Lingnan University

Course Syllabus

Course Title	:	Algorithms
Course Code	:	CDS3108
Recommended Study Year	:	Year 3
No. of Credits	:	3
Mode of Tuition	:	Sectional Approach
Class Contact Hours	:	3 hours per week
Category	:	Major Elective
Discipline	:	Nil
Prerequisite(s)	:	CDS2003 Data Structures and Object-Oriented Programming
Co-requisite(s)	:	Nil
Exclusion(s)	:	Nil
Exemption Requirement(s)	:	Nil

Brief Course Description:

Algorithms form the backbone of computer science, enabling efficient problem-solving and powering applications from data processing to artificial intelligence. This course provides a rigorous introduction to the design, analysis, and implementation of algorithms. Students will learn fundamental techniques such as divide-and-conquer, dynamic programming, greedy methods, and graph algorithms, as well as strategies for analysing algorithmic complexity using Big-O notation. Students will explore how algorithmic choices impact performance and scalability, and they will gain experience implementing algorithms in a programming language. Topics include sorting and searching, graph traversal, shortest paths, and optimization problems, along with discussions on NP-completeness and approximation algorithms. Through problem-solving exercises and programming assignments, students will develop the ability to design efficient algorithms for real-world challenges.

Aims:

This course aims to:

1. Introduce students to the core principles of algorithm design and analysis, emphasizing their critical role as the computational engine for data transformation, modelling, and knowledge discovery.

2. Provide a foundational understanding of core algorithmic paradigms and rigorous techniques for analysing correctness and complexity, bridging theoretical concepts with practical implementation.

3. Enable students to model real-world problems such as pattern matching, network analysis, and optimization using appropriate algorithmic frameworks and to evaluate solution efficiency.

4. Equip students with the skills to implement, adapt, and evaluate classical algorithms.

Learning Outcomes (LOs)

Upon successful completion of this course, students will be able to:

1. Analyse worst-case running times and space complexity of algorithms using asymptotic analysis. (PILO1)
2. Design algorithms using standard paradigms (divide-and-conquer, greedy methods, dynamic programming, graph traversal) and argue for their correctness. (PILO1, PILO2)
3. Implement and apply fundamental algorithms to problems such as clustering, similarity search, ranking, and network analysis. (PILO2, PILO4)
4. Recognize and model classical computational problems such as shortest path, minimum spanning tree, sequence alignment, and knapsack. (PILO4)
5. Evaluate trade-offs between different algorithmic approaches for a given problem, considering constraints of data scale and structure. (PILO8)

Indicative Contents:

Foundations of Algorithmic Analysis and Design

Introduction to asymptotic notation, recurrence relations, and techniques for rigorously analysing the time and space complexity of algorithms.

Core Algorithmic Paradigms for Data-Centric Problems

Design and analysis of fundamental algorithms based on core paradigms such as divide-and-conquer, greedy methods, and dynamic programming for solving optimization problems such as sorting, scheduling, sequencing, and knapsack.

Graph Algorithms and Network Flows

Model complex data relationships using graph structures and analyse algorithms for traversal, shortest paths, and network flow to solve problems in network analysis and connectivity.

Intractability and Approximation Algorithms

Introduction to the theory of NP-completeness and the design of approximation algorithms for computationally hard optimization problems.

Teaching Methods:

The teaching and learning activities for the course will include lectures, laboratory sessions, and group projects. Lectures will introduce foundational principles, algorithmic paradigms, and techniques for mathematical analysis and correctness proofs (LO1, LO2, LO4). Laboratory sessions will provide hands-on programming experience, allowing students to implement core algorithms, test them on datasets, and empirically analyse their performance characteristics (LO2-LO3). The group project will involve selecting a problem, researching and applying appropriate algorithmic solutions, and evaluating the trade-offs between different design choices in terms of scalability and effectiveness (LO2-LO5).

Measurement of Learning Outcomes:

Assessment / **LOs**	Class Attendance and Participation	Assignments	Midterm Exam	Group Project
	(15%)	(30%)	(25%)	(30%)
LO1	✓	✓	✓	
LO2		✓	✓	✓
LO3		✓	✓	✓
LO4	✓	✓	✓	✓
LO5	✓			✓

1. Class attendance and participation assess students' class attendance, engagement in learning activities, and their understanding of fundamental algorithmic concepts introduced in lectures (LO1), as well as their contribution of insights during discussions and completion of in-class exercises on core concepts (LO4, LO5).
2. Written assignments and programming assignments of implementing algorithms are designed to assess students' ability to analyse algorithmic complexity (LO1), design solutions using standard paradigms (LO2), implement and apply algorithms to practical problems (LO3), and recognize classical problem models (LO4).
3. The midterm exam evaluates students' independent ability to analyse worst-case time and space complexity (LO1), design, analyse, and apply algorithms (LO2, LO3), and recognize and model classical computational problems (LO4).
4. The group project assesses students' capability to collaboratively design an algorithmic solution (LO2), implement and apply it to a data science problem (LO3), model the problem within a classical framework (LO4), critically evaluate trade-offs between different approaches (LO5), and clearly communicate design and technical decisions through presentations and reports.

Assessment:

Class Attendance and Participation	15%
Assignments	30%
Midterm Exam	25%
Group Project	30%
Total	100%

Required/Essential Readings:

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to algorithms*. 3rd ed. MIT press.

Recommended/Supplementary Readings:

1. Miller, B., & Ranum, D. (2011). *Problem Solving with Algorithms and Data Structures Using Python*. 2nd ed. Franklin, Beedle & Associates.

Important Notes:

(1) Students are expected to spend a total of 9 hours (i.e. 3 hours of class contact and 6 hours of personal study) per week to achieve the course learning outcomes.
(2) Students shall be aware of the University regulations about dishonest practice in course work, tests and examinations, and the possible consequences as stipulated in the Regulations Governing University Examinations and Course Work. In particular, plagiarism, being a kind of dishonest practice, is "the presentation of another person's work without proper acknowledgement of the source, including exact phrases, or summarised ideas, or even footnotes/citations, whether protected by copyright or not, as the student's own work". Students are required to strictly follow university regulations governing academic integrity and honesty.
(3) Students are required to submit writing assignment(s) using Turnitin.
(4) To enhance students' understanding of plagiarism, a mini-course "Online Tutorial on Plagiarism Awareness" is available on http://pla.ln.edu.hk/.

Students must adhere to the University's guidelines and practices when using Generative Artificial Intelligence (GAI) tools. The official documents "Guidelines for Using GAI Tools at Lingnan University" and "Best Practices for Ethical and Responsible Use of GAI Tools in Course Assessments" can be found here: https://www.ln.edu.hk/tlc/generative-artificial-intelligence/gai-guidelines-best-practices.

Lingnan University

Course Rubrics

Rubric for Class Attendance and Participation (15%)

Criteria	Excellent (90-100)	Good (70-89)	Pass (50-69)	Failure (0-49)
Attendance and Participation (50%)	Student attends over 90% of all classes.	Student attends at least 80% of all classes.	Student attends at least 70% of all classes.	Student attends less than 70% of the classes.
Class Participation (50%)	Actively engages in all discussions and in-class exercises. Demonstrates clear understanding of algorithmic concepts. Frequently offers insightful questions and analyses, elevating class discussion.	Participates in most discussions and completes most exercises. Understand the course content fairly well, answer questions and communicate with teachers occasionally.	Minimal participation; completes basic exercises only. Can barely understand the course content, answer questions and communicate with teachers barely.	Rarely participates or engages with in-class activities. Cannot understand the course content, does not answer questions and communicate with teachers.

Rubric for Assignments (30%)

Criteria	Excellent (90-100)	Good (70-89)	Pass (50-69)	Failure (0-49)
Complexity Analysis & Theoretical Justification (50%)	Provides a precise and correct asymptotic analysis (time/space). Correctly argues for algorithm correctness and optimality where required.	Analysis is largely correct but may have minor inaccuracies or omissions. Correctness argument is present but may lack some rigor.	Analysis is attempted but contains major errors or misunderstandings. Justification is weak or incomplete.	Little to no meaningful analysis or justification is provided.
Algorithm Correctness & Implementation (30%)	Algorithm is successfully implemented for all test cases. Code is efficient and idiomatic.	Algorithm is mostly correct, with minor bugs or edge-case errors. Implementation is functional.	Algorithm has significant logical flaws or incomplete functionality but shows a basic grasp of the approach.	Implementation is largely incorrect, non-functional, or does not address the core problem.
Code Quality & Documentation (20%)	Code is exceptionally clean, well-structured, and thoroughly documented with insightful comments.	Code is readable and adequately documented. Some minor issues with style or comments may exist.	Code is poorly organized or sparsely documented, making it difficult to follow.	Code is unstructured, undocumented, and essentially unreadable.

Rubric for Midterm Exam (25%)

Criteria	Excellent (90-100)	Good (70-89)	Pass (50-69)	Failure (0-49)
Algorithmic Knowledge & Application (30%)	Demonstrates comprehensive and accurate understanding of all core paradigms (e.g., D&C, DP, Greedy). Correctly models given problems as classical algorithmic tasks (e.g., shortest path, knapsack) and applies appropriate techniques flawlessly.	Shows solid understanding of most paradigms and can correctly model standard problems. Application of techniques is largely correct but may contain minor errors in setup or edge cases.	Demonstrates basic, recognition-level knowledge of paradigms. Problem modelling is attempted but may be incomplete or partially misapplied, leading to flawed solution approaches.	Shows significant misunderstanding of core algorithmic concepts. Fails to correctly model problems or apply relevant techniques.
Problem-Solving & Correctness (30%)	Provided solutions are completely correct, elegant, and efficiently designed. Proofs of correctness or invariants are clearly stated and logically rigorous.	Solutions are largely correct and functional. Proofs or justification of approach are present but may lack some clarity or completeness.	Solutions contain significant errors or inefficiencies but show a basic, understandable approach. Justification is weak or missing.	Solutions are incorrect, irrelevant, or absent. Shows no viable problem-solving strategy.
Analysis & Critical Thinking (30%)	Asymptotic analysis is precise and correctly derived. Trade-offs between different potential approaches are insightfully evaluated. Recurrences are solved accurately and justified.	Analysis is mostly correct with minor errors. Some comparison of approaches is included but may lack depth. Recurrence solutions are mostly accurate.	Analysis is attempted but flawed or superficial. Little to no comparison of alternatives. Errors in solving recurrences.	No meaningful analysis provided. Fails to engage with complexity or trade-off concepts.
Communication &	Solutions are presented	Solutions are generally	Solutions are	Work is illegible,

Clarity (10%)	with exceptional clarity, structure, and mathematical notation. Explanations are concise, logical, and easy to follow.	clear and well-organized. Explanations are adequate but may occasionally be verbose or lack precision.	disorganized or confusing in parts. Explanations are unclear, hindering the evaluation of the underlying idea.	unstructured, or so poorly communicated that the intended solution cannot be discerned.

Rubric for Group Project (30%)

Criteria	Excellent (90-100)	Good (70-89)	Pass (50-69)	Failure (0-49)
Problem Analysis & Algorithmic Design (35%)	Problem is expertly modelled using formal computational frameworks. Algorithm selection/design is sophisticated, creatively adapted, and clearly justified against multiple alternatives with deep insight into trade-offs (e.g., time vs. space, accuracy vs. complexity, exact vs. approximate).	Problem is well-modelled. Algorithm choice is appropriate and logically justified, with a clear rationale for its selection over other obvious candidates. Trade-offs are identified and discussed adequately.	Problem modelling is simplistic or contains flaws. Algorithm choice is standard but justification is superficial, showing limited analysis of alternatives. Trade-offs are mentioned but not deeply analysed.	Problem is poorly understood or incorrectly modelled. Algorithm choice is inappropriate, trivial, or lacks any meaningful justification. No meaningful analysis of trade-offs.
Implementation & Empirical Evaluation (35%)	Implementation is robust, efficient, and meticulously tested on relevant datasets. Evaluation is comprehensive, using appropriate metrics and	Implementation is complete and functional. Evaluation is performed with relevant metrics but may lack some depth in analysis (e.g., missing	Implementation is buggy, incomplete, or poorly structured. Evaluation is superficial, uses inappropriate metrics, or lacks	Implementation is non-functional or trivial. Little to no empirical evaluation is conducted.

	controls to critically analyse performance, scalability, and boundary conditions.	scalability tests or comparative baselines).	meaningful interpretation of results.	
Presentation (20%)	Final presentation/report demonstrate exceptional clarity and logical rigor. Technical elements (notation, pseudocode, graphs) are used precisely. Scholarly conventions are followed.	Report/presentation is clear, organized, and effectively communicates the project's methodology and findings.	Report/presentation is disorganized, unclear, or omits key details about methods or results.	Report/presentation is poorly prepared, incoherent, or fails to document the work.
Collaboration & Contribution (10%)	Peer feedback indicates leadership, initiative, and technical contribution across all project phases (design, implementation, analysis, writing).	Peer feedback indicates reliable, consistent, and constructive contributions to the group's work.	Peer feedback indicates passive participation with minimal or uneven contribution to the final outcome.	Peer feedback indicates a lack of participation, poor communication, or negative impact on the team's progress.