Theory of Algorithms

CSCI 400

SEMESTER (fall 2016)

PROFESSOR/CLASS INFORMATION

Shieu-Hong Lin

(Course) Title: Theory of Algorithms
Location: LIB 141
E-Mail: shieu-hong.lin@biola.edu
Office Hours: M to Th 8:30-10:30am
MW 11:30am~1:00pm
Office Location: Grove 8
Meeting with Professor: **Make Appt via Email**

Term: fall, 2016
Dept. Website: http://csci.biola.edu
Course Code/#: CSCI 400
Class Days/Time: T Th 10:30-11:45pm
Credit Hours/Units: 3
Office Phone: 562 903-4741
Admin Assistant: Jerrianne Smith, x4741

COURSE DESCRIPTION

Various types of algorithms, analytic techniques for the determination of algorithmic efficiency, NP-complete problems, complexity hierarchies, and intractable problems.

COURSE ALIGNMENT WITH PROGRAM LEARNING OUTCOMES

This upper-division course is an elective for Computer Science majors. Successful completion of this course (see next section) will prepare students to demonstrate a developing proficiency toward the accomplishment of the PLO: modeling, analysis, and problem solving.

COURSE OBJECTIVES AND STUDENT LEARNING OUTCOMES

By the completion of this course including class participation, class assignments (referred to as "Tasks"), class readings and group interaction, the following objectives and learning outcomes will be assessed and demonstrated:

IDEA Objective #4: Developing specific skills, competencies, and points of view needed by professionals in the field most closely related to this course (Essential emphasis).

STUDENT LEARNING OUTCOMES (The learner will demonstrate that he or she has satisfactorily fulfilled IDEA Objective #4 by being able to):

- Able to understand the concepts of important algorithmic design paradigms such as divide and conquer, dynamic programming, branch-and-bound, and greedy algorithms.
- Able to utilize algorithmic paradigms to design efficient algorithms for reducing the computational time for problem solving in critical real world application domains.
- Able to prove the correctness of algorithms, analyze the computational complexity of algorithms, and implement the algorithms as functional computer programs.
- Understand the essence of basic complexity classes such as *N*, *NP*, *NP-Complete*, *Pspace-Complete*, the question of whether *P* equals *NP* and its implication, and the use of approximation algorithms and randomized algorithms to cope with time complexity.

REQUIRED TEXTS

J. Kleinberg and E. Tardos. *Algorithm Design*, Addison Wesley, 2005.

LEARNING TASKS (Assignments) & ASSESSMENT (Grading)

Description and Weighting of Assignments:

Task 1: Weekly Reading and Progress Report (15 assignments)

Weighting: 10%
Possible Points: 4 points each.
Report the (i) a numerical estimate of the amount of time he/she spent for the reading, (ii) a numerical percentage regarding the percentage of stuff in the reading actually read and understood, and (iii) whether the student has come to the class this week.

Reflection on the reading (2 points):
The student need to put down 1 to 2 paragraphs of his/her thoughts about the reading such as new insight you gained, interesting things encountered, questions of things you don't understand, and so forth.

Assessment:
For the effort part,
the student is expected to **(i)** have attended the class this week at least once (0.5 point), and **(ii) have either** gained a good understanding of **80% or more of the contents or** have spent at least **three hours** in the reading (1.5 points).

For the reflection part,
the student is expected to show substantial evidence of understanding or effort of trying to understand the contents in the reading.

Task 2: Homework and Programming Assignments

Weighting: 50%
Possible Points: 6 points each for programming assignments. 1-6 points for each Lab and Homework depending on the requirement in it.

Integrity rules for programming assignments:

- **Peer discussion is encouraged**: Peer discussion is encouraged to cultivate an open learning environment in the class, but you should carefully read the guidelines below to avoid any dishonest behavior and never step over the guidelines explicitly described in the following.
- **Never use code written by others**: Any copy-and-paste of code from other people's programs or from websites is viewed as cheating and you will get 0 points for the assignment.
- **Never circulate your code to others**: Peer discussion of code shown on the screen is acceptable for debugging purpose and for explanation of ideas. But you should never pass around your code (electronically or on paper) to others except for the TA and the instructor. Violating this rule is viewed as cheating in the class and the provider will receive 0 points for the assignment.
- **Never provide false or exaggerated results of test cases**: You need to describe results of test cases in the self-evaluation report**.** Providing false or exaggerated results of test cases in the report is viewed as cheating and you will receive 0 points for the assignment.
- **Demonstrate the credibility of your authorship of the work**: When you submit your code as your own work for points, you should make sure that you are able to explain your code and reconstruct your code from scratch without any outside help when requested. If you are not able to do that on your own when requested, you will get 0 points for the assignment and there will be an investigation.
- **Consequence of cheating in the class**: Cheatings end in 0 points for the assignments followed by discipline actions described in the student handbook.

Assessment: The student needs to submit (all related .cpp and .h files) together with a self-evaluation report. The self-evaluation report should describe results from sufficient test cases that are verified by a peer reviewer. We'll grade each programming assignment in a 0-6 scale based on the following rubric.

0. Nothing done **or missing the self-evaluation report or missing the integrity review** in the report.
1. Source code is completed but the code fails to compile successfully.
2. Source code can compile and do something required, but has serious bugs or miss a couple of key features.
3. Source code can compile and do most of the features required, but has many minor bugs or miss a key required feature.
4. Source code can compile and do all the features required, nearly fully functional, only a couple of minor bugs.
5. Source code can compile and do all the features required, fully functional, no bugs.
6. In addition to the points received above, get one more point if

   a. **the self-evaluation report contains sufficient descriptions of test cases used (0.25 point)**, and
   b. **the self-evaluation report indicates the results of the test cases were verified by a peer reviewer (0.25 point)**, and
   c. **the source code is well indented and commented to make it visually very readable (0.5 point)**.

Task 3: Exams (Quizzes, midterm and final exams)

Weighting: 40%

Description: The exams have both written components which mainly test the conceptual understanding of the theory of algorithms, and the programming components which test the skills of implementation.

Assessment: The written component will be graded based on the answers provided while the programming component will be graded based on the same rubric provided for the weekly programming assignments.

CLASS INFORMATION

1. Class Attendance:

Attendance: You are expected to attend the class regularly. Missing the class may seriously hamper your understanding of many key concepts and programming skills critically needed in your programming assignments. Class attendance is counted toward points for the weekly reading report.

2. Turning in Assignments:

Assignments are expected to be electronically submitted under the Canvas system. Due dates are all on Mondays. The submission link under Canvas may remain open for 2 more days after the due date as grace period.

3. Late Policy:

1 point will be deducted for late submission within 2 days of the due date while the submission link is still open. **You will receive no points after the submission on canvas is closed** unless it is something like a serious health issue with statements from the doctor as proof.

4. Computation of Final Grade:

Weekly Progress Report	10 %
Homework and Programming Assignments	50%
Exams	40 %
Total	**100%**

5. Final grades will be awarded on the following point system:

A	93%
A-	90%
B+	87%
B	84%
B-	80%
C+	77%
C	74%

C-	70%
D+	67%
D	64%
D-	60% to pass class

GENERAL INFORMATION

1. The GPA System used by the University Registrar's Office is:

A = 4.0	B = 3.0	C = 2.0	D = 1.0
A- = 3.66	B- = 2.66	C- = 1.66	D- = 0.66
B+ = 3.33	C+ = 2.33	D+ = 1.33	F = 0.0

2. Method of Instruction:

The following methods of instruction will be included in this course:

1. Lecture
2. Written Reports
3. Programming Assignments
4. Labs
5. Reading

3. Posting of Grades:

Grades for individual assignments will be posted under Biola's Canvas system. To access the records online, log on to http://canvas.biola.edu/ to make sure the records are accurate.

Tentative Schedule

- Week 1 Overview of Algorithmic Paradigms
- Weeks 2-3 Asymptotic notation & complexity analysis
- Weeks 4-5 More on greedy algorithms
- Weeks 6-8 More on divide and conquer
- Weeks 9-11 More on dynamic programming
- Weeks 12-13 More on applications
- Weeks 14-15 P, NP, and PSPACE and Approximation algorithms
- Final week Final Exam