```
class LRUCache {

    class Node {
        Node pre, next;
        int key, val;
        Node (int k, int v) {
            key = k;
            val = v;
        }
    }

    int cap;
    Node head, tail;
    Map<Integer, Node> map;
    public LRUCache(int capacity) {
        head = new Node(-1, -1);
        tail = new Node(-1, -1);
        head.next = tail;
        tail.pre = head;
        cap = capacity;
        map = new HashMap<>();
    }

    public int get(int key) {
        if (map.containsKey(key)) {
            Node node = map.get(key);
            node.pre.next = node.next;
            node.next.pre = node.pre;
            moveToTail(node);
            return node.val;
        } else {
            return -1;
        }
    }

    public void put(int key, int value) {
        if (get(key) != -1) {
            map.get(key).val = value;
        } else {
            Node node = new Node(key, value);

            if (cap == map.size()) {
                Node n = head.next;
                map.remove(n.key);
                head.next = head.next.next;
                head.next.pre = head;
            }
            map.put(key, node);
            moveToTail(node);
        }
    }

    private void moveToTail(Node node) {
        node.next = tail;
        node.pre = tail.pre;
        tail.pre.next = node;
        tail.pre = node;
    }
}

/**
 * Your LRUCache object will be instantiated and called as such:
 * LRUCache obj = new LRUCache(capacity);
 * int param_1 = obj.get(key);
 * obj.put(key,value);
 */
```