数据结构与算法

Data Structure and Algorithm

VIII. 链表

授课人：Kevin Feng

翻译 ：孙 兴

课前回顾

- 数据结构及算法
- 数学回顾
- 数组（Array）和数组列表（Array List）
- 递归 vs. 迭代
- 二分法搜索
- 分治法

CONTENTS

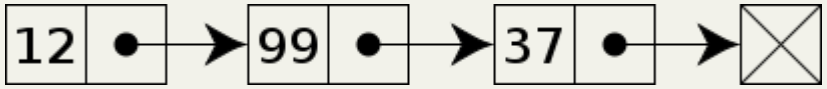
目录

- 链表
  - 数据成员（Data Member）
  - 操作（Operations）
  - 哈希（Hash）

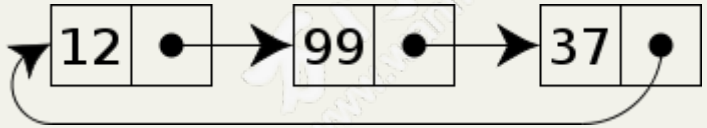
*译者注：对于CS专业常用的数据结构以及其他常用术语，ppt中会给出对应的中文翻译，但是译者认为中文的翻译并不能很好的诠释这些术语且有多种译法，所以对于此类术语保留了其英文。

基本思路

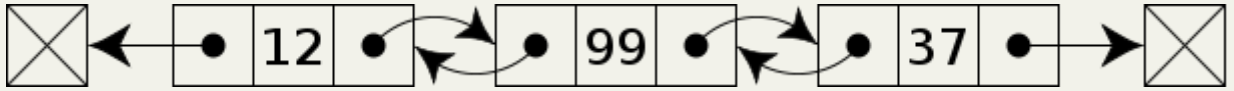
- 链表由一系列数据记录构成，在每个记录里有个区域包含一个指向下一个数据记录的索引（也就是一个链接）。



- 链表vs. 数组列表
  - 使用固定步数的操作可以在列表中任意节点做插入、删除操作。
  - 不允许随机访问
- 循环链表



- 双向链表



更多

- 哨兵节点
  - 在某些实现中，可能会在第一个数据记录之前和/或者最后一个数据记录之后添加一个额外的哨兵节点或者哑元节点。
  - 简化和加快一些列表处理的算法
- 链表vs. 动态数组
  - 都需要动态分配内存块
  - 动态数组：
    - 通过索引访问和分配非常快速，时间复杂度为0(1)
    - 添加元素（插入到数组的末尾）相对比较快，平摊时间复杂度为0(1)
    - 在动态数组里的任意位置添加和删除节点会很慢，时间复杂度为0(n)
    - 当需要对插入和删除做调整时，可能会出现不可预知的表现。
    - 会有一些未使用的空间
  - 链表：
    - 在列表中的任意位置做插入和删除都很快，时间复杂度为0(1)
    - 索引访问(随机访问)慢，时间复杂度为0(n)

抽象数据类型（ADT）列表操作

- 创建一个空列表
- 判定列表是否为空
- 确定列表中元素个数
- 在列表中给定位置添加一个元素
- 在列表中给定位置删除一个元素
- 删除列表中所有元素
- 在列表中取到给定位置上的元素
- 其他操作？
- 每一项操作的时间复杂度

其他操作

- void addFirst(E data)
- void addLast(E data)
- E getFirst()
- E getLast()
- E removeFirst()
- E removeLast()
- E peek()

节点类

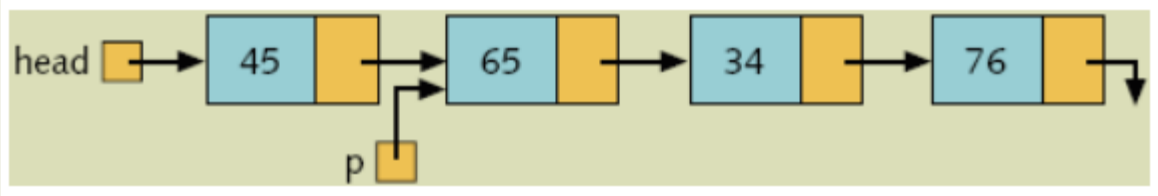
- 代表列表上的节点
- 有两个实例变量
  - 数据（整型，但也可以是其他类型）
  - 不允许随机访问

遍历链表

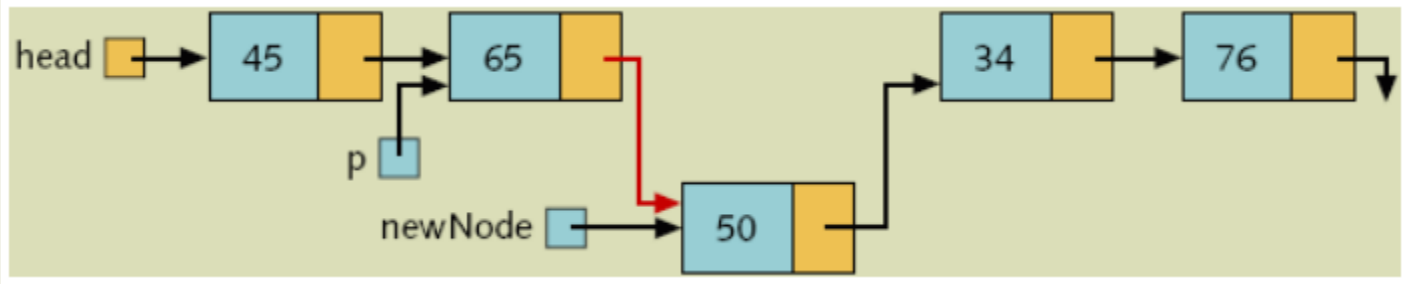
- 链表的基本操作：遍历next节点
  - 在列表中查找一个元素
  - 在列表中插入一个元素
  - 从列表中删除一列元素
- 不可以用head来遍历列表
  - 否则会丢失列表的一些节点
  - 可以使用和head相同类型的索引变量：current
- 哑元节点

插入

- 考虑如下的链表



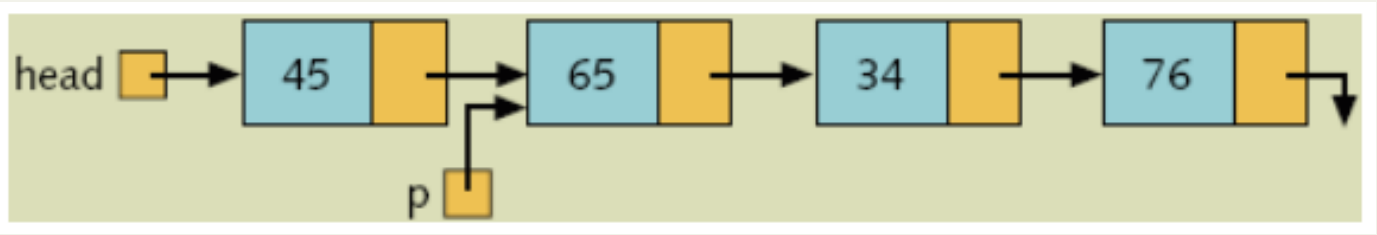
- 你想在节点P之后插入数据为50的一个新节点



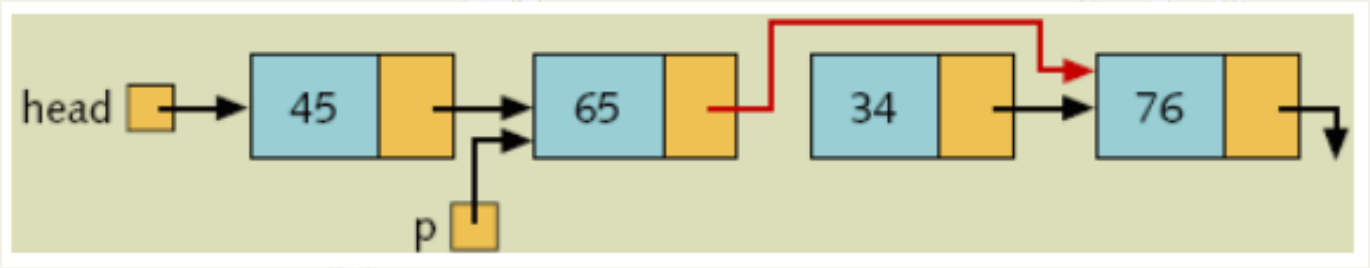
- *void addFirst(E data)*
- *boolean add(E e)*
- *void add(int index, E e)*

删除

- 考虑如下的链表



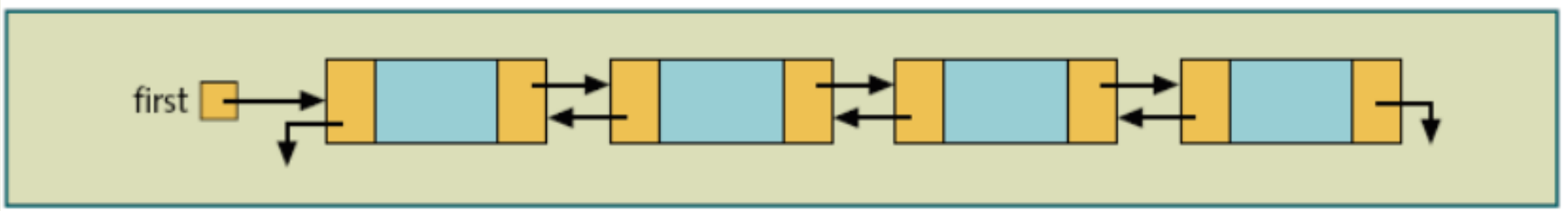
- 你想删除数据为34的节点



- *E remove(int index)*
- *boolean remove(Object o)*

双向链表

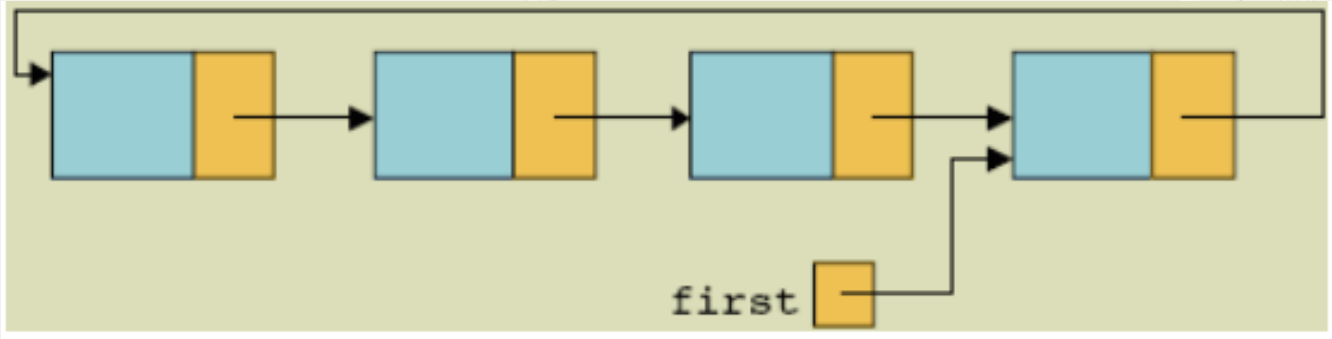
- 链表中的每个节点都有一个前向指针和一个后向指针
- 两个方向上都可以遍历双向链表



  - ***Node<E> next;***
  - ***Node<E> prev;***
- 双向链表在某些事情上更方便…

循环链表

- 链表中的最后一个节点指向第一个节点
- 把第一个点指向最后一个节点很方便



练习一

- 删除链表中的节点：除了结尾，只允许访问那个节点。
- 找到中间节点
- 是否有环：判定一个链表是否存在环
- 环的开始：给定一个循环链表，找到环的开始节点
- 删除倒数第N个节点：删除一个链表中的倒数第N个节点
- 分半：给定一个列表，把它分成两个列表，一个是前半部分，一个是后半部分。

练习二

- 合并两个有序链表
  - 合并两个有序链表，返回一个新的列表。新的列表是由这两个列表的节点拼接而成的。
- 两个链表的交集
  - 写一个可以找到两个链表交集开始节点的程序
- 链表插入排序
  - 用插入排序对一个链表做排序
- 链表排序
  - 用常数空间复杂度对链表排序，时间复杂度为$O(n \log n)$
- 分区链表
  - 给定一个链表和数值x，对链表做分区使得所有小于x的节点排在所有大于或等于x的节点之前

练习三

- 反转一个链表
- 反转链表II
  - 从位置m到n反转一个链表
- 成对交换节点
- 给定一个链表，对每两个相邻的节点作交换，返回head.

练习四

- 以k-group反转节点
  - 给定一个链表，以k个元素为一组进行反转，返回反转后的列表。
  - K是一个正整数，且小于等于链表的长度。如果最后余下的元素个数小于k个，则余下元素保持原状。
  - 例如：链表1->2->3->4->5->6->7, k=3, 则结果为：3->2->1->6->5->4->7
- 回文链表
  - 给定一个单链表，判定是否为回文
  - 可以用时间复杂度为0(n)，空间复杂度为0(1)的算法实现吗？

练习五

- 从有序链表中删除重复元素
  - 给定链表1->1->2, 返回1->2
  - 给定链表1->1->2->3->3, 返回1->2->3

- 从有序链表中删除重复元素II
  - 给定链表1->2->3->3->4->4->5，返回 1->2->5
  - 给定链表1->1->1->2->3，返回 2->3

Summary

总结

- 链表
  - 基础知识
  - 遍历，读取，添加，删除的时间复杂度
  - 双指针
  - 反转
  - 删除
  - 排序
  - 与其他数据结构合并
- 下节课：
  - 栈和队列

回顾

- 链表
  - 基础知识
  - 遍历，读取，添加，删除的时间复杂度
  - 双向指针/运算技术Two Pointers / Runner Technique
  - 反转
  - 删除
  - 排序
  - 与其他数据结构合并
- 下一结内容：
  - 栈和队列

数据结构与算法

VIII. 链表

结束