

Estudio sobre la Visualización de las Técnicas de Diseño de Algoritmos

Luis Fernández¹, J. Ángel Velázquez²

¹ Dpto. Lenguajes, Proyectos y Sistemas Informáticos,
Escuela Universitaria de Informática,
Universidad Politécnica de Madrid,
Ctra. Valencia km 7, 28031 Madrid
setillo@eui.upm.es

² Departamento de Lenguajes y Sistemas Informáticos,
Universidad Rey Juan Carlos,
C/ Tulipán s/n, 28933 Móstoles, Madrid
angel.velazquez@urjc.es

Abstract. Este trabajo presenta dos estudios sobre la visualización de técnicas de diseño de algoritmos. Se definió una hipótesis sobre las visualizaciones más útiles, basada en la experiencia docente de los autores, y se realizaron dos estudios para comprobar su validez. Por un lado, se examinaron 12 libros de texto sobre algoritmos para identificar las visualizaciones utilizadas con más frecuencia. Por otro, se realizó una encuesta a alumnos de una asignatura de algoritmos sobre las visualizaciones propuestas por los autores. Los resultados confirman la validez de la hipótesis, con algunas matizaciones. El trabajo proporciona una base empírica para el desarrollo de un sistema de visualización automática de programas basada en técnicas de diseño de algoritmos.

1 Introducción

La visualización de programas y algoritmos se ha utilizado en los últimos 25 años con un fin predominantemente docente. Sin embargo, su uso no se ha generalizado debido a dos razones principales [1][2]:

- falta de garantía de eficacia educativa, y
- excesivo esfuerzo de desarrollo para los profesores.

Se han adoptado varios enfoques para reducir el esfuerzo [3][4]. Un enfoque consiste en generar automáticamente visualizaciones de programas a partir del código fuente. Se suele aducir como inconveniente de este enfoque que produce visualizaciones menos expresivas o imaginativas que otras diseñadas *ad hoc*. Sin embargo, la experiencia muestra que muchas están estandarizadas de hecho, por lo que son suficientes (p.ej. véanse las de [5]).

Una forma de aumentar la expresividad de las visualizaciones automáticas consiste en restringirlas a dominios concretos. Dichas visualizaciones suelen estar diseñadas de forma más cuidada que en sistemas generalistas. Hay ejemplos de estos sistemas aplicados a código fuente [6], expresiones [7], grafos [8][9], etc. Actualmente estamos

desarrollando un sistema de visualización de programas basado en técnicas de diseño de algoritmos [10]. Inicialmente sólo abordaremos 4 técnicas de diseño: divide y vencerás, vuelta atrás, programación dinámica y algoritmos voraces.

En el artículo presentamos dos estudios realizados para seleccionar visualizaciones adecuadas. El apartado segundo comenta trabajos relacionados con nuestro enfoque. En el tercer apartado enumeramos nuestra hipótesis sobre las visualizaciones más útiles de técnicas de diseño. Los dos apartados siguientes describen los estudios realizados para validar dicha hipótesis: un estudio de las visualizaciones de técnicas de diseño utilizadas en 12 libros de texto y una encuesta realizada entre alumnos de una asignatura de algoritmos sobre sus preferencias de visualización. Por último, comentamos los resultados y resumimos las conclusiones y el trabajo en curso.

2 Trabajos Relacionados

Existen numerosos trabajos de visualización de estructuras de datos, destacando por su dificultad los orientados a grafos [11][12][13]. Nuestro enfoque es similar, pero de técnicas de diseño de algoritmos. No conocemos ninguna animación orientada a técnicas de diseño en general, aunque existen de algoritmos concretos desarrollados según alguna técnica de diseño. Por ejemplo, pueden encontrarse animaciones de algoritmos basados en divide y vencerás (p.ej. de ordenación [14]), en búsqueda en espacios de estados (p.ej. 8-puzzle [15]), etc.

Existe cierta similitud entre nuestro trabajo y otros sobre visualizaciones parciales de programas. El aspecto parcial más relacionado es la visualización de la recursión (véase un muestrario en [16]), ya que surge al visualizar varias técnicas de diseño.

3 Hipótesis y Metodología de Trabajo

Partimos de la hipótesis de que las visualizaciones más útiles de técnicas de diseño de algoritmos son algunas de las utilizadas por los autores tras varios años de docencia:

- Divide y vencerás. Dos visualizaciones: de la definición recursiva (véase Fig. 1) y del árbol de recursión.

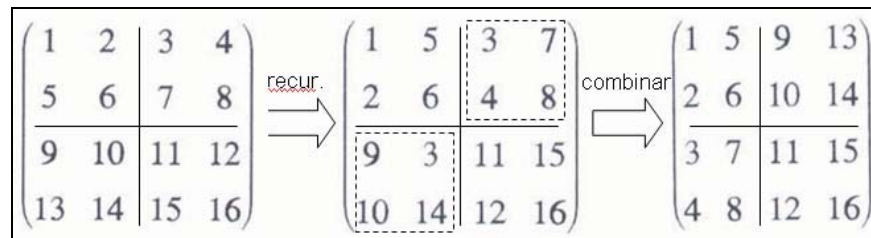


Fig. 1. Visualización de la definición recursiva para transponer una matriz

- Búsqueda en espacios de estados. Dos visualizaciones: árbol de búsqueda potencial y árbol de búsqueda generado (véase Fig. 2):

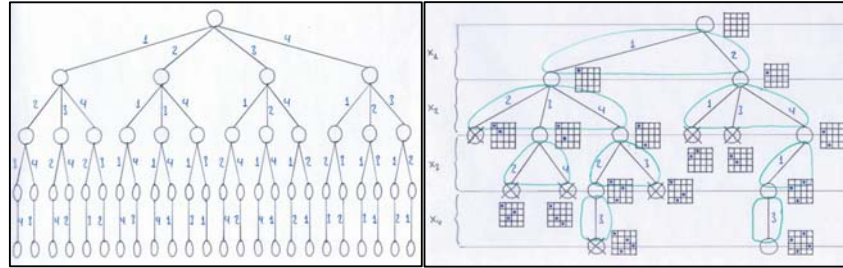


Fig. 2. Árboles de búsqueda potencial y generado para el problema de las 4 reinas

- Programación dinámica. Cuatro visualizaciones: árbol de recursión, grafo de dependencia, tabla de valores (véase Fig. 3) y tabla de decisiones:

p	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	7	7	7	7	7	11	11	11	11	15	15	15	15
2	0	0	0	0	0	4	4	4	4	8	8	8	8	8	12	12
3	0	0	0	0	0	4	4	4	4	8	8	8	8	8	12	12
4	0	0	0	0	0	4	4	4	4	4	4	4	4	4	4	4
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 3. Tabla de valores para el problema de la mochila 0/1

- Algoritmos voraces. No identificamos ninguna visualización general de la técnica.

Para comprobar esta hipótesis, se realizó un doble trabajo, que describimos en los apartados siguientes. Por un lado, se realizó una búsqueda bibliográfica de dichas visualizaciones y por otro, se realizó una encuesta entre alumnos de una asignatura de algoritmia para conocer su opinión con respecto a las visualizaciones seleccionadas.

4 Estudio de Visualizaciones en Libros de Texto

El primer estudio consistió en analizar las visualizaciones de técnicas de diseño presentes en 12 conocidos libros de texto [17][18][19][20][21][22][23][24][25][26][27][28]. La lista original era algo más larga, pero descartamos algunos libros por razones de disponibilidad. En ningún caso cuestionamos la catalogación que el autor realiza de los algoritmos por técnicas de diseño.

Cada libro se ha examinado para comprobar el uso de visualizaciones propias de técnicas de diseño de algoritmos, sean las propuestas por nosotros u otras distintas. Para buscar las visualizaciones, se siguieron tres criterios en el siguiente orden:

- Analizar el índice temático para determinar si existe un capítulo o apartado dedicado a cada técnica de diseño.
- Analizar el índice de figuras o algoritmos en busca de visualizaciones de ejercicios paradigmáticos de cada técnica de diseño (p.ej. *quicksort* para divide y vencerás).
- Analizar el índice cruzado en busca de referencias a técnicas de diseño o ejercicios paradigmáticos (p.ej. divide y vencerás, *divide & conquer*, *quicksort*, etc.).

4.1 Resultados Generales

Algunas conclusiones comunes a todas las técnicas de diseño son las siguientes:

- La inesperada ausencia de visualizaciones en textos con decenas de algoritmos desarrollados desde su análisis hasta el código.
- La también inesperada cantidad de visualizaciones específicas del dominio. Sin duda, este tipo de visualizaciones ayuda a entender mejor el enunciado, las estructuras de datos elegidas o el estado de la solución, pero no proporcionan una visualización del algoritmo conforme a su técnica de diseño. Hay numerosos ejemplos: las torres de Hanoi, códigos de Huffman, tres en raya, mochila, n -reinas, etc. En Fig. 4 y Fig. 5 se presentan dos ejemplos de visualizaciones [21].

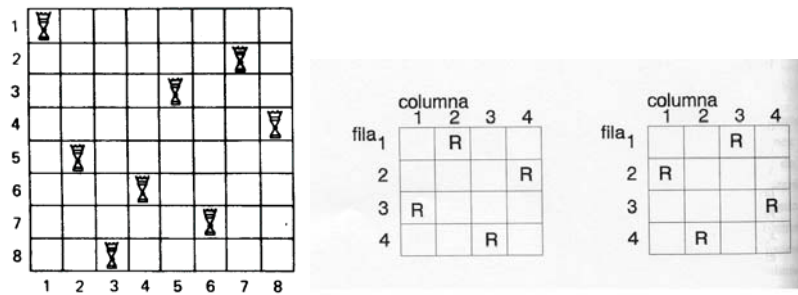


Fig. 4. Visualizaciones de varias soluciones del problema de las n -reinas

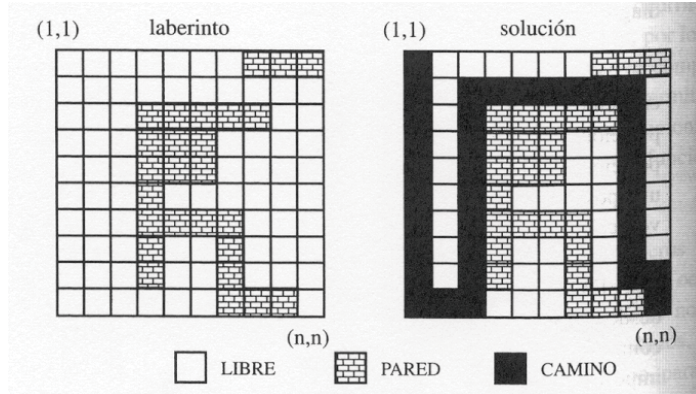


Fig. 5. Visualizaciones del estado inicial y el estado final del problema del laberinto

A continuación se presentan los resultados específicos de técnicas de diseño. Presentamos con algún detalle algunos resultados de divide y vencerás, pero sólo resumimos las conclusiones obtenidas para las demás técnicas.

4.2 Visualizaciones de Divide y Vencerás

En este apartado presentamos las características de las visualizaciones encontradas para el algoritmo de ordenación por mezcla (*mergesort*), por su amplio uso, y después resumimos los resultados encontrados para otros algoritmos.

La tabla 1 muestra los resultados de analizar las visualizaciones encontradas en 10 libros para la ordenación por mezcla.

Tabla 1. Tabla de visualizaciones de la ordenación por mezcla

Texto	Visualización
[18]	Secuencia de visualizaciones: vector dado desordenado, subvectores desordenados, subvectores ordenados y vector ordenado
[19]	Secuencia de visualizaciones: cambios de estado en el vector por la mezcla, desde el dado hasta el ordenado
[20]	Secuencia de visualizaciones: cambios de estado por la mezcla
[21]	Dos árboles “paralelos” con índices de llamadas: recursivas y a <i>mezclar</i>
[23]	Secuencia de visualizaciones: vector dado desordenado, vector con dos lados ordenados y vector ordenado
[24]	Árbol de recursión con contenidos ya ordenados
[25]	Dos árboles “paralelos”: con subvectores desordenados y ordenados
[26]	Árbol de recursión con índices para ordenar subvectores y su mezcla
[27]	Árbol de recursión con índices
[28]	Secuencia de visualizaciones: vectores desordenados y su mezcla hasta vectores ordenados (árbol de recursión implícito)

Básicamente encontramos las dos visualizaciones propuestas por nosotros:

- Visualización de la definición recursiva. Partiendo de un vector desordenado, se muestra cómo se parte en 2 subvectores, el resultado de ordenar cada uno por separado y su mezcla para producir el vector ordenado final.
- Visualización del árbol de recursión. Partiendo de un vector desordenado, se muestran todos los subvectores en que se parte y todas las mezclas hasta su ordenación final.

Resulta interesante que la segunda visualización presenta numerosas variantes:

- Visualizar la ordenación del vector mediante un solo árbol o dos árboles paralelos (el principal y otro para ilustrar la realización de la operación auxiliar).
- Mostrar el árbol de recursión o una secuencia de visualizaciones (árbol implícito).
- Mostrar los índices o el contenido de los subvectores.
- Mostrar el contenido dado o final de los subvectores (es decir, desordenado u ordenado).

Hemos encontrado visualizaciones del algoritmo de ordenación rápida (*quicksort*), así como de los algoritmos de *heapsort*, min-max, búsqueda binaria, Strassen, Fourier rápido, intercambio, mediana, torneo, transpuesta de una matriz y Fibonacci. En resumen, encontramos las dos visualizaciones básicas con las variantes descritas, junto con una variante más:

- Disposición ascendente o descendente del árbol de recursión.

4.3 Visualizaciones de Otras Técnicas de Diseño de Algoritmos

Hemos analizado las visualizaciones para vuelta atrás, programación dinámica y algoritmos voraces. Para vuelta atrás encontramos las dos visualizaciones propuestas, árbol de búsqueda completo y árbol de búsqueda generado. Para programación dinámica sólo hemos encontrado la visualización de las tablas de valores y de decisiones. Las tablas pueden contener algunos elementos gráficos más, sobre todo la dependencia entre celdas. Con respecto a los algoritmos voraces, no hemos encontrado ninguna visualización independiente del dominio, sino específicas: árboles de Huffman, algoritmos de Prim y Kruskal, problema de la mochila, etc.

5 Encuesta a Alumnos

Realizamos una encuesta a 20 alumnos de la asignatura “Diseño y Análisis de Algoritmos”, de tercer curso de Ingeniería Informática de la Universidad Rey Juan Carlos, en el curso 2004-2005. El objetivo fue conocer su opinión sobre las visualizaciones propuestas.

Algunas conclusiones son:

- Existe una homogénea y gran aceptación de las visualizaciones por parte de todos los alumnos, con una media de 4.2 por alumno en una escala de 1 a 5 y una

desviación típica de 0'9. Los valores mínimo y máximo varían entre 3'2 (valor superior al “aprobado”) y 5'0, respectivamente.

- Los resultados se resumen por técnicas de diseño en la tabla 1. Obsérvese que dos visualizaciones propuestas para programación dinámica –árbol de recursión y grafo de dependencia– se agrupan por separado porque se presentaron en un capítulo previo de la asignatura, bajo el epígrafe de eliminación de la recursividad múltiple. Por técnica, la puntuación varía desde una media mínima de 3'8 para el árbol de recursión en divide y vencerás hasta una media máxima de 4'5 para el grafo de dependencia en la eliminación de recursividad múltiple.
- No hay grandes diferencias entre visualizaciones de una misma técnica de diseño. De hecho, muchos alumnos dieron la misma puntuación a las distintas visualizaciones de una misma técnica, e incluso una única puntuación por técnica. En todo caso, sus preferencias son:
 - Divide y vencerás: definición recursiva mejor que árbol de recursión.
 - Vuelta atrás: árbol de búsqueda realmente generado mejor que el potencial.
 - Programación dinámica: tabla de valores mejor que visualizaciones de la eliminación de la recursividad, y éstas mejor que la tabla de decisiones.
 - Eliminación de recursividad múltiple: grafo de dependencia mejor que árbol de recursión.

Tabla 1. Datos de la encuesta por visualizaciones de las técnicas de diseño

Técnica de diseño	Visualización	Media	Desviación típica
Eliminación de la recursividad múltiple	Árbol de recursión	4'3	0,8
	Grafo de dependencia	4'5	0'7
Divide y vencerás	Árbol de recursión	3'8	0'8
	Definición recursiva	4'3	0'7
Vuelta atrás	Árbol de búsqueda potencial	3'9	0'8
	Árbol de búsqueda generado	4'4	0'6
Programación dinámica	Grafo de dependencia	4'1	1'2
	Tabla de valores	4'3	0'9
	Tabla de decisiones	3'9	1'1

La encuesta también ofreció la posibilidad de dar respuestas abiertas. Quizá podamos resumir la mayor parte de los comentarios en dos afirmaciones:

- Reclaman el uso generalizado de visualizaciones para ejercicios y ejemplos, y no sólo en la exposición teórica de una técnica de diseño sobre un ejemplo particular.
- Para algunos alumnos, basta con las visualizaciones para comprender mejor los algoritmos. Sin embargo, para otros las visualizaciones solas no son suficiente, sino que deben complementarse con explicaciones o con animaciones.
- Algunos alumnos sugieren que las visualizaciones deberían ser más detalladas. En otro caso, sugiere combinar dos visualizaciones distintas (grafo de dependencia y tabla de valores) para entender mejor el algoritmo (de programación dinámica).

6 Discusión

Un primer resultado es la ausencia de trabajos encontrados sobre visualización de programas para técnicas de diseño de algoritmos. Por esta razón, hemos contrastado nuestra hipótesis con libros de texto y con los alumnos. El valor de cada estudio es diferente: las visualizaciones de los libros de texto señalan tendencias de visualización (presencia o ausencia, más formatos), mientras que la encuesta tiene un valor refutativo, en su caso, de las visualizaciones.

La gran mayoría de las visualizaciones encontradas en libros son específicas del dominio. Ayudan a entender el algoritmo pero no ilustran su técnica de diseño. Este hecho es más evidente en los algoritmos voraces, donde no hemos podido identificar ninguna visualización general. Quizá este hallazgo dé cierto respaldo empírico a la propuesta de Nikander *et al.* [29] de visualizar estructuras de datos en 4 niveles de abstracción, donde la visualización del dominio corresponde al nivel superior.

Las siete visualizaciones propuestas gozan de respaldo bibliográfico, salvo el grafo de dependencia. Este desacuerdo se debe al enfoque adoptado por el segundo autor para la programación dinámica, basado en métodos de transformación de programas [30][31]. Por otro lado, consideramos necesario realizar un estudio bibliográfico más detallado para identificar los elementos gráficos relevantes en cada visualización.

En cuanto a la encuesta, no sorprenden los altos valores obtenidos por todas las visualizaciones, ya que una de las ventajas conocidas de la visualización es el aumento de motivación de los alumnos [1]. Los valores para las distintas visualizaciones de cada técnica de diseño son parecidos, aunque con matizaciones:

- Existen dos visualizaciones, el árbol de recursión y el grafo de dependencia, que se presentan dos veces cada una, obteniendo distintos resultados. Ambas tienen más aceptación en la eliminación de la recursividad múltiple que en las otras dos técnicas (divide y vencerás, programación dinámica). No tenemos datos completos para interpretar este desequilibrio. Puede deberse a que, en la eliminación de la recursividad múltiple, forman parte del propio método seguido. En el caso de divide y vencerás, además coincide con el método docente de los autores, que prima la definición recursiva frente al rastreo de su comportamiento.
- En vuelta atrás se da la situación contraria a la comentada en el punto anterior. Los alumnos prefieren el árbol realmente generado frente al árbol potencial. Aunque éste último es más útil para diseñar el algoritmo, el primero es más útil para rastrear su comportamiento. Quizás deba a la mayor dificultad de los algoritmos de vuelta atrás.
- Algunas preferencias pueden depender de la frecuencia de uso en el aula. Aparte del caso del árbol de recursión antes comentado, en programación dinámica se prefieren las tablas de valores, mucho más usadas que las tablas de decisiones.

En todo caso, conviene corroborar estas preferencias mediante más encuestas, aunque probablemente sería más realista comprobarlas sobre un prototipo funcionando.

7 Conclusiones y Trabajos Futuros

En el artículo hemos descrito un nuevo enfoque de visualización de programas, basado en las técnicas de diseño de algoritmos. Hemos hecho una propuesta de 7 visualizaciones adecuadas para las 4 técnicas de diseño más usadas en la docencia. Hemos validado la aceptación de la propuesta mediante las visualizaciones contenidas en libros de texto y una encuesta a alumnos. Los resultados han sido positivos, con varias matizaciones sobre el formato o sobre la aceptación de ciertas visualizaciones.

Actualmente estamos desarrollando un sistema para la generación automática de estas visualizaciones. Se ha desarrollado una arquitectura genérica y un prototipo de visualización del árbol de recursión [10]. El sistema permitirá que el usuario particularice cada visualización según los aspectos que más le interesen, como índices o contenidos en algoritmos de divide y vencerás.

Todavía queda trabajo pendiente sobre el diseño de las visualizaciones. Por un lado, hay que identificar claramente los elementos gráficos propios de cada una, como se ha visto en divide y vencerás. Conviene tener en cuenta que el estudio realizado es necesariamente parcial porque las animaciones introducen un factor extra sobre las visualizaciones estáticas. Asimismo, sería interesante analizar las diferencias entre distintos algoritmos de una misma técnica de diseño para comprender mejor la razón de la existencia de estas variantes. Otros dos retos son la visualización automática de la definición recursiva y el diseño de alguna visualización general para algoritmos voraces.

Agradecimientos

Este trabajo se ha financiado parcialmente con el proyecto TIN2004-07568 del Ministerio de Educación y Ciencia.

Referencias

1. Naps, T. et al.: Exploring the role of visualization and engagement in computer science education. *ACM SIGCSE Bulletin* 35, 2 (2003) 131-152
2. Naps, T. et al.: Evaluating the educational impact of visualization. *ACM SIGCSE Bulletin* 35, 4 (2003) 124-136
3. Hundhausen, C.D., Douglas, S.A.: Low-fidelity algorithm visualization. *Journal of Visual Languages and Computing* 13 (2002) 449-470
4. Ihanola, P., Karavirta, V., Korhonen, A., Nikander, J.: Taxonomy of effortless creation of algorithm visualization. *Proc. 2005 International Workshop on Computing Education Research*, ACM Press (2005) 123-133
5. LaFollette, P., Korsh, J., Sangwan, R.: A visual interface for effortless animation of C/C++ programs. *Journal of Visual Languages and Computing* 11 (2000) 27-48
6. Baecker, R., Marcus, A.: *Human Factors and Typography for More Readable Programs*, ACM Press (1990)
7. Velázquez Iturbide, J.Á., Pareja Flores, C., Urquiza Fuentes, J.: An approach to effortless construction of program animations. *Computers & Education* (en imprenta)

8. Di Batista, P., Eades, G., Tamassia, T., Tollis, I.: Graph Drawing: Algorithms for the Visualization of Graphs. Prentice-Hall (1999)
9. Herman, I., Melancon, G., Marshall, M.S.: Graph visualization and navigation in information visualization: A survey. *IEEE Trans. on Visualization and Computer Graphics* 6, 1 (2000) 24-43
10. Fernández Muñoz, L.: Visualización de programas para técnicas de diseño de algoritmos. Trabajo tutelado, Universidad Rey Juan Carlos (2005)
11. Khuri, S., Holzapfel, K.: EVEGA: An educational visualization environment for graph algorithms. *Proc. 6th Annual Conference on Innovation and Technology in Computer Science Education*, ACM Press (2001) 101-104
12. Lucas, J., Naps, T., Rößling, G.: VisualGraph – A graph class designed for both undergraduate students and educators. *Proc. 34th SIGCSE Technical Symposium on Computer Science Education*, ACM Press (2003) 167-171
13. Wu, M.: Teaching graph algorithms using online package IAPPGA. *ACM SIGCSE Bulletin* 37, 4 (2005) 64-68
14. Brummond, P.: The Complete Collection of Algorithm Animations. <http://www.cs.hope.edu/~alanim/ccaa> (2001)
15. MacDonald, P., Ciesielski, V.: Design and evaluation of an algorithm animation of state space search methods. *Computer Science Education* 12, 4 (2002) 301-324
16. Haynes, M.: Explaining recursion to the unsophisticated. *ACM SIGCSE Bulletin* 27, 3 (1995) 3-6
17. Aho, V., Hopcroft, J.E., Ullman, J.D.: Estructuras de datos y algoritmos. Addison-Wesley Iberoamericana (1988)
18. Brassard, G., Bratley, P.: Fundamentals of Algorithmics. Prentice-Hall (1996)
19. Manber, U.: Introduction to Algorithms: A Creative Approach. Addison-Wesley (1989)
20. Weiss, M.A.: Estructuras de datos y análisis de algoritmos. Addison-Wesley Iberoamericana (1995)
21. Galve Francés, J., González Moreno, J.C. Sánchez Calle, Á., Velázquez Iturbide, J.Á.: Algorítmica: diseño y análisis de algoritmos funcionales e imperativos. Ra-Ma (1993)
22. Gonnet, G.H., Baeza-Yates, R.: Handbook of Algorithms and Data Structures in Pascal and C. 2ª ed. Addison-Wesley (1991)
23. Baase, S., Van Gelder, A.: Computer Algorithms: Introduction to Design and Analysis. Addison Wesley (1988)
24. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. 2ª ed. The MIT Press (2001)
25. Goodrich, M.T., Tamassia, R.: Data Structures and Algorithms in Java. 2ª ed. John Wiley & Sons (2001)
26. Horowitz, E., Sahni, S.: Fundamentals of Computer Algorithms. Pitman (1978)
27. Martí, N., Ortega, Y., Verdejo, J.A.: Estructuras de datos y métodos algorítmicos: ejercicios resueltos. Prentice Hall (2004)
28. Sahni, S.: Data Structures, Algorithms and Applications in Java. McGraw-Hill (2000)
29. Nikander, J., Korhonen, A., Valanto, E., Virrantaus, K.: Visualization of spatial data structures on different levels of abstraction. *Proc. 4th Program Visualization Workshop (comunicación aceptada)*
30. Bird, R.S.: Tabulation techniques for recursive programs. *ACM Computing Surveys* 12, 4 (1980) 403-417
31. Pettorossi, A.: A powerful strategy for deriving efficient programs by transformation. *Proc. ACM Symposium on Lisp and Functional Programming*. ACM Press (1984) 273-281