University of Engineering and Technology
School of Computer Science
Syllabus of Course – Academic Period 2017-I

1. **Code and Name:** CS212. Analysis and Design of Algorithms
2. **Credits:** 4
3. **Hours of theory and Lab:** 2 HT; 4 HP;
4. **Professor(s)**
   Meetings after coordination with the professor
5. **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.

6. **Information about the course**

   (a) **Brief description about 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

   (b) **Prerrequisites:** CS210. Algoritmos y Estructuras de Datos. (4^{to} Sem)

   (c) **Type of Course:** Mandatory

7. **Competences**

   - 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?

8. Contribution to Outcomes

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**)

h) A recognition of the need for, and an ability to engage in life-long learning. (**Usage**)

i) An ability to use the techniques, skills, and modern computing tools necessary for computing practice. (**Usage**)

9. Competences (IEEE)

C1. An intellectual understanding and the ability to apply mathematical foundations and computer science theory.⇒ **Outcome a**

C2. Ability to have a critical and creative perspective in identifying and solving problems using computational thinking. ⇒ **Outcome b**

C3. An intellectual understanding of, and an appreciation for, the central role of algorithms and data structures.⇒ **Outcome b**

C5. Ability to implement algorithms and data structures in software.⇒ **Outcome i**

C6. Ability to design and implement larger structural units that utilize algorithms and data structures and the interfaces through which these units communicate.⇒ **Outcome i**

C9. Understanding of computing's limitations, including the difference between what computing is inherently incapable of doing vs. what may be accomplished via future science and technology.⇒ **Outcome a**

C16. Ability to identify advanced computing topics and understanding the frontiers of the discipline.⇒ **Outcome h**

10. List of topics

1. Basic Analysis
2. Algorithmic Strategies
3. Fundamental Data Structures and Algorithms
4. Basic Automata Computability and Complexity
5. Advanced Data Structures Algorithms and Analysis

11. Methodology and Evaluation
Methodology:

Theory Sessions:
The development of the theoretical sessions is focused on the student, through his active participation, solving problems related to the course with the individual contributions and discussing real cases of the industry. The students will develop throughout the course a project of application of the tools received in a company.

Lab Sessions:
Practical sessions are held in the laboratory. Laboratory practices are performed in teams to strengthen their communication. At the beginning of each laboratory the development of the practice is explained and at the end the main conclusions of the activity in group form are highlighted.

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

Reading:
Throughout the course different readings are provided, which are evaluated. The average of the notes in the readings is

considered as the mark of a qualified practice. The use of the UTEC Online virtual campus allows each student to access the course information, and interact outside the classroom with the teacher and with the other students.

Evaluation System:

.

12. Content

<table>
<tr><td colspan="2">Unit 1: Basic Analysis (10)</td></tr>
<tr><td colspan="2">Competences Expected: C1</td></tr>
<tr><td>Learning Outcomes</td><td>Topics</td></tr>
<tr><td>• Explain what is meant by “best”, “expected”, and “worst” case behavior of an algorithm [Assessment]
• In the context of specific algorithms, identify the characteristics of data and/or other conditions or assumptions that lead to different behaviors [Assessment]
• Determine informally the time and space complexity of simple algorithms [Assessment]
• State the formal definition of big O [Assessment]
• List and contrast standard complexity classes [Assessment]
• Use big O notation formally to give asymptotic upper bounds on time and space complexity of algorithms [Assessment]
• Use big O notation formally to give expected case bounds on time complexity of algorithms [Assessment]
• Explain the use of big omega, big theta, and little o notation to describe the amount of work done by an algorithm [Assessment]
• Use recurrence relations to determine the time complexity of recursively defined algorithms [Assessment]
• Solve elementary recurrence relations, eg, using some form of a Master Theorem [Assessment]</td><td>• Differences among best, expected, and worst case behaviors of an algorithm
• Asymptotic analysis of upper and expected complexity bounds
• Big O notation: formal definition
• Complexity classes, such as constant, logarithmic, linear, quadratic, and exponential
• Big O notation: use
• Recurrence relations
• Analysis of iterative and recursive algorithms
• Some version of a Master Theorem</td></tr>
<tr><td colspan="2">Readings : [KT05], [DPV06], [RS09], [SF13], [Knu97]</td></tr>
</table>

Unit 2: Algorithmic Strategies (30)	
Competences Expected: C2	
Learning Outcomes	**Topics**
• 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] • Use a greedy approach to solve an appropriate problem and determine if the greedy rule chosen leads to an optimal solution [Assessment] • Use a divide-and-conquer algorithm to solve an appropriate problem [Assessment] • Use dynamic programming to solve an appropriate problem [Assessment] • Determine an appropriate algorithmic approach to a problem [Assessment]	• Brute-force algorithms • Greedy algorithms • Divide-and-conquer • Dynamic Programming
Readings : [KT05], [DPV06], [RS09], [Als99]	

Unit 3: Fundamental Data Structures and Algorithms (10)

Competences Expected: C6

Learning Outcomes	Topics
• Implement basic numerical algorithms [Assessment] • Implement simple search algorithms and explain the differences in their time complexities [Assessment] • Be able to implement common quadratic and O(N log N) sorting algorithms [Assessment] • Discuss the runtime and memory efficiency of principal algorithms for sorting, searching, and hashing [Usage] • 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] • Solve problems using fundamental graph algorithms, including depth-first and breadth-first search [Assessment] • 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] • Describe the heap property and the use of heaps as an implementation of priority queues [Assessment] • Solve problems using graph algorithms, including single-source and all-pairs shortest paths, and at least one minimum spanning tree algorithm [Assessment]	• Simple numerical algorithms, such as computing the average of a list of numbers, finding the min, max, • Sequential and binary search algorithms • Worst case quadratic sorting algorithms (selection, insertion) • Worst or average case O(N log N) sorting algorithms (quicksort, heapsort, mergesort) • Graphs and graph algorithms – Representations of graphs (e.g., adjacency list, adjacency matrix) – Depth- and breadth-first traversals • Heaps • Graphs and graph algorithms – Shortest-path algorithms (Dijkstra's and Floyd's algorithms) – Minimum spanning tree (Prim's and Kruskal's algorithms)

Readings : [KT05], [DPV06], [RS09], [SW11], [GT09]

Unit 4: Basic Automata Computability and Complexity (2)

Competences Expected: C9

Learning Outcomes	Topics
• Define the classes P and NP [Familiarity] • Explain the significance of NP-completeness [Familiarity]	• Introduction to the P and NP classes and the P vs. NP problem • Introduction to the NP-complete class and exemplary NP-complete problems (e.g., SAT, Knapsack)

Readings : [KT05], [DPV06], [RS09]

Unit 5: Advanced Data Structures Algorithms and Analysis (8)	
Competences Expected: C16	
Learning Outcomes	**Topics**
• Understand the mapping of real-world problems to algorithmic solutions (eg, as graph problems, linear programs, etc) [Familiarity] • Select and apply advanced analysis techniques (eg, amortized, probabilistic, etc) to algorithms [Usage]	• Graphs (e.g, topological sort, finding strongly connected components, matching) • Number-theoretic algorithms (e.g., modular arithmetic, primality testing, integer factorization) • Randomized algorithms • Amortized analysis • Probabilistic analysis
Readings : [KT05], [DPV06], [RS09], [Tar83], [Raw92]	