School of
Computing and
Information Systems

Course Code:	**CS202**
Course Name:	**Design and Analysis of Algorithms**
When was the course design document last verified by the Course Manager:	**January 2024**

NOTE: The information given in this document is for reference only; the updates given during the class sessions and/or eLearn will supersede the information given in this document.

1. Synopsis

This course teaches students how to solve problems by designing efficient algorithms, and how to analyze the efficiency of algorithms. Students' earlier programming experiences, mathematics (discrete maths and linear algebra), and the mastery of data structures are necessary for this course. The materials as well as the assignments expect students to have proficient programming skills in Python.

Students will learn:

- The different paradigms of algorithm design such as greedy, divide and conquer, and dynamic programming.
- The analysis on the complexity of algorithms.
- Limits of algorithm design via the study of intractability including the reductions of given problems to known problems and the knowledge of NP completeness and NP hardness.
- More modern algorithm design concepts such as approximation to achieve more effective problem solving and more efficient solutions.

This course will go into the theoretical underpinnings of efficiency, algorithm correctness, and how algorithm design has a basis in identifying mathematical properties of the problem.

2. Prerequisites/Co-requisites

Prerequisite(s): CS201 Data Structures and Algorithms

Though not listed explicitly as prerequisites, the following backgrounds would be assumed:

- CS students should have cleared CS104 Mathematical Foundations of Computing, as the course assumes familiarity with mathematical concepts such as proofs.

3. Course Areas

IT Solution Development Core
IS Depth Electives
IS DCS Track
Adv Business Technology Major
Business Options
Econ Major Rel/Econ Options
Social Sciences/PLE Major-rel

4. Course Objectives

Upon finishing the course, a student will be able to:

- Understand the different paradigms of algorithm design
- Map real world problem to known computational problems
- Design and develop algorithms to solve computational problems
- Analyze the complexity and the correctness of a given algorithm

5. Competencies

- Understand the efficiency notations of functions, e.g., Big O, Omega, Theta
- Analyze the correctness and the complexity of an algorithm
- Design and develop algorithms to solve computational problems within the scope of this course
- Understand intractability and the scopes of NP completeness and NP hardness

6. Teaching Staff

Faculties:

- DAI Bing Tian <btdai@smu.edu.sg>
- LAU Hoong Chuin <hclau@smu.edu.sg>

Instructors:

- LIU Ziyuan <ziyuanliu@smu.edu.sg>

7. Course Assessments

Assessment Categories	**Weightage (%)**
Class Attendance & Participation	15
Assignments	36
Paper Presentations (Group)	9
Final Exam	40
Total	100

8. Course Assessment Details

1) Class Attendance and Participation
    - Class attendance – 5%
    - Class participation – 5%
    - Forum participation – 5%

2) Assignments
    - There will be three individual coding-based take-home assignments.

- Code will be evaluated through an online judge on the correctness and efficiency.

3) Group reading presentations
   - A group assignment in this course requires teams of 4 to 5 students to delve into an algorithmic problem based on a given research paper. The task is to investigate and analyze the problem, subsequently presenting their findings and the paper's content to their peers, thus demonstrating their grasp of the learning in this course.

9. Lesson Plan

Session	Topics	Assessment
1	Introduction and Recursion ● Recursion ● Python Programming (revisit) ● Mergesort	
2	Recurrences and Master theorem ● Analysis of Mergesort ● Big O notation ● Recurrence and substitution method ● Master theorem	Assignment 1 Release
3	Divide and Conquer and Dynamic Programming I ● Quicksort ● Selection ● Strassen's algorithm (optional) ● Rod cutting	
4	Dynamic Programming II ● Matrix chain multiplication ● Coin exchange ● Knapsack problem ● Maximum sub-matrix sum (optional)	
5	Dynamic Programming III and Greedy Algorithm ● Edit distance ● Interval scheduling ● Fractional Knapsack problem ● Huffman coding	Assignment 2 Release
6	Graph Algorithms I ● Graph DFS and BFS (Revisit) ● Connectivity and Kosaraju's algorithm ● Articulation points and bridges (optional)	Assignment 1 Due
7	Graph Algorithms II ● Union find ● Minimum spanning tree and greedy algorithm (Revisit) ● Shortest paths and dynamic programming	
8	**RECESS WEEK**	Assignment 3 Release

9	Flow and linear programming • Flow and bipartite matching • Simplex algorithm • Integer programming	Assignment 2 Due
10	NP Completeness and reducibility • NP Completeness • Reducibility • The clique problem	Release Group Reading materials
11	Approximation Algorithms • Knapsack problem revisited • Load balancing • Rectangle packing problem	
12	Search and Heuristics • Local search • TSP using heuristics • Simulated annealing	Assignment 3 Due
13	**Group reading presentation**	Presentation Slides Due
14	**STUDY WEEK**	
15	**FINAL EXAM**	

10. Resources

Main Reading:

- **Introduction to Algorithms** (3rd edition): Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. MIT Press 2009.
- **The Algorithm Design Manual** (2nd edition): Steven Skiena. Springer 2008.
- **Algorithm Design**: by Jon Kleinberg and Eva Tardos. Pearson 2005.
- **Algorithms**, S. Dasgupta, C. H. Papadimitriou, and U. V. Vazirani: http://algorithmics.lsi.upc.edu/docs/Dasgupta-Papadimitriou-Vazirani.pdf

11. Other Important Information

Academic Integrity

All acts of academic dishonesty (including, but not limited to, plagiarism, cheating, fabrication, facilitation of acts of academic dishonesty by others, unauthorized possession of exam questions, or tampering with the academic work of other students) are serious offences. All work (whether oral or written) submitted for purposes of assessment must be the student's own work. Penalties for violation of the policy range from zero marks for the component assessment to expulsion, depending on the nature of the offense.

When in doubt, students should consult the instructors of the course. Details on the SMU Code of Academic Integrity may be accessed at https://oasis.smu.edu.sg/Pages/DOS-WKLSWC/UCSC.aspx.

Copyright Notice

Accessibility and Accomodations

SMU strives to make learning experiences accessible for all. If you anticipate or experience physical or academic barriers due to disability, please inform SCIS Dean’s Office immediately. You are also welcome to contact the SMU Disability Support Services (dss@smu.edu.sg) if you have questions or concerns about academic provisions.

Emergency Preparedness for Teaching and Learning (EPTL)

As part of emergency preparedness, instructors may conduct lessons online via either the Zoom or WebEx platform during the term, to prepare students for online learning. During an actual emergency, students will be notified to access the Zoom or WebEx platform for their online lessons. The class schedule will mirror the current face-to-face class timetable unless otherwise stated.