第一周 - 链表及经典问题

(1) 链表的基础知识

链表的结构

- 节点
  - 数据域
  - 指针域
    - 实现方式包括地址、下标（相对地址）、引用
- 链状结构
  - 通过指针域的值形成了一个线性结构

访问链表的时间复杂度

链表不适合快速的定位数据，适合动态的插入和删除的应用场景。

- 查找节点O(n)
- 插入节点O(1)
- 删除节点O(1)

几种经典的链表实现方法

- 传统方法（节点+指针）
- 使用数组模拟
  - 指针域和数据域分离
  - 利用数组存放下标进行索引
- ……

(2) 链表的典型应用场景

- 操作系统内的动态内存分配
- LRU缓存淘汰算法
  LRU = Least Recently Used（近期最少使用）

缓存是一种高速的数据结构。

设备间存在速度差异，可以通过将使用较多的数据存放在高速区域，而将使用较少的内容存放在相对低速的区域的方式，来对系统进行优化。

(3) 经典面试题

链表的访问

- LeetCode #141 环状链表
  - 思路1：使用哈希表（额外的存储区）存储已经遍历过的节点
  - 思路2：双指针做法
    使用快慢指针 快指针一次向前2个节点 慢指针一次向前1个节点
    - 有环的链表中 快指针和慢指针最终一定会在环中相遇
    - 无环的链表中 快指针会率先访问到链表尾 从而终结检测过程
- LeetCode #142 环状链表II
  - 快指针走的路程是慢指针的2倍
  - 考虑快慢指针第一次相遇的情况（设此时慢指针走的路程为x）
    - 指定一个指针p放置在链表头部（p每次向前1个节点）
    - 再走一个路程为x的长度
      - 慢指针到达了2x的位置
      - 指针p到达了x的位置
      - 慢指针和p相遇了
        - 往前回放一下 在环的入口开始 慢指针和p已经相遇了
        - 慢指针和p重叠走了一段距离
- LeetCode #202 快乐数
  - 思路：转化为判断链表是否有环的问题
  - 收敛性的证明
    - 32位int的表示正整数大概是21亿(2^31-1)
      - 在这个范围内 各位数字平方和最大的数是1999999999 和为730
      - 根据鸽巢原理（pigeonhole's principle，也译作抽屉原理）在730次循环后必定出现重复

链表的反转

- LeetCode #206 反转链表
  - 思路1：迭代反转
    - 可以使用虚拟头节点来进行头插法
  - 思路2：递归反转（一次拆掉一个节点并递归处理剩余的子链表）
- LeetCode #92 反转链表II
  - 技巧：使用虚拟头结点（dummy head）
    - 通常用于链表的首地址有可能改变的情况
- LeetCode #25 K个一组翻转链表
  - 思路：先判断是否有K个元素 然后对这K个节点进行反转 最后拆装一下首尾部分
- LeetCode #61 旋转链表
  - 思路：把整个链表首尾相接 向后走K位后将环拆开
- LeetCode #24 两两交换链表中的节点
  思路与`LeetCode #25`完全一致，是`K = 2`的简单情形。

链表的节点删除

- LeetCode #19 删除链表的倒数第N个节点
  - 思路：找到前一个节点 删除后调整指针
- LeetCode #83 删除排序链表中的重复节点
- LeetCode #82 删除排序链表中的重复节点II

（4） 彩蛋习题及相关说明

> *求[0, 100000]内所有快乐数（定义参考LeetCode #202）的和。*

求出答案后，使用答案的数值替换下面链接中的对应部分。

https://xue.kaikeba.com/api/mentu/videos/答案.mp4