程序设计课程教学中的阶段回溯教学法

汪梅婷
燕山大学里仁学院，中国·河北 秦皇岛 066004

【摘　要】阶段回溯法从程序设计课程入门易、学透难的特点出发，将复习环节细分为近端回溯、章节回溯、综合回溯三个阶段，在分段回溯中实现知识的交叉覆盖，逐步构建知识网络图，以分段强化的方式逐渐加大知识的广度和深度。采用定量分析的方法，将学生分为实施组和参照组进行教学实践，数据分析显示，阶段回溯法适合程序设计课程，有利于教学效果的提升。

【关键词】阶段回溯教学法；计算机教学；教学改革
【课题】2019-2020年度河北省高等教育教学改革研究与实践项目,《计算机技术基础A》混合式课堂教学模式研究(2019GJJG585)；2020年燕山大学里仁学院教学改革与研究项目，《Python语言程序设计A》课程思政教学改革（LRJG202015）。

1　阶段回溯教学法

阶段回溯教学法是一种融复习与授课阶段的教学方法，主要用于不同目标的复习环节。采用知识重叠的方法，形成内容的交叉覆盖，通过分段强化的方式，在初学阶段、章节复习阶段、综合复习阶段进行前后关联且有差别的提升练习与扩展，在知识的多次回溯中逐步实现知识的牢固记忆、深入理解、灵活运用，以及编程能力的提升。

1.1 回溯时机与回溯点的选取

回溯是对已学内容的巩固和提高，回溯时间点的选择遵循适度、适量的原则，在内容全覆盖的基础上，实现对章节重点和难点有侧重的回溯，达到掌握基础知识、了解变化形式、适应复杂应用、编程解决实际问题的目标，以复习带动新知识的学习，在多阶段回溯中不断提高学习效果和编程能力。

回溯时机应选在记忆淡化之前进行，依据遗忘曲线理论，在初次学习之后的短期、中期、长期分阶段进行。不同阶段在回溯目标、知识覆盖面、用例难度、用例时长均有不同，都要符合教学大纲的要求。回溯点选取应覆盖教学大纲规定的必须掌握的全部内容，在课时允许的情况下适当覆盖课外扩展内容。覆盖的知识点主要从重点、易错点和编程常用点中选取，用例库应实现不同时间点的多次交叉覆盖，单点覆盖次数不应小于3次，可以采用单点独立练习或多点综合练习的方式。

1.2 回溯阶段划分

按照与初次讲解的间隔时长，回溯可以分为短期、中期、长期三种类型，分别对应近端回溯、章节回溯和综合回溯。

近端回溯属于短期回溯，适合初学时的课内回溯，在基本例题讲完后即可进行。近端回溯在初次记忆强化中发挥着不可忽视的作用，初次学习效果与近端回溯的开始时间、回溯频率呈正相关性。开始得越早，单点回溯频率越高，知识初次理解的程度越高，初次学习效果越好。

章节回溯属于中期回溯，时间点为一章内容讲解完毕后，结合相邻章节的内容来设计用例，既要体现知识的连贯性，又要体现综合性。通过对知识的再发现、再理解来达到深入理解、熟练掌握的目标，起到承上启下的作用。综合回溯属于长期回溯，在课程的后期阶段进行。综合回溯在知识网络的整体性、多章节内容的综合性、知识点的活学活用等方面较其他阶段更为突出。因此，在知识的广度和深度上要求高，回溯时应以实际知识掌握情况为基础，注重实效，满足教学目标的要求。

2　教学用例设计方法与实例

教学用例设计的质量与阶段回溯法的实施效果呈正相关，不同实施阶段的用例设计要求差别较大，用例难度随时间的推移渐进加深。下面以Python语言中的单重for循环为例，对理论教学中的不同阶段的回溯用例设计要求和方法进行探讨。

2.1 近端回溯用例设计

近端回溯是对已学内容的首次回溯，选在语法结构初次讲解后的课内进行，以语法中具有共性的易错点和难理解的变化点作为回溯的内容点为宜。回溯用例应与讲解用例有明显的区别，程序代码以覆盖多目标点、简短精巧、易读易讲、用时较短为宜，数量适中，能在规定学时内完成。

用例设计内容选取了结构化程序设计的三种基本结构之一的循环结构[1]，单重for循环是循环结构一章的重点，教学大纲对要求通过课程学习达到灵活使用单重for循环解决实际问题的目标。

【阶段划分】初次回溯——易错点。

【时间点】选在随堂练习、课程小结等环节，用于课内知识强化。

【用例目标】理解不同的缩进对执行过程的影响，能正确分析代码并推出运行结果，解决两个易错点：一是代码缩进，二是循环变量的取值范围。

【用例设计】假设代码中的变量均已正确定义，执行下列代码后的s的值。

```
第1行：s=0
第2行：for i in range(5):
第3行：    pass
第4行：s=s+i
第5行：print(s)
```

【用例分析】由于第2行和第4行代码的缩进相同，根据语法规则，第4行代码s=s+i不是第2行代码的循环体，只在for循环执行结束后执行一次。循环变量i的取值由range(5)确定，而range(5)的取值范围是0~4。第2行的循环结束时i=4，因此，s=0+4=4。

2.2 章节回溯设计

章节回溯的过程即为知识网络构建的过程，既要实现单章内容的综合深化，又要兼顾章间内容的综合，构建不同章节知识的内在联系，还要对后续待章节进行知识铺垫。用例设计方面以中等或中高难度为宜，代码量适中，以编程中的常用知识作用例为佳，覆盖的点要多，知识容量要大，体现语句用法的变化，且能与近端回溯用例形成联系与对比。简单回溯时与近端回溯用例有重复，多点回溯时与近端回溯有交叉。

【阶段划分】中期回溯——变化点。

【时间点】选在章节复习阶段进行。

【用例目标】远端回溯前几章的print()、range()、%运算符、if语句、条件表示，对本章的for循环、continue语句进行中期回溯，引入后续章节的list类型[2]，为讲解新内容做铺垫。回溯过程中串联多章知识，加深对知识内在联系的理解，逐步勾勒知识网络图。

【用例设计】假设代码中的变量均已正确定义，执行下列代码后的输出结果。

```
第1行：x=[1,2,3,4,5,6,7,8,9,10]
第2行：s=0
第3行：for i in range(1,10,2):
第4行：    if x[i]%2:
第5行：        continue
第6行：    else:
第7行：        s=s+i
第8行：    print(s,end=',')
```

【变化回溯点】一是range()用法，二是选择结构，三是数值做条件，四是列表的下标范围，五是continue语句对执行流程的影响，六是缩进对代码逻辑结构的影响，七是print()的参数变化。

【回溯点分析】远端回溯点涉及第3~8行。第3行的range(1,10,2)为变化点，步长值从常见的默认值1变为2，i的取值为1、3、5、7、9。第4行的条件为变化回溯，复习第2章的运算符与表达式，以及第3章的if语句，变化点为数值作条件。第8行的函数参数为变化点，从常见的输出后换行变为不换行，此处先输出变量s的值再后接一个逗号。近端回溯点包括第3~8行的单重for循环和continue语句，分析用例可以加深对continue语句改变for循环执行过程的理解，还能掌握利用与if条件的结合实现指定条件下的流程跳转。

【知识拓展】一是第3行，可以扩展到任意步长值的循环语句书写；二是第5行结合continue语句与break语句的区别展开讨论；三是第8行，由print()参数设置扩展至输出指定内容连接指定字符串。

【章节过渡】用例涉及选择结构、循环结构、组合数据类型共三章内容，未来铺垫点只有第1行，变量x的类型为下一章将要讲授的组合数据类型，借助该用例实现下一章内容的自然过渡。

2.3 综合回溯设计

综合回溯适合课程的总复习，是打通全书章节内容构建课程知识网络的关键一环，对编程技巧和知识的灵活运用提出了更高的要求，是所有回溯阶段中难度最大的，耗时较多。用例设计以中、高难度为主，与近端回溯、章节回溯均有交叉。用例覆盖面广，在与前期回溯相联系的基础上应体现出变化和对比。用例代码相对较长，理解上有难度，首次答对率偏低，单个用例的易错点应在2个以上，体现用例的高阶性。

【阶段划分】后期回溯——综合运用。

【时间点】选在期末综合复习阶段进行。

【用例目标】在总复习阶段对重点、难点和具有共性的易错点进行再次强化练习，与章节回溯阶段的用例进行差异对比，培养编程能力和技巧。

【用例设计】假设代码中的变量均已正确定义，执行下列代码后的输出结果。

```
第1行：x=[1,2,3,4,5,6,7,8,9,10]
第2行：s=0
第3行：for i in range(1,10,3):
第4行：    if x[i]//2>2:
第5行：        continue
第6行：    s=s+i
第7行：else:
第8行：    print(s,end='@')
```

【综合回溯点】一是list类型，二是循环结构，三是选择结构，四是运算符优先级与用法，五是表达式的求解，六是continue语句，七是for循环的else子句，八是缩进对代码逻辑结构的影响，九是print()的参数。

【回溯点分析】第1~3行为简单回溯，以复习语法为主。第4行为综合变化回溯，先定运算符的优先级再求解表达式的值，最后根据该值是否为零来判断条件成立与否。第6~8行是对比回溯，看似与章节回溯用例相似，实际差别较大。由于第7行无缩进，该else子句属于for循环，代码输出结果为12@。通过对比熟悉缩进对代码执行过程的影响，提高编程能力和技巧。

3 实施过程与效果分析

3.1 实施过程

以开设本课程的授课班级作为全体，以合班为单位，通过随机分组的方式将班级分为实施组和对照组，进行分阶段的教学实践对比研究。实践过程中，选取循环结构、指针等知识理解和应用难度较大的章节作为测试，以实施组的测验数据为样本，以单次测验成绩为最小单位进行随机抽样，进行不同回溯阶段的应用效果对比分析。

3.2 实施效果分析

（1）阶段对比数据分析。图1展示的是循环结构程序设计部分的答题数据对比情况。测试时间点有两个：时间点1为章节回溯阶段，时间点2为综合回溯阶段。两次数据采集时间差大于6周，能部分降低记答案对数据分析造成的影响。样本容量为个1个合班，由4个自然班组成。测验用题相同，试题容量为48道，由基础题、简单应用题、综合应用题组成。定量分析显示，经综合回溯后知识点掌握情况普遍提高。

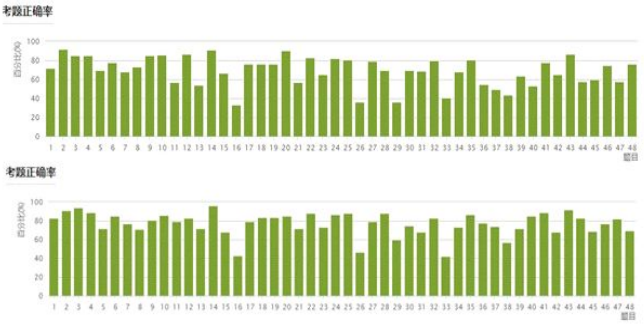

图1　实施组不同阶段答对率对比

（2）考核成绩变动分析。为了保证数据的可信度，选取同次上机考试系统自动评分的成绩作为分析数据来源，分别对实施组、对照组的各分数段成绩进行了对比分析，具体见表1。

表1　成绩分布数据

分数段	优秀90～100	良好80～89	中等70～79	及格60～69	不及格60以下
实施组	11.57%	29.75%	31.40%	23.15%	4.13%
对照组	9.41%	18.82%	34.12%	26.71%	10.94%

数据分析显示，在中高分数段，实施组的优良率占比达41.32%，而对照组仅为28.83%，差异显著。不及格率方面，实施组降至4.13%，比对照组低了6.81个百分点。以上分析表明，阶段回溯复习法在提高学习效果方面行之有效。

（3）课外环节任务点完成情况。本课程依托网络教学平台，提供了大量的课前预习、课后练习、习题讲解视频等资源。为了引导学生进行自主学习，在重点章节和习题讲解部分设置了任务点，各章节任务点的平均完成情况见表2。

表2　各章任务点完成情况

章节	1	2	3	4	5	6	7	8
实施组	96.30%	89.00%	90.78%	88.50%	82.30%	78.90%	90.25%	89.20%
对照组	91.70%	81.00%	87.56%	85.30%	81.00%	81.00%	83.60%	85.10%

注：表中数据为同一章中多个任务点的平均完成情况。

表2数据显示，实施组的非强制性任务的完成度除第6章外均优于对照组，数据分析表明实施组学生课外自主学习较参照组多，部分表明实施组知识渴求度的提高。

3.3 适应度数据分析

回溯教学法的学生适应度分析主要采用问卷调查、定量分析的方式进行。问卷从学生的主观感受、认可程度、参与度等方面进行设计。对实际回收的有效问卷进行数据分析显示，阶段回溯法的主观适应度达96.1%。

4 结束语

阶段回溯教学法是一种以教师为主导的引导式教学方法，通过不同时长间隔的分段回溯实现从知识的初识到综合运用的多方位提升。教学实践表明，阶段回溯教学法能与计算机程序设计课程注重动手编程能力培养的特点良好结合，适合程序设计课程。教学改革实践数据分析表明，在知识的广度、理解深度、实践运用、编程能力培养等方面均达到了预期效果，对其他课程有一定的参考价值。

参考文献:

[1]谭浩强.C程序设计(第五版).北京:清华大学出版社,2017.8.

[2]嵩天,礼欣,黄天羽. Python语言程序设计基础(第2版).北京:高等教育出版社,2017.2,pp:43-163.