

# 习题答疑课（三）

## 习题答疑课（三）

- 一、【中等】leetcode-146-LRU缓存机制
- 二、【困难】Leetcode-460-LFU缓存

## 一、【中等】leetcode-146-LRU缓存机制

1. 哈希表 + 链表解决  $O(1)$  读取，以及  $O(1)$  修改缓存数据节点位置的操作
2. 弯路1：一开始想到了哈希表，但忽略了利用链表实现  $O(1)$  修改
3. 弯路2：由于头尾指针会发生变化，忽略了虚拟节点的处理技巧

146. LRU 缓存机制

难度 中等 1124 收藏 分享 切换为英文 接收动态 反馈

运用你所掌握的数据结构，设计和实现一个 LRU（最近最少使用）缓存机制。实现 LRUCache 类：

- LRUCache(int capacity) 以正整数作为容量 capacity 初始化 LRU 缓存
- int get(int key) 如果关键字 key 存在于缓存中，则返回关键字的值，否则返回 -1。
- void put(int key, int value) 如果关键字已经存在，则变更其数据值；如果关键字不存在，则插入该组「关键字-值」。当缓存容量达到上限时，它应该在写入新数据之前删除最久未使用的数据值，从而为新的数据值留出空间。

进阶：你是否可以在  $O(1)$  时间复杂度内完成这两种操作？

示例：

输入  
["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"]  
[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]

输出  
[null, null, null, 1, null, -1, null, -1, 3, 4]

解释  
LRUCache lruCache = new LRUCache(2);

```
1 class LRUCache {
2 public:
3     LRUCache(int capacity) {
4
5     }
6
7     int get(int key) {
8
9     }
10
11    void put(int key, int value) {
12
13    }
14 };
15
16 /**
17  * Your LRUCache object will be instantiated
18  * with capacity 2.
19  * int param_1 = obj->get(key);
20  * obj->put(key,value);
21  */
```

Chrome 文件 编辑 视图 历史记录 书签 用户 标签页 窗口 帮助 武汉小说阅读网

146. LRU 缓存机制 - 力扣 (LeetCode) Online Judge

leetcode-cn.com/problems/lru-cache/

力扣 学习 题库 讨论 竞赛 求职 商店

题目描述 评论 (633) 题解 (854) 提交记录 C++ 智能模式

实现 LRUCache 类:

- LRUCache(int capacity) 以正整数作为容量 capacity 初始化 LRU 缓存
- int get(int key) 如果关键字 key 存在于缓存中, 则返回关键字的值, 否则返回 -1。
- void put(int key, int value) 如果关键字已经存在, 则变更其数据值; 如果关键字不存在, 则插入该组「关键字-值」。当缓存容量达到上限时, 它应该在写入新数据之前删除最久未使用的数据值, 从而为新的数据值留出空间。

进阶: 你是否可以在  $O(1)$  的时间复杂度内完成这两种操作?

示例

输入

```
["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"]
[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]
```

输出

```
[null, null, null, 1, null, -1, null, -1, 3, 4]
```

解释

```
LRUCache lruCache = new LRUCache(2);
lruCache.put(1, 1); // 缓存是 {1=1}
lruCache.put(2, 2); // 缓存是 {1=1, 2=2}
```

1 class LRUCache {  
2 public:  
3 LRUCache(int capacity) {  
4  
5 }  
6  
7 int get(int key) {  
8  
9 }  
10  
11 void put(int key, int value) {  
12  
13 }  
14 };  
15  
16 /\*\*  
17 \* Your LRUCache object will be instantiated with this capacity:  
18 \* LRUCache\* obj = new LRUCache(capacity);  
19 \* int param\_1 = obj->get(key);  
20 \* obj->put(key,value);  
21 \*/

## 二、【困难】Leetcode-460-LFU缓存

1. 编码较复杂, 基于十字链表实现 LFU 缓存机制
2. 将出现次数相同的节点, 存储在同一个 LRUCache 中
3. 将所有非空的 LRUCache, 按照代表的次数, 链接成一个大链表
4. 删除节点的操作, 简化成了: 删除第一个 LRUCache 中的头结点
5. 操作1: 删除一个节点
6. 操作2: 新增一个节点
7. 操作3: 将一个现有节点, 移动到下一个 LRUCache 中