Course Title and Code	CSC315 -Algorithms Analysis and Design

I. Course Identification and General Information

Course Title	Algorithms Analysis and Design	Course Code	CS315	Pre-requisite	CS211
Department	Computer Science	Course Level	7	Credit Hours	3(3+0)

II. Course Description/Topics: The following course topics will be covered.

- Dynamic programming - Assembly-line scheduling - Matrix-chain multiplication - Longest common subsequence - Optimal binary search trees
- Greedy algorithms - An activity-selection problem - Elements of the greedy strategy - Huffman codes - Theoretical foundations for greedy methods - A task-scheduling problem
- Amortized analysis - Aggregate analysis - Accounting method - Potential method - Dynamic tables
- B-Trees - Definition of B-trees - Basic operations on B-trees - Deleting a key from a B-tree
- Binomial heaps - Binomial trees and binomial heaps - Operations on binomial heaps
- Fibonacci heaps - Structure of Fibonacci heaps - Mergeable-heap operations - Decreasing a key and deleting a node - Bounding the maximum degree
- Data structures for disjoint sets - Disjoint-set operations - Linked-list representation of disjoint sets - Disjoint-set forests - Analysis of union by rank with path compression
- Elementary graph algorithms - Representations of graphs - Breadth-first search - Depth-first search - Topological sort - Strongly connected components
- Minimum spanning trees - Growing a minimum spanning tree - Algorithms of Kruskal and Prim
- Single-source shortest paths - Bellman-Ford algorithm - Single-source shortest paths in directed acyclic graphs - Dijkstra's algorithm - Difference constraints and shortest paths - Proofs of shortest paths
- All-pairs shortest paths - Matrix multiplication - The Floyd-Warshall algorithm - Johnson's algorithm
- Maximum Flow - Flow networks - The Ford-Fulkerson method - Maximum bipartite matching - Push-relabel algorithms - The relabel-to-front algorithm

III. Course Outcomes: Summary of the main learning outcomes for students enrolled in the course.

By the end of the course, one should be able to:

- Use dynamic programming and greedy to solve an appropriate problem - Eventual optimality.
- Understand the heap property and the use of heaps as an implementation of priority queues.
- Solve problems using fundamental graph algorithms, including depth-first and breadth-first search.
- Solve problems using graph algorithms, including single-source and all-pairs shortest paths, and at least one minimum spanning tree algorithm.
- Be able to implement a string-matching algorithm.
- Use recursive backtracking to solve a problem such as navigating a maze.
- 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.

IV. Required Text

- Cormen, T., C. Leiserson, R. Rivest and Clifford Stein, "*Introduction to Algorithms*", 2nd Ed. MIT Press,2002

V. Reference

- Goodrich, M.T. and R. Tamassia, "*Algorithms Design, Foundations, Analysis and Internet Examples*", John Wiley & Sons, 2002