<table>
<tr><td>Course Title</td><td colspan="5">Algorithms & Complexity</td></tr>
<tr><td>Course Code</td><td colspan="5">CSC662</td></tr>
<tr><td>Course Type</td><td colspan="5">Elective</td></tr>
<tr><td>Level</td><td colspan="5">Master (2nd cycle)</td></tr>
<tr><td>Year / Semester</td><td colspan="5">2nd year / 1st semester</td></tr>
<tr><td>Teacher’s Name</td><td colspan="5">TBA</td></tr>
<tr><td>ECTS</td><td>10</td><td>Lectures / week</td><td>3 hours / 14 weeks</td><td>Laboratories / week</td><td>N/A</td></tr>
<tr><td>Course Purpose and Objectives</td><td colspan="5">The first part of the cource introduce the students to the design and analysis of algorithms for computational problems, and how to think clearly about analyzing correctness and running time. The objective of the first part of the course is to provide the intellectual tools needed for designing and analyzing algorithms for new problems the students may face in the future. Specific algorithms for a variety of problems will be studied, such as greedy techniques, divide-and-conquer and others, as well as general design and analysis techniques.

The second part of the course includes advanced techniques in the design and analysis of algorithms. The algorithms are presented using a rigorous analytical style. We will be emphasizing various algorithmic paradigms such as dynamic programming, network flows, linear programming and rounding, randomized algorithms, local search and multiplicative weights update and NP and intractability. These techniques will be applied to a wide variety of (well motivated) discrete computational problems with a focus on combinatorial optimization.</td></tr>
<tr><td>Learning Outcomes</td><td colspan="5">After competing this course students should be able to:
Explain, use and discuss fundamental algorithms and algorithmic techniques.
Explain the use of big-O, Omega, and Theta notation to describe the amount of work done by an algorithm, and apply them to provide tight bounds on algorithmic complexity.
Create correctness proofs and estimate the running time of a given algorithm.
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.
Describe and discuss the basic idea behind the techniques, so that to be are able to develop algorithms for new problems where these techniques can be applied.</td></tr>
</table>

	- Describe and apply the algorithms discussed in class, prove their correctness, and analyze their time complexity in a mathematically rigorous manner. - Given a practical application, identify the computational issues and apply suitable algorithms to solve it effectively. - Identify, describe and use NP-complete problems - Discuss various issues on computability and complexity theory. - Prove a problem is NP-complete using reduction and identify the implications.		
Prerequisites	CSC615	Co-requisites	None
Course Content	**Analysis framework:** O, Θ, Ω notations Mathematical analysis: nonrecursive and recursive algorithms. Graphs, trees and their properties. Breadth- and depth-first search in graphs, topological sort, recurrences. **Divide-and-conquer:** Multiplication of Large Integers and Strassen's Matrix Multiplication, Closest-Pair and Convex-Hull Problems by Divide-and-Conquer. **Sorting and Selection:** Randomization, Median Finding, Quick Sort, Radix Sort, selection, Lower Bound for Sorting **Greedy technique:** Huffman's Codes, Minimum Spanning Tree algorithms: Kruskal's Algorithm, Prim's Algorithm, single pair Shortest Paths algorithm: Dijkstra's Algorithm. Scheduling to Minimize Lateness: An Exchange Argument, The Minimum Spanning Tree Problem, Huffman Codes and the Problem of Data Compression **Dynamic Programming:** Single Source Shortest Path algorithms: Warshall's and Floyd's Algorithms, Knapsack Problem, Optimal Binary Search Trees, The Knapsack Problem and Memory Functions **Iterative Improvement:** The Simplex Method, the Maximum-Flow Problem (Ford-Fulkerson method), Maximum Matching in Bipartite Graphs, the Stable Marriage Problem **Dynamic Programming:** Weighted Interval Scheduling: A Recursive Procedure, Weighted Interval Scheduling: Iterating over Sub-Problems, Segmented Least Squares: Multi-way Choices, Subset Sums and		

	Knapsacks: Adding a Variable, Shortest Paths in a Graph, Shortest Paths and Distance Vector Protocols, Negative Cycles in a Graph. **Network Flow:** Maximum Flows and Minimum Cuts in a Network, Disjoint Paths in Directed and Undirected Graphs, Airline Scheduling. **NP and Computational Intractability:** Polynomial-time Reductions, Efficient Certification and the Definition of NP, NP-Complete Problems, Sequencing Problems, Partitioning Problems, Graph Coloring, Numerical Problems, co-NP and the Asymmetry of NP, A Partial Taxonomy of Hard Problems **Extending the Limits of Tractability:** Finding Small Vertex Covers, Solving NP-hard Problem on Trees, Coloring a Set of Circular Arcs. **Local Search:** The Landscape of an Optimization Problem. The Metropolis Algorithm and Simulated Annealing. An Application of Local Search to Hopfield Neural Networks. Maximum Cut Approximation via Local Search **Approximation Algorithms:** Greedy Algorithms and Bounds on the Optimum: A Load Balancing Problem, the Vertex-cover problem, the traveling salesman problem, the set-cover problem, the vertex-coloring problem. The Center Selection Problem, the Set Cover. The Pricing Method: Vertex Cover. Linear Programming and Rounding: An Application to Vertex Cover.
Teaching Methodology	Face-to-Face
Bibliography	Jon Kleinberg and Éva Tardos. Algorithm Design. Addison-Wesley. S. Dasgupta, C. Papadimitriou, U. Vazirani,ALGORITHMS, McGraw-Hill. T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein INTRODUCTION TO ALGORITHMS, MIT Press. R. Johnsonbaugh, M. Schaefer, ALGORITHMS, Prentice Hall Sanjeev Arora and Boaz Barak, Computational Complexity, A Modern Approach, Cambridge University Press

Assessment	Examinations 60% Assignments 30% Class participation and Attendance 10% 100%
Language	English