BITTIGER

CLASS_1
LINEAR BASIC

Ben

我的一些小要求：

1. 周二，周四和作业题目 课下一定要自己动笔写写写写写写。

2. 积极的互动和反馈。

3. 全开放式课堂，诸事皆宜，百无禁忌。

Content of Class_1

String	*LinkedList*	*Array*
415. Add Strings	*203. Remove Linked List Elements*	*27. Remove Element*
242. Valid Anagram	*141. Linked List Cycle*	*283. Move Zeroes*

415. Add Strings

Given two non-negative integers `num1` and `num2` represented as string, return the sum of `num1` and `num2`.

Note:

1. The length of both `num1` and `num2` is < 5100.
2. Both `num1` and `num2` contains only digits `0-9`.
3. Both `num1` and `num2` does not contain any leading zero.
4. You **must not use any built-in BigInteger library** or **convert the inputs to integer** directly.

10 min

```
public String addStrings(String num1, String num2) {

}
```

```java
public String addStrings(String num1, String num2) {

    if(num1 == null || num1.length() == 0){
        return num2;
    }

    if(num2 == null || num2.length() == 0){
        return num1;
    }

    StringBuilder sb = new StringBuilder();
    int index1 = num1.length() - 1;
    int index2 = num2.length() - 1;
    int carry = 0;

    while(index1 >= 0 || index2 >= 0){
        int x = index1 < 0 ? 0 : num1.charAt(index1) - '0';
        int y = index2 < 0 ? 0 : num2.charAt(index2) - '0';

        sb.append((x + y + carry) % 10);
        carry = (x + y + carry) / 10;

        index1--;
        index2--;
    }

    if(carry == 1){
        sb.append('1');
    }

    return sb.reverse().toString();

}
```

corner/base case

诸位相加，溢出补零

检查carry位

203. Remove Linked List Elements

Remove all elements from a linked list of integers that have value ***val***.

Example

Given: 1 --> 2 --> 6 --> 3 --> 4 --> 5 --> 6, ***val*** = 6

Return: 1 --> 2 --> 3 --> 4 --> 5

10 min

```
public ListNode removeElements(ListNode head, int val) {

}
```

```
3    // Definition for singly-linked list.
4    public class ListNode {
5        int val;
6        ListNode next;
7        ListNode(int x) { val = x; }
8    }
```

```java
public class Solution {
    public ListNode removeElements(ListNode head, int val) {
        if(head == null){
            return null;
        }

        ListNode dummy = new ListNode(-1);
        dummy.next = head;

        ListNode cur = dummy;

        while(cur != null && cur.next != null) {
            if(cur.next.val == val){
                cur.next = cur.next.next;
            }else{
                cur = cur.next;
            }
        }

        return dummy.next;
    }
}
```

dummy node 用法

linked list 删除节点的方法

141. Linked List Cycle

Given a linked list, determine if it has a cycle in it.

Follow up:

Can you solve it without using extra space?

10 min

public boolean hasCycle(ListNode head) {

}

```
3    // Definition for singly-linked list.
4    public class ListNode {
5        int val;
6        ListNode next;
7        ListNode(int x) { val = x; }
8    }
```

```
3   public boolean hasCycle(ListNode head) {
4       if(head == null || head.next == null){
5           return false;
6       }
7
8       ListNode slow = head;
9       ListNode fast = head.next;
10
11      while(fast != null && fast.next != null){
12          if(slow == fast){
13              return true;
14          }
15          slow = slow.next;
16          fast = fast.next.next;
17      }
18
19      return false;
20  }
```

快慢指针 追及

移动快慢指针

27. Remove Element

Given an array and a value, remove all instances of that value in place and return the new length.

Do not allocate extra space for another array, you must do this in place with constant memory.

The order of elements can be changed. It doesn't matter what you leave beyond the new length.

Example:

Given input array *nums* = `[3,2,2,3]`, *val* = `3`

Your function should return length = 2, with the first two elements of *nums* being 2.

10 min

```
public int removeElement(int[] nums, int val) {

}
```

```
4   public int removeElement(int[] nums, int val) {
5       if(nums == null || nums.length == 0){
6           return 0;
7       }
8       int beg = 0;
9
10      for(int i = 0; i < nums.length; i++){
11          if(nums[i] != val){
12              nums[beg] = nums[i];
13              beg++;
14          }
15      }
16
17      return beg;
18  }
```

特定情况下移动beg 指针

283. Move Zeroes

Given an array `nums`, write a function to move all `0`'s to the end of it while maintaining the relative order of the non-zero elements.

For example, given `nums = [0, 1, 0, 3, 12]`, after calling your function, `nums` should be `[1, 3, 12, 0, 0]`.

Note:

1. You must do this **in-place** without making a copy of the array.
2. Minimize the total number of operations.

10 min

```
public void moveZeroes(int[] nums) {

}
```

```java
public void moveZeroes(int[] nums) {
    if(nums == null || nums.length == 0){
        return;
    }
    int pre = 0;

    for(int i = 0; i < nums.length; i++){
        if(nums[i] != 0){
            swap(nums, pre, i);
            pre++;
        }
    }

    return;
}

public void swap(int[] nums, int i, int j){
    int temp = nums[i];
    nums[i] = nums[j];
    nums[j] = temp;
}
```

把非零element向前移动

242. Valid Anagram

Given two strings *s* and *t*, write a function to determine if *t* is an anagram of *s*.

For example,

s = "anagram", *t* = "nagaram", return true.

s = "rat", *t* = "car", return false.

Note:

You may assume the string contains only lowercase alphabets.

10 min

```
public boolean isAnagram(String s, String t) {

}
```

```java
public boolean isAnagram(String s, String t) {
    if(s.equals(t)){
        return true;
    }

    if(s.length() != t.length()){
        return false;
    }

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

    for(int i = 0; i < s.length(); i++){
        map.put(s.charAt(i), map.getOrDefault(s.charAt(i), 0) + 1);
        map.put(t.charAt(i), map.getOrDefault(t.charAt(i), 0) - 1);
    }

    for(int cur : map.values()){
        if(cur != 0){
            return false;
        }
    }
    return true;
}
```

hash map / int array 统计词频

```java
public boolean isAnagram(String s, String t) {
    if(s.equals(t)){
        return true;
    }
    if(s.length() != t.length()){
        return false;
    }
    int[] map = new int[128];
    for(int i = 0; i < s.length(); i++){
        map[s.charAt(i) - 'a']++;
        map[t.charAt(i) - 'a']--;
    }
    for(int cur : map){
        if(cur != 0){
            return false;
        }
    }
    return true;
}
```

hash map / int array 统计词频

Homework

String	*LinkedList*	*Array*
49. Group Anagrams	*19. Remove Nth Node From End of List*	*66. Plus One*
344. Reverse String		

Q & A

Thank you