Catalog Description
Logical analysis of algorithms; reasoning about data structures and control structures; Topical material drawn from: seminumerical algorithms, searching and sorting, pattern matching, combinatorial algorithms, list processing, graphics.

Course Objectives
The basic goal of this course is to prepare a wiser consumer of data structures, algorithms, and heuristics. The course is in “Algorithms,” not “Algorithms appreciation.” So, the students are prepared to get your hands dirty in doing nontrivial analysis of some algorithms and applications.”

Prerequisites
CIS 275 (Discrete Math) and CIS 321 (Probability and Statistics).

Course Outcomes
After completing the course student should be able to do:

1. Foundations
   - Solve basic big-O [Abet (j)].
   - Derive asymptotic runtime bounds for reasonably straightforward pseudo-code with nested loop [Abet (j)].
   - Derive recurrence equations to express the runtime of recursive pseudo-code [Abet (j)].
   - Solve basic recurrence equations with the help of the Master Method [Abet (j)].
   - Do straightforward induction arguments over simple structures [Abet (j)].

2. Sorting
   - Know the standard comparison based sorts and their strengths and weaknesses [Abet (a, b, j)].
   - Know the standard linear-time sorts and the trade-offs involved in using them [Abet (a, b)].

3. Data Structures and their analysis
   - Know stacks, queues, linked lists, rooted trees; their standard implementations; and the analysis of these implementations [Abet (a, b, j)].
   - Understand the standard flavors of hash tables and the trade-offs involved in hashing [Abet (a, b)].
   - Understand both ordinary binary search trees and some variety of balanced binary search trees and their trade-offs [Abet (a, b, j)].

4. Standard Design and Analysis techniques
   - Know a variety of divide-and-conquer algorithms, how to analyze them, and how to apply the ideas to new situations [Abet (a, b, j)].
   - Know a variety of dynamic-programming algorithms, how to apply them via both memoization and tables, and recognize when a dynamic-programming approach might yield a good solution to a problem [Abet (a, b)].
Know a variety of greedy algorithms, know the basic ingredients of a greedy algorithm, and how to approach arguing the correctness of such algorithms [Abet (a, b)]

Know what is involved in an amortized analysis and be able to work through a banker’s method proof [Abet (a, b, j)]

5. Graph Algorithms

Know the basic representation of undirected and directed graphs. Know how to use depth-first and breath-first search of graphs and the analyses of these [Abet (a, b)]

Understand the shortest path problems and their applications [Abet (a, b)]

6. NP-Completeness

Understand P, NP, polynomial reduction, NP-hardness, and NP-Completeness. Know some standard NP-Complete problems and know the basics of an NP-hardness argument [Abet (a, b, j)]

Outcome Measurement
The course outcomes are measured through a combination of weekly homeworks, and in-class quizzes and exams.

Course Topics
Asymptotic notation, Recurrences, A review of basic data structures, Binary search trees, Dynamic programming, Heaps and binomial heaps, Greedy algorithms, Union-find forests, Depth-first search and its applications, Minimum spanning trees, Single-source shortest paths, All-pairs shortest paths, NP-completeness, and some selected topics

CAC Category Content
1 Data Structures
2 Algorithms
0 Software Design
0 Computer Organization & Architecture
0 Programming Languages