LRU Cache，最近最少使用缓存。在CPU的缓存机制中，如果要读出一条在cache中没有的数据并存到cache中时，需要淘汰一些已经在cache里的数据来腾出位置。LRU机制就是将最久没有使用过的数据淘汰的策略。

LRU有两个内置操作：

get(key):如果关键字key存在于缓存中，则获取关键字的值

put(key, value):如果关键字已存在，变更其数据值，如果不存在，插入该kv对。并且当缓存容量到达上限时，删除之前最久未使用的数据，然后插入新数据。

要求get和put操作都具有O(1)的时间复杂度。

对需求进行分析，首先需要O(1)进行get和put，至少需要一个哈希表。其次是需要满容量时删除最久未使用的数据，则需要存储元素的放入顺序，并且在put之后可以更新到最近使用的位置。通过数组实现的话，put之后要保证最近的元素在最后的位置，但数组删除元素之后需要整体移动到前一个，时间复杂度为O(N)。通过链表可以实现对放入顺序的存储，链表序即为插入顺序，但要实现O(1)删除元素，显然不可实现（因为要拿到该节点的前驱结点）。既然要得到前驱结点才能实现O(1)删除，那使用双向链表即可实现。

get(key)操作：判断key是否在哈希表中，如果在，返回该值，否则返回不存在

put(key, value)：判断key是否在哈希表中，如果在，更新value，如果不在，判断是否超出容量，如果达到容量上限，删除链表中第一个节点。将新的value插入到链表最后一个位置。