**[LRU Cache](https://leetcode.com/problems/lru-cache/)**

**class LRUCache {**

**HashMap<Integer , DLinkedList> cache;**

**DLinkedList head;**

**DLinkedList tail;**

**int count;**

**int capacity;**

**public LRUCache(int capacity) {**

**cache = new HashMap<>();**

**head = new DLinkedList(0 , 0);**

**tail = new DLinkedList(0 , 0);**

**count = 0;**

**this.capacity = capacity;**

**head.post = tail;**

**tail.pre = head;**

**}**

**public int get(int key) {**

**DLinkedList node = cache.get(key);**

**if(node == null) {**

**return -1;**

**}**

**this.moveToHead(node);**

**return node.value;**

**}**

**public void put(int key, int value) {**

**DLinkedList node = cache.get(key);**

**if(node == null) {**

**node = new DLinkedList(key, value);**

**cache.put(key , node);**

**count++;**

**if(count > capacity) {**

**DLinkedList temp = this.removeFromTail();**

**cache.remove(temp.key);**

**}**

**this.addNode(node);**

**}**

**else {**

**node.value = value;**

**this.moveToHead(node);**

**}**

**}**

**public void moveToHead(DLinkedList node) {**

**this.removeNode(node);**

**this.addNode(node);**

**}**

**public void addNode(DLinkedList node) {**

**head.post.pre = node;**

**node.post = head.post;**

**node.pre = head;**

**head.post = node;**

**}**

**public void removeNode(DLinkedList node) {**

**DLinkedList post = node.post;**

**DLinkedList pre = node.pre;**

**post.pre = pre;**

**pre.post = post;**

**}**

**public DLinkedList removeFromTail() {**

**DLinkedList node = tail.pre;**

**this.removeNode(node);**

**return node;**

**}**

**}**

**class DLinkedList {**

**DLinkedList post;**

**DLinkedList pre;**

**int key;**

**int value;**

**public DLinkedList(int key, int value) {**

**this.key = key;**

**this.value = value;**

**}**

**}**

Time complexity : O(1). Constant time

Space Complexity : O(1) , constant space