.es>
Jesus Lorés Vidal (Univ. de Lleida) <jesus@eup.udl.es>
Internet
Alonso Álvarez García (TID) <alonso@ati.es>
Llorenç Pagés Casas (Indra) <pages@ati.es>
Lenguaje Informática
M. del Carmen Ugarte (IBM) <cugarte@ati.es>
Lenguajes Informáticos
Andrés Marín López (Univ. Carlos III) <amarin@it.uc3m.es>
J. Ángel Velázquez (ESCET-URJC) <a.velazquez@escet.urjc.es>
Librerías de Informática
Alfonso Escolano (FIR-Univ. de La Laguna) <aescolano@ull.es>
Linguística Computacional
Xavier Gómez Guinovart (Univ. de Vigo) <xgg@uvigo.es>
Manuel Palomar (Univ. de Alicante) <mpalomar@dlsi.ua.es>
Mundo Estudiantil
Adolfo Vázquez Rodríguez (Rama de Estudiantes del IEEE-UCM)
<a.vazquez@ieee.org>
Profesión Informática
Rafael Fernández Calvo (ATI) <r/calvo@ati.es>
Miquel Sarríes Grifó (Ayto. de Barcelona) <msarries@ati.es>
Redes y servicios telemáticos
Luis Guijarro Coloma (DCOM-UPV) <lguijar@dc.com.upv.es>
Josep Solé Pareta (DAC-UPC) <pareta@ac.upc.es>
Seguridad
Javier Arellito (Redes y Sistemas, Bilbao) <jarellito@orion.deusto.es>
Javier López Muñoz (ETSI Informática-UMA) <jlm@cc.uma.es>
Sistemas de Tiempo Real
Alejandro Alonso, Juan Antonio de la Puente (DIT-UPM) <aalonso.jpunte@dit.upm.es>
Software Libre
Jesus M. González Barahona, Pedro de las Heras Quirós (GSYC-URJC) <[@gsyc.escet.urjc.es">jfgb.pheras@gsyc.escet.urjc.es](mailto:jfgb.pheras)>
Tecnología de Objetos
Jesus Garcia Molina (DIS-UM) <jmolina@correo.um.es>
Gustavo Rossi (LIFIA-UNLP, Argentina) <gustavo@sol.info.unpl.edu.ar>
Tecnologías para la Educación
Juan Manuel Dodero Beardo (UC3M) <jdodero@inf.uc3m.es>
Francisco Rivero Palomares (ATI) <frivero@wanadoo.es>
Tecnologías y Empresa
Pablo Hernández Medrano (Bluemart) <pablohm@bluemart.biz>
TIC para la Sanidad
Valentín Masero Vargas (DI-UNEX) <vmasero@unex.es>
TIC y Turismo
Andrés Aguayo Maldonado, Antonio Guevara Plaza (Univ. de Málaga)
<aguayo.guevara@icc.uma.es>

Las opiniones expresadas por los autores son responsabilidad exclusiva de los mismos. Novática permite la reproducción de todos los artículos, salvo los marcados con © o copyright, debiéndose en todo caso citar su procedencia y enviar a Novática un ejemplar de la publicación.

Coordinación Editorial, Redacción Central y Redacción ATI Madrid

Padilla 66, 3º dcha., 28006 Madrid
Tfn. 91 4029391; fax 91 3093685 <novatica@ati.es>
Composición, Edición y Redacción ATI Valencia
Av. del Reino de Valencia 23, 46005 Valencia
Tfn./fax 963330392 <cevalencia@ati.es>
Administración y Redacción ATI Cataluña
Via Laietana 41, 1º, 08003 Barcelona
Tfn. 93 4125235; fax 93 4127713 <cscregen@ati.es>
Redacción ATI Andalucía
Isaac Newton, s/n, Ed. Sadiel,
Isla Cartuja 41092 Sevilla, Tfn./fax 954460779 <cscreand@ati.es>
Redacción ATI Aragón
Lagasca 9, 3-B, 50006 Zaragoza
Tfn./fax 976235181 <cscreara@ati.es>
Redacción ATI Asturias-Cantabria
<gp-astucant@ati.es>
Redacción ATI Castilla-La Mancha
<gp-clmancha@ati.es>
Redacción ATI Galicia
Recinto Ferial s/n, 36540 Silleda (Pontevedra)
Tfn. 986581413; fax 986580162 <cscregal@ati.es>

Suscripción Ventas
<<http://www.ati.es/novatica/interes.html>>, o en ATI Cataluña o ATI Madrid
Publicidad
Padilla 66, 3º dcha., 28006 Madrid
Tfn. 91 4029391; fax 91 3093685 <novatica.publicidad@ati.es>
Imprenta
9 impressió S.A., Juan de Austria 66, 08005 Barcelona.
Depósito legal: B 15.154-1975 -- ISSN: 0211-2124; CODEN NOVAC E
Portada: Antonio Crespo Foix / © ATI 2004
Diseño: Fernando Agresta / © ATI 2004

editorial

ATI contra el cánón privado sobre soportes digitales

en resumen

Redes e historia

Rafael Fernández Calvo

monografía

Redes Inalámbricas: una nueva era en las Telecomunicaciones

(En colaboración con Upgrade).

Editores invitados: M. Ufuk Çağlayan, Vicente Casares Giner y Jordi Domingo i Pascual

Presentación. Redes de acceso inalámbricas:

hacia las comunicaciones móviles integradas

> 03

Vicente Casares Giner, Jordi Domingo Pascual

¿Cuál es la extensión óptima de un enlace inalámbrico?

> 06

M. Ufuk Çağlayan, Fikret Sivrikaya, Bülent Yener

Capacidad en Sistemas Celulares WCDMA: Métodos de Análisis

> 11

Luis Mendo Tomás

Perspectiva de la gestión de recursos radio en las redes celulares

> 15

Oriol Salent Roig, Jordi Pérez Romero, Ramón Agustí Comes

Estrategias de Gestión de Localización en la próxima generación

de Sistemas de Comunicaciones Móviles

> 20

Pablo García Escalle, Vicente Casares Giner

Movilidad IP: macromovilidad, micromovilidad, calidad de servicio y seguridad

> 28

Josep Mangles Bañalluy, Albert Cabellos Aparicio, René Serral Gracià, Jordi Domingo Pascual,

Antonio Gómez Skarmeta, Tomás P. de Miguel, Marcelo Bagnulo, Alberto García Martínez

Redes Inalámbricas ad hoc como tecnología de soporte para la Computación Ubicua

> 33

Juan Carlos Cano Escrivá, Carlos Miguel Tavares Calafate,

Manuel José Pérez Malumbres, Pietro Manzoni

Las WPAN en el trayecto hacia la 4G

> 39

Ramón Agüero Calvo, Johnny Choque Ollachica, José Ángel Irastorza Teja,

Luis Muñoz Gutiérrez, Luis Sánchez González

secciones técnicas

Bases de Datos

Diseño lógico de Almacenes de Datos: efectividad del diseño en estrella

> 44

Coral Calero Muñoz, Mario Piattini Velthuis, Manuel A. Serrano Martín

Enseñanza Universitaria de la Informática

La Asociación de Enseñantes Universitarios de Informática (AENUI)

> 47

Pedro Blesa Pons, Joe Miró Julià, Francisco Ruiz González

Informática Gráfica

Visualización de terreno en tiempo real

> 50

Cristina Rebollo Santamaría, Inmaculada Remolar Quintana, Miguel Chover Sellés

Ingeniería del Software

Buscando el Santo Grial de la Ingeniería del Software

> 54

Robert L. Glass

Interacción Persona-Computador

Patrones de interfaz de usuario para la navegación orientada a objetos

> 55

Pedro Juan Molina Moreno, Ismael Torres Boigues, Oscar Pastor López

Redes y servicios telemáticos

Análisis y Diseño de Políticas de Control de Admisión

en Redes Celulares Multiservicio

> 61

Vicent Pla Boscà, Vicente Casares Giner

Referencias autorizadas

> 68

sociedad de la información

programar es crear

Subcadenas en la secuencia "mira-y-di" (CUPCAM 2003, problema C, enunciado)

> 73

Óscar Martín Sánchez, Manuel Carro Liñares

Reconstrucción de árboles inclinados a partir de dos de sus recorridos

(CUPCAM 2003, problema B, solución)

> 74

Cristóbal Pareja Flores, Ángel Herranz Nieva

asuntos internos

Coordinación editorial / Programación de Novática

> 76

Normas de publicación para autores / Socios Institucionales

> 77

Monografía del próximo número: "UML (Unified Modeling Language)"

Cristóbal Pareja Flores¹, Ángel Herranz Nieva²

¹Depto. Lenguajes y Sistemas Informáticos, Universidad Complutense de Madrid; ²Facultad de Informática, Universidad Politécnica de Madrid

<cpareja@sip.ucm.es>, <aherranz@fi.upm.es>

1. Resumen del problema

El problema, brevemente, pedía encontrar el número de árboles binarios desviados a la izquierda cuyos recorridos por niveles y en 'inorden' eran dados como datos de entrada. En el problema se decía que "un árbol binario está desviado a la izquierda cuando todos sus niveles están alineados a la izquierda".

2. Fuerza bruta

La primera opción en la que se puede pensar es en la del uso de la tan socorrida fuerza bruta: generar árboles binarios, más o menos indiscriminadamente, con n nodos (donde n es la longitud de los recorridos) que respondan a uno de los recorridos, para comprobar luego si encajan en el otro recorrido. A priori, la ineficiencia estaría garantizada: sin nada de inteligencia en la generación de árboles, el algoritmo tendría una complejidad de orden factorial (!).

Dada la simplicidad del algoritmo creemos que merece la pena mostrar su esquema. El lenguaje de programación utilizado es Haskell 98 [1][2]:

```
> desdeInorden : String -> [ArbolChar]
>
> porNiveles : ArbolChar -> String
>
> estaDesviado : ArbolChar -> Bool
>
> solucion : (String, String)
>           -> [ArbolChar]
> solucion (niv,inord) =
>   (filter ((niv==) . porNiveles)
>    . filter (estaDesviado)
>    . desdeInorden) inord
```

donde las operaciones desdeInorden, porNiveles y estaDesviado son relativamente sencillas de implementar y el tipo ArbolChar se define en la sección 4.

3. Inteligencia

Un análisis más cauto revela propiedades del problema que permitirían realizar búsquedas en un dominio mucho más pequeño.

3.1. Sufijos viables en el recorrido por niveles

Empezamos observando lo siguiente:

En el último nivel del árbol solución, el carácter inicial es también el primero del recorrido en 'inorden' dado; y ése es el único principio posible de la última fila.

Reconstrucción de árboles inclinados a partir de dos de sus recorridos

Solución del problema B de los planteados en el I Concurso Universitario de la Comunidad Autónoma de Madrid (CUPCAM 2003). El enunciado de este problema apareció en el número 166 de Novática (noviembre-diciembre 2003, p. 71).

Buscando ese carácter en el recorrido por niveles, encontraremos distintas posibilidades (o posiblemente ninguna). Cada una de ellas da lugar a un "proyecto de solución" que consta de una lista de árboles (por ahora, triviales: árboles hoja), que son los del último nivel: $[t_1, t_2, \dots, t_k]$. Por ejemplo, el par de recorridos (niveles, 'inorden') siguiente ("121121121", "112211211")

indica que el último nivel empieza con un "1" (primer carácter del 'inorden'). Así que el último nivel está entre los siguientes, {"1", "121", "1121", "121121", "1121121", "121121121"}

que son los sufijos del recorrido por niveles que empiezan por un 1. Aparentemente, tenemos seis proyectos de solución prometedores.

3.2. Límites en la construcción de árboles

Pero ahora podemos hacer otra observación. Una lista de k árboles nivelados no puede unirse con menos de k-1 nodos intermedios y, de ellos, al menos la mitad han de estar en el primer nivel nuevo.

En nuestro ejemplo, si el último nivel fuera "121121", debería completarse con tres caracteres ("121") lo que es materialmente imposible. Esto limita los seis segmentos candidatos del recorrido por niveles según su longitud. En nuestro caso concreto, nos quedamos con los tres primeros candidatos: {"1", "121", "1121"}

La figura 1 muestra una búsqueda completa de árboles válidos para los recorridos estudiados como ejemplo aplicando el algoritmo esquematizado en esta sección e implementado en la siguiente.

4. Implementación

Representaremos los datos de entrada, los recorridos por niveles y en 'inorden', como pares de cadenas de caracteres:

```
> type Recorridos = (String, String)
```

Para representar los árboles nos apoyaremos en el siguiente tipo algebraico:

```
> data Arbol a =
>   Vacio
>   | Nodo (Arbol a) a (Arbol a)
Como la información de los nodos serán caracteres:
> type ArbolChar = Arbol Char
```

Por otra parte, con cada proyecto de solución $[t_1, t_2, \dots, t_k]$, el primer carácter c del nivel nuevo ha de ser tal que, junto con los dos primeros árboles y de la lista, ha de dar lugar a un árbol cuyo recorrido en 'inorden'

```
ordenCentral (Nodo c )
sea prefijo del 'inorden' global dado, donde ordenCentral se puede definir de esta forma:
```

```
> ordenCentral :: ArbolChar -> String
> ordenCentral Vacio =
>   ""
> ordenCentral (Nodo t1 c t2) =
>   ordenCentral t1
> ++ c : ordenCentral t2
```

Siguiendo con nuestro ejemplo, la elección de "1121" como último nivel daría lugar a un recorrido en 'inorden' de la forma "1?1... ", lo que no encaja con el 'inorden' dado.

Esta segunda restricción nos permite filtrar aún más los proyectos de solución.

Estas dos ideas determinan los posibles pares de segmentos en el recorrido por niveles: el del siguiente nivel y el resto, para los niveles posteriores:

```
> segmentos ::
>   ([ArbolChar], Recorridos)
>   -> [(ArbolChar)]
> segmentos (arbs, niv, inord) =
>   let l = length arbs
>       n = length niv
>       in [ (izda, dcha)
>           | k <- [div (l+1) 2
>                 .. div (n+1) 2],
>           let dcha = drop (n-k) niv,
>               esViable
>                 = (prefijo arbs $ head dcha)
>                   inord,
>               let izda = take (n-k) niv
>                   ]
```

donde las funciones prefijo y esViable se han programado en el siguiente código:

```
> prefijo :: [ArbolChar] -> Char
>         -> ArbolChar
> prefijo [t1] c = Nodo t1 c Vacio
> prefijo (t1:t2:_) c = Nodo t1 c t2
> esViable :: ArbolChar -> String
>          -> Bool
> esViable arbol cadena =
>   and $ zipWith (==)
```

“ El problema pedía encontrar el número de árboles binarios *desviados a la izquierda* ”

```
> (ordenCentral arbol)
> cadena

Ahora, las maneras de completar un nivel
vienen dadas por los posibles caracteres via-
bles según la función segmentos, ya descrita:

> completar ::
> ([ ArbolChar], Recorridos)
> -> [ ([ ArbolChar], Recorridos)]
> - PRE: las cadenas de caracteres
> - son no vacías
> completar (arbs, niv, inord) =
```

```
> let pares = segmentos (arbs,
>                       niv,
>                       inord)
> in [ (ponerFila arbs dcha,
>      izda,
>      inord)
>     | (izda, dcha) <- pares ]
```

```
> ponerFila :: [ ArbolChar] -> String
>           -> [ ArbolChar]
> ponerFila [] resto =
> [ Nodo Vacio c Vacio
> | c <- resto ]
> ponerFila [a] (r:resto) =
>   Nodo a r Vacio
>   : ponerFila [] resto
> ponerFila (a:b:bs) (r:resto) =
>   Nodo a r b : ponerFila bs resto
```

donde ponerFila empareja los demás árbo-
les, pudiendo quedar uno suelto y, también,
sobresalir en ese nivel, más árboles triviales,
hasta completar el substring que estamos
considerando:

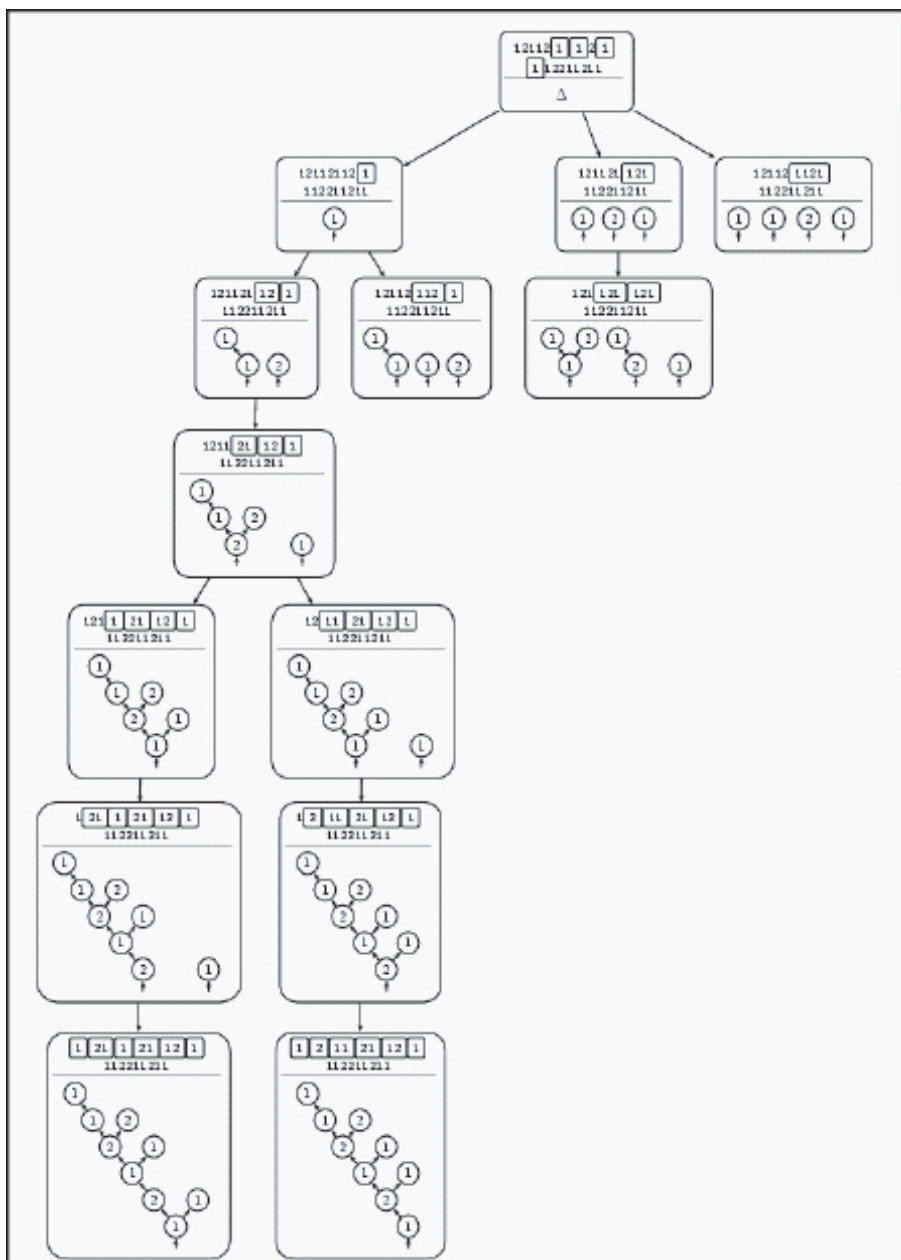


Figura 1. Búsqueda completa de árboles válidos para los recorridos.

Notas

[1] P. Hudak, J. Fasel, J. Peterson. *A gentle introduction to Haskell*. Informe Técnico YALEU/DCS/RR-90, Yale University, 1996.

[2] S. Peyton Jones, J. Hughes. *Report on the Programming Language Haskell 98. A Non-strict Purely Functional Language*, Febrero, 1999.