本周作业

简单：

用 add first 或 add last 这套新的 API 改写 Deque 的代码

分析 Queue 和 Priority Queue 的源码

[删除排序数组中的重复项](https://leetcode-cn.com/problems/remove-duplicates-from-sorted-array/)（Facebook、字节跳动、微软在半年内面试中考过）

[旋转数组](https://leetcode-cn.com/problems/rotate-array/)（微软、亚马逊、PayPal 在半年内面试中考过）

[合并两个有序链表](https://leetcode-cn.com/problems/merge-two-sorted-lists/)（亚马逊、字节跳动在半年内面试常考）

[合并两个有序数组](https://leetcode-cn.com/problems/merge-sorted-array/)（Facebook 在半年内面试常考）

[两数之和](https://leetcode-cn.com/problems/two-sum/)（亚马逊、字节跳动、谷歌、Facebook、苹果、微软在半年内面试中高频常考）

[移动零](https://leetcode-cn.com/problems/move-zeroes/)（Facebook、亚马逊、苹果在半年内面试中考过）

[加一](https://leetcode-cn.com/problems/plus-one/)（谷歌、字节跳动、Facebook 在半年内面试中考过）

中等：

[设计循环双端队列](https://leetcode.com/problems/design-circular-deque)（Facebook 在 1 年内面试中考过）

困难：

[接雨水](https://leetcode.com/problems/trapping-rain-water/)（亚马逊、字节跳动、高盛集团、Facebook 在半年内面试常考）

[26. 删除排序数组中的重复项](https://leetcode-cn.com/problems/remove-duplicates-from-sorted-array/)

class Solution {

    public int removeDuplicates(int[] nums) {

    if (nums.length == 0) return 0;

    int i = 0;

    for (int j = 1; j < nums.length; j++) {

        if (nums[j] != nums[i]) {

            i++;

            nums[i] = nums[j];

        }

    }

    return i + 1;

}

}

[189. 旋转数组](https://leetcode-cn.com/problems/rotate-array/)

class Solution {

    public void rotate(int[] nums, int k) {

        k = k % nums.length;

        int count = 0;

        for (int start = 0; count < nums.length; start++) {

            int current = start;

            int prev = nums[start];

            do {

                int next = (current + k) % nums.length;

                int temp = nums[next];

                nums[next] = prev;

                prev = temp;

                current = next;

                count++;

            } while (start != current);

        }

    }

}

[21. 合并两个有序链表](https://leetcode-cn.com/problems/merge-two-sorted-lists/)

class Solution {

    public ListNode mergeTwoLists(ListNode l1, ListNode l2) {

        ListNode prehead = new ListNode(-1);

        ListNode prev = prehead;

        while (l1 != null && l2 != null) {

            if (l1.val <= l2.val) {

                prev.next = l1;

                l1 = l1.next;

            } else {

                prev.next = l2;

                l2 = l2.next;

            }

            prev = prev.next;

        }

        // 合并后 l1 和 l2 最多只有一个还未被合并完，我们直接将链表末尾指向未合并完的链表即可

        prev.next = l1 == null ? l2 : l1;

        return prehead.next;

    }

}

[88. 合并两个有序数组](https://leetcode-cn.com/problems/merge-sorted-array/)

class Solution {

    public void merge(int[] nums1, int m, int[] nums2, int n) {

        int len1 = m - 1;

        int len2 = n - 1;

        int len = m + n - 1;

        while(len1 >= 0 && len2 >= 0) {

            // 注意--符号在后面，表示先进行计算再减1，这种缩写缩短了代码

            nums1[len--] = nums1[len1] > nums2[len2] ? nums1[len1--] : nums2[len2--];

        }

        // 表示将nums2数组从下标0位置开始，拷贝到nums1数组中，从下标0位置开始，长度为len2+1

        System.arraycopy(nums2, 0, nums1, 0, len2 + 1);

    }

}

[1. 两数之和](https://leetcode-cn.com/problems/two-sum/)

class Solution {

    public int[] twoSum(int[] nums, int target) {

        Map<Integer, Integer> map = new HashMap<>();

        for (int i = 0; i < nums.length; i++) {

            int complement = target - nums[i];

            if (map.containsKey(complement)) {

                return new int[] { map.get(complement), i };

            }

            map.put(nums[i], i);

        }

        throw new IllegalArgumentException("No two sum solution");

    }

}

[283. 移动零](https://leetcode-cn.com/problems/move-zeroes/)

class Solution {

    public void moveZeroes(int[] nums) {

        if(nums==null) {

            return;

        }

        //两个指针i和j

        int j = 0;

        for(int i=0;i<nums.length;i++) {

            //当前元素!=0，就把其交换到左边，等于0的交换到右边

            if(nums[i]!=0) {

                int tmp = nums[i];

                nums[i] = nums[j];

                nums[j++] = tmp;

            }

        }

    }

}

[66. 加一](https://leetcode-cn.com/problems/plus-one/)

class Solution {

    public int[] plusOne(int[] digits) {

        int len = digits.length;

        for(int i = len - 1; i >= 0; i--) {

            digits[i]++;

            digits[i] %= 10;

            if(digits[i]!=0)

                return digits;

        }

        digits = new int[len + 1];

        digits[0] = 1;

        return digits;

    }

}

[641. 设计循环双端队列](https://leetcode-cn.com/problems/design-circular-deque/)

class Node {

    int val;

    Node prev, next;

    Node(int val) {

        this.val = val;

        this.prev = null;

        this.next = null;

    }

}

class MyCircularDeque {

    int count;

    int capacity;

    Node head, tail;

    /\*\* Initialize your data structure here. Set the size of the deque to be k. \*/

    public MyCircularDeque(int k) {

        this.count = 0;

        this.capacity = k;

    }

    /\*\* Adds an item at the front of Deque. Return true if the operation is successful. \*/

    public boolean insertFront(int value) {

        if(this.count == this.capacity) {

            return false;

        }

        Node newNode = new Node(value);

        if(this.count == 0) {

            this.head = this.tail = newNode;

        } else {

            this.head.prev = newNode;

            newNode.next = this.head;

            this.head = this.head.prev;

        }

        this.count++;

        return true;

    }

    /\*\* Adds an item at the rear of Deque. Return true if the operation is successful. \*/

    public boolean insertLast(int value) {

        if(this.count == this.capacity) {

            return false;

        }

        Node newNode = new Node(value);

        if(this.count == 0) {

            this.head = this.tail = newNode;

        } else {

            this.tail.next = newNode;

            newNode.prev = this.tail;

            this.tail = this.tail.next;

        }

        this.count++;

        return true;

    }

    /\*\* Deletes an item from the front of Deque. Return true if the operation is successful. \*/

    public boolean deleteFront() {

        if(this.count == 0) {

            return false;

        }

        if(this.count == 1) {

            this.head = this.tail = null;

        } else {

            Node newHead = this.head.next;

            this.head.next = null;

            this.head = newHead;

            this.head.prev = null;

        }

        this.count--;

        return true;

    }

    /\*\* Deletes an item from the rear of Deque. Return true if the operation is successful. \*/

    public boolean deleteLast() {

        if(this.count == 0) {

            return false;

        }

        if(this.count == 1) {

            this.head = this.tail = null;

        } else {

            Node newTail = this.tail.prev;

            this.tail.prev = null;

            this.tail = newTail;

            this.tail.next = null;

        }

        this.count--;

        return true;

    }

    /\*\* Get the front item from the deque. \*/

    public int getFront() {

        return this.count > 0 ? this.head.val : -1;

    }

    /\*\* Get the last item from the deque. \*/

    public int getRear() {

        return this.count > 0 ? this.tail.val : -1;

    }

    /\*\* Checks whether the circular deque is empty or not. \*/

    public boolean isEmpty() {

        return this.count == 0;

    }

    /\*\* Checks whether the circular deque is full or not. \*/

    public boolean isFull() {

        return this.count == this.capacity;

    }

}

/\*\*

 \* Your MyCircularDeque object will be instantiated and called as such:

 \* MyCircularDeque obj = new MyCircularDeque(k);

 \* boolean param\_1 = obj.insertFront(value);

 \* boolean param\_2 = obj.insertLast(value);

 \* boolean param\_3 = obj.deleteFront();

 \* boolean param\_4 = obj.deleteLast();

 \* int param\_5 = obj.getFront();

 \* int param\_6 = obj.getRear();

 \* boolean param\_7 = obj.isEmpty();

 \* boolean param\_8 = obj.isFull();

 \*/

[42. 接雨水](https://leetcode-cn.com/problems/trapping-rain-water/)

class Solution {

public int trap(int[] height) {

    int ans = 0, current = 0;

    Deque<Integer> stack = new LinkedList<Integer>();

    while (current < height.length) {

        while (!stack.isEmpty() && height[current] > height[stack.peek()]) {

            int top = stack.pop();

            if (stack.isEmpty())

                break;

            int distance = current - stack.peek() - 1;

            int bounded\_height = Math.min(height[current], height[stack.peek()]) - height[top];

            ans += distance \* bounded\_height;

        }

        stack.push(current++);

    }

    return ans;

}

}