Creative Education Studies 创新教育研究, 2025, 13(4), 505-513
Published Online April 2025 in Hans. https://www.hanspub.org/journal/ces
https://doi.org/10.12677/ces.2025.134275

Hans汉斯

面向高阶性的数据结构与算法课程教学内容创新设计

——以图的应用为例

吴一尘*，杨国正，刘　俊，卢灿举，朱凯龙

国防科技大学电子对抗学院，安徽　合肥

收稿日期：2025年3月10日；录用日期：2025年4月18日；发布日期：2025年4月28日

摘　要

数据结构与算法是计算机相关专业一门重要的专业基础课程，针对当前课程教学中存在的课时少与内容多的突出矛盾，受数学领域“多题一解”理念启发，提出一种将多个算法整合成统一算法框架的数据结构与算法课程教学内容改革方法，采用统一算法框架开展课程教学，学生在掌握其中一种算法之后，能够快速完成对其余算法的学习，从而能够在有限的课时内高效完成课程教学内容的传授，提升高阶教学目标的达成度。

关键词

高阶性，多题一解，统一算法框架，图的应用

Innovative Design of Teaching Content for High-Order Data Structure and Algorithm Courses

—Taking the Application of Graphs as an Example

Yichen Wu*, Guozheng Yang, Jun Liu, Canju Lu, Kailong Zhu

College of Electronic Engineering, National University of Defense Technology, Hefei Anhui

Received: Mar. 10th, 2025; accepted: Apr. 18th, 2025; published: Apr. 28th, 2025

*通讯作者。

文章引用: 吴一尘, 杨国正, 刘俊, 卢灿举, 朱凯龙. 面向高阶性的数据结构与算法课程教学内容创新设计[J]. 创新教育研究, 2025, 13(4): 505-513. DOI: 10.12677/ces.2025.134275

Abstract

Data structures and algorithms is an important basic professional course for computer-related majors. In view of the prominent contradiction between the limited class hours and the extensive teaching content in the current curriculum teaching, inspired by the concept of "solving multiple problems with one solution" in the field of mathematics, a reform method of the teaching content of the data structures and algorithms course is proposed. This method integrates multiple algorithms into a unified algorithm framework. By adopting this unified algorithm framework for curriculum teaching, after students master one of the algorithms, they can quickly complete the learning of the remaining algorithms. In this way, the teaching content of the course can be efficiently imparted within the limited class hours, and the achievement level of high-order teaching objectives can be improved.

Keywords

High-Order Nature, Solving Multiple Problems with One Solution, Unified Algorithm Framework, Application of Graphs

1. 引言

2018 年 8 月，教育部印发《关于狠抓新时代全国高等学校本科教育工作会议精神落实的通知》[1]，提出各高校要全面梳理各门课程的教学内容，淘汰“水课”、打造“金课”。同年 11 月，时任教育部高等教育司司长吴岩在第十一届“中国大学教育论坛”上提出金课应满足“两性一度”要求，即高阶性、创新性和挑战度[2]。2019 年 10 月，教育部发布《关于一流本科课程建设的实施意见》[3]，提出全面开展一流本科课程建设，经过三年左右时间，建成万门左右国家级和万门左右省级一流本科课程(简称一流本科课程“双万计划”)。

在这样的大背景下，针对课程教学内容的优化研究成为教学改革的热点。数据结构与算法课程作为计算机相关专业的一门核心课程，许多学者开展了针对该课程的教学改革，其中，刘城霞[4]基于 OBE (Outcomes-Based Education)理念，从学习成果出发逆向改进课程教学内容，在课程理论教学部分增加问题引入和新技术介绍，同时增加实践环节的权重，在练习基本数据结构如何编程实现的基础上，侧重综合利用数据结构和算法解决具体问题。王彤等[5]设计了“线上验证型、设计型和线下应用探究式综合创新型”三层次的数据结构课程实验教学内容，培养学生解决复杂问题的综合能力和高级思维。王晓明[6]构建了“基础知识、应用实践和创新研究”三层次的数据结构课程内容体系，着力解决“教学内容缺挑战”的教学问题。王燕等[7]以建设高标准且以能力培养为导向的数据结构课程为目标，在现有教学内容的基础之上，扩充了决策树等知识点，增加了城市道路规划等应用案例。赵海丽等[8]针对教学内容多而分散，导致学生不会用的问题，按照在理解理论知识的基础上，总结归纳知识点之间的关系，并将相关联的知识点进行整合与升华，产生新的知识点的思路，重塑并构建了多层次的教学内容。

从上述研究工作可以看出，多数学者认为数据结构与算法课程教学内容的改革应以培养学生解决复杂工程问题的综合能力和计算思维为目标，这与“两性一度”中的高阶性不谋而合，大部分学者采用对

课程教学内容进行分层设计，在现有教学内容的基础之上引入新问题、新技术的做法，为课程教学效果的提升提供了有益参考。然而，如何在有限的课时内高效完成课程教学内容的传授，顺利达成既定的课程学习目标是极为现实且亟待解决的问题。本文以严蔚敏等[9]编著的《数据结构》(C 语言版 第 2 版)为参考教材，以其中“图的应用”小节为例，探讨数据结构与算法课程教学内容的创新设计，以期为解决该问题提供参考。

2. 教学中存在的问题

数据结构与算法课程是面向我校计算机相关专业的重要学科基础必修课，主要介绍表、树、图等基础数据结构和分治法、贪心法、动态规划法、回溯法等算法设计典型策略，培养学生利用计算思维采用程序化方法解决复杂工程问题的能力，教学内容与学时安排如表 1 所示。

Table 1. Teaching content and class hour arrangement of the data structures and algorithms course
表 1. 数据结构与算法课程教学内容与学时安排

教学内容	理论讲授学时	上机实验学时
基础知识	3	0
线性表	6	2
栈和队列	4	4
串、数组和广义表	3	0
树和二叉树	6	2
图	6	2
查找	6	4
排序	6	2
典型算法策略	8	0
合计	48	16

在课程教学过程中，课时少与内容多的矛盾比较突出。以“图”一章为例，其核心知识点包括图的定义和基本术语、图的存储结构、图的遍历以及图的应用。其中，图的存储结构主要介绍邻接矩阵和邻接表两种图的存储结构以及图的创建算法；图的遍历主要介绍图的深度和广度优先搜索遍历算法；图的应用主要介绍最小生成树、单源最短路径、拓扑排序和关键路径四个问题及其相应的求解算法。这一章总计需要学习 9 个算法，学生在短时间内学习如此大量的算法，多是“走马观花”“囫囵吞枣”，难以理解内化。按照布鲁姆认识领域教学目标分类设计测试内容考查学生的学习效果，结果显示高阶教学目标达成度不足，极少有学生能够达到“分析”及更高层次。例如，给定一张图，绝大部分学生能够根据 Prim 算法的基本思想画出图的最小生成树，部分学生能够编写程序，基于邻接矩阵/邻接表存储结构实现 Prim 算法，但几乎没有学生能够针对 Prim 算法采用不同存储结构实现时的效率差异展开分析。

3. 教学内容创新设计

针对上述课程教学中存在的问题，受数学领域“多题一解”理念启发，本文开展了数据结构与算法课程教学内容的创新设计，以“图的应用”小节为例，构建统一的算法框架整合 Prim 算法、Dijkstra 算法和拓扑排序算法的教学内容。基于这一统一算法框架，学生在掌握 Prim 算法之后，能够快速完成对其余两个算法的学习，并且能够借助框架中的关键函数，针对同一算法的不同实现方式进行效率差异分析。同时，学生得以“透过现象洞察本质”，探寻这些算法之间的共性，进而运用算法框架解决新的问题。

3.1. 构建统一算法框架

(1) 最小生成树问题

给定带权连通图 G = (V, E)，其中，V 和 E 分别是图 G 的顶点集合和边集合，包含图 G 中全部顶点的各边权值之和最小的生成树被称为图 G 的最小生成树。

求解该问题的 Prim 算法的基本思想是：定义顶点集合 U，存储已经加入最小生成树的顶点，初始时，从图 G 中任意挑选一个顶点加入集合 U。然后进行|V|−1 次迭代，每一次迭代从集合 V-U 中选择与集合 U 中的顶点有边相连并且权值最小的一个顶点，这条权值最小的边就是最小生成树中的一条边，将该顶点加入集合 U。

(2) 单源最短路径问题

对于带权连通图 G = (V, E)，给定顶点 a 和 b，a 到 b 的路径上各边权值之和称为这条路径的长度，a 到 b 可能不止一条路径，其中长度最小的路径称为 a 到 b 的最短路径。单源最短路径问题指的是，指定集合 V 中的一个顶点 s 作为源点，求从 s 出发到图 G 中其余各顶点的最短路径。

求解该问题的 Dijkstra 算法的基本思想是：定义顶点集合 U，存储已经求出最短路径的终点，初始时，将源点 s 加入集合 U。然后进行|V| − 1 次迭代，每一次迭代从集合 V-U 中选择从源点 s 出发直接到达该顶点或者仅经过集合 U 中的顶点最后到达该顶点的最短路径长度最小的顶点，将该顶点加入集合 U 中。

(3) 拓扑排序问题

对于有向无环图 G = (V, E)，对图 G 中的顶点进行拓扑排序，可以得到所有顶点的一个线性序列，该序列满足：对于顶点集 V 中的任意两个顶点 a 和 b，如果图 G 中存在从顶点 a 到 b 的一条路径，则该线性序列中顶点 a 必定在 b 之前。

拓扑排序算法的基本思想是：定义顶点集合 U，存储已经加入拓扑序列的顶点，初始时，从图 G 中挑选一个入度为 0 的顶点加入集合 U，将该顶点和以它为弧尾的弧从图 G 中删除。然后进行|V| − 1 次迭代，每一次迭代从集合 V-U 中选择一个入度为 0 的顶点加入集合 U，同时从图中删除该顶点和所有以它为弧尾的弧。

(4) 统一算法框架

通过对三个算法基本思想的分析可以发现，虽然这三个算法求解的问题并不相同，但从对图的操作的角度来看，却具有较大的相似性，本质上都可以看成是对图中顶点的选择，三个算法都将图中的顶点分成两个子集 U 和 V-U，在初始化操作之后进行|V|−1 次迭代，每一次迭代按照某种策略从集合 V-U 中选择一个顶点加入集合 U，因此，可以为这三个算法构建统一的算法框架，如算法 1 所示。

Algorithm 1. Unified algorithm framework for graph applications
算法 1. 图的应用统一算法框架

```
初始化顶点集合 U;
初始化图中所有顶点的状态;
K = 1;
while k <= |V| − 1 do
    采用某种选点策略从集合 V-U 中选择一个顶点;
    将选中的顶点加入集合 U;
    更新集合 V-U 中剩余顶点的状态;
    k 减 1;
```

3.2. 数据结构和关键函数

从算法 1 能够看出，在算法实现时需要解决以下 5 个问题：第 1 个问题是图的存储结构，第 2 个问题是针对图中的顶点，标记它属于集合 U 还是集合 V-U，第 3 个问题是记录集合 V-U 中的顶点针对算法所采用的选点策略的当前状态，第 4 个问题实现算法 1 中第 5 步对应的选点函数，第 5 个问题是实现算法 1 中第 7 步对应的顶点状态更新函数。

(1) 图的存储结构

本文采用教材中的邻接矩阵表示法存储图，用一个长度为|V|的一维数组 vexs 来存储顶点信息，用一个|V|行|V|列的二维数组 arcs 来存储边。例如，图中顶点 a 和 b 之间有一条权值为 3 的边，假设顶点 a 和 b 存储在数组 vexs 的第 1 和 2 号单元，则 arcs [1] [2] = 3。

(2) 标记顶点所属的集合

对于图中的所有顶点，需要标记它属于集合 U 还是集合 V-U，可以采用辅助数据结构来记录这一信息。定义一个长度为|V|的 bool 类型一维数组 visited 用来存储顶点所属的集合，例如，顶点 a 和 b 分别属于集合 U 和集合 V-U，假设顶点 a 和 b 存储在数组 vexs 的第 1 和 2 号单元，则 visited [1] = true，visited [2] = false。

(3) 记录顶点的当前状态

为了辅助选点函数的实现，对于集合 V-U 中的每一个顶点，需要记录其针对算法所采用的选点策略的当前状态。Prim 算法需要记录连接该顶点与集合 U 中顶点的权值最小的边；Dijkstra 算法需要记录从源点 s 出发直接到达该顶点或者仅经过集合 U 中的顶点最后到达该顶点的最短路径；拓扑排序算法需要记录该顶点的入度。

可以定义一个长度为|V|的一维辅助数组 status 来记录顶点的当前状态，根据算法需要记录的状态的不同，辅助数组的类型有所不同。拓扑排序算法需要记录顶点的入度，定义一个 int 类型数组即可；Prim 算法需要记录一条边，包含两个顶点以及权值共三项信息，其中，属于集合 V-U 的顶点利用数组 status 与数组 vexs 之间的下标映射来记录，因此，只需要记录这条边的属于集合 U 的顶点和权值；Dijkstra 算法需要记录一条路径，包含路径中经过的顶点以及路径长度两项信息，其中，对于路径中经过的顶点，可以只记录目标顶点的前驱顶点，最后通过反向推导的方式还原路径。根据上述分析，Prim 算法和 Dijkstra 算法可以定义类型相同的辅助数组 status，如表 2 所示，区别在于其中数据项的语义。

Table 2. The type definition of the auxiliary array status in Prim's algorithm and Dijkstra's algorithm
表 2. Prim 算法和 Dijkstra 算法中辅助数组 status 的类型定义

```
1:      typedef struct{
2:          int index;
3:          int length;
4:      }Status;
```

表 1 第 2 行定义的 index 在 Prim 算法中表示所记录的边依附的属于集合 U 的顶点在数组 vexs 中的下标，在 Dijkstra 算法中表示所记录的最短路径中目标顶点的前驱顶点在数组 vexs 中的下标；第 3 行定义的 length 在 Prim 算法中表示边的权值，在 Dijkstra 算法中表示最短路径的长度。

(4) 选点函数的实现

数组 status 存储了顶点针对选点策略的当前状态，因此，选点函数只需要对数组 status 进行遍历即可选出下一个要加入集合 U 的顶点。需要注意的是，随着顶点不断从集合 V-U 移动到集合 U 中，数组 status

中的部分记录对应的顶点不属于集合 V-U，不应该参与接下来的选择，因此，选点函数需要同时遍历数组 visited，将这部分顶点排除出选择范围。

Prim 算法根据数组 status 中记录的边的权值选点，Dijkstra 算法根据数组 status 中记录的最短路径的长度选点，从语义上来看选点策略是不同的，但从语法层面来看，都是在数组 status 中选择属于集合 V-U 并且 length 最小的顶点。根据上述分析，Prim 算法和 Dijkstra 算法可以定义相同的选点函数 Select，如算法 2 所示。

Algorithm 2. The point selection functions of Prim's algorithm and Dijkstra's algorithm
算法 2. Prim 算法和 Dijkstra 算法的选点函数

```
1:  int Select (Status status){
2:      int min = ∞, pos = 0;
3:      For (int i = 1; I <= |V|; i++){
4:          If (visited [i] == false && min > status [i]. length){
5:              min = status[i].length;
6:              pos = i;
7:          }
8:      }
9:      return pos;
10: }
```

拓扑排序算法根据数组 status 中记录的入度选点，只需要将算法 2 中第 4 至 7 行对应的 if 语句修改为 if (visited [i] == false && status [i] == 0){pos = i; break;}即可。算法 2 包含一个|V|次的 for 循环，时间复杂度为 O (|V|)。

(5) 顶点状态更新函数的实现

当将选中的顶点 u 加入集合 U 之后，该顶点的属于集合 V-U 的邻接点的状态可能会发生变化。例如，顶点 a 有 2 个邻接点 b、c，其中，顶点 b 属于集合 U，顶点 a、c 属于集合 V-U，当顶点 a 被选中之后，在 prim 算法中，边(c, a)有可能成为连接顶点 c 与集合 U 中顶点的权值最小的边；在 Dijkstra 算法中，从源点 s 到顶点 c 的最短路径可能会变成从源点 s 到顶点 a 的最短路径再加上边(a, c)；在拓扑排序算法中，顶点 c 的入度会减少 1。根据上述分析，顶点状态更新函数需要遍历被选顶点的所有邻接点，更新数组 status 中被选顶点的属于集合 V-U 的邻接点的状态信息。Prim 算法的顶点状态更新函数 Update 如算法 3 所示。

Algorithm 3. The vertex state update function of Prim's algorithm
算法 3. Prim 算法的顶点状态更新函数

```
1:  int Update (Graph G, Status status, int u){ //u 表示被选中的顶点
2:      For (int i = 1; I <= |V|; i++){
3:          If (visited [i] == false && G.arcs [u] [i] < status [i]. length){
4:              status [i]. length = G.arcs [u] [i];
5:              status [i]. index = u;
6:          }
7:      }
8:  }
```

Dijkstra 算法的顶点更新函数只需要将算法 3 中第 3 行对应的 if 判断条件修改为 if (visited [i] == false

&& status [u]. length + G.arcs [u] [i] < status [i]. length)即可。拓扑排序算法的顶点更新函数只需要将算法 3 中第 3 至 6 行对应的 if 语句修改为 if (visited [i] == false) {status [i]--;}即可。算法 3 包含一个|V|次的 for 循环，时间复杂度为 O (|V|)。

3.3. 算法效率分析优化

教材在介绍 Prim 算法、Dijkstra 算法和拓扑排序算法的时间复杂度时给出的结论是：Prim 算法和 Dijkstra 算法的时间复杂度为 O (|V|^2)，拓扑排序算法的时间复杂度为 O (|V| + |E|)。这一结论当然是正确的，但利用本文提出的统一算法框架和关键函数 Select、Update，可以针对这三个算法在采用不同实现方式时所表现出的时间效率差异进行更详细的分析。

算法 1 中包含一个|V| − 1 次的 for 循环，每一次循环都需要调用 Select 函数和 Update 函数，因此，算法 1 的时间复杂度均为 O (|V|^2)。由于算法 1 中|V| − 1 次的 for 循环是无法改变的，对算法 1 的时间复杂度优化只能从关键函数 Select 和 Update 入手。

Update 函数对被选顶点的属于集合 V-U 的邻接点的状态信息进行更新，算法 3 中第 2 行的 for 循环是遍历邻接矩阵 arcs 中被选顶点所在的行产生的时间开销，因此，如果改用邻接表存储图，这个 for 循环的执行次数会从|V|次变为|Eu|次，其中，|Eu|表示被选顶点的邻接点的个数，Update 函数的时间复杂度从 O (|V|)变为 O (|Eu|)。Select 函数通过遍历数组 status 来选择顶点，算法 2 中第 3 行的 for 循环是遍历数组 status 产生的时间开销，因此，改用邻接表存储图并不能改变这个 for 循环的执行次数，Select 函数的时间复杂度仍然是 O (|V|)。根据上述分析，采用邻接表存储图，算法 1 的时间复杂度为 O (|V|*(|V| + |Eu|))，整理之后得到 O (|V|^2 + |E|)，由于图的边数|E|最多和|V|^2 同数量级，因此，可以将时间复杂度进一步简化为 O (|V|^2)。根据上述分析能够得出两点结论，一是单纯改变图的存储结构并不能改变算法 1 的时间复杂度；二是 Select 函数是影响算法 1 时间效率的瓶颈。

在 Prim 算法和 Dijkstra 算法中，Select 函数从数组 status 中选择属于集合 V-U 并且 length 最小的顶点，如果将数组 status 建成一个小根堆，选点操作的时间复杂度会从 O (|V|)降低为 O (1)，在堆顶被选中后将其从堆中删除，从而保证堆中保存的都是集合 V-U 中的顶点对应的状态信息，因此不需要同时对数组 visited 进行遍历，删除堆顶之后需要对堆进行向下堆化，时间复杂度为 O (log|V|)。根据上述分析，将数组 status 建成一个小根堆，Select 函数的时间复杂度会从 O (|V|)降低为 O (log|V|)。

需要注意的是，这样做同时会影响 Update 函数的时间复杂度。一方面，堆不具备随机存取的能力，无法在 O (1)的时间访问需要更新状态的顶点，可以再引入一个散列表，建立从属于集合 V-U 的顶点到小根堆中对应单元的映射从而解决该问题。另一方面，每更新堆中一个顶点的状态，都需要对堆进行向上堆化，时间复杂度为 O (log|V|)。因此，采用邻接矩阵存储图时，UpdatePath 函数的时间复杂度会变成 O (|V|log|V|)，而采用邻接表存储图时，UpdatePath 函数的时间复杂度会变成 O (|Eu|log|V|)。

根据上述分析，对于 Prim 算法和 Dijkstra 算法，将辅助数组 status 组织成小根堆并引入一个散列表，采用邻接矩阵存储图时，算法 1 的时间复杂度为 O (|V|^{2}log|V|)，而采用邻接表存储图时，算法 1 的时间复杂度为 O (|E|log|V| + |V|log|V|)，由于连通图的边数|E|至少和|V|同数量级，因此，可以将时间复杂度进一步简化为 O (|E|log|V|)。由此可见，采用这种实现方式，当图比较稀疏(|E| < |V|^2)并且采用邻接表存储时，Prim 算法和 Dijkstra 算法的时间复杂度可以从 O (|V|^2)降低为 O (|E|log|V|)。

在拓扑排序算法中，Select 函数从数组 status 中选择入度为 0 的顶点，可以再引入一个栈或者队列，将入度为 0 的顶点压入栈或队列中，这样，Select 函数只需要从栈顶或队头取出一个顶点即可，时间复杂度从 O (|V|)降低为 O (1)，这样做并不会影响 UpdatePath 函数的时间复杂度。因此，采用邻接矩阵存储图时，拓扑排序算法的时间复杂度为 O (|V|*(1 + |V|))，整理之后得到 O (|V|^2)，而采用邻接表存储图时，拓

扑排序算法的时间复杂度为 O (|V|*(1 + |Eu|))，整理之后得到 O (|V| + |E|)，即教材中给出的结论。

上述实现方法中引入的辅助数据结构大小均与|V|同数量级，因此空间复杂度均为 O (|V|)。

3.4. 方法应用与拓展

无论算法求解的是什么问题，如果对图的操作是采用某种策略在图中进行顶点的选择，都可以统一到算法 1 的框架中。在多目标优化问题中，由于存在多个目标函数，通常会产生一组最优解，这组解中的任何一个都不比其它的解更好，被称为帕累托最优解。文献[10]提出了一种快速非支配排序算法求解帕累托最优解，该算法可以看成是采用和拓扑排序算法类似的选点策略在有向图中选择顶点，区别在于该算法的选点函数每次选择的是一组顶点，而不是一个顶点。

将种群中的 N 个解看成是图 G 中的 N 个顶点，对于解 p 和 q，如果 p 支配 q，则图 G 中存在弧<p, q>，反之如果 p 被 q 支配，则图 G 中存在弧<q, p>。统计图 G 中全部 N 个顶点的入度，将入度为 0 的顶点加入集合 F_1，这些顶点对应的解就是第一帕累托前沿的解，可以看成是算法 1 中初始化时加入集合 U 的起点，将集合 F_1 中的顶点从图 G 中删除，同时删除以这些顶点为尾的弧。接下来进入迭代过程：选择图 G 中所有入度为 0 的顶点加入集合 F_i，这些顶点对应的解就是第 i 帕累托前沿的解，将集合 F_i 中的顶点从图 G 中删除，同时删除以这些顶点为尾的弧。重复这一迭代过程直到图 G 中顶点为空。

4. 结论

根据课程教学计划，图一章有 6 个理论课时和 2 个实验课时，在以往的教学实施中，6 个理论课时只能讲完图的定义和基本术语、图的存储结构、图的遍历以及图的应用中的最小生成树问题，对单源最短路径问题、拓扑排序问题以及关键路径问题的学习通过占用 2 个实验课时来完成，并且受课时限制，在讲授中主要关注算法的基本思想和编程实现，对于算法效率直接给出结论，不做过多的讨论。2024 年秋季学期课程组调整了图一章的教学安排，采用线上线下混合式教学方法，将图的定义和基本术语调整为线上自学，利用 4 课时讲完图的存储结构以及图的遍历，然后利用 2 课时采用本文提出的统一框架讲授图的应用一节。第 1 课时介绍 Prim 算法的基本思想、算法框架和关键函数，课后通过教学视频、实训项目等形式布置学生自学 Dikstra 算法和拓扑排序算法。第 2 课时对这三个算法采用不同实现方式所产生的效率差异进行讨论，同时引导学生探寻这些算法之间的共性，课后通过文献资料、拓展实训等形式布置学生运用算法框架解决新的问题。通过这种方式在 6 课时内完成图一章的理论讲授，提升了高阶教学目标的达成度。

这种采用统一框架整合教学内容的方法也可以应用在其它章节。例如，在树和二叉树一章中，教材仅介绍了二叉树的中序非递归遍历算法，但无论是二叉树的先序、中序还是后序遍历，其相似之处在于都是沿着二叉树的外轮廓线进行周游，区别仅在于访问节点的时机不同，根据这一思路可以为二叉树的三种非递归遍历算法构建统一算法框架。

基金项目

本文得到安徽省教育厅高等学校省级质量工程教学研究重点项目“数据结构与算法课程高阶性教学研究”(2022jyxm1210)资助。

参考文献

[1] 中华人民共和国教育部. 关于狠抓新时代全国高等学校本科教育工作会议精神落实的通知[EB/OL]. 2018-08-27. http://www.moe.gov.cn/srcsite/A08/s7056/201809/t20180903_347079.html, 2025-02-18.

[2] 西安电子科技大学电子工程学院本科教学办公室. 建设中国金课——吴岩司长在“中国大学教学论坛”上的讲话

[EB/OL]. 2018-11-30. https://see.xidian.edu.cn/html/news/9768.html, 2025-02-18.

[3] 中华人民共和国教育部. 关于一流本科课程建设的实施意见[EB/OL]. 2019-10-30. http://www.moe.gov.cn/srcsite/A08/s7056/201910/t20191031_406269.html, 2025-02-18.

[4] 刘城霞. 基于 OBE 的数据结构教学的可持续改进研究[J]. 教育进展, 2021, 11(3): 728-734.

[5] 王彤, 鲍玉斌, 杨雷, 张立立, 刘佳良. 数据结构实验金课建设教学探索与实践[J]. 实验室研究与探索, 2021, 40(5): 184-192.

[6] 王晓明. 新工科建设背景下的数据结构课程改革与实践[J]. 教育进展, 2022, 12(5): 1514-1518.

[7] 王燕, 罗佳琪, 王曙燕, 潘晓英, 曾艳, 白琳. 能力导向的数据结构课程“五环节”混合教学改革[J]. 计算机教育, 2023(1): 183-186.

[8] 赵海丽, 胡克用, 王李冬. 以能力培养为导向的数据结构课程多层次循环教学新生态构建[J]. 计算机教育, 2024(2): 76-80.

[9] 严蔚敏, 李冬梅, 吴伟民. 数据结构(C 语言版) [M]. 第 2 版. 北京: 人民邮电出版社, 2022.

[10] Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002) A Fast and Elitist Multiobjective Genetic Algorithm: Nsga-II. *IEEE Transactions on Evolutionary Computation*, **6**, 182-197. https://doi.org/10.1109/4235.996017