CMP2202 Analysis and Design of Algorithms

Period per Week			Contact Hour per Semester	Weighted Total Mark	Weighted Exam Mark	Weighted Continuous Assessment Mark	Credit Units
LH	PH	TH	CH	WTM	WEM	WCM	CU
45	00	00	45	100	40	100	3

Rationale

Algorithms are fundamental to computer engineering. The real-world performance of any software or hardware system depends on two things: (1) the algorithms chosen, and (2) the suitability and efficiency of the implementation. Good algorithm design is, therefore, crucial for the performance of all systems. Moreover, the study of algorithms provides insight into the intrinsic nature of the problem as well as possible solution techniques independent of programming language, computer hardware, or any other implementation aspect.

Objectives

- Explain the role and purpose of algorithms in Computer Engineering
- Indicate how algorithms are part of many different computer applications
- Provide some knowledge themes such as relating complexity with algorithms

Course Content

1. History and Overview

- Indicate some reasons for studying analysis, complexity, and algorithmic strategies
- Highlight some people that contributed or influenced the area of algorithms
- Mention some basic algorithms and some reasons for their differences
- Highlight how the use of theory influences algorithms
- Indicate how algorithms are part of many different computer applications
- Provide some knowledge themes such as relating complexity with algorithms
- Contrast complexities of different algorithmic strategies
- Explore some additional resources associated with algorithms
- Explain the purpose and role of algorithms in computer engineering

2. Basic Algorithmic Analysis

- Asymptotic analysis of upper and average complexity bounds
- Identifying differences among best, average, and worst case behaviors
- Big “O,” little “o,” omega, and theta notation
- Empirical measurements of performance
- Time and space tradeoffs in algorithms Using recurrence relations to analyze recursive algorithms

3. Algorithmic Strategies

- Brute-force/exhaustive search algorithms
- Greedy algorithms ; Divide-and-conquer
- At least one of: Backtracking, branch-and-bound, heuristics

4. Computing algorithms

- Simple numerical algorithms
- Sequential and binary search algorithms
- Sorting algorithms
- Hash tables, including collision-avoidance strategies

- Binary search trees
- Representations of graphs (adjacency list, adjacency matrix)
- Depth- and breadth-first traversals
- Shortest-path algorithms (Dijkstra's and Floyd's algorithms)
- Transitive closure (Floyd's algorithm)
- Minimum spanning tree (Prim's and Kruskal's algorithms)
- Topological sort

5. Distributed algorithms

- Concurrency
- Scheduling
- Fault tolerance

6. Algorithmic complexity

- Tractable and intractable problems
- Definition of the classes P and NP NP-completeness (Cook's theorem)
- Standard NP-complete problems
- Uncomputable functions
- The halting problem
- Implications of uncomputability
- Deterministic ;finite Automata (DFA)
- Non-deterministic finite Automata (NFA)
- Equivalence of DFA's and NFA's
- Context-free grammars
- Pushdown automata (PDA)

Learning Outcomes

- Design algorithms using the brute-force, greedy, and divide-and-conquer strategies.
- Design an algorithm using at least one other algorithmic strategy from the list of topics for this unit.
- Use and implement the fundamental abstract data types—specifically including hash tables, binary search trees, and graphs—necessary to solve algorithmic problems efficiently.
- Solve problems using efficient sorting algorithms, and fundamental graph algorithms, including depth-first and breadth-first search, single-source and all- pairs shortest paths, transitive closure, topological sort, and at least one minimum spanning tree algorithm.
- Demonstrate the following abilities: to evaluate algorithms, to select from a range of possible options, to provide justification for that selection, and to implement the algorithm in simple programming contexts.

Recommended and Reference Books

[1] Michael T. Goodrich, Roberto Tamassio, David Mount, 1995. *Data Structures and Algorithms in C++,* John Wiley, ISBN 0-471-20208-8

[2] Robert Secidgewick, 2001. *Algorithms in C++,* Addison – Wesley, ISBN 0201510596

[3] Nell Dale, 2003. *C++ Data Structures,* Jones and Bartlett Publishers

[4] Thomas H. Cormen,Clara Lee, Erica Lin, 2002. *Introduction to Algorithms,*

Second Edition, The MIT Press
[5] Dasgupta, C. H. Papadimitriou, and U. V. Vazirani, *Algorithms*