Education Reform and Development, 2025, Volume 7, Issue 10
https://ojs.bbwpublisher.com/index.php/ERD
Online ISSN: 2652-5372
Print ISSN: 2652-5364

Research on Strategies for Strengthening Computer Professional Thinking Training in Discrete Mathematics Courses

Lijuan Yao*

Taiyuan University, Taiyuan 030032, Shanxi, China

**Author to whom correspondence should be addressed.*

Abstract: Discrete mathematics, as a core foundational course in computer science, covers set theory, logical reasoning, graph theory, algebraic structures, etc., and is a key carrier for building computer science thinking. At present, there is a problem in discrete mathematics teaching in some universities that emphasizes theoretical derivation over thinking transformation, which makes it difficult for students to effectively connect the course knowledge with subsequent professional courses and engineering practice. This paper analyzes the necessity of strengthening computer professional thinking training in discrete mathematics courses, and elaborates on the core value of discrete mathematics in the cultivation of computer professional thinking from three dimensions: supporting core course learning, cultivating problem modeling ability, and enhancing algorithm design literacy, providing theoretical support for the optimization of teaching strategies for subsequent courses.

Keywords: Discrete Mathematics; Computer Science; Thinking training

Online publication: November 12, 2025

1. Introduction

Discrete mathematics is a fundamental core course for computer science and technology. Its content system covers key modules such as logical reasoning, set theory, graph theory, and algebraic systems, which directly determine the depth of understanding and application ability of students' computer professional knowledge. With the rapid development of fields such as artificial intelligence, big data, and cybersecurity, the computer industry's demand for professional thinking is constantly increasing. It requires not only a solid theoretical foundation but also core thinking abilities such as abstract modeling [1], logical analysis, and algorithm design. However, in the current teaching of some discrete mathematics courses, there is still a phenomenon of being mainly theoretical and disconnected from professional practice, which makes it difficult for students to transform discrete mathematics knowledge into professional thinking ability and affects subsequent course learning and career development. Therefore, an in-depth analysis of the necessity of strengthening computer

professional thinking training in discrete mathematics courses is of great significance for optimizing course teaching and improving the quality of talent cultivation.

2. The necessity of strengthening computer professional thinking training in Discrete Mathematics courses

2.1. The foundation of knowledge transfer that supports the study of core computer courses

The knowledge system of discrete mathematics serves as the theoretical foundation for many core courses in computer science, and the impact of its thinking training directly affects students' understanding and application abilities in subsequent courses. In the data structures course, the concept of set theory provides theoretical support for the definition of data structures such as linear lists, trees, and graphs, and the contents of path analysis and topological sorting in graph theory are directly applied to the algorithm design of graph structures; In the Database Principles course, relational algebra serves as the theoretical foundation of the query language for relational databases, and its logical reasoning thinking directly determines students' ability to optimize SQL statements; In artificial intelligence courses, propositional logic and predicate logic are the core tools for knowledge representation and reasoning. If students cannot form rigorous logical thinking through discrete mathematics training, it will be difficult for them to understand the principles of key technologies such as neural networks and expert systems [2].

In addition, discrete mathematics thinking methods play a crucial role in courses such as operating systems, compiler principles, and network security. Strengthening professional thinking training in discrete mathematics courses can help students establish intrinsic connections among knowledge, form a systematic knowledge system, provide effective knowledge transfer ability for subsequent core course learning, and avoid the learning predicament of "disconnection between theory and application" [3]. For example, when students study syntactic analysis in compiler principles, if they have mastered the logical connection between finite automata and regular expressions through discrete mathematics training, they can quickly understand the construction principle of the parser, achieve efficient transfer and application of knowledge, and improve the efficiency and effectiveness of course learning.

2.2. Develop the core ability of abstract modeling of computer problems

One of the core tasks of computer science is to transform practical problems into computable mathematical models, and discrete mathematics is precisely the key carrier for cultivating this ability of abstract modeling. In engineering practice within the field of computer science, whether it involves software system development, algorithm design, or data processing, practical problems must be abstracted into discrete mathematical models first. Solutions are then derived through theoretical analysis and calculation. For example, in the problem of network routing optimization, nodes and links in the network need to be abstracted as vertices and edges in graph theory, the routing selection problem is transformed into the shortest path solution problem, and the optimal routing solution is obtained through Dijkstra's algorithm or Floyd's algorithm [4]; In logistics distribution path planning, elements such as delivery points, distribution routes, and time costs need to be abstracted into weighted graph models, and the distribution scheme is optimized through algorithms related to the minimum spanning tree in graph theory or the traveling salesman problem. Set theory, graph theory, algebraic structures, and other contents in the discrete mathematics course provide students with thinking

methods and tools for abstract problems and model construction. Strengthening professional thinking training in the course can guide students to learn to extract key elements from practical problems, ignore irrelevant details, construct mathematical models using symbols, concepts, and methods of discrete mathematics, and gradually form the thinking path of "problem abstraction - model construction - solution verification" [5]. This ability of abstract modeling is not only a core requirement for computer science learning, but also a necessary ability for students to engage in software development, system design, and other jobs in the future. Without this ability, students will have difficulty dealing with complex engineering practice problems and can only remain at the level of simple code writing, unable to achieve the transformation from "technical executor" to "problem solver."

2.3. Enhance literacy in computer algorithm design and optimization

Algorithms are at the core of computer science, and discrete mathematics is the theoretical foundation of algorithm design and optimization. Strengthening professional thinking training in discrete mathematics courses can effectively enhance students' algorithmic literacy. Logical reasoning in discrete mathematics provides methods for proving the correctness of algorithms. Through training in propositional logic and predicate logic, students can learn to verify the correctness of algorithms using methods such as mathematical induction and proof by contradiction to avoid system failures caused by logical loopholes in algorithms. The concepts in set theory and graph theory provide ideas for algorithm design. For example, data deduplication algorithms can be designed based on the intersection and union operations of sets, and path finding and topological sorting algorithms can be designed based on the depth-first and breadth-first search of graphs [6]. The concepts of groups, rings, and fields in algebraic structures provide theoretical support for the design of cryptographic algorithms. For example, the security of the RSA encryption algorithm depends on the mathematical properties of the large number factorization problem [7]. In the teaching of discrete mathematics, through the derivation and analysis of classical algorithms, students can be guided to understand the underlying logic of algorithm design and master the thinking method of "problem transformation - mathematical modeling - algorithm implementation - optimization improvement" [8]. For example, when explaining the shortest path algorithm, by analyzing the applicable scenarios and time complexity of Dijkstra's algorithm and Bellman-Ford's algorithm, students can learn to choose the appropriate algorithm based on the characteristics of the problem and improve its efficiency through optimization. This cultivation of algorithmic literacy can not only enhance students' performance in course design and graduation projects, but also lay the foundation for their future work in high-end fields such as algorithm research and development and artificial intelligence, avoiding the shortcoming of "only being able to call existing algorithms but unable to design and optimize them."

3. Strategies for strengthening computer professional thinking in Discrete Mathematics courses

3.1. Modularized thinking: Breaking down complex problems to build discrete cognition

In discrete mathematics teaching, with the core idea of "breaking down complex problems into independent modules", students are guided to develop the awareness of "discretization" and "modularization" problem-solving through specific teaching content. For example, when teaching the chapter on combinatorial counting, instead of directly explaining the complex counting problem, break it down into sub-problems such as

permutation and combination calculation, recurrence relation analysis, and application of the principle of tolerance and exclusion, allowing students to gradually master the method of problem decomposition from the whole to the part [9]. At the same time, in combination with computer programming scenarios to enhance module cognition, after explaining theoretical knowledge, introduce module decomposition cases in programming design, such as breaking down core algorithms into independent modules like input data processing, logical judgment operations, and output result generation, to help students understand the correspondence between the "subproblems" in discrete mathematics and the "functional modules" in programming [10]. Through this "theoretical dissection + practical mapping" teaching approach, students overcome the fear of complex problems, learn to analyze discrete mathematics problems and computer application problems with modular thinking, form a "disassembly - solution - integration" problem-solving path, and lay the foundation for subsequent handling of professional problems such as complex algorithm design and system development.

3.2. Hierarchical and systematic thinking training: Build a logical hierarchy based on modular teaching

Design hierarchical teaching paths around core teaching modules such as mathematical logic and graph theory to guide students to sort out problems in logical hierarchy and cultivate systematic thinking. In the teaching of the mathematical logic chapter, start with the basic concepts of propositional logic, gradually transition to quantitative analysis of predicate logic, and then extend to the rule application of logical reasoning, allowing students to perceive the logical hierarchy of "basic concepts - advanced analysis - comprehensive application" during the learning process; In graph theory teaching, first explain the basic definition and representation of graphs, then delve into algorithmic analysis such as graph traversal and path finding, and finally, in combination with practical scenarios such as network topology design, guide students to handle problems in the hierarchy of "structure definition - algorithm design - scenario application" [11]. At the same time, this hierarchical thinking is combined with computer science practice. For example, when explaining cases related to program design, students are guided to divide the program development process into different levels such as requirements analysis, module design, code implementation, test optimization, etc. Each level corresponds to different knowledge modules in discrete mathematics. For example, in the requirements analysis stage, functional logic is sorted out based on mathematical logic. In the module design stage, graph theory is used to optimize the interrelationships between modules. Through this teaching approach, students can gradually develop the habit of handling problems logically and improve their ability to systematically analyze and solve professional problems [12].

3.3. Axiomatic thinking enhancement: Deepening the ability of abstract derivation based on Algebraic systems

In the teaching of more abstract chapters such as algebraic systems, with the goal of "strengthening axiomatic thinking", focus on explaining the basic operation rules and the axiomatic derivation process to help students master the method of deriving complex problems from abstract modules. The teaching process is no longer limited to formula memorization and operation training, but starts from the axiom definitions of the algebraic system, such as the basic operation rules of groups, rings, and fields, and elaborately deduces the theorem proving process under the axiom system, allowing students to understand the logical derivation chain of "axioms - theorems - inferences". For example, in the teaching of group theory, starting from the four axioms

of group closure, associative law, identity element, and inverse element, the relevant properties such as the order and subgroup of the group are derived, guiding students to think about how to abstract the core features of the algebraic structure through axiom definitions [13]. At the same time, in combination with abstract problems in computer science, such as the equivalent transformation of logical formulas and the application of solving algebraic equations, let students try to use axiomatic thinking to derive solutions to problems and understand the intrinsic connection between abstract axioms in discrete mathematics and professional problems [14]. Through this teaching model, students can gradually break away from their reliance on specific examples, learn to analyze complex problems from abstract axioms, enhance their abstract thinking and logical reasoning abilities, and lay the foundation for subsequent study of specialized courses such as cryptography and artificial intelligence that rely on abstract reasoning.

3.4. Synergistic optimization of practice cases and curriculum integration: Lowering the learning threshold and strengthening application cognition

Through the collaborative design of "practical case integration" and "course connection optimization", discrete mathematics teaching is more in line with the learning path of computer science, and students' application cognition is strengthened. In terms of adjusting the sequence of course content, the traditional teaching model of "theory first, then application" is broken. Modules closely related to the computer major, such as mathematical logic and set theory, are introduced first, and then gradually transition to abstract theoretical modules such as algebraic systems and combinatorial mathematics. For example, propositional reasoning in mathematical logic is explained first, and conditional judgment statements in programming are combined to help students understand logical relationships. Then delve into the quantitative analysis of predicate logic to lower the threshold for learning abstract theories [15]. At the same time, incorporate computer application scenario cases into the teaching of each module, such as introducing state transition analysis cases when teaching algorithm design-related content, and explaining state transition logic in programs in combination with knowledge of finite automata; In graph theory teaching, use network topology optimization as a case to have students design optimization schemes using knowledge of graph path finding, minimum spanning tree, etc. Through the teaching method of "learning relevant modules first + integrating practical cases", students can intuitively perceive the specific applications of discrete mathematics in professional fields such as programming, algorithm design, network optimization, etc., alleviate the resistance to abstract theories, and at the same time achieve a smooth connection between discrete mathematics and subsequent professional courses, and enhance the coherence and effectiveness of knowledge application [10].

4. Conclusion

Discrete mathematics, as the core vehicle for cultivating thinking in computer science, is directly related to the formation of the core competence of talents in terms of its teaching quality. This paper clarifies the necessity of strengthening professional thinking training from four aspects: supporting core courses, cultivating modeling ability, enhancing algorithmic literacy, and adapting to industry demands, and then proposes four strategies: modular dissection, hierarchical teaching, axiomatic derivation, and coordinated connection between practice and courses. These strategies are both in line with the disciplinary characteristics of discrete mathematics and closely meet the practical needs of computer science, and can provide path references for solving the teaching predicament of discrete mathematics, which is "difficult to learn theoretically and disconnected

from application." In the future, we can further explore the implementation methods and effect evaluation mechanisms of the strategies in combination with specific teaching scenarios, continuously optimize the teaching mode, contribute to the improvement of the quality of talent cultivation in the computer major, and provide high-quality talents with solid thinking ability for the development of the industry.

Funding

2025 Shanxi Province Higher Education Teaching Reform and Innovation Project, "Reform and Practice of Discrete Mathematics Curriculum for Application-Oriented Undergraduate Computer Major under the Background of '101 Plan'" (Project No.: J20250295)

Disclosure statement

The author declares no conflict of interest.

References

[1] He A, Pan L, Li T, 2025, Research on Teaching Innovation Reform of Discrete Mathematics Oriented by Computational Cognitive Thinking. Educational Teaching Forum, 2025(20): 67–71.

[2] Zhang S, Gao G, Guo X, 2024, Research on Inquiry-Based Teaching Mode Based on Computational Thinking Cultivation. Journal of Henan Institute of Science and Technology, 44(4): 8–16.

[3] Zhao J, Fan L, Zhang T, et al., 2024, Research on Teaching Mode of Discrete Mathematics Based on Computational Thinking. Neijiang Science and Technology, 45(3): 52–53 + 56.

[4] Lu S, Liu Q, 2023, Journal of Guangxi Radio and Television University. Journal of Guangxi Radio and Television University, 34(4): 83–87.

[5] Dong L, Yang M, Wen Z, 2023, Practice and Analysis of Teaching Reform in Discrete Mathematics. Modern Vocational Education, 2023(19): 129–132.

[6] Chen W, Zhou X, 2023, A Teaching Exploration of the Integration of Programming and Discrete Mathematics. Computer Education, 2023(3): 76–80.

[7] He C, Liu D, 2023, Computational Thinking and Course-Based Ideological and Political Education in Discrete Mathematics Curriculum. Computer Education, 2023(2): 79–82.

[8] Zhou X, Qiao H, Li L, 2022, Cultivation of Computational Thinking in Combinatorial Counting Teaching in Discrete Mathematics Curriculum. Computer Education, 2022(5): 1–5.

[9] Zhang R, Zhang F, Yang G, 2021, Research on Stratified Teaching of Discrete Mathematics Based on Computational Thinking Ability Cultivation. Zhengzhou Normal Education, 10(6): 70–72.

[10] Jia J, Li W, 2021, Teaching Reform of Discrete Mathematics Based on Computational Thinking for Ability Development. Computer Education, 2021(9): 152–155.

[11] Wu Z, 2021, Research on the Necessity of Embedding Applied Thinking into Discrete Mathematics Teaching. Science & Technology Vision, 2021(16): 29–30.

[12] Liu F, 2021, Cultivating Computational Thinking in Propositional Logic of Discrete Mathematics Curriculum. Computer Education, 2021(4): 151–154.

[13] Hou Y, 2020, Analysis of Discrete Mathematics Teaching in the Era of Big Data. New Curriculum Teaching

(Electronic Edition), 2020(22): 116–117.

[14] Zhang Y, Bao Y, 2020, Research on Computational Thinking Oriented Discrete Mathematics Teaching Mode. Science & Technology Wind, 2020(21): 38–39.

[15] Jin Y, Hu Y, Tian Q, et al., 2020, A Case Study of Discrete Mathematics Teaching: Taking Equivalence Relations as an Example. Educational Teaching Forum, 2020(29): 264–266.