Codigo de asignatura: 4769B4			
Nombre del programa académico	Maestría en Ingeniería Eléctrica		
Nombre completo de la asignatura	Métodos y Modelos Computacionales		
Número de créditos ECTS por categoría	Ciencias naturales y matemáticas	Módulos profesionales y especiales	Humanidades y ciencias sociales y económicas
	3	3	1
Semestre y año de actualización	2026-1		
Semestre y año en que se imparte	2026-1		
Tipo de asignatura	[ ] Obligatoria [X] Electiva		
Director o contacto del programa	Andrés Escobar Mejía		
Coordinador o contacto de la asignatura	Germán Andrés Holguín Londoño		

Descripción y contenidos

1. Breve descripción

Este curso describe las técnicas básicas para el análisis de la complejidad de un algoritmo. Se incluyen técnicas de diseño y análisis de algoritmos recursivos e iterativos fundamentales en tareas como búsqueda, ordenamiento, operaciones de conjuntos, algoritmos gráficos, algoritmos de manipulación matricial, transformaciones lineales y no lineales, problemas NP-completos, y otras tareas fundamentales en ciencias de la computación, que son transversales a la mayoría de los campos de la investigación científica.

2. Objetivo del curso:

Se espera que al finalizar este curso el estudiante esté en la capacidad de diseñar algoritmos eficientes para la solución de problemas fundamentales en ciencias de la computación. Comprender los fundamentos básicos de la teoría de la complejidad y aplicarlos en el análisis de algoritmos. Tomar decisiones en el diseño de soluciones computacionales a problemas de ingeniería. Dar solución algorítmica a las tareas básicas que aparecen en la mayoría de los campos de la investigación científica utilizando Python como lenguaje de programación de base. Se corresponde con los siguientes Resultados de Aprendizaje del Programa (RAP3, RAP4, RAP5, RAP6, RAP7, RAP8, RAP9, RAP10, RAP11, RAP12, RAP13)

3. Resultados de aprendizaje. Los propósitos de formación en el estudiante de posgrado son:

RA1: Entender qué es complejidad y los fundamentos básicos de la teoría de la complejidad. Se corresponde con los RAP: RAP8, RAP10, RAP11, RAP12.
RA2: Analizar y determinar la complejidad de un algoritmo. Se corresponde con los RAP: RAP8, RAP10, RAP11, RAP12.
RA3: Diseñar algoritmos fundamentales con complejidad óptima. Se corresponde con los RAP: RAP3, RAP7, RAP8, RAP10.
RA4: Capacidad para implementar algoritmos en Python. Se corresponde con los RAP: RAP7, RAP10.
RA5: Capacidad para utilizar Python para solucionar problemas de Ingeniería. Se corresponde con los RAP: RAP1, RAP3, RAP4, RAP7, RAP8, RAP10.
RA6: Capacidad de trabajo en equipo. Se corresponde con los RAP: RAP9, RAP13.
RA7: Capacidad de análisis y síntesis de información. Se corresponde con los RAP: RAP11, RAP12, RAP13.
RA8: Planear y ejecutar proyectos de desarrollo. Se corresponde con los RAP: RAP9, RAP10, RAP12.

4. Contenido

T1: Introducción a los métodos y modelos computacionales.
T2: Problemas NP-Completos.
T3: Algoritmos de ordenamiento.
T4: Programación lineal.
T5: Teoría de Grafos.
T6: Programación dinámica y búsqueda avara.
T7: Optimización irrestricta.
T8: Optimización no lineal con restricciones.

5. Requisitos. Los definidos en requisito de admisión de la IES.

Competencias:
1. Capacidad de programación en cualquier lenguaje de programación.
2. Conocimientos básicos de cálculo integral.

6. Recursos

Herramientas informáticas:

- Python, PyCharm, Spyder, Anaconda. Todas herramientas de software libre.

Bibliografía

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Third Edition (3rd. ed.). The MIT Press. 2009.

[2] Erwin Kreyzig. Advanced Engineering Mathematics. Tenth Edition. John Wiley & Sons Inc Editorial. 2011.

[3] Michael R Garey; David S Johnson. Computers and Intractability, A Guide to the Theory of NP-Completeness. Bell Telephone Laboratories. 1979.

[4] Martin D Davis; Ron Sigal; Elaine J Weyuker. Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science. Academic Press. 1994.

7. Herramientas técnicas de soporte para la enseñanza

- Ejercicios de programación de funciones básicas en Python.
- Ejercicios de comparación de complejidades utilizando Python.
- Ejemplos en clase sobre complejidad computacional.
- Ejemplos en clase sobre algoritmos de búsqueda y ordenamiento.
- Ejemplos de estructuras de datos en Python.
- Ejemplos de optimización utilizando Python.
- Ejercicios de problemas NP-Completos.
- Otras herramientas se presentan en 6.

8. Trabajos en laboratorio y proyectos

- Asignación del curso.

- Trabajos en clase de programación y análisis.

9. Métodos de aprendizaje

Cátedra magistral. Se efectúa planteamiento y debates sobre problemas y diseños propuestos.
Aula extendida. Se dejan temáticas específicas para ser estudiadas y profundizadas en trabajo independiente.
Aprendizaje basado en problemas. Se presentan problemas reales de aplicación al diseño de autómatas.
Trabajos colaborativos. Se desarrollan actividades independientes, personalizadas y grupales en forma de trabajos prácticos.
Investigación formativa. Se fomenta la investigación a través de actividades que permitan la construcción u organización de conocimiento.

10. Métodos de evaluación

Para la obtención de la nota definitiva se realizan diferentes pruebas mediante informes escritos y sustentaciones individuales y grupales de las asignaciones requeridas durante el semestre, las cuales están previstas:
- Examen 1: (T1), (T2), y (T3). (20%). Se evalúan los resultados de aprendizaje (RA1, RA2).
- Examen 2: (T4), (T5), y (T6). (20%). Se evalúan los resultados de aprendizaje (RA3, RA4)