

Análisis de Algoritmos

Ingeniería Informática, EPS-UAM

Información general

Organización del curso:

- 13-15 (mínimo-máximo) semanas docentes:
- 30-33 clases teóricas.
- 9-12 clases de problemas
- 26-30 clases prácticas

Objetivos

Conocimientos teóricos

El curso está orientado a estudiantes que quieran profundizar en algoritmos y técnicas básicas de resolución de problemas computacionales. En concreto, y sobre una selección de problemas de interés en computación teórica y práctica, se abordarán:

1. Conceptos básicos sobre el problema en cuestión y cuestiones afines.
2. Formulación de los correspondientes algoritmos y su pseudocódigo.
3. Análisis de la corrección de los algoritmos.
4. Análisis y discusión de las estructuras de datos más adecuadas.
5. Desarrollo manual de los algoritmos sobre ejemplos pequeños.
6. Programación en lenguaje C de los algoritmos.
7. Análisis del rendimiento en el caso peor.

Conocimientos prácticos

Alcance de un nivel alto de programación en C, a través de la programación de los algoritmos tratados en clase y sus estructuras de datos y la organización del código en bibliotecas.

Recomendaciones

Conocimientos previos

Para un buen aprovechamiento del curso, es sumamente recomendable haber aprobado MTP II y EDI II. El curso puede seguirse de no ser así, pero requerirá un esfuerzo extra considerable, lo que hará más difícil no sólo la superación del mismo, sino también la de otras asignaturas en las que el alumno esté matriculado.

Asistencia a las clases

Se considera imprescindible la asistencia continua a las clases de teoría y problemas, y a las clases de prácticas. Durante el curso se fomentará la asistencia a las clases de teoría y problemas y se considerará cómo incorporar dicha asistencia y el seguimiento continuado de la asignatura a la calificación final.

La entrega de las prácticas propuestas es obligatoria. A su vez, se controlará la asistencia a las clases prácticas, procurándose que la misma influya de alguna manera en la calificación final.

Esfuerzo del estudiante

Independientemente del trabajo de preparación de exámenes, se recomienda:

- Un repaso individual de 45-60 minutos por cada clase teórica.
- La resolución de unos 6 problemas por semana lectiva.

Programa resumido

Parte I: Algoritmos en Grafos

1. Revisión de Algoritmos elementales en grafos
2. Algoritmos Codiciosos: Árboles abarcadores mínimos. Problema del Viajante.
3. Algoritmos Recursivos: Búsqueda en Profundidad. Conectividad. Circuitos Eulerianos y Hamiltonianos. Aplicaciones.
4. Algoritmos de Programación Dinámica. Distancias mínimas.

Parte II: Técnicas generales de diseño de algoritmos

1. Programación Dinámica
2. Recursividad
3. Algoritmos Codiciosos

Bibliografía

- Weiss, Data structures and algorithm analysis in C, Benjamin Cummings.
- Cormen, Leiserson, Rivest, Introduction to algorithms, The MIT Press--Mc Graw Hill.
- Baase, Computer algorithms, Addison-Wesley.
- Aho, Hopcroft, Ullman, The design and analysis of computer algorithms, Addison-Wesley.
- Aho, Hopcroft, Ullman, Data Structures and Algorithms, Addison-Wesley.
- Kernighan, Ritchie, The C programming language, Prentice hall.
- Kernighan, Pike, The practice of programming, Addison Wesley.

Procedimiento de evaluación

Evaluación de contenidos teóricos:

- A. Examen intermedio: 3-4 cuestiones teóricas y prácticas a desarrollar en 1,5 horas.
- B. Examen escrito final: 6-8 cuestiones teóricas y prácticas a desarrollar en 3 horas.

Cálculo de la nota teórica: máximo entre

$$0,65*(Nota\ final/10) + 0,35*(Nota\ intermedia/10).$$

$$Nota\ teórica\ final /10.$$

Evaluación de contenidos prácticos:

Se propondrán 4 prácticas de entrega obligatoria. La nota final se obtendrá de su evaluación.

Nota final:

$$Nota\ teórica\ final \times 0.70 + Nota\ de\ prácticas \times 0.30$$

Importante: la Teoría y las Prácticas SE APROBARÁN POR SEPARADO.

Programa detallado

Parte I: Algoritmos en Grafos

1. Revisión de Algoritmos elementales en grafos

1.1. Revisión de conceptos básicos

Nodos, ramas, grafos

Representación de grafos: matriz de adyacencia, listas de adyacencia

Tamaño de grafos, coste de algoritmos

1.2. Problemas de distancia mínima

Grafos ponderados

Coste de caminos

Tipos de problemas de distancia mínima

Proceso por Colas y distancia mínima en grafos no ponderados

1.3. Revisión del algoritmo de Dijkstra

Colas de prioridad

Algoritmo de Dijkstra

Caso peor del algoritmo de Dijkstra

Corrección del algoritmo de Dijkstra

2. Árboles abarcadores mínimos

2.1. Revisión de árboles abarcadores mínimos

Concepto

Algoritmo de Prim

2.2. Algoritmo de Kruskal

Introducción, ejemplos

Pseudocódigo y operaciones básicas

2.3. TAD Conjunto Disjunto

Concepto, primitivas: Unión, Find

Primitivas, Unión por alturas

Tablas como EdDs para bosques

- Unión por rangos y Find con compresión de caminos
- Rendimiento de primitivas y de Kruskal sobre bosques
- 2.4. Corrección de Prim y Kruskal
 - Particiones, cruces y cruces mínimos
 - Psc generalizado de determinación de árboles abarcadores mínimos
 - Corrección de Prim y Kruskal
- 3. Grafos de tareas
- 4. Aplicaciones de Búsqueda en Profundidad
 - 4.1. Introducción
 - Conceptos básicos
 - Problemas de conexión
 - Revisión de búsquedas en amplitud y en profundidad
 - Conectividad fuerte en grafos dirigidos (opcional)
 - 4.2. Grafos biconexos
 - Puntos de articulación y grafos biconexos
 - Orden y ascenso de vértices
 - Algoritmo de determinación de puntos de articulación
 - 4.3. Circuitos eulerianos
 - Los puentes de Königsberg (y el dibujo de casitas!!)
 - Existencia de circuitos y caminos eulerianos
 - Algoritmo de determinación de circuitos eulerianos
 - 4.4. Circuitos hamiltonianos y el Problema del Viajante
 - Circuitos hamiltonianos
 - Dificultad, problemas NP y NP-completos
 - El Problema del Viajante (TSP)
 - Solución del TSP y obtención de circuitos hamiltonianos
 - Soluciones aproximadas del TSP
- 5. Flujos en grafos

- 5.1. Introducción
 - Capacidades y redes de flujo
 - Flujos, valor de un flujo
 - Capacidades residuales y redes residuales
- 5.2. Algoritmo de Ford-Fulkerson
 - Caminos aumentadores
 - Aumento de flujos
 - Algoritmo de Ford-Fulkerson
- 5.3. Cortes mínimos y flujos máximos
 - Capacidades y flujos sobre cortes
 - Propiedades de cortes, cortes mínimos
 - Teorema del flujo máximo y del corte mínimo
- 5.4. Algoritmo de Edmonds-Karp
 - Rendimiento general del algoritmo de Ford-Fulkerson
 - Implementaciones del algoritmo de Ford-Fulkerson
 - Algoritmo de Edmonds-Karp
 - Eficacia del algoritmo de Edmonds-Karp

Parte II: Técnicas generales de diseño de algoritmos

1. Programación dinámica

El problema de la suma y la criptografía de clave pública

Criptografía de clave pública

Problemas NP-completos y criptografía de clave pública

El problema de la suma

Claves públicas y el problema de la suma: algoritmo de Merkle-Hellman

Ordenación óptima en el producto de matrices

Planteamiento, ejemplos

Subestructura óptima y solución recursiva

Subestructura óptima y solución de programación dinámica (PD)

Algoritmo PD de determinación del orden óptimo

Problemas resolubles por programación dinámica

Existencia de subestructura óptima

Solapamiento de subproblemas

Solución bottom-up

Árboles binarios de búsqueda óptimos

Planteamiento, ejemplos, solución codiciosa

Subestructura óptima del coste medio de búsquedas

Algoritmo PD de determinación del árbol óptimo

Distancias mínimas sobre todos los vértices en grafos densos (opcional)

Planteamiento, solución por reiteración del algoritmo de Dijkstra

Subestructura óptima de las distancias mínimas entre vértices

Algoritmo PD de cálculo de distancias mínimas

Comparación de rendimientos

Búsquedas aproximadas en cadenas (opcional)

Distancia entre cadenas

Subestructura óptima de las distancias entre cadenas

Algoritmo PD de cálculo de distancias entre cadenas

Búsqueda aproximada de cadenas

2. Recursividad

El problema de Selección

El problema general de Selección, el problema de la mediana

Análisis de soluciones elementales

Algoritmo QuickSelect I: caso medio lineal

Algoritmo QuickSelect II: caso peor lineal

Problemas resolubles mediante recursividad

Resolución recursiva de problemas

Rendimiento general de soluciones recursivas

Multiplicación de matrices

Análisis del algoritmo estándar, caso mejor

Análisis de solución recursiva estándar

Algoritmo de Strassen

La Transformada Rápida de Fourier

Introducción: multiplicación de polinomios

Multiplicación desde valores

Coste de algoritmo elemental

Raíces complejas de la unidad, fórmula de Euler, transformada de Fourier discreta

Transformada de Fourier rápida (FFT)

Rendimiento de la FFT

Multiplicación de números, algoritmo de Schönhage-Strassen

Distancias mínimas entre conjuntos (opcional)

Planteamiento del problema

Solución recursiva

Análisis de la solución recursiva

3. Algoritmos codiciosos

Secuenciación lineal en colas de trabajos

Introducción y planteamiento

Secuenciación codiciosa de trabajos

Optimalidad de la secuenciación codiciosa

Problemas resolubles mediante algoritmos codiciosos

Problemas de optimización

Subestructura óptima implícita

Optimización local y soluciones codiciosas

Codificación Huffman

Compresión de archivos y códigos de bits

Códigos prefijo

Algoritmo de Huffman

Corrección del algoritmo de Huffman

Algoritmos codiciosos y de programación dinámica

Problemas de la mochila 0-1 y fraccionario

Solución codiciosa del problema fraccionario de la mochila

Solución codiciosa errónea del problema 0-1 de la mochila

Solución PD del problema 0-1 de la mochila

NP-completitud del problema de la mochila

Bibliografía

- Weiss, Data structures and algorithm analysis in C, Benjamin Cummings.
- Cormen, Leiserson, Rivest, Introduction to algorithms, The MIT Press--Mc Graw Hill.
- Baase, Computer algorithms, Addison-Wesley.
- Aho, Hopcroft, Ullman, The design and analysis of computer algorithms, Addison-Wesley.
- Aho, Hopcroft, Ullman, Data Structures and Algorithms, Addison-Wesley.
- Kernighan, Ritchie, The C programming language, Prentice hall.
- Kernighan, Pike, The practice of programming, Addison Wesley.