



九章算法基础班

第四讲 线性数据结构Ⅱ

课程版本 : v2.0 张三疯 老师



扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: [ninechapter](#)

知乎专栏 : <http://zhuanlan.zhihu.com/jiuzhang>

微博: <http://www.weibo.com/ninechapter>

官网: www.jiuzhang.com

九章课程不提供视频，也严禁录制视频的侵权行为
否则将追求法律责任和经济赔偿
请不要缺课

本节重点

- 链表 (Linked list) 及其操作
- 算法的时间复杂度 (Time complexity)
- 栈 (Stack) 及其操作
- 队列 (Queue) 及其操作

课程回顾

- 什么是数据结构 (data structure)

- 数据，结构，操作
- 线性数据结构

- 操作

- CRUD
- 增查改删



列表 (List)

- Python的基本数据结构之一
 - 任意对象的**有序**集合
 - list中的元素不一定是同一类型，非常灵活

```
list_1 = [12, 15.6, True, 'hello', ['a', 'b']]  
list_2 = [1, 2, 3, 4]  
list_3 = list('hello') # ['h', 'e', 'l', 'l', 'o']
```

列表 (List)

- list的常见操作
 - 增 (Create) : +, *, append, insert, extend
 - 查 (Read) : 迭代 (iteration), 索引, 切片 (slice), in, index, count
 - 改 (Update) : 索引赋值 , 切片赋值
 - 删 (Delete) : pop , remove , del
 - 其他 : len, sort, reverse

元组 (Tuple)

- tuple的操作
 - 对比list，tuple没有修改自身元素的操作
 - 任何对于tuple的修改都会报错
- 思考：tuple存在的意义？

字符串 (String)

- Python的基本数据结构之一
 - 字符的有序集合
 - 固定长度，不可变 (immutable) !!!
 - 可以使用单引号或者双引号，在代码中保持统一

```
str_1 = 'Hello world!'
str_2 = 'Jiuzhang'
str_3 = "spam's"
```

引用 (Reference)

- 什么是引用 (Reference) ?
 - 引用好比遥控器，对象好比电视机
 - 赋值操作 (引用型变量 = 对象)
- Python中所有变量都是引用
 - 存储的是对象的地址
 - id, is

引用 (Reference)

- 赋值操作和函数传参都是复制地址 (Copy address)
- 修改引用 vs. 修改对象
 - list和tuple内部存储的也都是引用

```
my_list = [20, 16, 34, 51, 66]
my_list = my_list[1:4]
my_list[2] = 11
```



链表 (Linked list) 及其操作

链表 (Linked list)

- 什么是链表 (linked list)

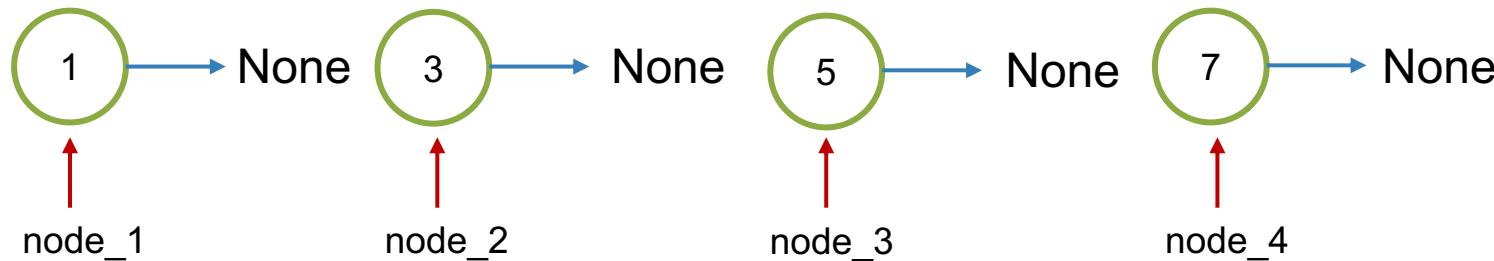
- 由节点构成的列表
- 线性的数据结构
- 自定义数据结构

```
1 class ListNode:  
2  
3     def __init__(self, val):  
4         self.val = val  
5         self.next = None
```

链表 (Linked list)

- 手动建立链表

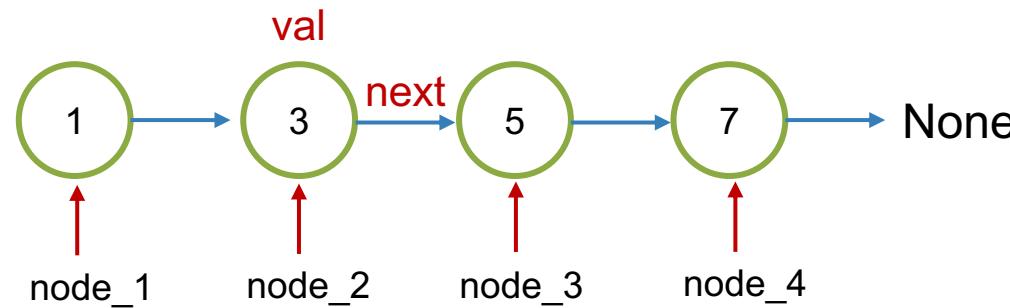
```
13 node_1 = ListNode(1)
14 node_2 = ListNode(3)
15 node_3 = ListNode(5)
16 node_4 = ListNode(7)
```



链表 (Linked list)

- 手动建立链表

```
18 node_1.next = node_2  
19 node_2.next = node_3  
20 node_3.next = node_4
```

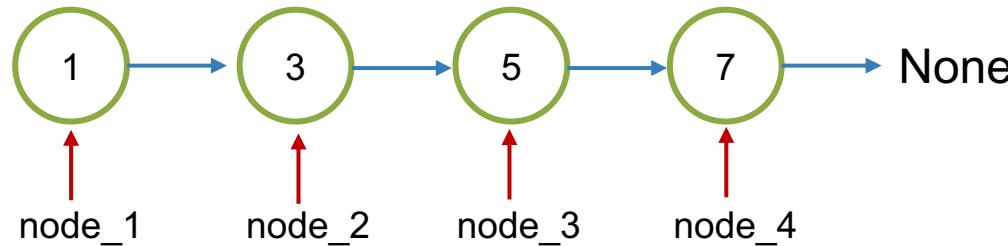


链表操作

- 链表的操作
 - 遍历 (traverse)
 - 插入 (insert)
 - 查找 (find)
 - 更新 (update)
 - 删除 (delete)

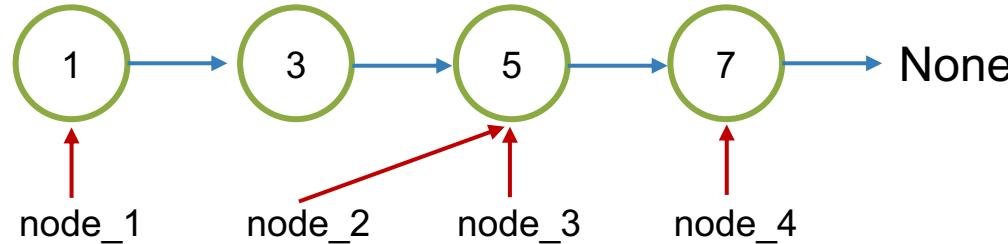
- 遍历 (traverse)

```
36 cur = node_1
37 while cur is not None:
38     print(cur.val, end=' ')
39     cur = cur.next
```



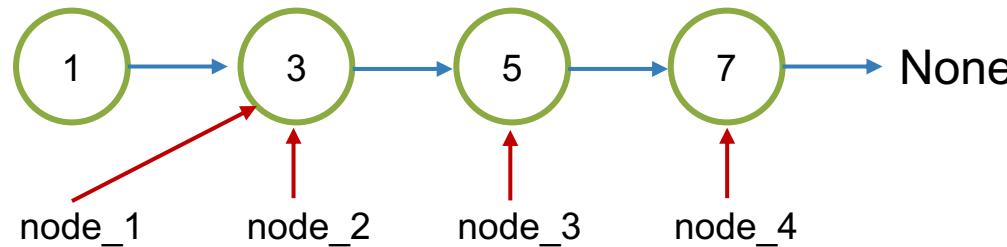
- 遍历 (traverse)
 - 测试一

```
50 node_1.next = node_2  
51 node_2.next = node_3  
52 node_3.next = node_4  
53  
54 node_2 = node_3
```



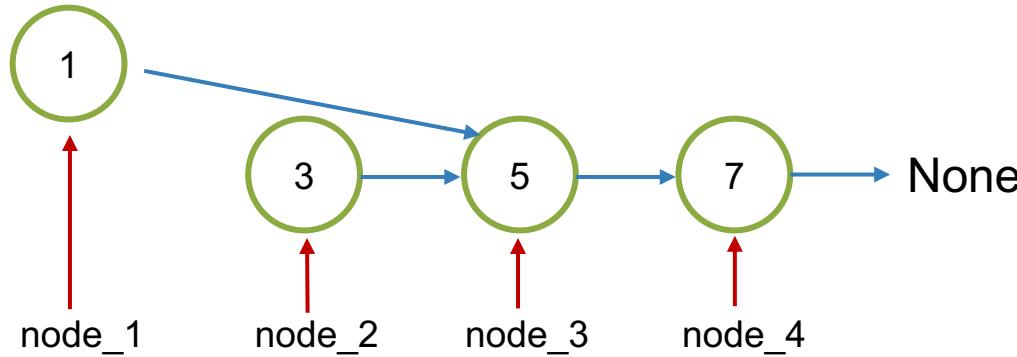
- 遍历 (traverse)
 - 测试二

```
70 node_1.next = node_2  
71 node_2.next = node_3  
72 node_3.next = node_4  
73  
74 node_1 = node_2
```



- 遍历 (traverse)
 - 测试三
 - 删除节点

```
90 node_1.next = node_2  
91 node_2.next = node_3  
92 node_3.next = node_4  
93  
94 node_1.next = node_3
```

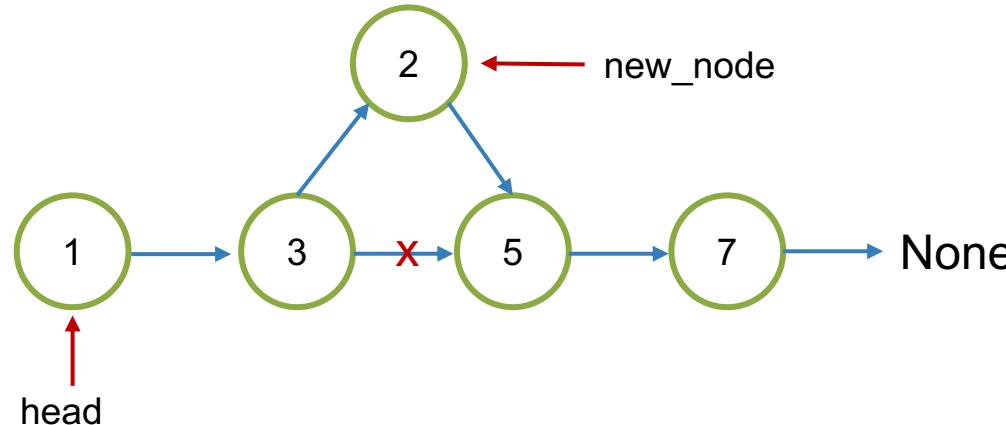


链表操作

- 基于ListNode实现一个Linked List
 - 插入 : add(location, val)
 - 查找 : get(location)
 - 更新 : set(location, val)
 - 删除 : remove(location)

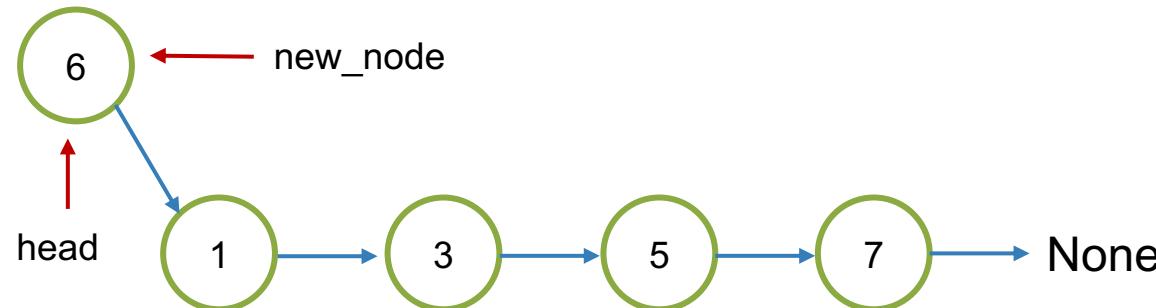
- 插入
 - 插入位置在中间

add(2, 2)



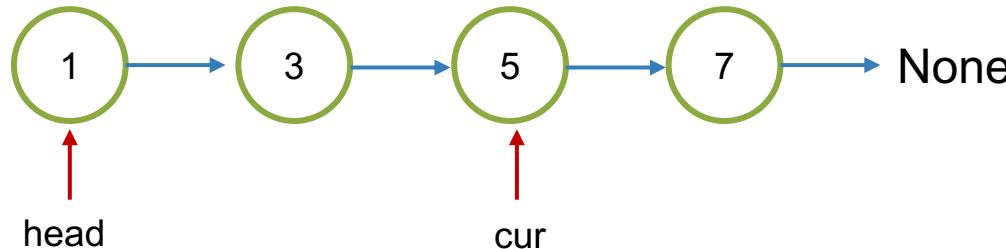
- 插入
 - 插入位置在头部

add(0, 6)



- 查找
 - 遍历到指定位置

get(2)



- 练习一：面试真题
 - Remove Nth Node From End of List
 - <https://www.lintcode.com/en/problem/remove-nth-node-from-end-of-list/>
 - <https://www.jiuzhang.com/solution/remove-nth-node-from-end-of-list/>

算法的时间复杂度

时间复杂度 (Time complexity)

- 评估算法时间效率的标准
- 算法的“执行时间”和输入问题规模之间的关系
 - 执行时间：不是实际的时间
 - 输入问题规模：具体问题具体分析



- 循环执行的次数

- 实际：不确定
- 最坏： n (数组长度)
- 最好：1
- 复杂度： $O(n)$

```
1 def find_number(nums, target):  
2     for num in nums:  
3         if num == target:  
4             return True  
5     return False
```

时间复杂度 (Time complexity)



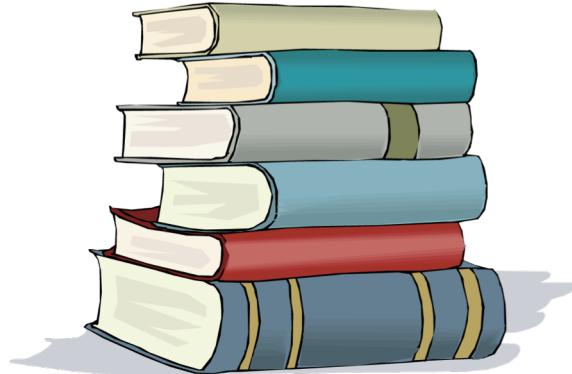
- List vs. Linked list

| | list | linked list |
|--------------|------|-------------|
| add(0, v) | O(n) | O(1) |
| add(n, v) | O(1) | O(n) |
| add / remove | O(n) | O(n) |
| get | O(1) | O(n) |
| set | O(1) | O(n) |

栈 (Stack) 及其操作

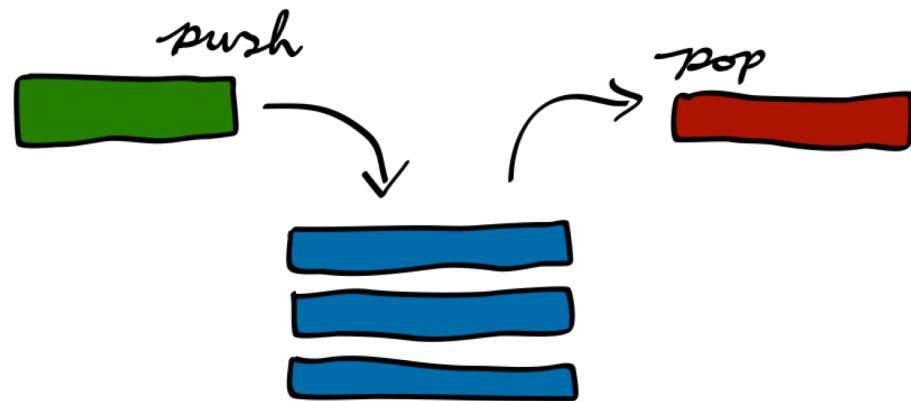
栈 (Stack)

- 什么是栈 (stack)
 - 栈是一种后进先出 (last in first out , LIFO) 的线性数据结构



栈 (Stack)

- 栈的操作
 - push(val)
 - pop()
 - peek()
 - is_empty()



栈 (Stack)

| Stack Operation | Stack Contents | Return Value |
|--------------------|------------------------------|--------------|
| s.is_empty() | [] | True |
| s.push(100) | [100] | |
| s.push('jiuzhang') | [100, 'jiuzhang'] | |
| s.peek() | [100, 'jiuzhang'] | 'jiuzhang' |
| s.push(True) | [100, 'jiuzhang', True] | |
| s.size() | [100, 'jiuzhang', True] | 3 |
| s.is_empty() | [100, 'jiuzhang', True] | False |
| s.push(8.1) | [100, 'jiuzhang', True, 8.1] | |
| s.pop() | [100, 'jiuzhang', True] | 8.1 |
| s.pop() | [100, 'jiuzhang'] | True |
| s.size() | [100, 'jiuzhang'] | 2 |

栈 (Stack)

- 栈的实现
 - 基于list实现stack
 - list的尾部相当于stack的栈顶



- 练习二：面试真题
 - Valid Parentheses
 - <https://www.lintcode.com/en/problem/valid-parentheses/>
 - <https://www.jiuzhang.com/solution/valid-parentheses/>

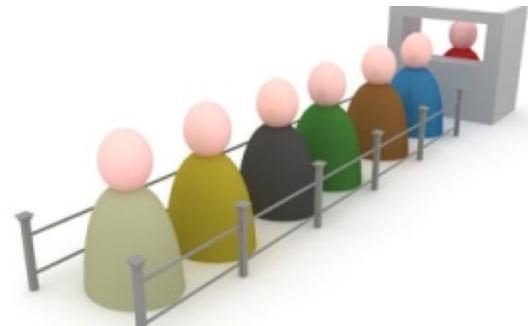
- 栈的应用
 - 操作系统 (OS) 用来保存函数调用的状态
 - 小视频



队列 (Queue) 及其操作

队列 (Queue)

- 什么是队列 (queue)
 - 队列是一种先进先出 (first in first out , FIFO) 的线性数据结构



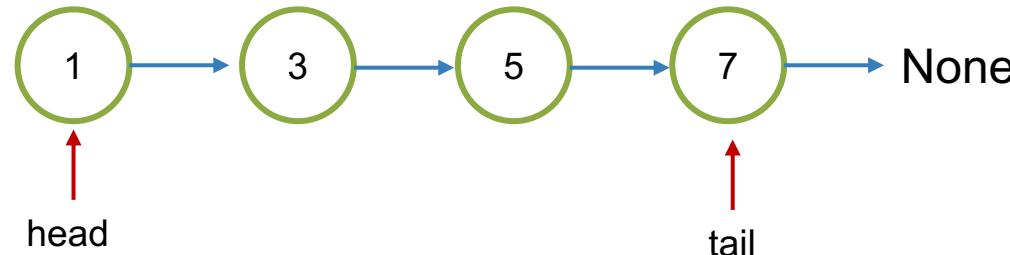
队列 (Queue)

- 队列的操作
 - enqueue(val) : 进队列
 - dequeue() : 出队列
 - size() : 队列中元素个数
 - is_empty() : 队列是否为空



队列 (Queue)

- 队列的实现
 - 使用LinkedList实现队列 (为什么不用List？)
 - 使用queue模块



- 练习三：面试真题
 - Implement Queue by Linked List
 - <https://www.lintcode.com/en/problem/implement-queue-by-linked-list/>
 - <https://www.jiuzhang.com/solution/implement-queue-by-linked-list/>

队列 (Queue)

- Python中提供了queue模块

```
1 from queue import Queue
2
3 que = Queue(maxsize=100) # maxsize < 1 表示队列长度无限
4
5 for i in range(20):
6     que.put(i)
7
8 print(que.qsize())
9
10 while not que.empty():
11     print(que.get())
```

- 队列的应用
 - Message queue 消息队列
 - BFS 广度优先搜索 (Breadth-First-Search)

总结

- 链表及其操作
 - 面试重点
- 算法的时间复杂度分析
 - 基本内功
- 栈和队列



扫描二维码关注微信/微博
获取最新面试题及权威解答

微信: [ninechapter](#)

微博: <http://www.weibo.com/ninechapter>

官网: www.jiuzhang.com



谢谢大家