Universidad de Navarra

Estructura de Datos y Algoritmia

Guía docente 2026-27

PRESENTACIÓN

Breve descripción:

- **Titulación**: Grado en Ingeniería en Inteligencia Artificial
- **Módulo/Materia**: Fundamentos de Computación / Algoritmia y Optimización
- **ECTS**: 4
- **Curso, semestre**: 2º, 2
- **Carácter**: OB
- **Profesorado**:
  - Rosquete De Mora, Daniel / Profesor externo
  - Ruete Serrano, Pilar / Personal de apoyo a la docencia
- **Idioma**: Castellano
- **Horas semanales**: 3 horas aprox.
- **Duración**: 12 semanas (36 horas totales)
- **Lenguaje de programación**: C++
- **Prerrequisitos**: Fundamentos de Programación, programación orientada a objetos, programación básica en C++

La asignatura de Estructura de Datos y Algoritmos es una piedra angular en un campo donde la eficiencia y la capacidad de procesar grandes volúmenes de datos son vitales, comprender cómo organizar los datos y diseñar algoritmos eficientes es fundamental.

Este curso se centra en el estudio de las estructuras de datos fundamentales y las técnicas algorítmicas esenciales, con un énfasis particular en el análisis de su eficiencia. Abordaremos la notación de la complejidad asintótica (Notación O Grande), el diseño y análisis de algoritmos recursivos, y procederemos al análisis exhaustivo de diversas estructuras de datos como herramientas intrínsecas para la resolución eficaz de problemas computacionales. Se dotará al estudiante de las habilidades necesarias para elegir la estructura de datos y el algoritmo más adecuados para una tarea específica, valorando siempre el coste computacional (tiempo y espacio) de la solución propuesta. La asignatura combina la teoría con la práctica intensiva, a través de la implementación de los conceptos estudiados. Se garantizará el uso de tecnologías de vanguardia durante la práctica de esta asignatura.

RESULTADOS DE APRENDIZAJE (Competencias)

R18 - Valorar la eficiencia de las soluciones algorítimicas propuestas para la resolución de problemas de programación.

R34 - Utilizar las estructuras de datos como herramienta necesaria para el diseño de algoritmos eficientes a nivel fundamental

Al completar esta asignatura, el estudiante será capaz de:

1. **Analizar la eficiencia:** Comprender y aplicar las herramientas de análisis de la complejidad algorítmica, en particular la Notación O Grande, para determinar el coste temporal y espacial de un algoritmo.
2. **Comparar algoritmos:** Evaluar y comparar diferentes algoritmos o implementaciones de una misma funcionalidad en función de su eficiencia.
3. **Diseñar algoritmos recursivos:** Entender los principios de la recursividad y diseñar, implementar y analizar algoritmos recursivos.
4. **Seleccionar estructuras de datos:** Identificar las propiedades y el comportamiento de las estructuras de datos fundamentales (listas, pilas, colas, tablas hash, árboles,

grafos y tensores) y seleccionar la más adecuada para modelar y resolver un problema dado de manera eficiente.
5. **Implementar estructuras y algoritmos:** Codificar las estructuras de datos y algoritmos estudiados en un lenguaje de programación adecuado, prestando atención a la eficiencia de la implementación.
6. **Aplicar técnicas algorítmicas:** Utilizar algoritmos clásicos (ordenación, búsqueda, recorrido de grafos, etc.) y adaptar sus principios a nuevos problemas.
7. **Resolver problemas algorítmicos:** Abordar problemas computacionales, descomponerlos, elegir las herramientas (estructuras y algoritmos) más eficientes y desarrollar una solución correcta y eficiente.

Valorar la eficiencia: Incorporar la eficiencia algorítmica como un criterio fundamental en el proceso de diseño y desarrollo de software.

PROGRAMA

El programa de la asignatura se estructura en tres módulos principales, diseñados para proporcionar una comprensión sólida de las estructuras de datos y algoritmos, con un enfoque práctico en C++ y una constante atención a la eficiencia computacional. Las horas indicadas para cada módulo son aproximadas y pueden ajustarse en función del ritmo del curso y las necesidades del grupo.

Módulo 1: Fundamentos y Optimización en C++ (Aprox. 10-12 horas)

Este módulo sienta las bases para el análisis y la implementación eficiente de algoritmos utilizando C++. Se revisan aspectos del lenguaje relevantes para el rendimiento y se introducen las herramientas formales para el análisis de la complejidad.

Introducción

- El papel de las Estructuras de Datos y Algoritmos en la computación y la IA.
- Importancia de la eficiencia (tiempo y espacio).
- Por qué C++ para el estudio de la eficiencia fundamental.

Aspectos de C++ para la eficiencia

- Gestión de memoria: Stack vs Heap. Punteros y referencias.
- Uso eficiente de const.
- Introducción a la gestión dinámica de memoria (new, delete, std::unique_ptr, std::shared_ptr).
- Principios básicos de profiling y medición de rendimiento en C++.

Recursividad y ecuaciones de recurrencia

- Principios y diseño de algoritmos recursivos.
- Implementación de recursividad en C++.
- Backtracking

Análisis de la complejidad algorítmica

- Medición del tiempo de ejecución y uso de memoria.
- Análisis asintótico: tasa de crecimiento.
- Notación O Grande (O).
- Análisis de bucles y estructuras de control.
- Casos peor, mejor y promedio.

- Análisis de la complejidad de algoritmos recursivos mediante ecuaciones de recurrencia (Método de sustitución, Árbol de recursión).

Módulo 2: Estructuras de datos fundamentales (Aprox. 12-14 horas)

Se exploran las estructuras de datos esenciales, analizando sus implementaciones típicas en C++ y el coste de sus operaciones para elegir la más adecuada en la resolución de problemas.

Conceptos de estructuras de datos

- Tipos de Datos Abstractos (TDA) vs. Implementaciones concretas.

Estructuras lineales

- Arrays y std::vector: Implementación, operaciones, análisis de coste. memoria contigua.
- Listas enlazadas (Simples, Dobles): Implementación, operaciones, análisis de coste. Comparativa con arrays.
- Pilas (std::stack): TDA, implementaciones (vector, lista), operaciones, análisis.
- Colas (std::queue, std::deque): TDA, implementaciones, operaciones, análisis. Colas de doble extremo (std::deque).

Tablas Hash (std::unordered_map, std::unordered_set)

- Funciones Hash y dispersión.
- Colisiones y estrategias de resolución (Encadenamiento, Direccionamiento Abierto).
- Factor de carga y análisis de coste de operaciones (promedio y peor caso).

Árboles

- Conceptos generales de árboles.
- Árboles Binarios. Recorridos (Preorden, Inorden, Postorden, por niveles).
- Árboles Binarios de Búsqueda (BST): Operaciones de búsqueda, inserción y borrado. Análisis de coste.
- La necesidad de árboles equilibrados (mención de AVL/Rojo-Negro para garantizar el peor caso logarítmico).Añadir en el módulo 3, dentro de la sección de Grafos:

Árbol de expansión mínima

- Algoritmo de Kruskal: Unión de componentes conexas mediante aristas de menor peso. Análisis de coste.
- Algoritmo de Prim: Construcción del árbol añadiendo vértices adyacentes de menor peso. Análisis de coste y uso de colas de prioridad.

Montículos (Heaps y std::priority_queue)

- Propiedades del Montículo Binario.
- Operaciones (inserción, extracción del máximo/mínimo).
- Análisis de coste. Aplicaciones (colas de prioridad, Heap Sort).

Universidad de Navarra

Módulo 3: Algoritmos y Estructuras Avanzadas (Aprox. 10-12 horas)

Este módulo aborda algoritmos clásicos y relevantes, así como estructuras de datos más complejas, incluyendo la fundamental estructura del Tensor en el contexto de IA.

Algoritmos de ordenación

- Análisis comparativo de algoritmos eficientes: Merge Sort (Divide y Vencerás, análisis recurrente).
- Quick Sort (Diseño, elección del pivote, análisis promedio y peor caso).
- Algoritmos de Ordenación no basados en comparación (Radix Sort): Aplicabilidad y análisis.
- Uso eficiente de std::sort.

Algoritmos de búsqueda

- Búsqueda Lineal y Búsqueda Binaria. Análisis de coste en diferentes estructuras.

Grafos

- Conceptos fundamentales de la teoría de grafos.
- Representaciones (Matriz de Adyacencia vs. Lista de Adyacencia). Análisis de espacio y tiempo para operaciones básicas.
- Algoritmos de Recorrido: Búsqueda en Amplitud (BFS) y Búsqueda en Profundidad (DFS). Implementación y análisis de coste.
- Introducción a algoritmos sobre grafos (ej: concepto de Dijkstra para camino más corto si el tiempo lo permite, mostrando el uso de colas de prioridad).

El tensor como estructura de datos

- Definición: Generalización de arrays multidimensionales (escalar, vector, matriz, tensor).
- Representación de datos en tensores (imágenes, series temporales, datos en batches).
- Conexión con estructuras de datos (implementación subyacente basada en arrays/vectores).
- Operaciones básicas sobre tensores (acceso, broadcasting, reducción, operaciones elemento a elemento).
- Mención de librerías de tensores en C++ (Eigen, APIs de TF/PyTorch) como herramientas para la computación eficiente con esta estructura.

Introducción a paradigmas algorítmicos (conceptos):

- Mención breve de la idea detrás de la Programación Dinámica y Algoritmos Voraces (Greedy) como técnicas avanzadas para la resolución de problemas optimizados. (Énfasis en la idea, no implementación exhaustiva dada la limitación de tiempo).
- Buenas prácticas en programación

ACTIVIDADES FORMATIVAS

La metodología docente combinará diferentes tipos de actividades para facilitar la adquisición de conocimientos y competencias:

Universidad de Navarra

- **Clases teóricas:** Presentación de los conceptos fundamentales, definiciones, propiedades de las estructuras de datos y algoritmos, y técnicas de análisis de complejidad.
- **Clases de laboratorio (Prácticas):** Implementación de las estructuras de datos y algoritmos estudiados en C++. Desarrollo de programas que resuelvan problemas utilizando las técnicas algorítmicas aprendidas, haciendo hincapié en la eficiencia de la implementación.
- **Trabajo autónomo del estudiante:** Estudio individual de los contenidos teóricos, resolución de ejercicios propuestos, preparación de las sesiones de laboratorio y desarrollo de las tareas o proyectos.
- **Tutorías:** Espacio para resolver dudas, discutir conceptos y obtener seguimiento individualizado del aprendizaje.

EVALUACIÓN

CONVOCATORIA ORDINARIA

Distribución de Calificaciones:

- Evaluación Teórica (50%)
  - Examen final en cada módulo: 10%, 20% y 20%
- Evaluaciones de trabajos (40%)
  - Proyecto de Implementación 1 (Estructuras de datos fundamentales): 20%
  - Proyecto de Implementación 2 (Algoritmos y Estructuras Avanzadas): 20%

El examen final será el **25 de mayo**. 10%

CONVOCATORIA EXTRAORDINARIA

- El examen de la convocatoria extraordinaria será el **19 de junio**.

HORARIOS DE ATENCIÓN

- **Prof. Daniel Rosquete**

Jueves: 20:30 a 21:30

Online

- **Prof. Pilar Ruete**

Lunes: 10:00 a 11:30

Despacho: 109 Edificio: Urdaneta Planta: Primera

BIBLIOGRAFÍA Y RECURSOS

- Introduction to Algorithms -Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest y Clifford Stein Localízalo en la Biblioteca (online y papel)
- Un recorrido por C++ - Bjarne Stroustrup Localízalo en la Biblioteca
- C++20 - The Complete Guide: First Edition - Nicolai M. Josuttis Localízalo en la Biblioteca