**Peruvian Computing Society (SPC)**
School of Computer Science
Sillabus 2021-I

## 1. COURSE

CS212. Analysis and Design of Algorithms (Mandatory)

## 2. GENERAL INFORMATION

| | | |
|---|---|---|
| **2.1 Credits** | : | 4 |
| **2.2 Theory Hours** | : | 2 (Weekly) |
| **2.3 Practice Hours** | : | 4 (Weekly) |
| **2.4 Duration of the period** | : | 16 weeks |
| **2.5 Type of course** | : | Mandatory |
| **2.6 Modality** | : | Face to face |
| **2.7 Prerrequisites** | : | • CS210. Algorithms and Data Structures. ($4^{th}$ Sem)<br>• CS211. Theory of Computation. ($4^{th}$ Sem) |

## 3. PROFESSORS

Meetings after coordination with the professor

## 4. INTRODUCTION TO THE COURSE

An algorithm is, essentially, a well-defined set of rules or instructions that allow solving a computational problem. The theoretical study of the performance of the algorithms and the resources used by them, usually time and space, allows us to evaluate if an algorithm is suitable for solving a specific problem, comparing it with other algorithms for the same problem or even delimiting the boundary between Viable and impossible. This matter is so important that even Donald E. Knuth defined Computer Science as the study of algorithms. This course will present the most common techniques used in the analysis and design of efficient algorithms, with the purpose of learning the fundamental principles of the design, implementation and analysis of algorithms for the solution of computational problems

## 5. GOALS

- Develop the ability to evaluate the complexity and quality of algorithms proposed for a given problem.
- Study the most representative, introductory algorithms of the most important classes of problems treated in computation.
- Develop the ability to solve algorithmic problems using the fundamental principles of algorithm design learneds.
- Be able to answer the following questions when a new algorithm is presented: How good is the performance ?, Is there a better way to solve the problem?

## 6. COMPETENCES

**a)** An ability to apply knowledge of mathematics, science. (**Assessment**)

**b)** An ability to design and conduct experiments, as well as to analyze and interpret data. (**Assessment**)

## 7. SPECIFIC COMPETENCES

**a10)** Make a computational analysis that allows calculating the execution time of a given algorithm.

**a11)** Use mathematical techniques that allow to delimit sums and to solve recurrences that reflect the computational costs of an algorithm.

**b4)** Identify and efficiently apply various algorithmic strategies and data structures for the solution of a problem given certain space and time constraints.

**b11)** Understand the difference between an NP-difficult problem and one that has a polynomial solution.

**b12)** Given a problem with a polynomial solution, identify whether it can be solved by a voracious strategy, by a dynamic scheduling strategy or by a strategy of divide and conquer taking into account the size of the input.

## 8. TOPICS

| **Unit 1: Basic Analysis (10)** | |
|---|---|
| **Competences Expected: a** | |
| **Topics** | **Learning Outcomes** |
| • Differences among best, expected, and worst case behaviors of an algorithm<br>• Asymptotic analysis of upper and expected complexity bounds<br>• Complexity classes, such as constant, logarithmic, linear, quadratic, and exponential<br>• Asymptotic Notation<br>• Analysis of iterative and recursive algorithms<br>• Inductive proofs and correctness of algorithms<br>• Master Theorem and Recursion Trees | • Explain what is meant by "best", "expected", and "worst" case behavior of an algorithm [Assessment]<br>• Determine informally the time and space complexity of different algorithms [Assessment]<br>• List and contrast standard complexity classes [Assessment]<br>• Explain the use of big omega, big theta, and little o notation to describe the amount of work done by an algorithm [Assessment]<br>• Analyze worst-case running times of algorithms using asymptotic analysis [Assessment]<br>• Use recurrence relations to determine the time complexity of recursively defined algorithms [Assessment]<br>• Solve elementary recurrence relations, eg, using some form of a Master Theorem [Assessment]<br>• Argue the correctness of algorithms using inductive proofs [Assessment] |
| **Readings :** [KT05], [DPV06], [RS09], [SF13], [Knu97] | |

| **Unit 2: Algorithmic Strategies (30)** | |
|---|---|
| **Competences Expected: a,b** | |
| **Topics** | **Learning Outcomes** |
| • Brute-force algorithms<br>• Greedy algorithms<br>• Divide-and-conquer<br>• Dynamic Programming | • For each of the strategies (brute-force, greedy, divide-and-conquer, recursive backtracking, and dynamic programming), identify a practical example to which it would apply [Assessment]<br>• Use a greedy approach to solve an appropriate problem and determine if the greedy rule chosen leads to an optimal solution [Assessment]<br>• Use a divide-and-conquer algorithm to solve an appropriate problem [Assessment]<br>• Use dynamic programming to solve an appropriate problem [Assessment]<br>• Determine an appropriate algorithmic approach to a problem [Assessment] |
| **Readings :** [KT05], [DPV06], [RS09], [Als99] | |

| **Unit 3: Fundamental Data Structures and Algorithms (6)** | |
|---|---|
| **Competences Expected: a,b** | |
| **Topics** | **Learning Outcomes** |
| • Graphs and graph algorithms<br>– Maximum and minimum cut problem<br>– Local search<br>• Cache oblivious algorithms<br>• Number theory and cryptography | • Discuss factors other than computational efficiency that influence the choice of algorithms, such as programming time, maintainability, and the use of application-specific patterns in the input data [Familiarity]<br>• Solve problems using fundamental graph algorithms, including depth-first and breadth-first search [Assessment]<br>• Demonstrate the ability to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in a particular context [Assessment]<br>• Solve problems using graph algorithms, including single-source and all-pairs shortest paths, and at least one minimum spanning tree algorithm [Assessment] |
| **Readings :** [KT05], [DPV06], [RS09], [SW11], [GT09] | |

| Unit 4: Basic Automata Computability and Complexity (2) | |
|---|---|
| **Competences Expected: a,b** | |
| **Topics** | **Learning Outcomes** |
| • Introduction to the P and NP classes and the P vs. NP problem<br>• Introduction to the NP-complete class and exemplary NP-complete problems (e.g., SAT, Knapsack)<br>• Reductions | • Define the classes P and NP [Familiarity]<br>• Explain the significance of NP-completeness [Familiarity] |
| **Readings :** [KT05], [DPV06], [RS09] | |

| Unit 5: Advanced Data Structures Algorithms and Analysis (12) | |
|---|---|
| **Competences Expected: a,b** | |
| **Topics** | **Learning Outcomes** |
| • Graphs (e.g, topological sort, finding strongly connected components, matching)<br>• Randomized algorithms<br>• Amortized analysis<br>• Probabilistic analysis<br>• Approximation Algorithms<br>• Linear Programming | • Understand the mapping of real-world problems to algorithmic solutions (eg, as graph problems, linear programs, etc) [Familiarity]<br>• Select and apply advanced analysis techniques (eg, amortized, probabilistic, etc) to algorithms [Usage] |
| **Readings :** [KT05], [DPV06], [RS09], [Tar83], [Raw92] | |

## 9. WORKPLAN

### 9.1 Methodology

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

### 9.2 Theory Sessions

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

### 9.3 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.

## 10. EVALUATION SYSTEM

********* EVALUATION MISSING ********

## 11. BASIC BIBLIOGRAPHY


[Als99] H. Alsuwaiyel. *Algorithms: Design Techniques and Analysis*. World Scientific, 1999. ISBN: 9789810237400.

[DPV06] S. Dasgupta, C. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill Education, 2006. ISBN: 9780073523408.

[GT09] Michael T. Goodrich and Roberto Tamassia. *Algorithm Design: Foundations, Analysis and Internet Examples*. 2nd. John Wiley & Sons, Inc., 2009. ISBN: 0470088540, 9780470088548.

[Knu97] D.E. Knuth. *The Art of Computer Programming: Fundamental algorithms Vol 1*. Third Edition. Addison-Wesley, 1997. ISBN: 9780201896831. URL: `http://www-cs-faculty.stanford/~knuth/taocp.html`.

[KT05] Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN: 0321295358.

[Raw92] G.J.E. Rawlins. *Compared to What?: An Introduction to the Analysis of Algorithms*. Computer Science Press, 1992. ISBN: 9780716782438.

[RS09] Thomas H. Cormen; Charles E. Leiserson ; Ronald L. Rivest and Clifford Stein. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 0262033844.

[SF13] R. Sedgewick and P. Flajolet. *An Introduction to the Analysis of Algorithms*. Pearson Education, 2013. ISBN: 9780133373486.

[SW11] R. Sedgewick and K. Wayne. *Algorithms*. Pearson Education, 2011. ISBN: 9780132762564.

[Tar83] Robert Endre Tarjan. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, 1983. ISBN: 0-89871-187-8.