Course number and name: **CS 07340: Design and Analysis of Algorithms**
Credits and contact hours: 3 credits / 3 contact hours
Course Coordinator: Andrea Lobo
Instructional Materials: The Design and Analysis of Algorithms, Anany Levitin, 2012.

Specific course information

Catalog description: In this course, students will learn to design and analyze efficient algorithms for sorting, searching, graphs, sets, matrices, and other applications. Students will also learn to recognize and prove NP-Completeness.

Prerequisites: CS 07210 Foundations of Computer Science **and** CS 04222 Data Structures and Algorithms

Type of Course: ☒ Required ☐ Elective ☐ Selected Elective

Educational objectives for the course

1. **algorithm complexity.** Students have analyzed the worst-case runtime complexity of algorithms including the quantification of resources required for computation of basic problems.
    - ABET (6) Apply computer science theory and software development fundamentals to produce computing-based solutions.

2. **algorithm design.** Students have applied multiple algorithm design strategies.
    - ABET (6) Apply computer science theory and software development fundamentals to produce computing-based solutions.

3. **classic algorithms.** Students have demonstrated understanding of algorithms for several well-known computer science problems
    - ABET (6) Apply computer science theory and software development fundamentals to produce computing-based solutions.

4. **NP complete.** Students have written NP-completeness proofs.
    - ABET (6) Apply computer science theory and software development fundamentals to produce computing-based solutions.

Required list of topics to be covered

1. Brute Force and Exhaustive Search
2. Mathematical preliminaries
3. Complexity classes, Big O, upper and lower bounds
4. Worst-case algorithm analysis: worst, best, average; time, storage, communications, numbers of processors
5. Recurrence relations and analysis of recursive algorithms
6. Divide and Conquer algorithm design strategy

7. Dynamic Programming algorithm design strategy
8. Greedy algorithm design strategy
9. Backtracking, and Backtracking with Branch and Bound algorithm design
10. Hill climbing algorithm design strategy
11. Advanced Data Structures: Graphs, Heaps, Union-Find
12. NP-Completeness, complexity classes P and NP, Intractability
13. Classic problems, such as sorting, searching, MST, making change, Knapsack, SAT, Sudoku, string matching, Clique, Independent Set

Optional list of topics that could be covered

1. Approximation algorithms
2. Randomized algorithms
3. Balanced trees