Article

Exploration of Teaching Reform for Data Structures and Algorithms Course Empowered by AI

Huanhuan Fang[1,*], Bingke Wei[1] and Bingjie Wei[1]

[1] School of Computer Science and Artificial Intelligence (Industrial Software), Guangdong University of Science and Technology, Dongguan, China
* Correspondence: Huanhuan Fang, School of Computer Science and Artificial Intelligence (Industrial Software), Guangdong University of Science and Technology, Dongguan, China

Abstract: With the rapid development of artificial intelligence technologies, AI-empowered education has become an important direction for teaching reform in higher education institutions. Based on an analysis of the current teaching situation and key challenges of the Data Structures and Algorithms course, this paper proposes an AI-enabled teaching reform model for the course. The proposed model establishes a four-in-one AI-enabled teaching reform framework consisting of "Intelligent Diagnosis–Personalized Learning–Intelligent Learning Assistance–Multi-dimensional Evaluation." It systematically investigates intelligent instructional content development, AI teaching assistant system construction, interactive algorithm visualization teaching, and the design of a multi-dimensional evaluation system. Specific reform measures and implementation pathways are also presented. The findings of this study provide important references for promoting teaching reform in computer-related courses and constructing intelligent curriculum systems.

Keywords: artificial intelligence; data structures; algorithms; teaching innovation; personalized learning; intelligent tutoring

1. Introduction

With the rapid advancement of large language model technologies, exemplified by systems demonstrating advanced conversational and reasoning capabilities, artificial intelligence is increasingly reshaping pedagogical practices and instructional methodologies across higher education [2]. National educational policy frameworks have emphasized the strategic integration of intelligent technologies into teaching and learning processes, advocating for evidence-informed approaches that enhance instructional effectiveness, broaden access to high-quality learning resources, and support adaptive learning pathways. In this evolving landscape, educators are actively exploring how AI tools can be systematically incorporated into course design, delivery, and assessment—particularly in foundational disciplines where conceptual abstraction and procedural complexity pose persistent challenges for learners.

Data Structures and Algorithms serves as a cornerstone course in computer science curricula, essential for developing rigorous computational thinking, systematic problem decomposition skills, and proficiency in designing efficient software solutions [3]. Its curriculum integrates theoretical foundations—including asymptotic analysis, graph theory, and recursive paradigms—with hands-on implementation tasks requiring precise coding discipline and debugging acumen. However, the inherent abstraction of data abstractions (e.g., trees, hash tables, heaps), the layered dependencies among algorithmic concepts, and the necessity of bridging formal logic with executable code create significant cognitive load for learners. Conventional instructional strategies often struggle to accommodate heterogeneous learner profiles: students entering with varying levels of prior programming exposure, mathematical maturity, or spatial-reasoning aptitude may experience divergent rates of conceptual uptake. Furthermore, instructor capacity

Received: 11 May 2026
Revised: 23 June 2026
Accepted: 04 July 2026
Published: 07 July 2026

constraints frequently limit timely formative feedback during laboratory sessions; automated grading tools often assess only syntactic correctness or output matching, neglecting process-oriented competencies such as algorithmic intuition, trade-off analysis, and iterative refinement. Assessment mechanisms remain predominantly summative and exam-centric, offering limited insight into longitudinal skill development, collaborative reasoning, or metacognitive awareness—dimensions critical to professional readiness in computing fields.

This study develops and validates an AI-enhanced pedagogical framework specifically tailored for Data Structures and Algorithms instruction. Grounded in learning science principles and human-centered AI design, the framework addresses four interdependent dimensions: first, intelligent content generation that dynamically adapts explanations, analogies, and illustrative examples based on real-time learner interaction patterns and knowledge state modeling; second, an AI-powered teaching assistant capable of delivering context-aware scaffolding—such as hint sequencing, misconception detection, and stepwise solution guidance—while preserving pedagogical intent and avoiding solution disclosure; third, interactive algorithm visualization modules that enable multi-perspective exploration (e.g., structural, temporal, and operational views) synchronized with live code execution and parameter manipulation; and fourth, a holistic evaluation architecture integrating automated code analysis, peer-assisted rubric scoring, reflective self-assessment prompts, and longitudinal competency mapping. The resulting model supports scalable personalization without compromising conceptual rigor, fosters active sense-making over passive consumption, and establishes measurable benchmarks for evaluating both learner outcomes and instructional efficacy [4]. By operationalizing these components within authentic classroom settings, the study contributes actionable insights for advancing intelligent, learner-centered computing education.

2. Analysis of Current Teaching Situation and Key Challenges

Data Structures and Algorithms is characterized by a high degree of abstraction and systematicity. Core concepts—including linked lists, stacks, queues, trees, and graphs—are fundamentally logical constructs rather than concrete physical entities, which presents substantial cognitive demands on learners [3]. Students often struggle to visualize dynamic memory allocation, pointer manipulation, and recursive traversal patterns without robust foundational experience in programming fundamentals. Moreover, the subject exhibits strong conceptual interdependence: mastery of foundational topics such as memory management, recursion, and modular design directly influences comprehension of advanced structures like balanced trees, hash tables, and graph algorithms. For instance, insufficient familiarity with recursive problem decomposition and pointer-based memory modeling frequently impedes students' ability to grasp binary tree operations or implement depth-first search effectively. Equally important is the course's dual emphasis on theoretical reasoning and practical implementation—students must not only articulate algorithmic correctness and termination conditions but also translate abstract logic into executable code, rigorously analyze time and space complexity using asymptotic notation, and debug non-trivial implementations under constrained computational environments.

To gain a deeper understanding of the current teaching situation of the Data Structures and Algorithms course, this study analyzes students' learning conditions based on recent teaching practices across multiple institutions. The findings reveal several persistent pedagogical challenges [4]. First, considerable heterogeneity exists among learners in prior programming exposure, mathematical maturity, analytical reasoning capacity, and self-regulated learning strategies; this diversity makes uniform pacing and content delivery ineffective for inclusive learning outcomes. Traditional lecture-based instruction tends to prioritize content coverage over active knowledge construction, limiting opportunities for real-time feedback, peer discussion, and guided inquiry. Second, laboratory sessions often fail to cultivate algorithmic thinking autonomy: many students replicate template solutions without engaging in problem decomposition, design iteration,

or critical evaluation of alternative approaches, thereby weakening their capacity for adaptive problem solving. Third, formative assessment mechanisms remain underdeveloped—reliance on infrequent assignments, summative exams, and sporadic oral questioning provides fragmented and delayed insights into individual conceptual gaps, misconceptions, or procedural errors. Fourth, evaluation frameworks continue to emphasize terminal performance over longitudinal growth; final examinations dominate grading weightings, while continuous assessment components—such as code walkthroughs, reflective journals, collaborative debugging tasks, and incremental project milestones—constitute only a minor portion of overall evaluation. This imbalance inadvertently reinforces outcome-oriented learning behaviors and overlooks essential competencies including computational thinking fluency, resilience in debugging, and metacognitive awareness of learning progress.

3. Design of AI-Enabled Teaching Reform for Data Structures and Algorithms Course

3.1. Overall Framework Design

To address the pedagogical challenges identified in the instruction of the Data Structures and Algorithms course—such as heterogeneous student preparedness, abstract conceptual barriers, and limited real-time feedback mechanisms—this study proposes a holistic, AI-integrated teaching framework structured around four interdependent functional pillars: Intelligent Diagnosis, Personalized Learning Pathways, Intelligent Learning Assistance, and Multi-dimensional Evaluation. This integrated architecture is designed to support adaptive, evidence-informed instructional practices while maintaining rigorous academic standards and aligning with contemporary educational technology principles. The framework operates as a closed-loop system wherein diagnostic insights inform personalization, assistance tools scaffold learning progression, and evaluation data feed back into instructional refinement. As illustrated in Figure 1, each component is systematically coordinated to ensure coherence across the entire teaching–learning–assessment cycle.

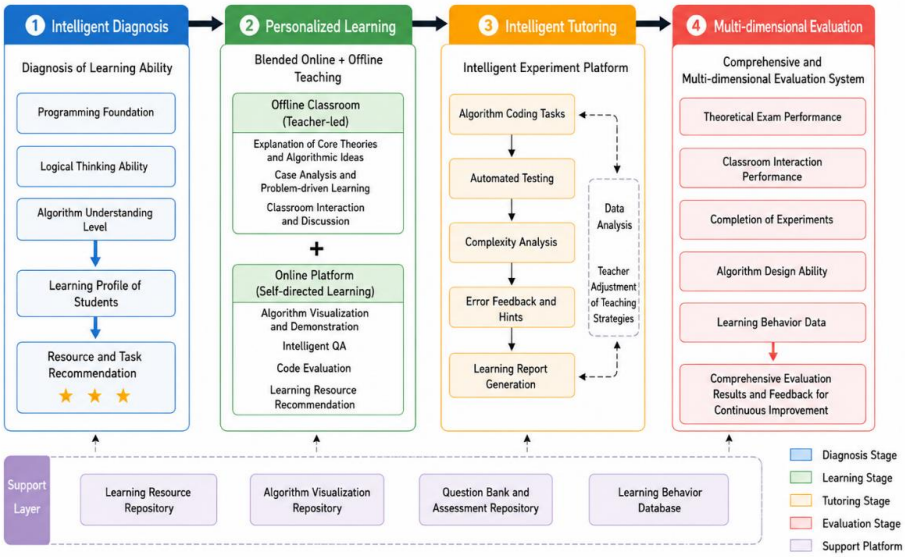

Figure 1. Framework for AI-Empowered Data Structures and Algorithms Teaching Reform

At the initial stage of course implementation, the learning capability diagnosis module conducts a comprehensive baseline assessment of students' foundational competencies. This includes evaluating prior programming experience through syntactic and semantic code analysis, assessing logical reasoning capacity via structured problem-solving tasks, and gauging algorithmic conceptual understanding using validated cognitive task models. The resulting learner profiles are dynamically updated throughout

the semester to reflect evolving competencies. Based on these profiles, the system generates differentiated learning pathways—recommending tiered resources such as annotated pseudocode examples, interactive simulation modules, scaffolded coding exercises, and concept reinforcement quizzes—tailored to individual readiness levels and preferred learning modalities.

During the teaching implementation stage, a blended pedagogical model integrates synchronous face-to-face instruction with asynchronous digital engagement [5]. In physical classroom sessions, instructors emphasize conceptual clarity, theoretical rigor, and metacognitive strategy development—utilizing carefully curated case studies, incremental algorithm walkthroughs, and collaborative problem decomposition activities to deepen comprehension. Concurrently, the online learning platform delivers algorithm visualization tools that render dynamic step-by-step execution traces, context-aware intelligent tutoring responses to student queries, automated code correctness and style validation, and adaptive resource curation based on real-time interaction analytics. These functionalities collectively foster self-regulated learning, promote reflective practice, and extend instructional reach beyond scheduled class hours.

During the practical teaching stage, an intelligent laboratory platform serves as the primary environment for hands-on algorithm implementation and analysis. Upon submission of programming solutions, the system executes multi-layered automated assessment—including functional correctness testing across diverse input sets, asymptotic time and space complexity estimation using static and dynamic analysis techniques, syntactic and semantic error classification, and constructive feedback generation aligned with Bloom's taxonomy levels. Comprehensive learning reports synthesize performance trends, common misconception patterns, and skill progression metrics [5]. These actionable insights empower instructors to implement responsive pedagogical interventions—such as targeted mini-lessons, peer-led debugging workshops, or adaptive assignment sequencing—thereby enhancing instructional agility and responsiveness.

During the teaching evaluation stage, a comprehensive and multi-dimensional evaluation system is implemented to capture the full spectrum of student development. Rather than relying solely on summative examinations, this system aggregates longitudinal data from multiple authentic assessment contexts: participation quality in Socratic classroom dialogues, consistency and depth of engagement in online discussion forums, iterative improvement observed in laboratory submissions, demonstrable growth in algorithm design fluency across increasingly complex problem domains, and behavioral indicators such as persistence in debugging, strategic resource utilization, and collaborative contribution logs. Such triangulated evidence yields a nuanced, competency-based portrait of each learner's intellectual growth, critical thinking maturation, and computational problem-solving proficiency [6].

The entire teaching framework is grounded in the educational philosophy of 'teacher-led, student-centered, and AI-assisted', which affirms the irreplaceable role of expert pedagogical judgment while leveraging artificial intelligence as a scalable, responsive, and data-informed enhancement to human instruction [7]. It prioritizes epistemic equity by reducing knowledge access disparities, cultivates metacognitive awareness through transparent learning analytics, and sustains academic integrity via robust assessment design. Crucially, the framework does not replace instructor agency but rather augments it—enabling more precise differentiation, richer formative feedback, and empirically guided curriculum iteration—thus realizing a synergistic integration of traditional pedagogical wisdom and intelligent learning infrastructure.

3.2. Implementation Path and Stage Division

To ensure the smooth implementation of AI-enhanced pedagogical innovation for the Data Structures and Algorithms course, a phased and systematic approach is adopted, comprising four sequential yet interdependent stages: Infrastructure Construction, Platform Application, Deep Integration, and Continuous Optimization, as illustrated in

Figure 2. This structured progression enables iterative capacity building, evidence-informed decision-making, and sustainable scalability across diverse teaching contexts [8].

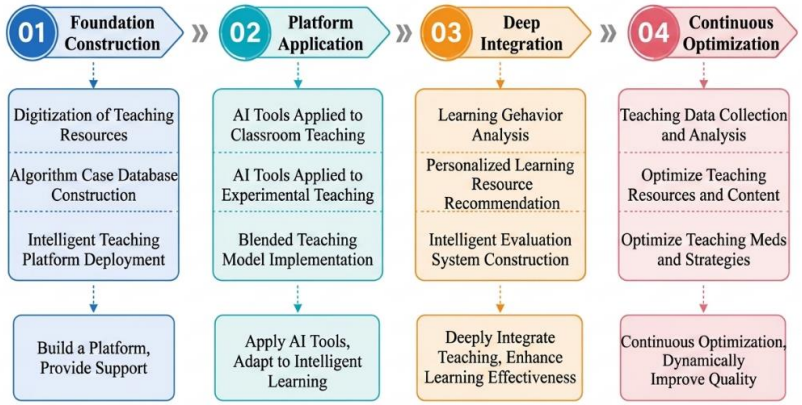

Figure 2. Implementation Path of Teaching Reform

The first stage centers on foundational capacity development, encompassing comprehensive digital curation of curriculum materials, systematic construction of a domain-specific algorithm case repository featuring real-world problem scenarios and step-by-step solution pathways, and deployment of a secure, scalable intelligent teaching platform compliant with national educational information security standards. The second stage advances practical adoption by embedding AI-powered tools—such as automated code feedback engines, interactive visualization modules, and adaptive practice environments—into both lecture-based instruction and hands-on laboratory sessions. Through a carefully scaffolded blended learning model, instructors support students in progressively developing digital literacy and metacognitive awareness within intelligent learning ecosystems [9]. The third stage achieves holistic integration by leveraging AI across the full instructional cycle: real-time analysis of student interaction patterns to infer conceptual understanding, dynamic generation of personalized learning pathways aligned with individual knowledge gaps and cognitive load profiles, and implementation of multi-dimensional assessment mechanisms that evaluate not only correctness but also algorithmic reasoning depth, efficiency awareness, and debugging resilience. In the final stage, longitudinal teaching analytics—including engagement metrics, performance trajectories, and qualitative feedback—are systematically analyzed to refine resource design, adjust pedagogical strategies, and enhance platform functionality, thereby fostering a self-improving, data-driven quality assurance loop for sustained excellence in computer science education.

3.3. Specific Reform Measures

3.3.1. Development of Intelligent Teaching Content

The development of teaching content constitutes a foundational element in the ongoing evolution of curriculum design. Traditional instruction in Data Structures and Algorithms has historically relied heavily on static, textbook-driven knowledge frameworks, which often exhibit limited responsiveness to rapid technological advancements and evolving industrial practice. This inertia can lead to curricular lag—where theoretical coverage outpaces real-world application contexts—and may hinder students' ability to transfer foundational concepts into contemporary software engineering environments [10]. To align with the competencies demanded by the artificial intelligence era, course content must be systematically reoriented toward enhanced intelligence integration, demonstrable practical applicability, and stronger engineering relevance. This reorientation entails not only updating core topics but also embedding contextual authenticity, computational thinking scaffolds, and domain-specific problem-solving paradigms. The intelligent teaching content development scheme proposed in this study—illustrated in Figure 3—operates through three interlocking dimensions:

pedagogical structure, thematic expansion, and resource architecture. Each dimension is designed to reinforce conceptual mastery while cultivating adaptive learning behaviors and technical fluency essential for next-generation computing professionals.

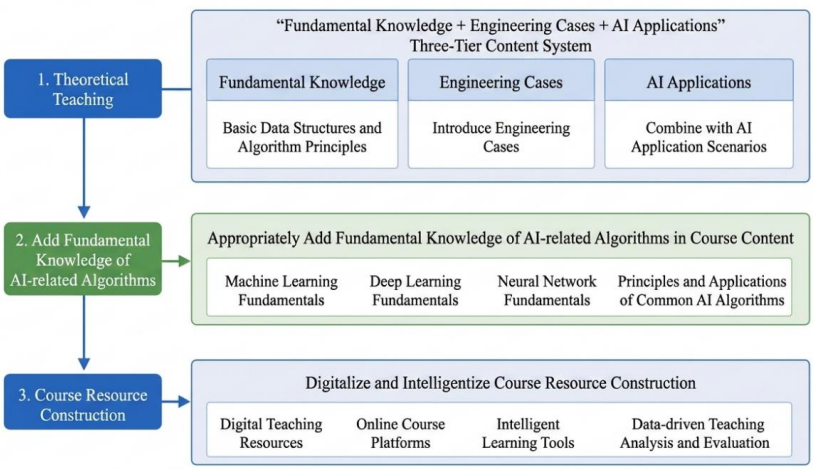

Figure 3. Proposal for Intelligent Teaching Content Construction

First, theoretical instruction adopts a rigorously scaffolded three-layer content structure: 'Fundamental Knowledge + Engineering Cases + AI Applications.' This layered approach ensures that students first internalize canonical data structures—such as arrays, linked lists, trees, and graphs—and core algorithmic paradigms—including recursion, divide-and-conquer, dynamic programming, and greedy strategies—before engaging with authentic implementation challenges. Engineering cases are carefully selected from widely deployed systems to illustrate functional relevance; for instance, graph theory concepts are contextualized through analysis of real-time navigation system path-planning logic, including considerations of edge-weight optimization, scalability under traffic dynamics, and latency constraints in distributed routing protocols. Such case-based anchoring strengthens cognitive retention, fosters metacognitive awareness of algorithm selection criteria, and bridges abstract formalism with observable system behavior.

Second, the curriculum intentionally incorporates algorithmic knowledge domains traditionally associated with artificial intelligence research, thereby expanding students' conceptual repertoire beyond classical deterministic methods. When covering search algorithms, for example, the syllabus extends beyond breadth-first and depth-first traversal to include heuristic search frameworks such as A* and IDA*, emphasizing admissible and consistent heuristics, time-space trade-offs, and empirical evaluation metrics. Similarly, probabilistic reasoning elements—like Monte Carlo simulation techniques—are introduced alongside randomized algorithms, enabling learners to appreciate uncertainty modeling and statistical convergence properties. These integrations are not additive supplements but rather synergistic extensions that highlight continuity between foundational computer science principles and emerging AI methodologies, reinforcing conceptual coherence across disciplinary boundaries.

Third, curriculum resources are developed using digital-native, intelligence-enhanced methodologies to support differentiated learning pathways [11]. An intelligent learning resource repository is constructed as a modular, interoperable ecosystem comprising annotated lecture slides, validated laboratory assignments with auto-graded test suites, curated exercise banks categorized by Bloom's taxonomy level, interactive algorithm visualization modules with adjustable parameters and step-through execution tracing, cloud-based integrated development environments for immediate code experimentation, and adaptive learning-path recommendation engines powered by student interaction analytics. These components collectively enable just-in-time scaffolding, self-paced progression, and formative feedback loops—empowering learners to diagnose knowledge gaps, revisit prerequisite concepts, and advance confidently along

personalized trajectories aligned with their academic background, career aspirations, and cognitive preferences.

3.3.2. Construction of AI Teaching Assistant System

The AI teaching assistant system constitutes a core element of pedagogical innovation enabled by artificial intelligence technologies. Within conventional classroom settings, instructors frequently encounter significant challenges in delivering timely responses to student inquiries and offering individualized academic support—particularly when managing large cohorts with diverse learning paces and needs. By contrast, AI-powered teaching assistants substantially augment instructional capacity through scalable, consistent, and context-aware support mechanisms. The system developed in this study comprises two functionally integrated components: an intelligent question-answering module and a learning analytics module, both designed to operate synergistically within the existing course infrastructure without disrupting established curricular workflows or assessment protocols.

The intelligent question-answering module leverages domain-specific course knowledge bases—curated from syllabi, lecture notes, textbooks, and verified educational resources—and aligns them with robust large language model capabilities to deliver accurate, pedagogically appropriate responses. It supports continuous, real-time interaction across multiple sessions, enabling students to refine queries, request clarifications, and explore conceptual linkages in a natural conversational manner. Response generation adheres to strict academic integrity standards, prioritizing factual accuracy, logical coherence, and alignment with disciplinary conventions; all outputs undergo validation against authoritative source materials prior to deployment [12].

The learning analytics module systematically processes anonymized, aggregated behavioral data—including time-on-task metrics, completion rates for interactive learning activities, performance trends across formative assessments, laboratory exercise outcomes, and patterns in response errors categorized by cognitive demand level (e.g., recall, application, analysis). These multidimensional insights are synthesized into actionable diagnostic reports that highlight strengths, emerging competencies, and targeted areas requiring reinforcement. Instructors utilize these reports not only to inform adaptive interventions but also to iteratively refine course design, resource allocation, and scaffolding strategies in evidence-informed ways.

Collectively, the implementation of this AI teaching assistant system facilitates a paradigm shift from generalized, one-size-fits-all instruction toward a responsive, learner-centered approach grounded in empirical observation and continuous feedback loops. This transition enhances pedagogical precision, optimizes instructor workload distribution, strengthens student engagement, and fosters more equitable learning opportunities across varied academic backgrounds. Furthermore, the system's architecture is modular and extensible, allowing seamless integration with institutional learning management platforms and future upgrades in AI capabilities while maintaining full compliance with national educational data governance standards [13].

3.3.3. Experimental Platform Upgrade

Traditional laboratory teaching frequently encounters persistent pedagogical challenges, including constrained experimental scope, delayed instructional feedback, and suboptimal levels of student motivation and sustained participation. These limitations hinder the development of robust problem-solving skills and practical technical proficiency. To systematically mitigate these constraints, this study implements a comprehensive upgrade of the experimental teaching platform through the strategic integration of intelligent educational technologies—such as adaptive learning algorithms, real-time code analysis engines, and data-driven progress analytics—thereby transforming static lab sessions into dynamic, responsive, and learner-centered experiences.

First, experimental content design adopts a dual-layered pedagogical architecture: foundational verification-oriented experiments and authentic project-based learning

modules aligned with industry practice. The foundational tier emphasizes structured reinforcement of core concepts in data structures and algorithmic design, enabling students to develop precise implementation competence through guided, stepwise exercises. The advanced tier integrates realistic engineering requirements—including modular system design, version-controlled collaboration, performance benchmarking, and documentation standards—drawn from contemporary software development workflows. This scaffolded progression ensures that learners progressively internalize theoretical principles while cultivating professional habits, teamwork competencies, and solution-oriented thinking essential for complex technical environments [14].

Second, a cloud-native online programming platform is deployed alongside an intelligent assessment infrastructure featuring multi-dimensional evaluation criteria. Students execute coding tasks, run test suites, inspect runtime behavior, and iteratively refine solutions entirely within the browser-based environment. The assessment engine performs syntactic validation, semantic correctness checking, edge-case coverage analysis, and memory-efficiency profiling; it then generates granular, actionable feedback—including pinpointed error localization, contextualized debugging hints, and comparative performance metrics against reference implementations. This automated, continuous feedback loop reduces instructor grading burden, accelerates learning iteration cycles, and fosters metacognitive awareness by making implicit reasoning processes explicit and traceable.

Through this holistic platform enhancement, laboratory instruction evolves from isolated skill drills into an integrated, competency-oriented ecosystem where formative assessment, reflective practice, and iterative improvement are embedded throughout the learning journey. The revised framework strengthens alignment between conceptual knowledge acquisition and applied technical fluency, supports differentiated learning pathways, and enables fine-grained tracking of individual growth trajectories across cognitive, procedural, and attitudinal domains—ultimately reinforcing the seamless interplay between theoretical instruction and experiential learning in computer science education.

3.3.4. Multi-Dimensional Evaluation System

Traditional course evaluation methods have long centered on summative assessment via final examination scores, which inherently limits the scope of insight into students' longitudinal learning trajectories and holistic competency development. This narrow focus often overlooks critical dimensions such as procedural understanding, adaptive problem-solving, collaborative engagement, and reflective growth—elements increasingly emphasized in contemporary pedagogical frameworks. To overcome these limitations, this study develops an AI-integrated multi-dimensional evaluation system grounded in educational measurement theory and learning analytics principles. The system comprises four interrelated and mutually reinforcing dimensions: Theoretical Knowledge Evaluation, Practical Ability Evaluation, Learning Process Evaluation, and Innovation Capability Evaluation. Each dimension is designed with distinct yet complementary assessment mechanisms, ensuring alignment with both curriculum objectives and real-world skill expectations in computing disciplines [12].

Theoretical knowledge evaluation employs a balanced mix of formative and summative instruments—including scheduled quizzes, structured classroom questioning, concept-mapping exercises, and comprehensive final assessments—to gauge students' conceptual clarity, logical reasoning, and retention of foundational principles. Practical ability evaluation emphasizes authentic application through algorithmic design tasks, iterative coding assignments, and scenario-based debugging challenges; key indicators include laboratory performance quality, project completion rigor, code efficiency metrics, and adaptability in modifying solutions under constraint variations. Learning process evaluation leverages granular behavioral data captured by the intelligent teaching platform—such as time-on-task distribution across modules, frequency and depth of peer or instructor interactions, patterns in self-assessment responses, submission timeliness,

and progression trends in diagnostic knowledge checks—to construct dynamic, individualized learning profiles. Innovation capability evaluation fosters exploratory thinking through scaffolded open-ended projects, interdisciplinary design challenges, and participation in academic competitions; it assesses originality in solution formulation, integration of cross-domain concepts, willingness to experiment with alternative approaches, and articulation of design rationale—all within ethically grounded and technically sound boundaries.

By integrating quantitative precision with qualitative nuance, this multi-dimensional evaluation system enables educators to move beyond static, point-in-time judgments toward continuous, evidence-informed feedback loops. It supports differentiated instruction, early identification of learning bottlenecks, and timely intervention strategies while cultivating student agency through transparent criteria and reflective self-monitoring tools [6]. Moreover, the system's design prioritizes fairness, consistency, and developmental appropriateness—ensuring that assessment outcomes meaningfully inform both instructional refinement and learner growth. Ultimately, it contributes to a more responsive, equitable, and pedagogically robust educational ecosystem aligned with national priorities for high-quality talent development in science and technology fields.

4. Expected Outcomes and Evaluation Plan

4.1. Expected Teaching Outcomes

Through the integration of interactive visualization technologies—such as dynamic flowcharts, step-by-step code execution animations, and real-time data structure rendering—and intelligent learning resources—including adaptive learning paths, contextualized concept explanations, and multimodal content delivery—the cognitive load associated with mastering abstract algorithmic concepts can be significantly reduced. This pedagogical approach fosters deeper conceptual understanding by enabling learners to observe, manipulate, and predict algorithm behavior across diverse input scenarios, thereby strengthening mental models of computational processes. Furthermore, the incorporation of intelligent question-answering systems, immediate formative feedback on misconceptions, and integrated online programming environments supports continuous engagement, promotes metacognitive reflection, and cultivates sustained motivation for both in-class participation and self-directed after-class practice.

AI-powered educational tools also contribute meaningfully to instructional efficiency by automating routine yet time-intensive tasks, such as syntax- and logic-based assessment of student-submitted code, identification of common error patterns, generation of diagnostic reports, and provision of targeted remedial suggestions. This automation alleviates administrative burdens on instructors, allowing them to redirect their professional expertise toward higher-order teaching responsibilities—including curriculum refinement, scaffolding design for diverse learner profiles, development of authentic assessment tasks, and delivery of individualized academic support—thereby enhancing the overall quality, responsiveness, and equity of pedagogical practice within computing education.

4.2. Evaluation Indicators and Methods

To scientifically evaluate the effectiveness of the AI-integrated teaching initiative, this study constructs a comprehensive, multidimensional teaching evaluation framework grounded in pedagogical theory and empirical educational practice. The framework is designed to capture both quantitative performance metrics and qualitative experiential dimensions, ensuring balanced assessment across cognitive, behavioral, and affective domains. It aligns with contemporary standards for evidence-based educational evaluation and emphasizes validity, reliability, and contextual relevance in higher education settings.

Regarding learning outcomes, the evaluation incorporates four core indicators: course pass rate, distinction rate (defined as scores ≥90%), demonstrated proficiency in algorithmic problem decomposition and implementation, and depth of conceptual

understanding as measured through standardized knowledge mapping assessments. These indicators are tracked longitudinally across multiple academic cohorts to isolate the influence of instructional interventions from confounding variables such as prior preparation or cohort-level differences. Statistical analysis includes paired t-tests and effect size estimation to quantify meaningful shifts in academic achievement following the integration of AI-supported instructional strategies.

For learning behavior assessment, the system monitors four observable and quantifiable engagement metrics: frequency and quality of in-class interaction (e.g., question-asking, peer collaboration), cumulative duration of structured online learning activities within the institutional LMS, completion rate of mandatory laboratory exercises, and volume and temporal distribution of code submissions in automated grading environments. Behavioral data are aggregated at the individual and class levels to identify patterns of sustained engagement, self-regulation, and adaptive learning strategies, thereby offering insights into how AI tools influence students' metacognitive development and persistence in computational tasks.

Practical ability is evaluated through three rigorously defined criteria: rubric-based assessment of final project deliverables—including functionality, documentation quality, scalability, and adherence to software engineering principles; measurable improvement in time/space complexity optimization across iterative coding assignments; and documented participation in externally validated programming contests or open-source contributions. This tripartite approach ensures that practical competence is assessed not only through summative outputs but also through process-oriented growth and real-world applicability, reflecting industry-aligned competency frameworks widely adopted in computer science education.

Teaching satisfaction is gauged via triangulated qualitative feedback collected through structured, anonymous questionnaires administered at mid-term and end-of-term intervals, supplemented by semi-structured interviews with representative samples of students and instructors. The instrument covers four thematic domains: perceived quality and accessibility of digital teaching resources, usability and pedagogical utility of embedded AI tools (e.g., intelligent tutoring systems, automated feedback engines), perceived effectiveness of instructor-facilitated interactive activities, and adequacy of academic and technical support services throughout the learning journey [4, 8].

5. Conclusions and Future Prospects

This study introduces an AI-integrated pedagogical framework specifically designed for the Data Structures and Algorithms course, responding to persistent instructional challenges—including heterogeneous student preparedness, abstract conceptual barriers, limited real-time feedback on coding practice, and difficulties in assessing algorithmic reasoning beyond syntactic correctness. The framework operationalizes artificial intelligence across four interlocking dimensions: (1) adaptive learning path generation, which dynamically adjusts content sequencing and scaffolding based on individual diagnostic assessments; (2) intelligent code analysis and explanation, delivering context-aware feedback on algorithmic logic, time/space complexity implications, and common misconception patterns; (3) interactive, step-through visualizations that render abstract data transformations—such as heapify operations or graph traversal states—into perceptually grounded, manipulable representations; and (4) a multi-layered evaluation architecture integrating formative micro-assessments, automated rubric-aligned code reviews, and reflective self-assessment prompts. Collectively, these components establish continuous, responsive support across the full instructional cycle—from initial concept exposure through iterative practice to summative synthesis.

Future work will prioritize three interconnected avenues. First, robust domain-specific knowledge representation remains foundational: expanding and semantically enriching the underlying course ontology—including formalized algorithmic patterns, prerequisite dependency graphs, and common error taxonomies—will enhance the precision and pedagogical fidelity of AI-generated guidance. Second, cultivating

responsible AI literacy requires deliberate curricular integration—not merely tool access, but structured metacognitive activities that scaffold students' critical appraisal of AI outputs, awareness of limitations in code generation or explanation, and development of independent verification strategies. Third, longitudinal empirical investigation is needed to assess how sustained use of such frameworks influences deep conceptual retention, transfer to novel problem contexts, and long-term computational thinking dispositions—particularly across diverse learner profiles and institutional settings.

Funding: This work is supported by the Quality Engineering Project of Guangdong University of Science and Technology under the project number GKZLGC2025019.

References

1. L. I. Ruiz-Rojas and P. Acosta-Vargas, "Transforming learning: Use of the 4PADAFE instructional design methodology and generative artificial intelligence in designing MOOCs for innovative education," *Sustainability*, vol. 18, no. 6, p. 2683, 2026.
2. X. Ding, "Teaching reform and exploration of data structure course based on artificial intelligence," in *2024 International Conference on Artificial Intelligence and Digital Libraries (AIDL)*, Dec. 2024, pp. 18–21.
3. H. Li, L. Xiao, K. Lu, D. Li, Z. Zhang, and Q. Xia, "Exploring the Application of Large Language Models (LLMs) in Data Structure Instruction: An Empirical Analysis of Student Learning Outcomes in Computer Science," *Information*, vol. 17, no. 4, p. 353, 2026.
4. S. AlBlooshi, "Artificial intelligence in higher education, opportunities, and challenges: A review," in *Frontiers in Education*, vol. 10, p. 1683968, Jan. 2026.
5. J. Schumann et al., "Enabling open and FAIR catalysis data with standardized data structures," *Nature Catalysis*, vol. 9, no. 3, pp. 225–229, 2026.
6. F. Winterstein, S. Bayliss, and G. A. Constantinides, "High-level synthesis of dynamic data structures: A case study using Vivado HLS," in *2013 International Conference on Field-Programmable Technology (FPT)*, Dec. 2013, pp. 362–365.
7. X. Tang, Y. Zhou, K. Xu, Q. Zhang, and W. Pedrycz, "Enhancement of the classification performance of fuzzy C-Means with a nonlinear transformation strategy for data structures," *IEEE Transactions on Cybernetics*, 2026.
8. M. E. d'Imperio, "Data structures and their representation in storage," *Annual Review in Automatic Programming*, vol. 5, pp. 1–75, 1969.
9. X. Bosch-Capblanch et al., "An Open-Source Systematic Reviews Integrated System (OSSYRIS)–Streamlining Processes and Standardising Data Structures," *Cochrane Evidence Synthesis and Methods*, vol. 4, no. 4, p. e70088, 2026.
10. I. Dziedziczak, M. Kęsy, and J. Stępniewski, "The Strategic Importance of Data Structures in Accounting Information Systems: A Theoretical, Regulatory and Technological Perspective," *European Research Studies Journal*, vol. 29, no. 2, pp. 79–100, 2026.
11. B. Chandrasekaran and A. Goel, "From numbers to symbols to knowledge structures: Artificial intelligence perspectives on the classification task," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 3, pp. 415–424, 1988.
12. I. D. Pop and B. Iudean, "AI-Assisted Diagnosis of Students' Misconceptions on Memory Allocation and Dynamic Data Structures in C++," in *2026 IEEE International Conference on Software Analysis, Evolution and Reengineering–Companion (SANER-C)*, Mar. 2026, pp. 1–9.
13. W. Ertel, *Introduction to Artificial Intelligence*. Springer Nature, 2024.
14. S. J. Russell, *Artificial Intelligence: A Modern Approach*. Pearson Education, Inc., 2010.