UNIVERSIDAD NACIONAL DE INGENIERÍA

SÍLABO

INFORMACIÓN GENERAL

ASIGNATURA	:ANÁLISIS Y DISEÑO DE ALGORITMOS
DOCENTE	:GERSON GARRIDO
HORAS POR SEMANA	:06 (TEORÍA: 04, LABORATORIO: 02)

SUMILLA

El curso es de naturaleza teórico-práctica y enseña a diseñar algoritmos para problemas nuevos, a demostrar que son correctos y a reconocer los problemas para los que quizá no exista un algoritmo rápido. El estudiante modela problemas escritos en palabras, elige una técnica, la justifica y la implementa. Se organiza en cuatro partes: **demostrar y dividir** (correctitud, recurrencias y divide y vencerás); **greedy y grafos** (argumentos de intercambio, árboles de expansión mínima y DFS); **programación dinámica** (subproblemas, secuencias, árboles y caminos mínimos); y **flujo e intratabilidad** (flujo máximo, reducciones, NP-completitud y qué hacer con un problema difícil). El laboratorio se desarrolla en C++, con un juez automático, en un entorno común.

COMPETENCIAS

- Demuestra que un algoritmo es correcto, con invariantes, inducción y argumentos de intercambio.
- Plantea y resuelve la recurrencia del costo de un algoritmo recursivo.
- Diseña algoritmos con divide y vencerás, greedy y programación dinámica, y justifica por qué la técnica elegida sirve.
- Modela un problema escrito en palabras como un problema de grafos o de flujo.
- Aplica y demuestra los algoritmos clásicos de árboles de expansión mínima, caminos mínimos, componentes fuertemente conexas y flujo máximo.
- Implementa en C++ una solución que cumple los límites de tiempo, usando la biblioteca estándar.
- Demuestra que un problema es NP-completo con una reducción, y explica qué significa eso en la práctica.
- Elige una estrategia para un problema difícil: un caso especial, búsqueda con poda, aproximación o un solucionador.

UNIDADES DE APRENDIZAJE

Unidad	Contenido	Semanas
I	Demostrar y dividir	1–3
II	Greedy y grafos	4–6
III	Programación dinámica	7–10
IV	Flujo e intratabilidad	11–16

PROGRAMACIÓN SEMANAL

Cada semana de clases comprende dos sesiones de teoría de 2 horas cada una y una sesión de laboratorio de 2 horas. Cada capítulo de los apuntes abre con una ruta de lectura: qué leer antes de clase (15 a 20 minutos), qué se explica en clase y qué repasar después. Los temas marcados con (*) pueden pasar a lectura autónoma si se pierden clases por feriados.

Sem.	Teoría	Laboratorio
1	Presentación. Demostrar un algoritmo: invariantes e inducción. Matching estable: Gale-Shapley	Asignar postulantes a carreras
2	Recurrencias y teorema maestro. Mergesort demostrado. Contar inversiones	Comparar rankings
3	Selección en tiempo lineal. Karatsuba. Par más cercano	El par más cercano
4	Cota inferior para ordenar. Quicksort aleatorizado (*). Greedy: selección de intervalos y argumentos de intercambio	*Práctica 1*
5	Huffman demostrado. Minimizar el retraso máximo. Dijkstra demostrado	Rutas con a lo sumo k peajes
6	Árboles de expansión mínima: propiedad del corte, Kruskal y Prim. DFS: orden topológico y componentes fuertemente conexas	Diseñar una red. Los prerrequisitos de una malla
7	Programación dinámica: de la memoización a la tabla. Mochila. Distancia de edición y subsecuencia común más larga	*Práctica 2*
8	**Examen parcial**	
9	Programación dinámica sobre intervalos, DAG y árboles	Optimizar sobre una jerarquía
10	Bellman-Ford como programación dinámica. Floyd-Warshall. Máscaras de bits (*)	Arbitraje con tipos de cambio
11	Flujo máximo: Ford-Fulkerson. Teorema de flujo máximo y corte mínimo. Edmonds-Karp	*Práctica 3*
12	Matching bipartito. Caminos disjuntos. Modelar con flujo: asignación y selección de proyectos	Asignar estudiantes a secciones
13	P, NP y reducciones polinomiales. Conjunto independiente, cobertura de vértices y SAT	Resolver sudokus con un solucionador SAT
14	NP-completitud: Cook-Levin enunciado, reducciones desde 3-SAT, coloreo y suma de subconjuntos	*Práctica 4*
15	Qué hacer con un problema difícil: poda y aproximación. Programación lineal como idea (*). Repaso	El viajero: exacto frente a heurística
16	**Examen final**	

METODOLOGÍA

- **Teoría:** clases expositivas con algoritmos seguidos paso a paso y demostraciones construidas en clase. Antes de cada semana, el estudiante lee la parte del capítulo que su ruta de lectura marca como previa.
- **Laboratorio:** una sesión semanal en un entorno común para todos los estudiantes, donde resuelven en C++ un problema escrito en palabras y lo comprueban con un juez automático, que da la respuesta de inmediato.
- **Prácticas:** cuatro prácticas individuales, en las semanas 4, 7, 11 y 14, con una demostración escrita y un problema en C++. Ocupan la sesión de laboratorio de su semana, y el problema de esa semana queda como ejercicio opcional.
- Los términos técnicos se presentan en español cuando así se dicen (grafo, flujo, arista) y en inglés cuando la traducción suena forzada (greedy, backtracking).

NORMAS DEL CURSO

- **Trabajo:** individual. Se puede discutir ideas, pero cada estudiante escribe su propio código y sus demostraciones. Copiar soluciones de otras personas o de internet cuenta como plagio.

Herramientas permitidas

En este curso no se usa IA generativa en ningún trabajo: laboratorios, prácticas ni exámenes. Las demás herramientas se permiten en laboratorios y prácticas, dentro del entorno del curso.

Herramienta	¿Se permite?
Mensajes del compilador, `gdb`, valgrind y sanitizers	Sí
Documentación oficial (cppreference, manuales del entorno) y búsqueda en la web	Sí, para consultar el lenguaje y la biblioteca. Copiar una solución ajena es plagio
Autocompletado del editor que no genera soluciones enteras (nombres, firmas, paréntesis)	Sí
Correctores de ortografía, traductores y herramientas de accesibilidad	Sí
Formateadores de código y linters	Sí
IA generativa para pedir explicaciones, sin que escriba código	No
IA generativa que escribe código o demostraciones	No

Cómo se declara lo usado: al final de la entrega, una nota breve con la documentación y las páginas consultadas, salvo las del compilador y el editor. Cualquier entrega puede llevar una explicación oral: quien entrega debe poder explicar cualquier parte de su código y de su demostración.

BIBLIOGRAFÍA

- Apuntes del curso, un capítulo por semana, en el sitio del curso. No se requiere comprar ningún libro.
- Erickson, J. *Algorithms*, 1.ª edición, 2019. Licencia CC BY 4.0. `https://jeffe.cs.illinois.edu/teaching/algorithms/`
- Cormen, T., Leiserson, C., Rivest, R. y Stein, C. *Introduction to Algorithms*, 4.ª edición. MIT Press, 2022. Texto de referencia.

- Sitio del curso: guías semanales, apuntes, diapositivas, código y enunciados.