UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA FACULTAD DE INGENIERÍA PROGRAMA ANALÍTICO DE LA UNIDAD DE APRENDIZAJE: ANÁLISIS DE ALGORITMOS		
	DES:	**Ingeniería**
	Programa académico	**Doctorado en Ingeniería**
	Tipo de materia (Obli/Opta):	Optativa
	Clave de la materia:	DI24OP20
	Semestre:	1, 2, 3
	Área en plan de estudios (B, P y E):	G, E
	Total de horas por semana:	4
	Teoría: Presencial o Virtual	2
	Laboratorio o Taller:	0
	Prácticas:	2
	Trabajo extra-clase:	6
	Créditos Totales:	10
	Total de horas semestre (x 16 sem):	160
	Fecha de actualización:	Marzo 2024
	Prerrequisito (s):	Ninguno

DESCRIPCIÓN DEL CURSO

El curso aporta los principios fundamentales del análisis y diseño de algoritmos para comprender cómo se aplican en una amplia gama de problemas. Permite el estudio de diversos tipos de problemas y el diseño y/o aplicación de algoritmos para resolverlos de manera eficiente.

COMPETENCIAS A DESARROLLAR

GESTIÓN DE PROYECTOS

Coordina y administra de forma responsable, proyectos que atiendan criterios de sustentabilidad y que contribuyan a mejorar la calidad de vida.

GESTIÓN DEL CONOCIMIENTO

Demuestra conocimientos y habilidades para la búsqueda, análisis crítico, síntesis y procesamiento de información para su transformación en conocimiento, con actitud ética.

COMUNICACIÓN CIENTÍFICA

Difunde con responsabilidad ética y social el conocimiento científico, tecnológico, artístico y/o humanístico que produce de forma objetiva para aportar ideas y hallazgos científicos.

INVESTIGACIÓN

Desarrolla investigación original, tecnología y/o innovaciones en procesos, servicios o productos que contribuyan a la solución de problemas, mejoren la convivencia, generen oportunidades para el desarrollo sustentable y propicien una mejor calidad de vida.

DISEÑO Y GESTIÓN DE INFRAESTRUCTURAS SOSTENIBLES PARA EL DESARROLLO

El doctorando diseña y gestiona infraestructuras seguras, eficientes y sostenibles que promueven el desarrollo socioeconómico y ambiental, integrando conocimientos de áreas como infraestructura para el transporte, estructura y materiales, computación e hidrología. Este diseño y gestión considera la sostenibilidad en todos sus aspectos y se rige por altos estándares éticos y profesionales.

DOMINIOS	OBJETOS DE ESTUDIO	RESULTADOS DE APRENDIZAJE	METODOLOGÍA	EVIDENCIAS
Identifica áreas de oportunidad, actores y fuentes de financiamiento para proyectos de desarrollo sostenible. Identifica y articula necesidades de conocimiento para comprender la complejidad computacional.	**1. INTRODUCCIÓN AL ANÁLISIS DE ALGORITMOS** 1.1 Complejidad Computacional 1.2 Orden de crecimiento asintótico 1.3 Tiempos de ejecución más comunes 1.4 Problemas P y NP 1.5 Problemas NP-Complete y NP-Hard 1.6 Problemas PSPACE	Los estudiantes serán capaces de explicar los conceptos de complejidad computacional y orden de crecimiento asintótico. Los estudiantes podrán analizar y comparar los tiempos de ejecución de algoritmos comunes. Los estudiantes comprenderán la diferencia entre problemas P y NP, así como los problemas NP-Complete y NP-Hard. Los estudiantes serán capaces de aplicar el teorema maestro para resolver recurrencias y calcular la complejidad de algoritmos recursivos.	Clases magistrales para introducir los conceptos fundamentales de complejidad computacional y orden de crecimiento asintótico. Ejercicios prácticos para calcular la complejidad temporal y espacial de algoritmos simples.	Los estudiantes deberán resolver ejercicios teóricos y prácticos para calcular la complejidad temporal y espacial de algoritmos simples. Realizar análisis de algoritmos básicos para comprender los conceptos de orden de crecimiento asintótico, tiempos de ejecución y problemas P y NP.
Se comunica con propiedad sobre los conceptos de complejidad computacional y orden de crecimiento asintótico. Desarrolla el pensamiento científico y aplica procesos metodológicos en el análisis de algoritmos.	**2. ALGORITMOS DIVIDE Y VENCERÁS** 2.1 Algoritmo Merge-Sort 2.2 Resolviendo recursiones 2.2.1 Árboles de recursiones 2.2.2 Inducción matemática 2.2.3 El teorema maestro 2.3 Conteo de inversiones	Los estudiantes serán capaces de implementar el algoritmo Merge-Sort y comprender su complejidad. Los estudiantes podrán resolver recurrencias utilizando técnicas como árboles de recursiones e inducción matemática. Los estudiantes serán capaces de aplicar el algoritmo de la multiplicación y calcular	Enseñanza mediante ejemplos concretos de algoritmos divide y vencerás, como Merge-Sort y la multiplicación de matrices. Sesiones de resolución de problemas donde los estudiantes trabajan en grupos para diseñar algoritmos divide y vencerás para problemas específicos.	Los estudiantes deberán implementar algoritmos divide y vencerás, como Merge-Sort y algoritmos recursivos, y comparar su eficiencia en términos de complejidad temporal. Resolver problemas prácticos utilizando la técnica divide y vencerás, como encontrar el par de puntos más cercano.

	2.4 Encontrar el par de puntos más cercano 2.5 Algoritmo de la multiplicación 2.6 Convoluciones y la transformada rápida de Fourier	convoluciones utilizando la transformada rápida de Fourier. Los estudiantes podrán diseñar algoritmos eficientes para problemas como el conteo de inversiones y encontrar el par de puntos más cercano.		
Establece alianzas estratégicas para la gestión de proyectos de infraestructura sostenible. Accede y analiza información pertinente sobre teoría de grafos y algoritmos.	**3. TEORÍA DE GRAFOS** 3.1 Definiciones básicas 3.2 Conectividad en grafos 3.3 Grafos bipartitos 3.4 Conectividad en grafos dirigidos 3.5 Directed Acyclic Graphs (DAG) y Orden Topológico	Los estudiantes comprenderán las definiciones básicas de grafos y su aplicación en la resolución de problemas. Los estudiantes serán capaces de determinar la conectividad en grafos y grafos dirigidos. Los estudiantes podrán resolver problemas utilizando grafos bipartitos y Directed Acyclic Graphs (DAG) con el orden topológico. Los estudiantes serán capaces de aplicar algoritmos de grafos en la solución de problemas prácticos, como el emparejamiento bipartita y encontrar caminos disjuntos en grafos bipartitos.	Clases teóricas para introducir los conceptos fundamentales de grafos y sus aplicaciones. Resolución de problemas prácticos utilizando algoritmos de grafos en sesiones prácticas.	Los estudiantes deberán analizar grafos simples para identificar propiedades como conectividad, ciclos y componentes. Resolver problemas prácticos relacionados con grafos, como encontrar caminos más cortos o identificar ciclos en un grafo.
Administra los recursos del proyecto con criterios de sustentabilidad.	**4. ALGORITMOS GREEDY** 4.1 Calendarización de intervalos	Los estudiantes comprenderán el concepto de algoritmos greedy y su aplicación en la solución de	Discusión de estrategias greedy y su aplicación en problemas prácticos. Estudio de casos reales donde se	Implementar algoritmos greedy para resolver problemas de optimización, como el algoritmo de Dijkstra para

	4.2 Minimización de latencia 4.3 Árboles de expansión mínimos 4.4 El algoritmo de Dijkstra 4.5 El algoritmo de Kruskal	problemas de optimización. Los estudiantes serán capaces de aplicar algoritmos greedy en problemas como la calendarización de intervalos, minimización de latencia y construcción de árboles de expansión mínimos. Los estudiantes podrán implementar algoritmos de Dijkstra y Kruskal para resolver problemas de camino más corto y encontrar árboles de expansión mínimos, respectivamente.	utilicen algoritmos greedy para optimización.	encontrar el camino más corto en un grafo. Evaluar la calidad de las soluciones obtenidas mediante algoritmos greedy y compararlas con soluciones óptimas en términos de eficiencia y calidad.
Transforma la información en conocimiento aplicable a través de la programación dinámica y el análisis de flujo en redes.	**5. PROGRAMACIÓN DINÁMICA** 5.1 El paradigma de la programación dinámica (estructuras de memorización y traslape de sub-problemas). 5.2 Calendarización de intervalos ponderados. 5.3 Mínimos cuadrados segmentados. 5.4. El problema de la mochila (knapsack). 5.5 Alineamiento de secuencias.	Los estudiantes serán capaces de aplicar el paradigma de la programación dinámica para resolver problemas de optimización. Los estudiantes podrán resolver problemas como la calendarización de intervalos ponderados y el alineamiento de secuencias utilizando programación dinámica. Los estudiantes comprenderán el concepto de traslape de sub-problemas y estructuras de memorización en la programación dinámica.	Enseñanza basada en ejemplos para introducir el concepto de programación dinámica. Resolución de problemas utilizando programación dinámica en sesiones prácticas.	Implementar algoritmos de programación dinámica para resolver problemas clásicos, como el problema de la mochila o el alineamiento de secuencias. Analizar la complejidad temporal de los algoritmos de programación dinámica y compararla con otras estrategias de resolución de problemas.

Divulga conocimiento científico sobre grafos y algoritmos a través de ponencias y artículos.	**6. FLUJO EN REDES** 6.1 Los problemas de máximo flujo y mínimo corte 6.2 El algoritmo de Ford-Fulkerson 6.3 Elección de mejores caminos de aumentación (versión "capacity scaling" de Ford-Fulkerson) 6.4 Emparejamiento bipartita 6.5 Caminos disjuntos en grafos bipartitas	Los estudiantes comprenderán los problemas de máximo flujo y mínimo corte en redes. Los estudiantes serán capaces de aplicar el algoritmo de Ford-Fulkerson y sus variantes para resolver problemas de flujo máximo en redes. Los estudiantes podrán resolver problemas de emparejamiento bipartito y encontrar caminos disjuntos en grafos bipartitos utilizando algoritmos de flujo en redes.	Clases magistrales para introducir los conceptos de flujo máximo y mínimo corte. Resolución de problemas prácticos utilizando algoritmos de flujo en redes.	Resolver problemas de flujo máximo y mínimo corte en redes utilizando el algoritmo de Ford-Fulkerson. Modelar problemas prácticos en forma de redes y aplicar algoritmos de flujo para encontrar soluciones óptimas.
Participa en la investigación y aplicación de algoritmos de aproximación para la solución de problemas prácticos.	**7. ALGORITMOS DE APROXIMACIÓN** 7.1 El problema de la selección de centros 7.2 El problema Vertex Cover (cobertura de vértices o nodos) 7.3 Solución del problema Vertex Cover usando programación lineal	Los estudiantes comprenderán el concepto de algoritmos de aproximación y su aplicación en la resolución de problemas NP-hard. Los estudiantes serán capaces de aplicar algoritmos de aproximación en problemas de selección de centros y cobertura de vértices. Los estudiantes podrán utilizar técnicas de programación lineal para resolver problemas de optimización como el problema de Vertex Cover.	Clases teóricas para introducir los conceptos de algoritmos de aproximación. Estudio de casos donde se apliquen algoritmos de aproximación para resolver problemas prácticos.	Implementar algoritmos de aproximación para resolver problemas NP-hard, como el problema del Vertex Cover. Evaluar la calidad de las soluciones obtenidas mediante algoritmos de aproximación y compararlas con soluciones óptimas conocidas.

Aplica conceptos matemáticos y de ingeniería para el desarrollo sostenible en la resolución de problemas de infraestructura.	**8. ALGORITMOS ALEATORIOS Y BÚSQUEDA LOCAL** 8.1 Solución de contenciones 8.2 El problema del corte mínimo global 8.3 Diccionarios Hashing 8.4 El algoritmo gradiente descendiente 8.5 El Algoritmo de templado simulado	Los estudiantes comprenderán el funcionamiento de algoritmos aleatorios y su aplicación en la resolución de problemas prácticos. Los estudiantes serán capaces de implementar algoritmos de búsqueda local, como el gradiente descendiente y el algoritmo de templado simulado. Los estudiantes podrán resolver problemas como el corte mínimo global y la solución de contenciones utilizando algoritmos aleatorios y de búsqueda local.	Discusión de conceptos teóricos de algoritmos aleatorios y búsqueda local. Implementación práctica de algoritmos aleatorios y de búsqueda local en problemas específicos.	Implementar algoritmos aleatorios, como el algoritmo de templado simulado, para resolver problemas de optimización. Evaluar la calidad de las soluciones obtenidas mediante algoritmos aleatorios y de búsqueda local y compararlas con soluciones obtenidas por otros métodos.

FUENTES DE INFORMACIÓN	EVALUACIÓN DE LOS APRENDIZAJES
• Kleinberg, J., & Tardos, E. (2005). Algorithm Design. • Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. • Dasgupta, S., Papadimitriou, C. H., & Vazirani, U. V. (2006). Algorithms. • Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms.	• Promedio de todos los proyectos de cada tema 100%

Cronograma del avance programático

Objetos de aprendizaje	Semanas															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1. Introducción al análisis de algoritmos																
2. Algoritmos divide y vencerás																
3. Teoría de grafos																
4. Algoritmos greedy																
5. Programación dinámica																
6. Flujo en redes																
7. Algoritmos de aproximación																
8. Algoritmos aleatorios y búsqueda local																