Universidad Nacional de Colombia (UNAL) Sede Manizales
Programa Profesional de Administración de Sistemas Informáticos
SILABO

CS311. Programación Competitiva (Obligatorio)

2022-II

1. Información general		
1.1 Escuela	:	Sistemas de Información
1.2 Curso	:	CS311. Programación Competitiva
1.3 Semestre	:	6^{to} Semestre.
1.4 Prerrequisitos	:	CS212. Análisis y Diseño de Algoritmos. (5^{to} Sem)
1.5 Condición	:	Obligatorio
1.6 Modalidad de aprendizaje	:	Presencial
1.7 horas	:	2 HT; 2 HP; 2 HL;
1.8 Créditos	:	4

2. Profesores

3. Fundamentación del curso

La Programación Competitiva combina retos de solucionar problemas con el añadido de poder competir con otras personas. Enseña a los participantes a pensar más rápido y desarrollar habilidades para resolver problemas, que son de gran demanda en la industria. Este curso enseñará la resolución de problemas algorítmicos de manera rápida combinando la teoría de algoritmos y estructuras de datos con la práctica la solución de los problemas.

4. Resumen

1. Introducción 2. Estructuras de datos 3. Paradigmas de diseño 4. Gráfos 5. Tópicos avanzados 6. Problemas de dominio especifico

5. Objetivos Generales

- Que el alumno utilice técnicas de estructuras de datos y algoritmos complejos.
- Que el alumno aplique los conceptos aprendidos para la aplicación sobre un problema real.
- Que el alumno investigue la posibilidad de crear un nuevo algoritmo y/o técnica nueva para resolver un problema real.

6. Contribución a los resultados (*Outcomes*)

Esta disciplina contribuye al logro de los siguientes resultados de la carrera:

1) Analizar un problema computacional complejo y aplicar los principios computacionales y otras disciplinas relevantes para identificar soluciones. (**Usar**)

6) Aplicar fundamentos de teoria de ciencias de la computación y desarrollo de software para producir soluciones basados en computación. (**Usar**)

7. Contenido

UNIDAD 1: Introducción (20)

Competencias:

Contenido	**Objetivos Generales**
• Introducción a la Programación competitiva • Modelo computacional • Complejidad algoritmica • Problémas sobre búsqueda y ordenamiento • Recursión y recurrencia • Estrategia divide y conquista	• Reconocer y sabes como usar los recursos del modelo de computación RAM (Random Access Machine). [Usar] • Determinar el tiempo y espacio de complejidad de algoritmos. [Usar] • Determinar relaciones de recurrencia para algoritmos recursivos.[Usar] • Resolver problemas de búsqueda y ordenamiento.[Usar] • Aprender a seleccionar los algoritmos adecuados para problemas de tipo divide y conquista.[Usar] • Diseñar nuevos algoritmos para la resolución de problemas.[Usar]

Lecturas: Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)

UNIDAD 2: Estructuras de datos (20)

Competencias:

Contenido	**Objetivos Generales**
• Problemas sobre arrays y strings • Problemas sobre listas enlazadas • Problemas sobre pilas, colas • Problemas sobre arboles • Problemas sobre Hash tables • Problemas sobre Heaps	• Reconocer las distintas estructuras de datos sus complejidades usos y restricciones. [Usar] • Identificar el tipo de estructura de datos adecuado a la resolución del problema. [Usar] • Reconocer tipos de problemas asociado a operaciones sobre estructuras de datos como búsqueda, inserción, eliminación y actualización.[Usar]

Lecturas: Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)

UNIDAD 3: Paradigmas de diseño (20)

Competencias:

Contenido	**Objetivos Generales**
• Fuerza bruta • Divide y conquista • Backtracking • Greedy • Programación Dinamica	• Aprender los distintos paradigmas de resolución de problemas.[Usar] • Aprender a seleccionar los algoritmos adecuados para distintos problemas según el tipo de paradigma.[Usar]

Lecturas: Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)

UNIDAD 4: Gráfos (20)	
Competencias:	
Contenido	**Objetivos Generales**
• Recorrido de gráfos • Aplicaciones y problemas sobre gráfos • Camino mas corto • Redes y flujos	• Identificar problemas clasificados como problemas de grafos. [Usar] • Aprender a seleccionar los algoritmos adecuados para problemas de grafos (recorrido, MST, camino mas costo, redes y flujos) y conocer sus soluciones eficientes. [Usar]
Lecturas: Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)	

UNIDAD 5: Tópicos avanzados (20)	
Competencias:	
Contenido	**Objetivos Generales**
• Teoria de números • Probabilidad y combinaciones • Algoritmos para manejos de strings (tries, string hashing, z-algorithm) • Geometria y sweep line algorithms, segment trees	• Aprender a elegir los algoritmos adecuados para problemas sobre teoria de números y matemáticas ya que son importantes en programación competitiva. [Usar] • Aprender a seleccionar los algoritmos adecuados para problemas sobre probabilidades y combinaciones, manejos de strings y geometría computacional. [Usar]
Lecturas: Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)	

UNIDAD 6: Problemas de dominio especifico (20)	
Competencias:	
Contenido	**Objetivos Generales**
• Latencia y rendimiento • Paralelismo • Redes • Almacenamiento • Alta disponibilidad • Caching • Proxies • Equilibradores de carga • Almacenamiento clave-valo • Replicar y compartir • Elección del líder • Limitación de la tasa • Registro y monitoreo	• Aprender a diseñar sistemas para diferentes problemas de dominio especifico aplicando conocimiento sobre redes, computación distribuida, alta disponibilidad, almacenamiento y arquitectura de sistemas. [Usar]
Lecturas: Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)	

8. Metodología

El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.

El profesor del curso presentará demostraciones para fundamentar clases teóricas.

El profesor y los alumnos realizarán prácticas

Los alumnos deberán asistir a clase habiendo leído lo que el profesor va a presentar. De esta manera se facilitará la comprensión y los estudiantes estarán en mejores condiciones de hacer consultas en clase.

9. Evaluar

Evaluación Continua 1 : 20 %

Examen parcial : 30 %

Evaluación Continua 2 : 20 %

Examen final : 30 %

References

Aziz, A., T.H. Lee, and A. Prakash (2012). *Elements of Programming Interviews: The Insiders' Guide*. ElementsOfProgrammingInterviews.com. ISBN: 9781479274833.

Cormen, T. H. et al. (2009). *Introduction to Algorithms*. MIT Press.

Halim, Steven (2013). *Competitive Programming*. 3 rd. Lulu.

Kulikov, Alexander S. (2019). *Learning Algorithms Through Programming and Puzzle Solving*. Active Learning Technologies.

Laaksonen, Antti (2017). *Guide to Competitive Programming: Learning and Improving Algorithms Through Contests*. Stringer.

Miguel A. Revilla, Steve Skiena (May 2003). *Programming Challenges: The Programming Contest Training Manual*. Springer. ISBN: 978-0387001630.