NATIONAL UNIVERSITY OF ENGINEERING
COLLEGE OF SCIENCES
COMPUTER SCIENCE PROGRAM

CC262 - ALGORITHMS

I. GENERAL INFORMATION

CODE	: CC262 - Algorithms
SEMESTER	: 4
CREDITS	: 4
HOURS PER WEEK	: 6 (Theory – Practice - Laboratory)
PREREQUISITES	: CM102 Introduction to Programming, CM254 Introduction to Discrete Mathematics
CONDITION	: Mandatory

II. COURSE DESCRIPTION

Develop the student's capacity for abstraction. Introduce and develop structures and algorithms. The course will also offer an introduction to the historical and social context of computer science and a review of the scope of this discipline.

III. LEARNING UNITS

Basics:

1. Algorithms' Role in Computing

I: Algorithms.
II: Algorithms as a technology.

2. Starting

I: Classification by insertion.
II: Analyzing algorithms.
III: Designing algorithms.

3. Growth of Functions

I: Asymptotic notation.
II: Standard notation and common functions.

4. Divide and Conquer

I: The maximum subrray problem.
II: The Strassen algorithm for matrix multiplication.
III: The substitution method to resolve recurrences.
IV: The tree recursion method to resolve recurrences.
V: The master method to resolve recurrences.
VI: Testing the master theorem.

5. Probabilistic analysis and random algorithms

I: The hiring problem.
II: Indicator of random variables.
III: Random algorithms.
IV: Probabilistic analysis and additional uses of the indicator of random variables.

Order statistics and classification

6. HeapSort

I: Pile.
II: Maintaining the pile property.
III: Building a heap.
IV: HeapSort algorithm.
V: Queues' property.

7. QuickSort

I: QuickSort description.
II: QuickSort performance.
III: A random version of QuickSort.
IV: QuickSort analysis.

8. Real-time sorting

I: Lower limits for sorting.
II: Sorting by counting.
III: Basic sorting.
IV: Cube sorting.

9. Medium and order statistics

I: Minimum and maximum.
II: Selection in expected linear time.
III: Selection in the worst case of linear time.

Advanced design and analysis techniques

10. Dynamic programming

I: Cutting bar.
II: Matrix multiplication chain.
III: Dynamic programming elements.
IV: Longest common subsequence.

V: Optimal tree of binary search.

11. Greedy algorithms

I: A problem of activity selection.
II: Greedy strategy elements.
III: Huffman codes.
IV: Matroids and greedy methods.
V: A problem of programming tasks as a matroid.

12. Amortized analysis

I: Added analysis.
II: The accounting method.
III: The potential method.
IV: Dynamic tables.

13. Graph algorithm

I: Graph representations.
II: Search first in amplitude.
III: Search first in depth.
IV: Topologic sorting.
V: Strongly connected components.

14. Minimum expansion tree.

I: Growth of a minimum expansion tree.
II: Kruskal and Prim algorithms.

15. Shorter paths of a single provider

I: Bellaman-Ford algorithm.
II: Shorter paths of a single provider in acyclic directed graphs.
III: Dijkstra algorithm.
IV: Restriction differences and shorter paths.
V: Testing properties of shorter paths.

16. Every pair of shorter paths

I: Shorter paths and matrix multiplication.
II: Floyd-Warshall algorithm.
III: Johnson algorithm for scattered graphs.

17. Maximum Flow

I: Network flow.
II: The Ford-Fulkerson method.
III: Maximum coincidence in bipartite graphs.
IV: Re-labeling algorithm.
V: Algorithms of re-labeling forward.

IV. BIBLIOGRAPHY

- H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein. Introducción to Algoritmos, 3rd Edition. MIT Press. September 2009. ISBN-10: 0-262-03384-4, ISBN-13: 978-0-262-03384-8.