LeetCode 第 146 号问题：LRU缓存机制

> *本文首发于公众号「图解面试算法」，是 图解 LeetCode 系列文章之一。*
>
> *同步博客：https://www.algomooc.com*

题目来源于 LeetCode 上第 146 号问题：LRU缓存机制。题目难度为 Hard，目前通过率为 15.8% 。

题目描述

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。它应该支持以下操作： 获取数据 `get` 和 写入数据 `put` 。

获取数据 `get(key)` - 如果密钥 (key) 存在于缓存中，则获取密钥的值（总是正数），否则返回 -1。 写入数据 `put(key, value)` - 如果密钥不存在，则写入其数据值。当缓存容量达到上限时，它应该在写入新数据之前删除最近最少使用的数据值，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:

```
LRUCache cache = new LRUCache( 2 /* 缓存容量 */ );

cache.put(1, 1);
cache.put(2, 2);
cache.get(1);       // 返回  1
cache.put(3, 3);    // 该操作会使得密钥 2 作废
cache.get(2);       // 返回 -1 (未找到)
cache.put(4, 4);    // 该操作会使得密钥 1 作废
cache.get(1);       // 返回 -1 (未找到)
cache.get(3);       // 返回  3
cache.get(4);       // 返回  4
```

题目解析

这道题是让我们实现一个 LRU 缓存器，LRU是Least Recently Used的简写，就是最近最少使用的意思。

这个缓存器主要有两个成员函数，get和put。

其中 get 函数是通过输入 key 来获得 value，如果成功获得后，这对 (key, value) 升至缓存器中最常用的位置（顶部），如果 key 不存在，则返回 -1 。

而 put 函数是插入一对新的 (key, value)，如果原缓存器中有该 key，则需要先删除掉原有的，将新的插入到缓存器的顶部。如果不存在，则直接插入到顶部。

若加入新的值后缓存器超过了容量，则需要删掉一个最不常用的值，也就是底部的值。

具体实现时我们需要三个私有变量，cap , l 和 m，其中 cap 是缓存器的容量大小，l 是保存缓存器内容的列表，m 是 HashMap，保存关键值 key 和缓存器各项的迭代器之间映射，方便我们以 O(1) 的时间内找到目标项。

然后我们再来看 get 和 put 如何实现。

其中，get 相对简单些，我们在 m 中查找给定的key，若不存在直接返回 -1；如果存在则将此项移到顶部。

对于 put ，我们也是现在 m 中查找给定的 key，如果存在就删掉原有项，并在顶部插入新来项，然后判断是否溢出，若溢出则删掉底部项(最不常用项)。

动画描述

代码实现

```
class LRUCache{
public:
    LRUCache(int capacity) {
        cap = capacity;
    }

    int get(int key) {
        auto it = m.find(key);
        if (it == m.end()) return -1;
        l.splice(l.begin(), l, it->second);
        return it->second->second;
    }

    void put(int key, int value) {
        auto it = m.find(key);
        if (it != m.end()) l.erase(it->second);
        l.push_front(make_pair(key, value));
        m[key] = l.begin();
        if (m.size() > cap) {
            int k = l.rbegin()->first;
            l.pop_back();
            m.erase(k);
        }
    }

private:
    int cap;
    list<pair<int, int>> l;
    unordered_map<int, list<pair<int, int>>::iterator> m;
};
```