Course Code	Course Title
COMP 3711	**Design and Analysis of Algorithms**

Course Description

Techniques for designing algorithms, proving their correctness, and analyzing their running times. Topics covered include: sorting, selection, heaps, balanced search trees, divide-and-conquer, greedy algorithms, dynamic programming, and graph algorithms. Prerequisite(s): (COMP 2011 OR COMP 2012 OR COMP 2012H) AND (COMP 2711 OR COMP 2711H OR MATH 2343). Exclusion(s): COMP 3711H

List of Topics

Syllabus:
Techniques for designing algorithms, proving their correctness, and analyzing their running times.

Topics covered include:
sorting, selection, heaps, balanced search trees, divide-and-conquer, greedy algorithms, dynamic programming, and graph algorithms.

Textbooks

Cormen, C. Leiserson, R. Rivest, C. Stein. Introduction to Algorithms, MIT Press.

Reference books

N/A

Grading Scheme

Written assignments (All assignments must be submitted online.)	30%
Midterm	30%
Final examination	40%
Total	100%

Course Intended Learning Outcomes

1. Describe fundamental concepts and techniques for determining the asymptotic behavior of real-valued functions defined in natural numbers.
2. Explain recurrence equations and solve common recurrences using a variety of techniques.

3. Analyze an algorithm described in plain language or some form of pseudocode in terms of its time (or space) efficiency as a function of the size of a problem instance.
4. Explain how various data structures, including trees, heaps and disjoint set structures, influence the time efficiency of algorithms.
5. Apply general algorithmic design and analysis techniques to solving problems, including greedy, divide-and-conquer and dynamic programming.
6. Identify randomization in algorithms and analyze basic randomized algorithms such as randomized quicksort and selection.

Assessment Rubrics

1. Describe fundamental concepts and techniques for determining the asymptotic behavior of real-valued functions defined in natural numbers.	Demonstrates thorough understanding of the definitions and usage of asymptotic analysis of real-valued functions. Is able to use them to describe the running time of very complicated algorithms.	Demonstrates sufficient understanding of the definitions and usage of asymptotic analysis of real-valued functions. Is able to use them to describe the running time of basic algorithms.	Demonstrates some preliminary understanding of the definitions and usage of asymptotic analysis of real-valued functions.	Demonstrates deficient understanding of the definitions and/or usage of asymptotic analysis of real-valued functions.
2. Explain recurrence equations and solve common recurrences using a variety of techniques.	Demonstrates thorough understanding of recurrence relations and how to solve recurrences that are major modifications of the common varieties taught in class.	Demonstrates sufficient understanding of recurrence relations and how to solve recurrences that are slight modifications of the common varieties taught in class.	Demonstrates basic understanding of what a recurrence relation models and how to solve simple recurrence relations.	Demonstrates deficient knowledge of how to formulate and solve even basic recurrence equations.
3. Analyze an algorithm described in plain language or some form of pseudocode in terms of its time (or space) efficiency as a function of the size of a problem instance	Demonstrates ability to fully analyze the efficiency of complicated algorithms described in plain-language/pseudocode.	Demonstrates sufficient ability to fully analyze the efficiency of simple algorithms described in plain-language/pseudocode and at least partially analyze more complicated algorithms.	Demonstrates preliminary understanding of how to analyze efficiency of an algorithm described in plain-language/pseudocode.	Does not understand how to translate a plain-language/pseudocode description of an algorithm into a format amenable to analysis

4. Explain how various data structures, including trees, heaps and disjoint set structures, influence the time efficiency of algorithms.	Demonstrates strong understanding of the influence of the usage of data structures on the time efficiency of algorithms. Is usually able to use this knowledge to formulate accurate equations describing this time efficiency	Demonstrates sufficient understanding of the influence of the usage of data structures on the time efficiency of algorithms. Is often able to use this knowledge to formulate accurate equations describing this time efficiency.	Demonstrates preliminary understanding of the direct influence of the usage of various data structures on the time efficiency of algorithms.	Has very limited understanding of the influence of the usage of data structures on the time efficiency of algorithms.
5. Apply general algorithmic design and analysis techniques to solving problems, including greedy, divide-and-conquer and dynamic programming.	Is usually able to apply general algorithmic design and analysis techniques to deriving the most efficient algorithms to solve problems	Is usually able to apply general algorithmic design and analysis techniques to deriving efficient algorithms to solve problems	Is sometimes unable to apply general algorithmic design and analysis techniques to deriving efficient algorithms to solve simple problems.	Is usually unable to apply general algorithmic design and analysis techniques to deriving efficient algorithms to solve problems.
6. Identify randomization in algorithms and analyze basic randomized algorithms such as randomized quicksort and selection.	Is able to identify randomization in algorithms and to analyze more complicated randomized algorithms	Is able to identify randomization in algorithms and to usually analyze basic randomized algorithms	Is able to identify randomization in algorithms and to sometimes analyze basic randomized algorithms.	Is often unable to identify randomization in algorithms and to analyze basic randomized algorithms