Técnicas de diseño de algoritmos

GUÍA DOCENTE

PROFESORES

- Luis Javier Rodríguez Fuentes

  Profesor Agregado
  Departamento de Electricidad y Electrónica
  Facultad de Ciencia y Tecnología ZTF/FCT, UPV/EHU
  Barrio Sarriena s/n 48940 Leioa
  e-mail: luisjavier.rodriguez@ehu.eus

- Amparo Varona Fernández

  Profesora Titular de Universidad
  Departamento de Electricidad y Electrónica
  Facultad de Ciencia y Tecnología ZTF/FCT, UPV/EHU
  Barrio Sarriena s/n 48940 Leioa
  e-mail: amparo.varona@ehu.eus

OBJETIVOS

El diseño de algoritmos constituye uno de los cimientos de las ciencias de la computación, y su análisis y optimización resulta fundamental para una mayor eficiencia y precisión en tareas de cálculo, estimación, modelado, etc. Los algoritmos son parte esencial de las herrramientas informáticas necesarias para el procesamiento de la información en múltiples disciplinas científicas e incluso humanísticas. El material que se presenta en este curso trata de proporcionar una amplia información, tanto teórica como práctica, de diferentes técnicas para diseñar algoritmos, junto con las herramientas que permiten medir su eficiencia. El objetivo del curso no es aportar soluciones conocidas a problemas concretos, sino presentar y analizar las técnicas algorítmicas básicas que permitan abordar el desarrollo de programas correctos y eficientes para resolver problemas no triviales.

En el curso se abordan las principales técnicas de diseño de algoritmos (divide y vencerás, algoritmos voraces, algoritmos de búsqueda exhaustiva y programación dinámica), exponiendo sus características generales, sus ventajas e inconvenientes, evaluándolas desde el punto de vista de su complejidad computacional y aportando numerosos ejemplos que ponen de manifiesto cuándo y cómo aplicar cada técnica. El criterio de bondad de un algoritmo se basa en su eficiencia, expresando ésta como una medida del consumo de recursos (tiempo y memoria), por lo que el estudio de la complejidad computacional (asintótica, esto es, para problemas de tamaño muy grande) es una parte importante del curso. En este sentido, se aprenderán una serie de técnicas que permitirán evaluar las distintas soluciones a un problema y elegir aquélla que haga un mejor uso de los recursos disponibles en cada caso.

PRERREQUISITOS Y RECOMENDACIONES

El curso está pensado para estudiantes o titulados de un grado científico-técnico (Física, Matemáticas, Ingeniería, etc.) con unos conocimientos básicos de programación, álgebra y análisis.

- Se presentan diferentes técnicas y estructuras de datos avanzadas para la resolución de problemas, por lo que se debe conocer y tener práctica en programación estructurada y en el uso de métodos recursivos. Los ejemplos se desarrollan usando el lenguaje de programación Python, por lo que también es muy recomendable tener experiencia previa en el manejo del mismo.
- En lo que respecta a la parte matemática del curso, se debe estar familiarizado con los conceptos de límite, combinatoria y resolución de sistemas de ecuaciones.

Aunque no foman parte del núcleo del curso, en caso de disponer de tiempo o en una versión extendida del curso, se introducirían conceptos avanzados de la teoría de complejidad computacional y algoritmos aproximados, y se estudiarían casos de interés desde el punto de vista científico-técnico: string matching, generación de números pseudo-aleatorios, Transformada Rápida de Fourier (FFT), sistemas criptográficos, etc.

COMPETENCIAS

Competencias genéricas

- Introducir el análisis de la complejidad de algoritmos.
- Presentar y estudiar los esquemas de diseño de los algoritmos más comunes.
- Desarrollar habilidades en la aplicación práctica de las técnicas de análisis y diseño de algoritmos.
- Analizar la eficiencia de diversas técnicas para resolver una serie de problemas de referencia.
- Diseñar y analizar nuevos algoritmos.

Competencias específicas

- Reconocer los esquemas algorítmicos básicos: divide y vencerás, algoritmos voraces, búsqueda exhaustiva y programación dinámica.
- Conocer algoritmos clásicos de resolución de ciertos problemas fundamentales e identificar las situaciones en las que cabe utilizar dichos algoritmos.
- Ser capaz de particularizar los esquemas algorítmicos generales para resolver nuevos problemas.
- Razonar los fundamentos teóricos de los esquemas algorítmicos.
- Ser capaz de analizar la complejidad computacional de un algoritmo: costes, casos, notación asintótica, etc.
- Disponer de criterios que, durante la etapa de especificación, diseño e implementación, permitan escoger la alternativa más adecuada y argumentar dicha elección de forma razonada.

TEMARIO

1. Introducción
    - Ideas generales sobre diseño de algoritmos
    - Análisis de la eficiencia computacional
    - Algoritmos simples de búsqueda y ordenación
2. Divide y vencerás
    - Recursividad: ideas generales, ejemplos y análisis
    - Divide y Vencerás: esquema de diseño y complejidad temporal
    - Algoritmos de búsqueda y ordenación de tipo *Divide y vencerás*
3. Algoritmos voraces
    - Esquema de diseño
    - Ejemplos de algoritmos voraces
    - Algoritmos voraces sobre grafos

4. Búsqueda exhaustiva o de fuerza bruta
    - Vuelta atrás (*Backtracking*): esquema de diseño
    - Vuelta atrás: ejemplos
    - Juegos: búsqueda minimax y poda alfa-beta
    - Ramificación y poda
5. Programación dinámica
    - Ideas generales: precondicionamiento y principio de optimalidad
    - Esquema de diseño
    - Ejemplos de algoritmos de programación dinámica
    - Algoritmos sobre grafos: caminos de coste mínimo (Floyd)

DESCRIPCIÓN DE LOS TEMAS

- En el Tema 1 se presentan las ideas generales sobre diseño de algoritmos. Se introducen conceptos básicos del análisis de algoritmos, tamaño de los problemas, notación asintótica, órdenes de eficiencia, etc. Se hace un análisis detallado sobre ejemplos de algoritmos simples de búsqueda y ordenación.
- En el Tema 2 se presenta la técnica de diseño de algoritmos conocida como *Divide y Vencerás*. En primer lugar se introduce el concepto de recursividad junto con el análisis de complejidad temporal asociada. A continuación se describe el esquema general de la técnica y se presentan ejemplos de algoritmos de búsqueda (dicotómica) y ordenación (*quicksort*) mucho más eficientes que los estudiados en el Tema 1.
- En el Tema 3 se presentan los llamados *Algoritmos voraces*. En primer lugar, se exponen los principios de diseño. Se continúa con ejemplos prácticos sobre el problema del cambio de monedas y el problema de la mochila (continuo). En este tema se introduce el tipo de datos *Grafo* y se presentan varios algoritmos voraces sobre grafos: árboles de recubrimiento de coste mínimo, el problema del viajante (ciclos hamiltonianos) y la búsqueda de caminos de coste mínimo (algoritmo de Dijkstra).
- En el Tema 4 se presenta la *Búsqueda exhaustiva o de fuerza bruta*. En primer lugar, se introducen los principios de diseño de la *Vuelta atrás (Backtracking)* que opera de manera recursiva, explorando en profundidad un árbol de búsqueda. Esta técnica se aplica sobre varios ejemplos: el problema de las n reinas, suma de conjuntos de enteros, el problema de la mochila (discreto), el problema del viajante. A continuación, se introduce el concepto de *juego* con la *búsqueda minimax*. Finalmente, se profundiza en la *ramificación y poda* que generaliza una búsqueda en anchura sobre el árbol de búsqueda y se aplica sobre el problema de la mochila (discreto) y sobre el problema de la asignación de tareas.
- En el Tema 5 se presenta la técnica conocida como *Programación dinámica*. Se introducen los principios de diseño y se aplica sobre varios problemas prácticos: el problema de la mochila (discreto), devolver el cambio con el número mínimo de monedas, parentizado óptimo en la multiplicación de matrices y el problema de obtención de los caminos de coste mínimo en un grafo ponderado.

METODOLOGÍA DOCENTE

Para conseguir el máximo aprovechamiento del material del curso, se propone seguir el siguiente esquema:

- Estudiar en profundidad el material teórico proporcionado para cada tema (que incluye ejemplos prácticos significativos, incluido el código Python de las soluciones).
- Resolver y estudiar los ejercicios del Bloque 1, en los que se deberá aplicar la técnica de diseño presentada en cada tema. Junto a los enunciados, se suministran una o más soluciones de cada ejercicio.
- Resolver los ejercicios del Bloque 2, para los que se dan unas pautas de resolución (pero no las soluciones).

- Por último, como recurso de autoevaluación, en cada tema se plantea un examen con varios ejercicios, que van acompañados de las soluciones. Resolver los ejercicios del examen y comprobar su grado de corrección permitirá a los estudiantes verificar la comprensión de los conceptos y el grado de asimilación de las técnicas de diseño estudiadas en cada tema.

CRONOGRAMA

El tiempo recomendable para el estudio de los contenidos de la asignatura es de 15 semanas (un cuatrimestre), a razón de un mínimo de 4 horas/semana, distribuidas de la siguiente manera:

- Semanas 1-2 (8 horas) Tema 1: Introducción
- Semanas 2-4 (14 horas) Tema 2: Divide y vencerás
- Semanas 5-8 (14 horas) Tema 3: Algoritmos voraces
- Semanas 8-12 (14 horas) Tema 4: Búsqueda exhaustiva o de fuerza bruta
- Semanas 12-15 (14 horas) Tema 5: Programación dinámica

BIBLIOGRAFÍA

- G. Brassard, P. Bratley. ”Fundamentals of Algorithmics”. Prentice-Hall, 1995.
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. ”Introduction to Algorithms” (3rd Edition). MIT Press, 2009.
- Steven Skiena. ”The Algorithm Design Manual” (2nd Edition). Springer, 2008.
- Vernon L. Ceder. ”The Quick Python Book (2nd Edition, covers Python 3). Manning Publications, 2010.
- David M. Beazley. ”Python. Essential Reference (4th Edition). Addison-Wesley, 2009.

Recursos de AUTO-APRENDIZAJE en INTERNET (en inglés)

- http://www.algorist.com/
- http://www.cs.princeton.edu/ wayne/kleinberg-tardos/
- http://ww3.algorithmdesign.net/
- http://compgeom.cs.uiuc.edu/ jeffe//teaching/algorithms/
- Páginas del **MIT OpenCourseWare**:

  [URL raíz] = http://ocw.mit.edu/courses/electrical-engineering-and-computer-science

  - **[URL raíz]** /6-00sc-introduction-to-computer-science-and-programming-spring-2011/index.htm
  - **[URL raíz]** /6-046j-introduction-to-algorithms-sma-5503-fall-2005/index.htm
  - **[URL raíz]** /6-006-introduction-to-algorithms-fall-2011
  - **[URL raíz]** /6-006-introduction-to-algorithms-spring-2008 /index.htm
  - **[URL raíz]** /6-046j-design-and-analysis-of-algorithms-spring-2012/index.htm
  - **[URL raíz]** /6-854j-advanced-algorithms-fall-2005/index.htm
  - **[URL raíz]** /6-854j-advanced-algorithms-fall-2008