

**Universidad Católica San Pablo (UCSP)**  
**Escuela Profesional de**  
**Ciencia de la Computación**  
**SILABO**



**CS311. Programación Competitiva (Obligatorio)**

**1. Información general**

1.1 Escuela	:	Ciencia de la Computación
1.2 Curso	:	CS311. Programación Competitiva
1.3 Semestre	:	6 <sup>to</sup> Semestre.
1.4 Prerrequisitos	:	CS212. Análisis y Diseño de Algoritmos. (5 <sup>to</sup> Sem)
1.5 Condición	:	Obligatorio
1.6 Modalidad de aprendizaje	:	Virtual
1.7 horas	:	2 HT; 2 HP; 2 HL;
1.8 Créditos	:	4

**2. Profesores**

**Titular**

- Rensso Victor Hugo Mora Colque <rvhmora@ucsp.edu.pe>  
– Master en Ciencia de la Computación, Universidade Federal de Ouro Preto, Brasil, 2012.

**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 específico

**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:

- a) Aplicar conocimientos de computación y de matemáticas apropiadas para la disciplina. (**Usar**)
- b) Analizar problemas e identificar y definir los requerimientos computacionales apropiados para su solución. (**Usar**)
- i) Utilizar técnicas y herramientas actuales necesarias para la práctica de la computación. (**Usar**)
- j) Aplicar la base matemática, principios de algoritmos y la teoría de la CS en el modelamiento y diseño de sistemas. (**Usar**)

7. Contenido	
<b>UNIDAD 1: Introducción (20)</b>	
<b>Competencias: a,b</b>	
Contenido	Objetivos Generales
<ul style="list-style-type: none"> <li>• Introducción a la Programación competitiva</li> <li>• Modelo computacional</li> <li>• Complejidad algorítmica</li> <li>• Problemas sobre búsqueda y ordenamiento</li> <li>• Recursión y recurrencia</li> <li>• Estrategia divide y conquista</li> </ul>	<ul style="list-style-type: none"> <li>• Reconocer y saber como usar los recursos del modelo de computación RAM (Random Access Machine). [Usar]</li> <li>• Determinar el tiempo y espacio de complejidad de algoritmos. [Usar]</li> <li>• Determinar relaciones de recurrencia para algoritmos recursivos.[Usar]</li> <li>• Resolver problemas de búsqueda y ordenamiento.[Usar]</li> <li>• Aprender a seleccionar los algoritmos adecuados para problemas de tipo divide y conquista.[Usar]</li> <li>• Diseñar nuevos algoritmos para la resolución de problemas.[Usar]</li> </ul>
<b>Lecturas:</b> Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)	
<b>UNIDAD 2: Estructuras de datos (20)</b>	
<b>Competencias: a,b</b>	
Contenido	Objetivos Generales
<ul style="list-style-type: none"> <li>• Problemas sobre arrays y strings</li> <li>• Problemas sobre listas enlazadas</li> <li>• Problemas sobre pilas, colas</li> <li>• Problemas sobre arboles</li> <li>• Problemas sobre Hash tables</li> <li>• Problemas sobre Heaps</li> </ul>	<ul style="list-style-type: none"> <li>• Reconocer las distintas estructuras de datos sus complejidades usos y restricciones. [Usar]</li> <li>• Identificar el tipo de estructura de datos adecuado a la resolución del problema. [Usar]</li> <li>• Reconocer tipos de problemas asociado a operaciones sobre estructuras de datos como búsqueda, inserción, eliminación y actualización.[Usar]</li> </ul>
<b>Lecturas:</b> Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)	
<b>UNIDAD 3: Paradigmas de diseño (20)</b>	
<b>Competencias: a,b</b>	
Contenido	Objetivos Generales
<ul style="list-style-type: none"> <li>• Fuerza bruta</li> <li>• Divide y conquista</li> <li>• Backtracking</li> <li>• Greedy</li> <li>• Programación Dinamica</li> </ul>	<ul style="list-style-type: none"> <li>• Aprender los distintos paradigmas de resolución de problemas.[Usar]</li> <li>• Aprender a seleccionar los algoritmos adecuados para distintos problemas según el tipo de paradigma.[Usar]</li> </ul>
<b>Lecturas:</b> Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)	

<b>UNIDAD 4: Gráfos (20)</b>	
<b>Competencias: a,b</b>	
<b>Contenido</b>	<b>Objetivos Generales</b>
<ul style="list-style-type: none"> <li>• Recorrido de gráfos</li> <li>• Aplicaciones y problemas sobre gráfos</li> <li>• Camino mas corto</li> <li>• Redes y flujos</li> </ul>	<ul style="list-style-type: none"> <li>• Identificar problemas clasificados como problemas de grafos. [Usar]</li> <li>• Aprender a seleccionar los algoritmos adecuados para problemas de grafos (recorrido, MST, camino mas costo, redes y flujos) y conocer sus soluciones eficientes. [Usar]</li> </ul>
<b>Lecturas:</b> Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)	

<b>UNIDAD 5: Tópicos avanzados (20)</b>	
<b>Competencias: a,b</b>	
<b>Contenido</b>	<b>Objetivos Generales</b>
<ul style="list-style-type: none"> <li>• Teoria de números</li> <li>• Probabilidad y combinaciones</li> <li>• Algoritmos para manejos de strings (tries, string hashing, z-algorithm)</li> <li>• Geometria y sweep line algorithms, segment trees</li> </ul>	<ul style="list-style-type: none"> <li>• Aprender a elegir los algoritmos adecuados para problemas sobre teoria de números y matemáticas ya que son importantes en programación competitiva. [Usar]</li> <li>• Aprender a seleccionar los algoritmos adecuados para problemas sobre probabilidades y combinaciones, manejos de strings y geometría computacional. [Usar]</li> </ul>
<b>Lecturas:</b> Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)	

UNIDAD 6: Problemas de dominio específico (20)	
Competencias: a,b	
Contenido	Objetivos Generales
<ul style="list-style-type: none"> <li>• Latencia y rendimiento</li> <li>• Paralelismo</li> <li>• Redes</li> <li>• Almacenamiento</li> <li>• Alta disponibilidad</li> <li>• Caching</li> <li>• Proxies</li> <li>• Equilibradores de carga</li> <li>• Almacenamiento clave-valor</li> <li>• Replicar y compartir</li> <li>• Elección del líder</li> <li>• Limitación de la tasa</li> <li>• Registro y monitoreo</li> </ul>	<ul style="list-style-type: none"> <li>• Aprender a diseñar sistemas para diferentes problemas de dominio específico aplicando conocimiento sobre redes, computación distribuida, alta disponibilidad, almacenamiento y arquitectura de sistemas. [Usar]</li> </ul>
<b>Lecturas:</b> Cormen et al. (2009), Halim (2013), Kulikov (2019), Miguel A. Revilla (2003), Laaksonen (2017), Aziz, Lee, and Prakash (2012)	

8. Metodología
<p>El profesor del curso presentará clases teóricas de los temas señalados en el programa propiciando la intervención de los alumnos.</p> <p>El profesor del curso presentará demostraciones para fundamentar clases teóricas.</p> <p>El profesor y los alumnos realizarán prácticas</p> <p>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.</p>

9. Evaluar
<p><b>Evaluación Continua 1</b> : 20 %</p> <p><b>Examen parcial</b> : 30 %</p> <p><b>Evaluación Continua 2</b> : 20 %</p> <p><b>Examen final</b> : 30 %</p>

## 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.