哈尔滨工业大学(深圳)

《数据结构》实验报告

实验一

线性结构及其应用

学　院: 计算机科学与技术学院

姓　名: 吴彦乔

学　号: 210110410

专　业: 计算机

日　期: 2022-05-12

一、问题分析

1. 题目首先要求升序收集学生的信息并降序排列，之后要求反转链表，最后要求找出两个相交的链表的相交结点。其中每个学生的信息包括学号和成绩信息，需要一个结构体作为整体储存相关信息。但是如果使用数组等顺序存储的方式来存储数据，每次存储一位学生的信息都要进行大量移动操作，并且学生数量未知，用顺序存储可能造成资源的浪费。这时利用链表的插入和删除操作时间复杂度仅为 O（1）的良好特性，并且可以根据实际情况动态扩展内存，因此选择线性链表为存储结构。
2. 题目中操作考验的分别是对一个栈的压入、弹出、找栈顶、判断栈是否为空以及两个栈去完成对队列的增队尾、删队头、找队列头、判断队列是否为空的操作。其中每个节点的信息包括存储的数值和栈顶信息，需要一个结构体作为整体储存相关信息。此题用 top 表示栈顶的同时还可以将其作为数组下标，非常灵活方便。两个栈表示队列时，将栈 stack_in 的元素逐个弹出并压入栈 stack_out 再弹出时仍然保证了先进先出、后进后出的特点。选用链表而不用数组的理由同第一题。操作时要随时注意这两个栈之间的关联。

二、详细设计

2.1 设计思想

用自然语言描述解决问题的方案

1、选择线性链表做为存储结构。

链表的每一个节点包含学生的两个信息（StuID[11]-学号和 Grade-成绩）和一个指向下一个节点的指针，节点与节点之间通过降序连接起来。每次要插入一个元素，先动态申请内存，因为是按照成绩升序输入，所以直接采用头插法，在上一个节点前面插入该元素（要考虑插入位置为链表头和尾的特殊情况）。这样就可以按降序建立一个链表，不用每次增加元素都要进行排序，十分方便。

2、链表的反转

反转则利用两个指针，第一个指向链表头节点，第二个指向下一个节点，先复制原头节点的数据，让头指针指向 NULL。之后采用循环让第二个指针指向原来的下一个节点，之后将第二个指针变成新的头节点，原来的头节点变成首元节点，最后将两个指针均指向反转完成之后的最后一个节点，不断循环。循环后便成功以 O(n)的时间复杂度构建了一个反转链表。

3、找出两个相交的链表的相交结点

分别用两个指针指向两个链表头指针，遍历两个链表，同时计数得到两个链表的节点数，作差。再次分别将两个指针指向两个链表头指针，并让节点数多的链表的指向头指针的指针向后移动节点数的差值数个节点，然后两个指针同时开始逐个向后移动，直到这两个指向节点存储的数相等为止，并输出此节点。

4、用数组实现栈

压入：每压入一个元素，创建新节点，存储新节点数据并令表示栈顶的 top 加 1，即数组储存的元素数减 1

弹出：每弹出一个元素，令表示栈顶的 top 减 1，即数组储存的元素数减 1，每次都先判断栈是否为空

找栈顶：即输出数组元素中下标为 top 的元素（初始化 top 为-1），每次都先判断栈是否为空

判断栈是否为空：即判断 top 是否为-1（初始化 top 为-1），是-1 则为空，反之不为空。

5、两个栈实现队列（两个栈分别叫 stack_in 和 stack_out）

增队尾：和用数组实现栈的压入操作相同

删队头：若 stack_out 不为空则对 stack_out 直接进行弹出操作；若 stack_out 为空，则先将 stack_in 中元素从栈顶逐个弹出并压入 stack_out 中，直到 stack_in 为空结束循环，并删掉 stack_out 中数组元素中下标为 top 的元素（初始化 top 为-1），每次都先判断栈是否为空

找队列头：若 stack_out 不为空则直接输出数组元素中下标为 top 的元素（初始化 top 为-1）；若 stack_out 为空，则采用相同操作使 stack_in 元素全部压入 stack_out 之中，之后再输出数组元素中下标为 top 的元素，每次都先判断栈是否为空。

判断队列是否为空：即判断 stack_in 和 stack_out 两个栈的 top 是否均为-1 是-1 则为空，反之不为空。

2.2 存储结构及操作

(1) 存储结构（一般为自定义的数据类型，比如单链表，栈等。）

1.链表节点结构体 StudentLinkedListNode

内容包括 StuID[11]（学号-字符串）,Grade(成绩-int 型)，指向结构体 StudentLinkedListNode 的指针 next

2.链表节点结构体 Stack

内容包括 data (数据-int 型-也是 DataType 型) , top (表示栈顶-int 型)

链表节点结构体 Queue

内容包括 stack_in(指向第一个栈的 Stack 型的结构体指针)，stack_out(指向第二个栈的 Stack 型的结构体指针)

(2) 涉及的操作(一般为自定义函数,可不写过程,但要注明该函数的含义。)

1.

函数名	参数	功能	返回值
studentLinkedListCreate	学生学号和成绩	新建一个链表 node	新节点地址
studentLinkedListAdd	链表头和要插入的新节点指针	按照降序插入学生的成绩情况	链表头指针
outputStudentLinkedList	链表头指针	输出该表的成绩情况	无
printLinkedListNode	给定节点指针 p	打印单个节点	无
reverseLinkedList	链表头指针	反转链表	链表头指针
findCrossBeginNode	指向两个链表的头指针	找到相交的第一个结点	指向第一个结点指针

2.

参数中 S 为栈、Q 为队列、e 为输入的数、seq 为用于拷贝节点数据的数组名

函数名	参数	功能	返回值
GetTop	Stack 型结构体 S，DataType 型指针 e	获取栈顶元素，不删除该元素	成功返回 1，否则返回 0
StackEmpty	Stack 型结构体 S	判断栈是否空	为空返回 1，否则返回 0
Push	Stack 型结构体指针 S，DataType 型 e	向栈插入一个元素	成功返回 1，否则返回 0
Pop	Stack 型结构体指针 S，DataType 型 e	从栈中弹出一个元素	成功返回 1，否则返回 0
StackToArray	Stack 型结构体 S，DataType 型指针 seq	获取栈的一个数组拷贝	无
GetHead	Queue 型结构体 Q，DataType 型指针 e	获取队列头 (不删除元素)	如果成功获取返回 1，否则返回 0
QueueEmpty	Queue 型结构体 Q	判断队列是否为空	为空返回 1，否则返回 0
EnQueue	Queue 型结构体指针 Q，DataType 型 e	入队操作，将元素插入队列	插入成功返回 1，否则返回 0

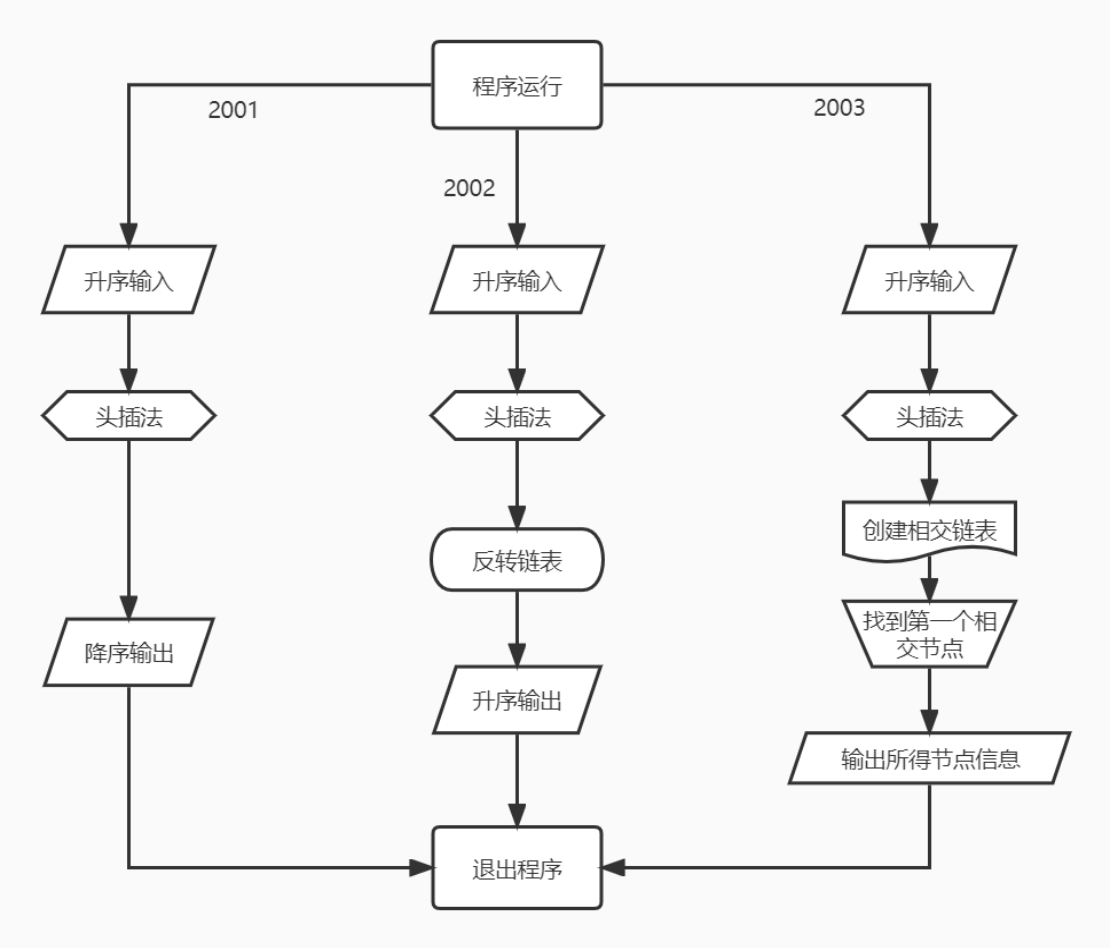
DeQueue	Queue 型结构体指针 Q，DataType 型指针 e	出队操作，从队列中取出一个元素	成功取出返回 1，否则返回 0
QueueToArray	Queue 型结构体 Q，DataType 型指针 seq	获取队列的一个数组拷贝	无

2.3 程序整体流程

画出整体流程，及核心算法流程。

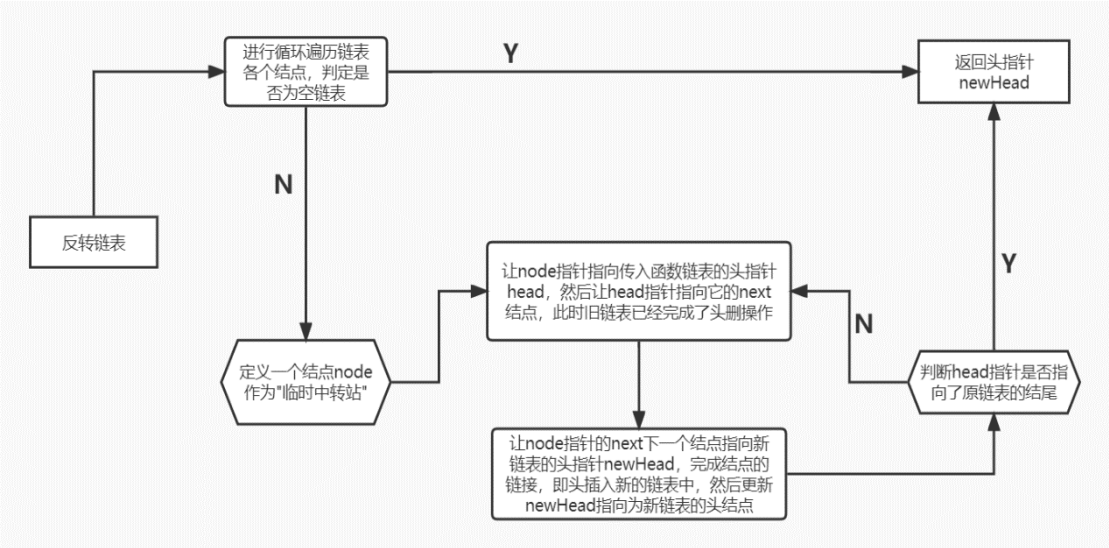
1.

整体流程：

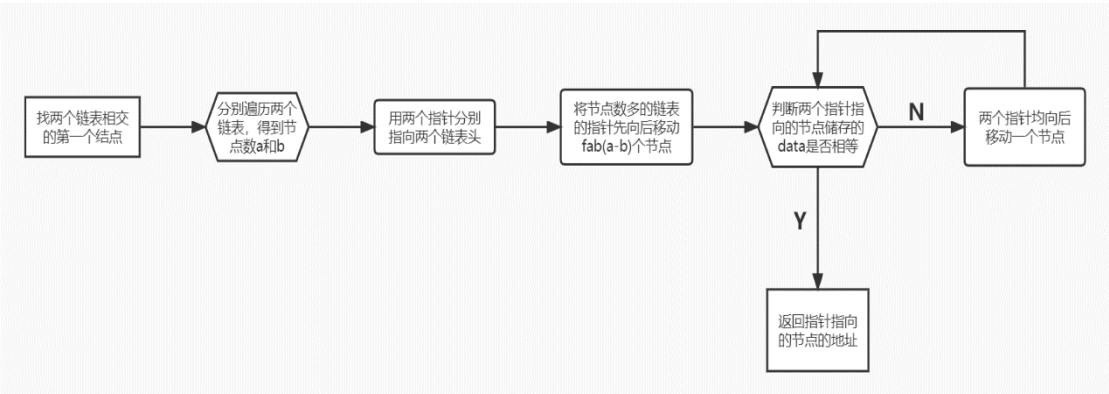


核心算法流程：

（1） 反转链表 reverseLinkedList

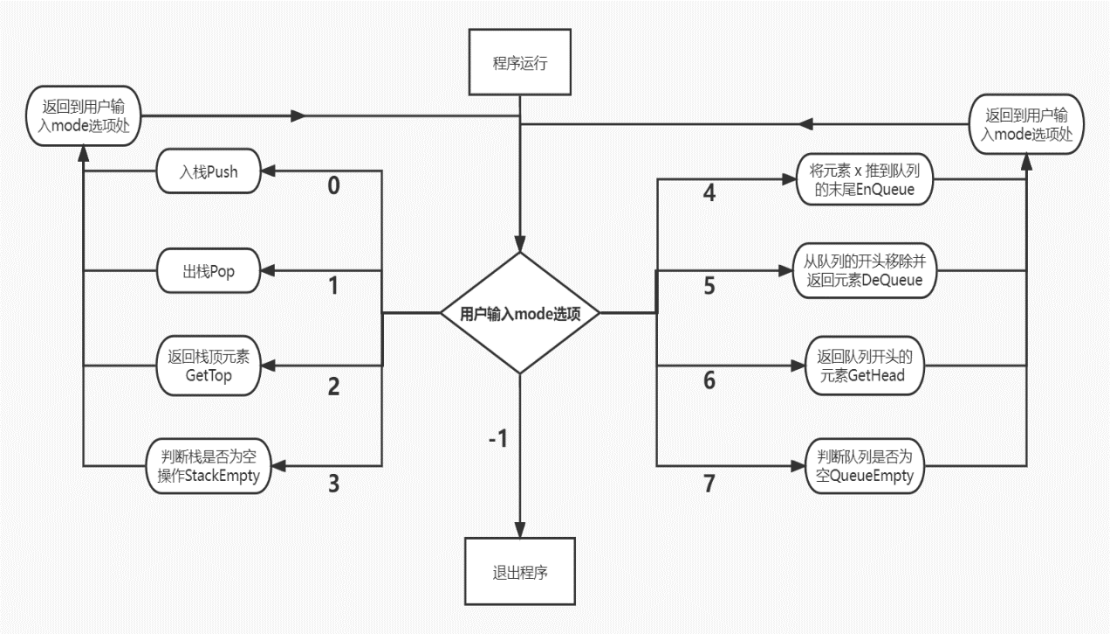


（2） 找到相交的第一个结点 findCrossBeginNode



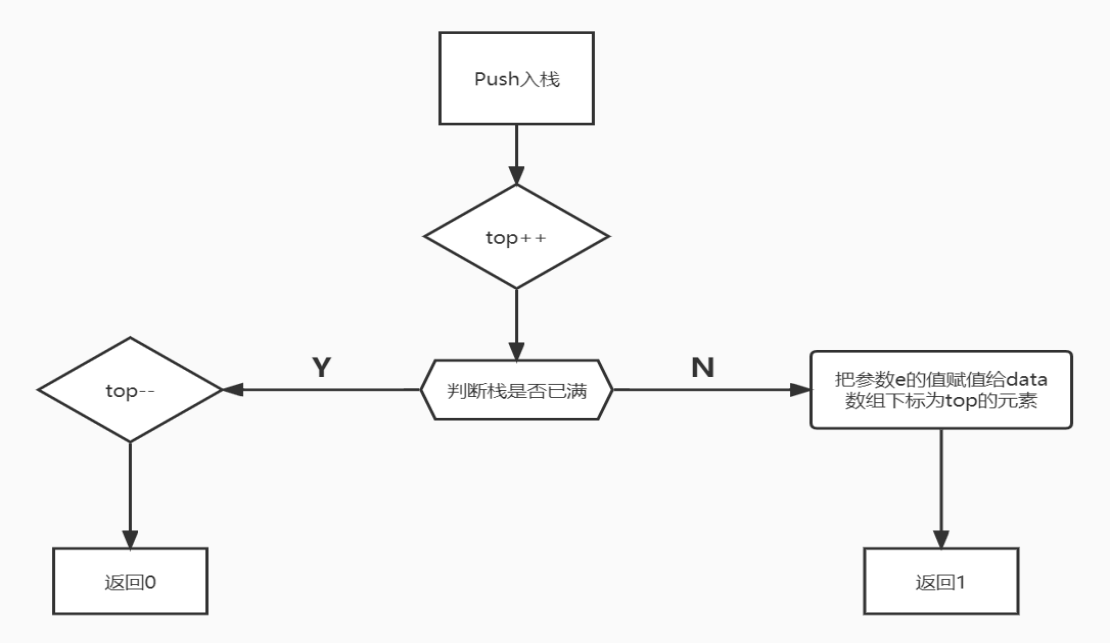
2.

整体流程：

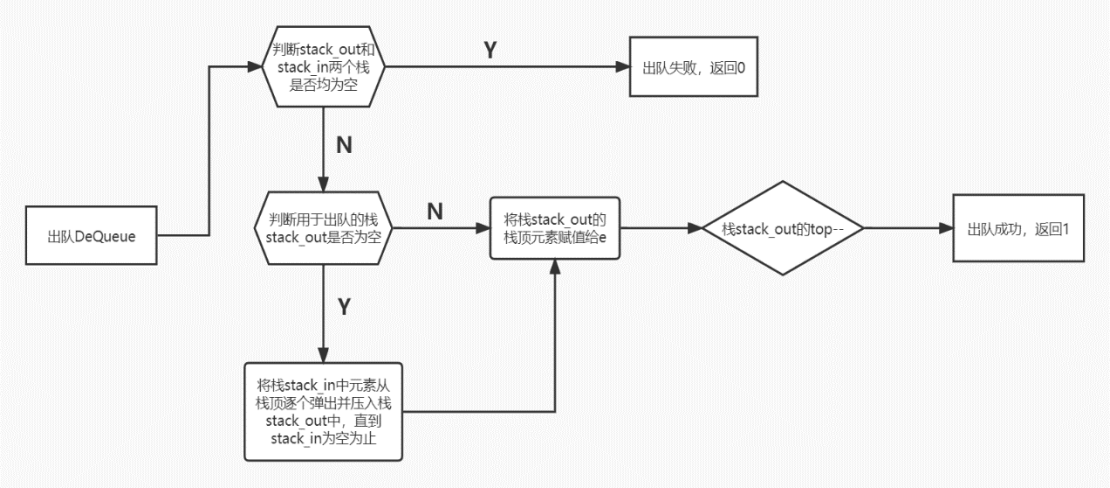


核心算法流程：

（1） 入栈 Push

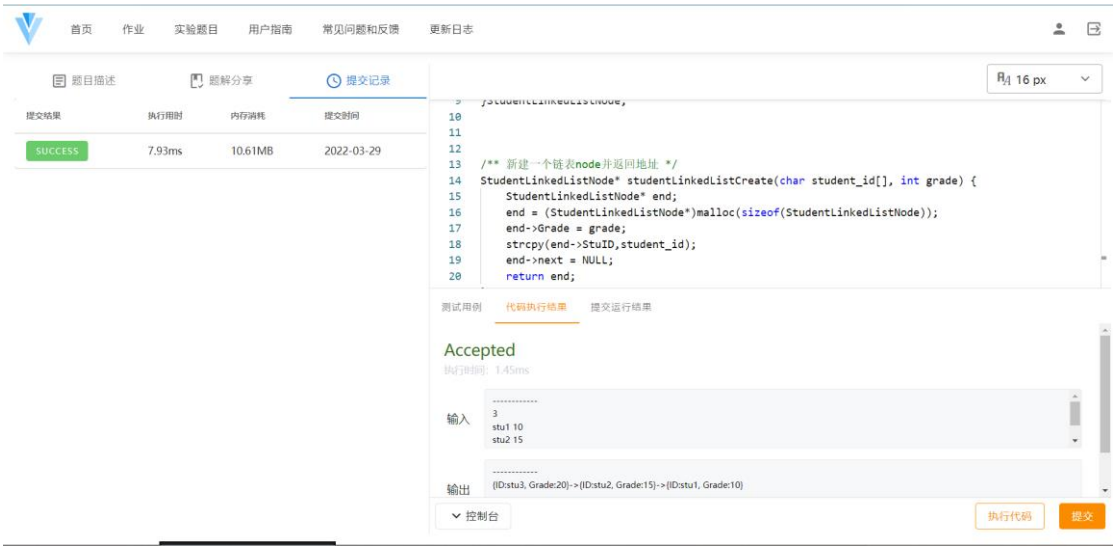


（2） 出队 DeQueue

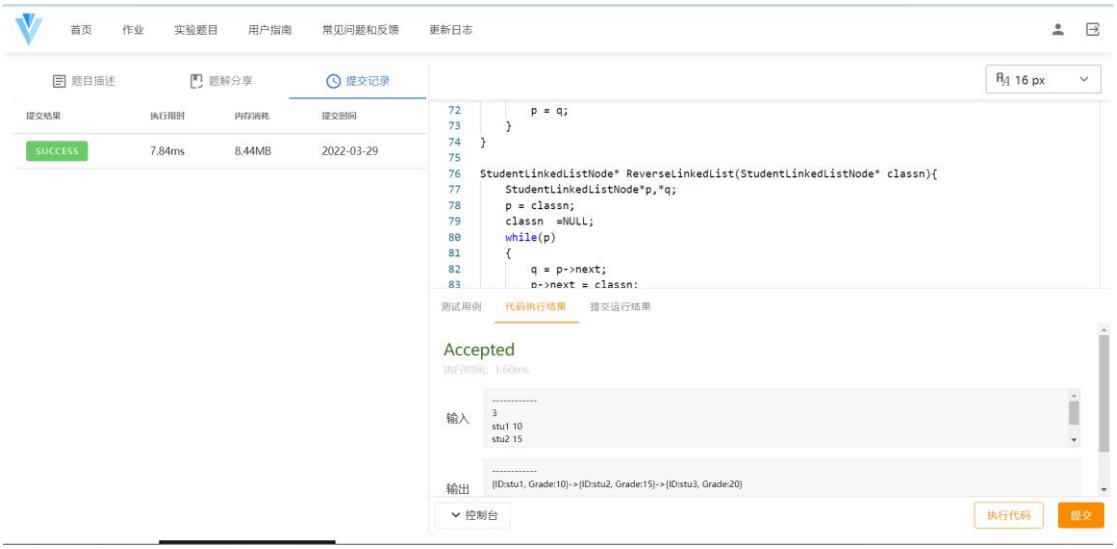


三、运行结果

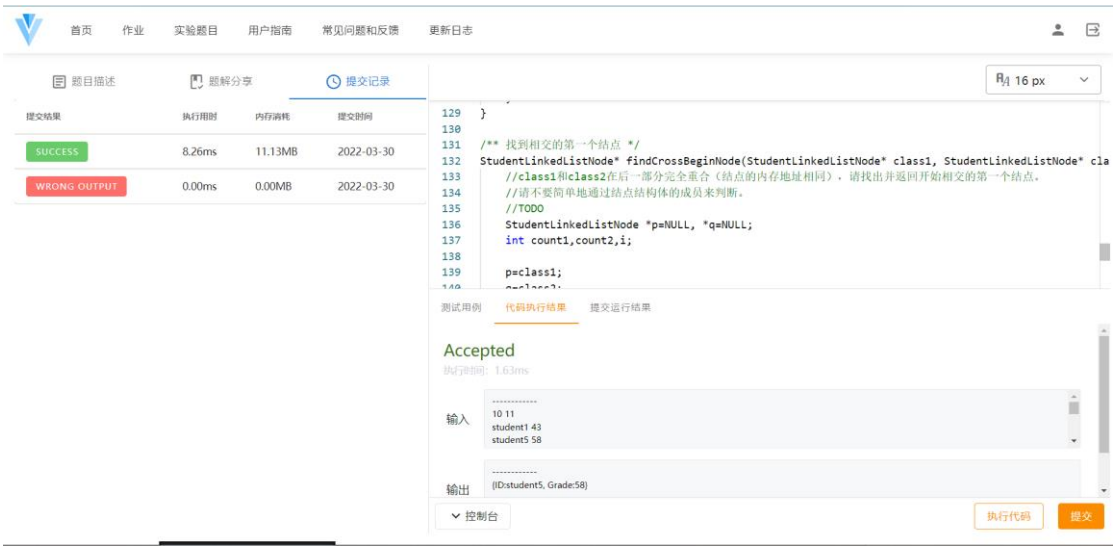
2001．建立链表



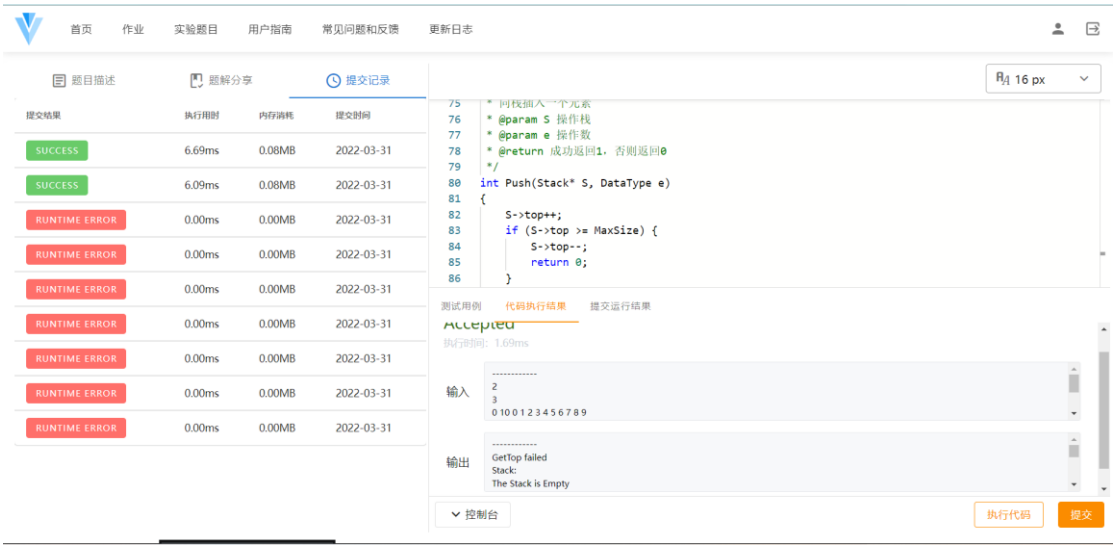
2002．反转链表



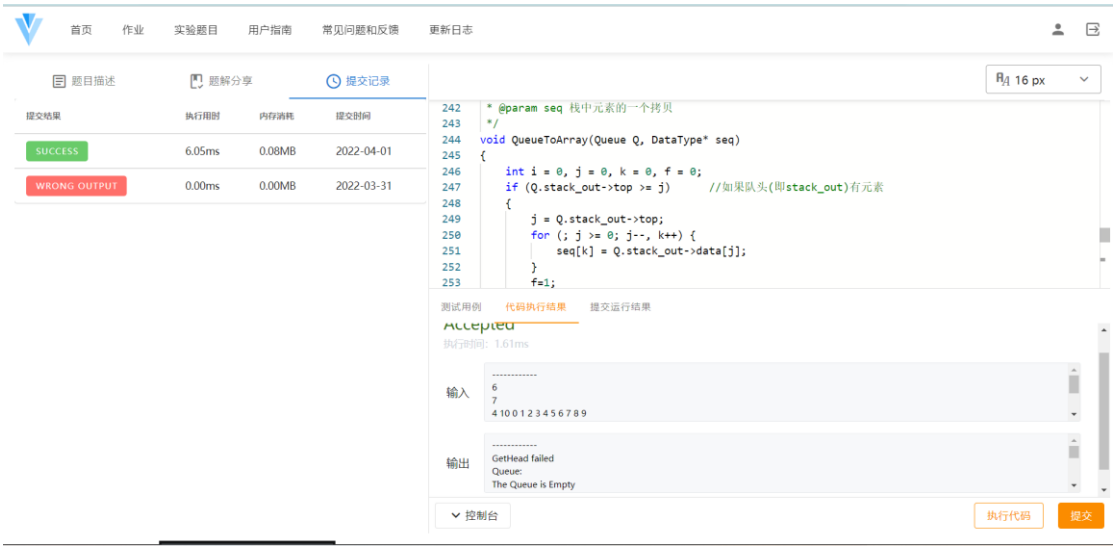
2003．链表交点



2004．数组实现栈



2005．栈实现队列



四、总结

该实验涉及到的数据结构和算法，以及遇到的问题和收获。

本次实验是基于可动态扩展的线性链表完成的。在设计实验程序以及最后的报告总结时反复地亲身体会到了这一数据结构的优越性：

1.首先是动态的数据结构：链表是一种动态的数据结构，因此它可以在运行时通过分配和取消分配内存来不断地增长和缩小。所以不需要给出链表的初始长度。 2.易于插入和删除：在链表中进行插入和删除节点是真的很容易。与数组不同，我们不必在每次插入或删除元素后都移位那么多元素。在链表中，我们只需要更新或更改节点下一个指针中的地址即可时间复杂度仅为 O（1）。

3.内存利用率高：由于链表的大小可以在运行时增加或减少，因此没有内存浪费。在使用数组时，经常存在大量的内存浪费：就像我们声明一个大小为 20 的数组而实际只存储了 6 个元素，那么就浪费了 14 个元素的空间。反观链表则没有这样的问题。

但是，程序仍存在改进之处。比如在反转链表操作时，我们只能用循环遍历整个单向链表，这个操作的时间复杂度达到了 O（n），当需要处理的元素非常多时所消耗的时间也相应的会很长。此时我们可以采用双向链表，在每个节点多储存上一个节点的地址（头节点除外）。这样相对单向链表所占用的空间仅增加了约 1/2，而时间复杂度却能减至 O（1），在处理大量数据时拥有很大的优势。

此次程序设计的过程中，我收获颇丰，除了让我明白程序设计需要扎实的基

础知识和能力之外，更重要的是学会了如何去完成一个任务，懂得了去享受程序设计。每当遇到问题，要冷静，耐心地一行一行排除障碍，直到最后获取成功，一种自信心、成就感便油然而生，这就是程序设计的乐趣。有时候也需要虚心请教，从别人的身上真的能学习到自己尚未熟练掌握的知识点。同时，在开发中我也学会了如何在一个大框架内填充逻辑，运用所学的知识，将数据结构化成真正实用的工具，极大方便了针对任务的开发任务。除此以外，我还学会了如何更好地与别人沟通、如何更好地去陈述自己的观点、如何利用流程图清楚地表明实验过程……将所学知识与实际相应用、理论与实际相结合，让我大开眼界。内心充满万千感慨地总结此次实验报告，也对以前所学知识进行了一次初审。这次程序设计实验在短短的一个星期中就能让我初步从理性回到感性的重新认识，也让我初步的认识这个代码的宏大世界，对于今后工作所应把握的技能和方向也大有启发！相信这些宝贵的经验会成为我今后成功的重要的基石，这一切都使我对未来的学习充满了希望！