



Ucayali State University (UNU)  
School of Computer Science  
Syllabus 2023-I

## 1. COURSE

CS311. Competitive Programming (Mandatory)

## 2. GENERAL INFORMATION

2.1 Credits	:	4
2.2 Theory Hours	:	2 (Weekly)
2.3 Practice Hours	:	2 (Weekly)
2.4 Duration of the period	:	16 weeks
2.5 Type of course	:	Mandatory
2.6 Modality	:	Blended
2.7 Prerequisites	:	CS212. Analysis and Design of Algorithms. (5 <sup>th</sup> Sem)

## 3. PROFESSORS

Meetings after coordination with the professor

## 4. INTRODUCTION TO THE COURSE

Competitive Programming combines problem-solving challenges with the fun of competing with others. It teaches participants to think faster and develop problem-solving skills that are in high demand in the industry. This course will teach you to solve algorithmic problems quickly by combining theory of algorithms and data structures with practice solving problems.

## 5. GOALS

- That the student uses techniques of data structures and complex algorithms..
- That the student apply the concepts learned for the application on a real problem.
- That the student investigate the possibility of creating a new algorithm and / or new technique to solve a real problem.

## 6. COMPETENCES

- a) An ability to apply knowledge of mathematics, science. (**Usage**)
- b) An ability to design and conduct experiments, as well as to analyze and interpret data. (**Usage**)
- i) An ability to use the techniques, skills, and modern computing tools necessary for computing practice. (**Usage**)
- j) Apply the mathematical basis, principles of algorithms and the theory of Computer Science in the modeling and design of computational systems in such a way as to demonstrate understanding of the equilibrium points involved in the chosen option. (**Usage**)

## 7. TOPICS

<b>Unit 1: Introduction (20)</b>	
<b>Competences Expected: a,b</b>	
<b>Topics</b>	<b>Learning Outcomes</b>
<ul style="list-style-type: none"> <li>• Introduction to Competitive Programming</li> <li>• Computational model</li> <li>• Runtime and space complexity</li> <li>• Recurrence and recursion</li> <li>• Divide and conquer</li> </ul>	<ul style="list-style-type: none"> <li>• Identify and learn how to use the resources in the Random Access Machine (RAM) computational model. [Usage]</li> <li>• Compute the runtime and space complexity for written algorithms. [Usage]</li> <li>• Compute the recurrence relations for recursive algorithms. [Usage]</li> <li>• Solve problems related to searching and sorting. [Usage]</li> <li>• Learning to select the right algorithms for divide-and-conquer problems. [Usage]</li> <li>• Design new algorithms for real-world problem solving.[Usage]</li> </ul>
<b>Readings :</b> [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

<b>Unit 2: Data structure (20)</b>	
<b>Competences Expected: a,b</b>	
<b>Topics</b>	<b>Learning Outcomes</b>
<ul style="list-style-type: none"> <li>• Arrays and strings problems</li> <li>• Linked lists problems</li> <li>• Stacks and queues problems</li> <li>• Trees problems</li> <li>• Hash tables problems</li> <li>• Heaps problems</li> </ul>	<ul style="list-style-type: none"> <li>• Recognize different data structures, their complexities, uses and restrictions.[Usage]</li> <li>• Identify the type of data structure appropriate to the resolution of the problem. [Usage]</li> <li>• Recognize types of problems associated with operations on data structures such as searching, inserting, deleting and updating.[Usage]</li> </ul>
<b>Readings :</b> [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

<b>Unit 3: Algorithmic Design Paradigms (20)</b>	
<b>Competences Expected: a,b</b>	
<b>Topics</b>	<b>Learning Outcomes</b>
<ul style="list-style-type: none"> <li>• Brute force</li> <li>• Divide and conquer</li> <li>• Backtracking</li> <li>• Greedy</li> <li>• Dynamic Programming</li> </ul>	<ul style="list-style-type: none"> <li>• Learning the different algorithmic design paradigms.[Usage]</li> <li>• Learning to select the right algorithms for different problems applying different algorithmic design paradigms.[Usage]</li> </ul>
<b>Readings :</b> [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

<b>Unit 4: Graphs (20)</b>	
<b>Competences Expected: a,b</b>	
<b>Topics</b>	<b>Learning Outcomes</b>
<ul style="list-style-type: none"> <li>• Graphs transversal</li> <li>• Graphs applications</li> <li>• Shortest path</li> <li>• Networks and flows</li> </ul>	<ul style="list-style-type: none"> <li>• Identify problems classified as graph problems. [Usage]</li> <li>• Learn how to select the right algorithms for network problems (transversal, MST, shortest-path, network and flows). [Usage]</li> </ul>
<b>Readings :</b> [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

<b>Unit 5: Advanced topics (20)</b>	
<b>Competences Expected: a,b</b>	
<b>Topics</b>	<b>Learning Outcomes</b>
<ul style="list-style-type: none"> <li>• Number theory</li> <li>• Probabilities and combinations</li> <li>• String algorithms (tries, string hashing, z-algorithm)</li> <li>• Geometric algorithms</li> </ul>	<ul style="list-style-type: none"> <li>• Learning to select the right algorithms for problems in number theory and mathematics as they are important in competitive programming. [Usage]</li> <li>• Learning to select the right algorithms for problems about probabilities and combinations, strings and computational geometry. [Usage]</li> </ul>
<b>Readings :</b> [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

<b>Unit 6: Domain specific problems (20)</b>	
<b>Competences Expected: a,b</b>	
<b>Topics</b>	<b>Learning Outcomes</b>
<ul style="list-style-type: none"> <li>• Latency and throughput</li> <li>• Parallelism</li> <li>• Networks</li> <li>• Storage</li> <li>• High availability</li> <li>• Caching</li> <li>• Proxies</li> <li>• Load balancers</li> <li>• Key-value stores</li> <li>• Replicating and sharing</li> <li>• Leader election</li> <li>• Rate limiting</li> <li>• Logging and monitoring</li> </ul>	<ul style="list-style-type: none"> <li>• Learning to design systems for different domain-specific problems by applying knowledge about networks, distributed computing, high availability, storage and system architecture.[Usage]</li> </ul>
<b>Readings :</b> [Cor+09], [Hal13], [Kul19], [Mig03], [Laa17], [ALP12]	

## 8. WORKPLAN

### 8.1 Methodology

Individual and team participation is encouraged to present their ideas, motivating them with additional points in the different stages of the course evaluation.

### 8.2 Theory Sessions

The theory sessions are held in master classes with activities including active learning and roleplay to allow students to internalize the concepts.

### 8.3 Practical Sessions

The practical sessions are held in class where a series of exercises and/or practical concepts are developed through problem solving, problem solving, specific exercises and/or in application contexts.

## 9. EVALUATION SYSTEM

\*\*\*\*\* EVALUATION MISSING \*\*\*\*\*

## 10. BASIC BIBLIOGRAPHY

- [ALP12] A. Aziz, T.H. Lee, and A. Prakash. *Elements of Programming Interviews: The Insiders' Guide*. ElementsOfProgrammingInterviews.com, 2012. ISBN: 9781479274833. URL: <https://books.google.com.pe/books?id=y6FLBQAAQBAJ>.
- [Cor+09] T. H. Cormen et al. *Introduction to Algorithms*. MIT Press, 2009.
- [Hal13] Steven Halim. *Competitive Programming*. 3 rd. Lulu, 2013.
- [Kul19] Alexander S. Kulikov. *Learning Algorithms Through Programming and Puzzle Solving*. Active Learning Technologies, 2019.
- [Laa17] Antti Laaksonen. *Guide to Competitive Programming: Learning and Improving Algorithms Through Contests*. Stringer, 2017.
- [Mig03] Steve Skiena Miguel A. Revilla. *Programming Challenges: The Programming Contest Training Manual*. Springer, May 2003. ISBN: 978-0387001630.