ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202604_9(4).0006

Teaching Reform of a Data Structures Course Oriented Toward the Cultivation of Complex Engineering Problem-Solving Ability

Qi Jiang [1, *], Gang Zhou [1], Guofang Liu [1], Zheng Li [1], Shishi Liao [2]

[1] School of Computer Science, Sichuan University Jinjiang College, Meishan 620860, China

[2] School of Electrical and Electronic Information Engineering, Sichuan University Jinjiang College, Meishan 620860, China

Abstract

The Data Structures course is a core foundational component in the curriculum system of computer science education, characterized by high levels of abstraction, strong logical interconnections, and intensive practical requirements. To address the limitations of traditional teaching—such as the insufficient integration between theoretical instruction and programming practice, weak knowledge transfer ability among students, and inadequate support for formative assessment—this paper proposes a teaching reform framework for the Data Structures course aimed at cultivating the ability to solve complex engineering problems with the assistance of artificial intelligence. The proposed framework is structured around progressive competency development, supported by modular reconstruction of teaching content. It adopts a case-driven approach, hierarchical task design, and integrated practical training as its primary implementation pathways. Artificial intelligence technologies are embedded into multiple stages of the teaching process, including pre-class diagnostics, in-class support, programming practice, and learning feedback. As a result, a four-stage progressive teaching system is established, encompassing "conceptual understanding – structural implementation – algorithm analysis – comprehensive application."

Keywords

Data Structures; Teaching Reform Framework; Artificial Intelligence Technologies; Competency-Oriented Education; Formative Assessment.

1. Introduction

The Data Structures course plays a pivotal role in the curriculum system of computer science programs, serving as a bridge between foundational and advanced courses. It encompasses core topics such as linear lists, stacks and queues, trees, graphs, searching, and sorting. The course not only requires students to master fundamental data structures and associated algorithms, but also to develop capabilities in data abstraction, complexity analysis, and program implementation.

However, in current teaching practice, several challenges remain prevalent. These include the high level of abstraction of course content, the disconnect between theoretical instruction and experimental training, students' insufficient ability to transfer and apply knowledge in integrated contexts, and the limited effectiveness of formative assessment. With the rapid integration of generative artificial intelligence (AI) into educational settings, a new issue has emerged: how to leverage AI as a supportive tool for cognitive construction, code debugging, and learning feedback, rather than allowing it to function merely as an answer generator that replaces independent thinking. Addressing this issue has become a critical task in the reform of Data Structures education.

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202604_9(4).0006

2. Related Work and Problem Statement

2.1. Related Work

In recent years, research on the reform of Data Structures courses has primarily focused on approaches such as Outcome-Based Education (OBE) [1], deliberate practice [2], project-based learning [3], flipped classrooms [4], and smart classroom environments [5]. Other studies, conducted earlier, have explored the role of visualization techniques, online programming platforms, automated feedback, and learning analytics tools in the teaching of algorithms and data structures [6].

Overall, existing studies have provided valuable insights for optimizing course instruction. However, several limitations remain. These include insufficient restructuring of the internal logical organization of course content, unclear boundaries in the application of AI as an instructional aid, and the lack of a well-coordinated mechanism between formative assessment and competency development.

2.2. Practical Challenges

First, students tend to remain at the level of conceptual memorization and code imitation, and often struggle to grasp the intrinsic connections among problem abstraction, data structure selection, algorithm design, and complexity analysis.

Second, there is often a disconnect between classroom instruction and experimental practice. While students may be able to articulate algorithmic ideas, they frequently demonstrate insufficient competence in independent programming, boundary condition handling, and performance comparison.

Third, although course assessment typically includes components such as class participation, in-class quizzes, practical projects, and a final written examination, the effectiveness of formative assessment is still constrained in the absence of clearly defined evaluation criteria and a continuous feedback mechanism.

3. Overall Design of the Teaching Reform

To address the practical challenges identified, this paper proposes an overall teaching reform framework based on five guiding principles: competency orientation, modular reconstruction, integrated practice, AI empowerment, and coordinated assessment. The course is structured into four progressive levels-“conceptual understanding structural implementation algorithm analysis comprehensive application”-to establish a clear competency development pathway. A closed-loop instructional process is implemented through case introduction, in-class simulation, experimental verification, and post-class extension activities.AI is leveraged to assist with learning diagnostics, case generation, code debugging, and feedback provision, thereby enhancing the precision and personalization of course delivery. Fig.1 illustrates the overall framework of the teaching reform.

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202604_9(4).0006

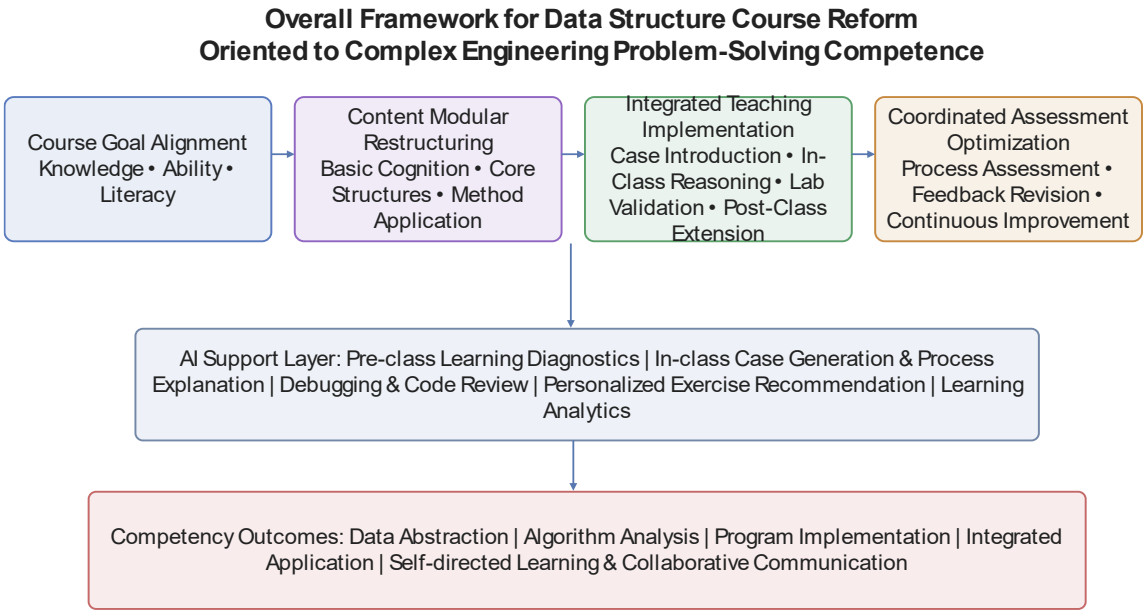

Figure 1. Overall Framework of Data Structures Course Teaching Reform

4. Implementation Pathways of the Teaching Reform

4.1. Restructuring Teaching Content Along a Progressive Competency Pathway

The course content is no longer organized in a linear, chapter-by-chapter manner, but is structured around the process of competency development. In the conceptual understanding stage, the emphasis is on fundamental concepts, distinctions between logical and storage structures, and methods for algorithm analysis. The structural implementation stage focuses on hands-on programming practice with typical data structures such as sequential lists, linked lists, stacks, queues, trees, and graphs. During the algorithm analysis stage, students examine differences among various methods in terms of time complexity, space complexity, and applicable scenarios. Finally, the comprehensive application stage requires students to integrate structure selection, algorithm design, and result analysis within real-world problem contexts. Table 1 presents the core course modules and their corresponding competency development objectives.

Table 1. The core course modules and their corresponding competency development objectives

Module	Core Content	Corresponding Competency	Representative Practice Tasks
Introduction and Algorithm Analysis	Basic concepts of data structures, abstract data types, time and space complexity	Develop overall course understanding and cultivate awareness of algorithm analysis	Concept differentiation, preliminary complexity evaluation
Linear Lists, Stacks, and Queues	Sequential and linked storage, LIFO and FIFO principles	Master basic structure implementation and operation design	Insertion, deletion, search; push/pop; enqueue/dequeue
Trees and Graphs	Trees and binary trees, storage and traversal of graphs	Enhance recursive thinking and complex structure modeling abilities	Binary tree construction, DFS/BFS traversal
Searching and Sorting	Binary search, binary search trees, insertion sort, quick sort	Develop ability to compare methods and apply them comprehensively	Efficiency comparison of multiple approaches, integrated design

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202604_9(4).0006

4.2. Establishing the "Case Introduction → In-Class Simulation → Experimental Verification → Post-Class Extension" Instructional Chain

In the case introduction stage, instructors introduce problem scenarios such as grade management, task scheduling, browser history tracking, and campus navigation to help students understand the practical context for different data structures. The in-class simulation stage emphasizes board analysis, pseudocode construction, and explanation of key code segments. During the experimental verification stage, students are required to complete program implementation, debugging, and complexity analysis based on predefined practice tasks. Finally, the post-class extension stage promotes knowledge transfer through comparative, reflective, and integrative assignments. Table 2 presents the correspondence among course modules, practice tasks, and competency development objectives.

Table 2. The correspondence among course modules, practice tasks, and competency development objectives

Teaching Module	Representative Classroom Cases	Experimental / Assignment Tasks	Key Competencies Trained
Linear Lists	Student grade management, address book management	Insertion, deletion, and search in sequential and linked lists	Abstract modeling, basic implementation
Stacks and Queues	Browser back functionality, task scheduling	Stack full/empty checks, push/pop, enqueue/dequeue	Logical reasoning, boundary handling
Trees and Binary Trees	File directory structures, organizational charts	Construct binary trees and implement traversal and printing	Recursive thinking, structural analysis
Graphs	Campus path planning, social network modeling	Depth-first and breadth-first traversal	Complex structure modeling, pathfinding
Searching and Sorting	Grade retrieval, log sorting	Binary search, binary search tree, insertion sort, quick sort	Algorithm comparison, performance analysis

4.3. Integrating Artificial Intelligence Across the Entire Teaching–Learning–Assessment Process

In this study, AI is positioned as a cognitive scaffold and a feedback tool, rather than as an answer-generating machine that replaces students in completing learning tasks. In the pre-class stage, instructors can use AI to analyze preparatory tests and past assignments to identify students' frequently encountered weaknesses. During the in-class stage, AI can rapidly generate teaching cases aligned with real-world scenarios, provide algorithm execution explanations, and visualize code execution traces, helping students lower cognitive barriers. In the experimental stage, AI can assist with code debugging, boundary condition checking, and multi-solution comparisons; however, students are required to review, modify, and justify AI-generated outputs. In the assessment stage, instructors can leverage AI to summarize common errors in quizzes, experiments, and reports, thereby improving feedback efficiency. Table 3 outlines the key implementation points of AI-assisted teaching, learning, and assessment. And Fig. 2 illustrates the closed-loop implementation pathway of "teaching–learning–assessment."

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202604_9(4).0006

Table 3. The key implementation points of AI-assisted teaching, learning, and assessment

Stage	AI Support	Implementation Focus	Notes / Considerations
Pre-Class	Analysis of preparatory tests, identification of weak points	Adjust classroom focus and task levels according to students' foundational differences	Avoid relying solely on score-based grouping; integrate knowledge point diagnostics
In-Class	Case generation, process explanation, pseudocode optimization	Use visualizations to illustrate abstract concepts and algorithm execution logic	Teachers must ensure accuracy and appropriate difficulty
Experimental	Code debugging, boundary checks, complexity hints	Assist students in identifying common issues such as pointers, recursive exits, and traversal markers	Require students to submit corrections and explanations to prevent direct copying of AI outputs
Assessment	Summarization of common assignment errors, tiered practice assignment push	Improve feedback efficiency and support personalized learning	AI-generated conclusions should be reviewed by instructors before being used for teaching decisions

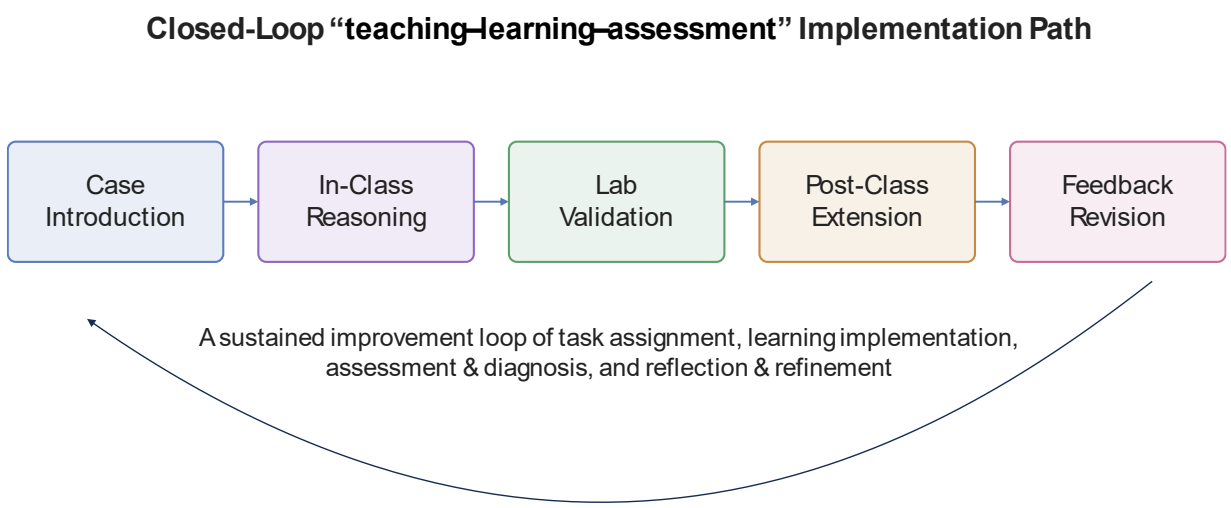

Figure 2. The closed-loop implementation pathway of "teaching–learning–assessment"

4.4. Adopting a Practice Organization Approach Combining Tiered Tasks and Group Collaboration

Considering the significant differences in students' prior knowledge and abilities, practical tasks in the course are designed at three levels: basic, intermediate, and advanced. The basic level focuses on core experimental tasks to ensure that students meet the fundamental course requirements. The intermediate level requires students to compare different data structures and algorithms, analyzing their respective advantages, disadvantages, and complexity differences. The advanced level emphasizes solving more comprehensive and realistic problems, in which students are expected to complete structure selection, algorithm design, program implementation, and result analysis. In addition, group discussions, in-class presentations, and experimental reports are incorporated to cultivate students' collaboration skills and structured expression abilities.

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202604_9(4).0006

4.5. Optimizing Formative Assessment and Establishing a Continuous Improvement Mechanism

Course assessment consists of both formative evaluation and a final written examination. Formative assessment not only considers class attendance and learning engagement, but also places greater emphasis on students' staged performance in classroom interaction, in-class quizzes, practical projects, group presentations, and extracurricular tasks. The key to the reform lies in establishing a closed-loop mechanism of "assessment-feedback-revision-reassessment", ensuring that formative assessment functions as an effective tool for promoting learning rather than merely measuring outcomes. Table 4 presents the correspondence between formative assessment indicators and competency dimensions.

Table 4. The correspondence between formative assessment indicators and competency dimensions

Assessment Component	Suggested Weight (%)	Main Evaluation Content	Corresponding Competency Dimension
Classroom Participation and Real-Time Interaction	10	Problem understanding, quality of responses, engagement in discussions	Conceptual understanding, communication skills
In-Class Quizzes	20	Mastery of core concepts, algorithm analysis, and immediate application	Fundamental cognition, analytical judgment
Practical Projects and Code Implementation	30	Program correctness, coding standards, boundary handling, and complexity analysis	Implementation ability, problem-solving skills
Group Presentations	20	Rationale of solution design, clarity of expression, quality of collaboration	Collaborative communication, comprehensive analysis
Assignments and Reflective Learning	20	Knowledge transfer, information organization, reflective revision	Self-directed learning, continuous improvement

5. Expected Outcomes and Impact Analysis of the Teaching Reform

The effectiveness of the teaching reform should not be evaluated solely based on a single examination score, but rather through a multidimensional analysis encompassing knowledge acquisition, competency development, learning behaviors, and classroom ecology.

First, modular restructuring and case-driven instruction are expected to help students develop a clearer knowledge framework, enhancing their understanding of the relationships among logical structures, storage structures, and complexity analysis. Second, the integration of practice and tiered tasks can improve students' abilities in program implementation, debugging, and comprehensive application, enabling them to progress from "being able to write code" to "being able to design solutions." Third, formative assessment combined with a feedback loop is likely to increase students' classroom engagement and foster greater awareness of self-directed learning. Finally, the appropriate integration of artificial intelligence demonstrates positive effects in areas such as case generation, code debugging, and learning feedback. However, it also places higher demands on instructors' ability to supervise and validate AI outputs, as well as on students' critical awareness in evaluating and using AI-assisted results. Table 5 presents the expected outcomes of the reform and their corresponding evidence.

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202604_9(4).0006

Table 5. Dimensions of Reform Effectiveness and Observable Evidence

Analytical Dimension	Expected Change	Observable Evidence
Knowledge Structure	Transition from fragmented memorization to systematic understanding	Classroom questioning, concept differentiation, explanations in lab reports
Practical Ability	Improvement in program implementation, debugging, and boundary handling	Quality of lab completion, code standardization, records of error correction
Comprehensive Application	Ability to select appropriate data structures and algorithms based on problem contexts	Integrated tasks, case analysis, group presentations
Learning Behavior	Increased participation and awareness of self-directed learning	Classroom interaction, assignment completion rates, platform learning logs, reflective reports
Technical Literacy	Development of the ability to critically and appropriately use AI tools	AI usage documentation, code revision explanations, offline validation performance

6. Conclusion and Future Work

The key to reforming the Data Structures course lies not in merely introducing additional technological tools, but in reorganizing knowledge, tasks, practical activities, and assessment around the course objectives. The AI-assisted teaching reform framework proposed in this study establishes a progressive instructional system-conceptual understanding-structural implementation-algorithm analysis-comprehensive application-through curriculum logic restructuring, instructional chain design, practice organization, and optimization of assessment mechanisms. This approach not only reflects the professional characteristics of core computer science courses but also addresses the practical needs arising from the integration of artificial intelligence into educational settings.

Future work may focus on quantifying the effectiveness of the reform through comprehensive stage-based assessments, evaluation of experimental task quality, analysis of learning behaviors, and survey or interview data, thereby supporting continuous iteration and improvement of the teaching approach.

Acknowledgements

This work was supported by the 2024 Talent Cultivation Quality and Teaching Reform Project of Sichuan University Jinjiang College (Project No. 2024JG014).

References

[1] S.S. You, H. Dai, B.K. Bao: Exploration of C language algorithms and data structures course reform driven by large models and OBE educational philosophy, Han Zi Culture, 2024(22). (In Chinese)

[2] C. Yang, S. Zhou, X. Zhang: Exploration of a "deliberate practice" teaching system for data structures courses, Computer Education, 2024(11), p.105-110. (In Chinese)

[3] X. Yan: Exploration of teaching methods for computer science data structures courses, Computer Education, 2024(11), p.101-104. (In Chinese)

[4] B. Long, Y. Yan: Research on generative AI-enabled reform of machine learning courses, Computer Era, 2025(11), p.83-87. (In Chinese)

ISSN: 2637-6067 DOI: 10.6918/IJOSSER.202604_9(4).0006

[5] J. Chen, E. Chen, C. Xie: Application of smart classroom teaching model in higher education computer science courses, Western Quality Education, 2025, 11(06), p.145-148+177. (In Chinese)

[6] G. Kogan, H. Chassidim, I. Rabaev: The efficacy of animation and visualization in teaching data structures: a case study, Educational Technology Research and Development, vol.72 (2024) No.4, p.2349-2372.