TWO POINTERS

PROBLEM SET

- No.19 Remove Nth Node From End of List
- No.86 Partition List
- No.141 Linked List Cycle
- No.142 Linked List Cycle II
- No.143 Reorder List
- No.167 Two Sum II - Input array is sorted
- No.283 Move Zeroes
- No.287 Find the Duplicate Number
- No.344 Reverse String
- No.345 Reverse Vowels of a String
- No.349 Intersection of Two Arrays
- No.350 Intersection of Two Arrays II
- No.457 Circular Array Loop

NO. 19

REMOVE NTH NODE FROM END OF LIST

PROBLEM DESCRIPTION

- Given a linked list, remove the nth node from the end of list and return its head
- For example
  - Given linked list: 1->2->3->4->5, and n = 2.
  - Return 1->2->3->5.

IDEA

- Fast-Slow pointers
  - Fast pointer goes n step first
  - Both fast and slow pointers move forward
  - Once fast pointer reaches the end, slow pointer arrives the nth node counting from the end

```
for _ in xrange(n): fast = fast.next
while fast.next:
        fast, slow = fast.next, slow.next
slow.next = slow.next.next
```

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/19_Remove_Nth_Node_From_End_of_List

NO. 86

PARTITION LIST

PROBLEM DESCRIPTION

- Given a linked list and a value x, partition it such that all nodes less than x come before nodes greater than or equal to x. You should preserve the original relative order of the nodes in each of the two partitions.
- For example
  - Given 1->4->3->2->5->2 and x = 3
  - Return 1->2->2->4->3->5.

IDEA

- Maintain 2 lists
  - smallerList
  - largerList
- Combine these 2 lists
  - largerList.next = None
  - smallerList.next = largerListHead.next

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/86_Partition_List

NO. 141

LINKED LIST CYCLE

PROBLEM DESCRIPTION

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

IDEA

- Fast-Slow pointers
  - fast = fast.next.next
  - slow = slow.next
- If fast meets slow, there must be a circle

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/141_Linked_List_Cycle

NO. 142

LINKED LIST

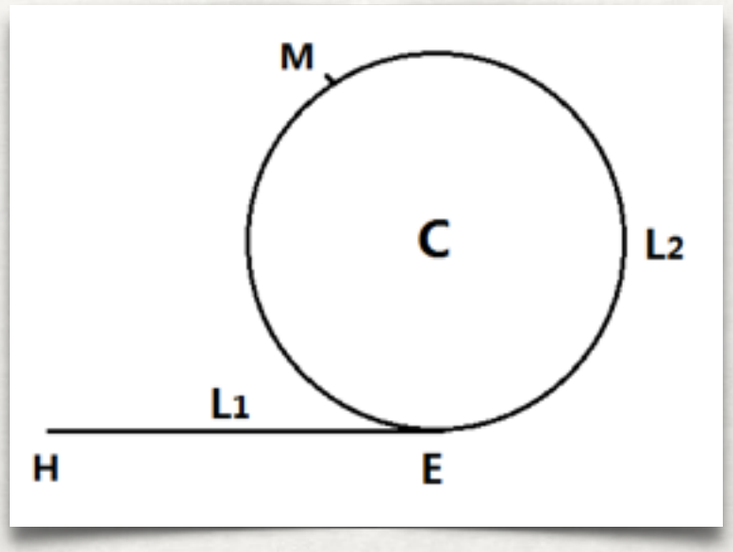
CYCLE II

PROBLEM DESCRIPTION

- Given a linked list, return the node where the cycle begins. If there is no cycle, return null.

IDEA

- Fast and Slow pointers will meet at M
- Slow: L1 + L2
- // Assume fast ran a circle already
- Fast: L1 + L2 + 1 * C = 2 * (L1 + L2)
- C = L1 + L2
- L1 = C - L2



SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/142_Linked_List_Cycle_II

NO. 143

REORDER LIST

PROBLEM DESCRIPTION

- Given a singly linked list L: L0→L1→…→Ln-1→Ln, reorder it to: L0→Ln→L1→Ln-1→L2→Ln-2→…
- For example,
  - Given {1,2,3,4}, reorder it to {1,4,2,3}.

IDEA

- Find the mid point, separate list into two halves
- Reverse the second half
- Concat these two halves

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/143_Reorder_List

NO. 167

TWO SUM II - INPUT ARRAY IS SORTED

PROBLEM DESCRIPTION

- Given an array of integers that is already sorted in ascending order, find two numbers such that they add up to a specific target number.
- Example
- Input: numbers={2, 7, 11, 15}, target=9
- Output: index1=1, index2=2

IDEA

- Head-Tail pointers
  - head to the front, tail to the end
- If arr[head] + arr[tail] > target: tail -= 1
- If arr[head] + arr[tail] < target: head += 1
- Otherwise, we found the answer

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/167_Two_Sum_II

NO. 283

MOVE ZEROES

PROBLEM DESCRIPTION

- 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],
  - Return [1, 3, 12, 0, 0].

IDEA

- Two pointers
  - zeroPos
  - cursor
- Cursor traverses from begin to end, if cursor value is not 0
  - zeroPos moves forward
  - Swap cursor value and zeroPos value // 自己跟自己換無妨
- If cursor value is 0
  - zeroPos stays for the swap in the future

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/283_Move_Zeroes

NO. 287

FIND THE DUPLICATE NUMBER

PROBLEM DESCRIPTION

- Given an array nums containing n + 1 integers where each integer is between 1 and n (inclusive), prove that at least one duplicate number must exist. Assume that there is only one duplicate number, find the duplicate one.
- You must not modify the array (assume the array is read only).
- You must use only constant, O(1) extra space.
- Your runtime complexity should be less than O(n2).
- There is only one duplicate number in the array, but it could be repeated more than once.

IDEA

- Use fast-slow pointers lead to TLE
  - node i points to node nums[i] to form a single linked list
  - Others are exactly the same as No. 142

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/287_Find_the_Duplicate_Number

NO. 344

REVERSE STRING

PROBLEM DESCRIPTION

- Write a function that takes a string as input and returns the string reversed.
- Example: Given s = "hello", return "olleh".

IDEA

- Head-Tail pointers change their values
  - head ptr moves forward
  - tail ptr moves backward

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/344_Reverse_String

NO. 345

REVERSE VOWELS OF A STRING

PROBLEM DESCRIPTION

- Write a function that takes a string as input and reverse only the vowels of a string.
- Example 1:
  - Given s = "hello", return "holle".

IDEA

- Convert input string to list
- Head pointer points to the front
- Tail pointer points to the end
- Head can not surpass tail
- If head is not vowel, moves forward. Otherwise, stays
- If tail is not vowel, move backward. Otherwise, stays
- If head is vowel and tail is vowel as well, swap

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/345_Reverse_Vowels_of_a_String

NO. 349

INTERSECTION OF TWO ARRAYS

PROBLEM DESCRIPTION

- Given two arrays, write a function to compute their intersection.
- Example:
  - Given nums1 = [1, 2, 2, 1], nums2 = [2, 2], return [2].

IDEA

- Convert both lists to sets
- Traverse one set, if element exists in another, add to ans list

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/349_Intersection_of_Two_Arrays

NO. 350

INTERSECTION OF TWO ARRAYS II

PROBLEM DESCRIPTION

- Given two arrays, write a function to compute their intersection.
- Example:
  - Given nums1 = [1, 2, 2, 1], nums2 = [2, 2], return [2, 2].

IDEA

- User Python sugar coating built-in function Counter()
- Counter(list1) & Counter(list2) to get intersection
- use .element() to get intersection elements

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/350_Intersection_of_Two_Arrays_II

NO. 467 (DRAFT)

CIRCULAR ARRAY LOOP

PROBLEM DESCRIPTION

- You are given an array of positive and negative integers. If a number n at an index is positive, then move forward n steps. Conversely, if it's negative (-n), move backward n steps. Assume the first element of the array is forward next to the last element, and the last element is backward next to the first element. Determine if there is a loop in this array. A loop starts and ends at a particular index with more than 1 element along the loop. The loop must be "forward" or "backward'.

- Example 1:

  - Given the array [2, -1, 1, 2, 2], there is a loop, from index 0 -> 2 -> 3 -> 0.

- Example 2:

  - Given the array [-1, 2], there is no loop.

IDEA

- Similar to No.141 Linked List Cycle
- Fast-slow pointers
  - fast goes 2 step forwards or backwards
  - slow goes 1 step forwards or backwards
  - if fast meets slow, there is a circle
- Consider early reject conditions
  - Mark visited node
  - A -> B -> C -> A
    - if C can’t reach A, A&B can’t reach A either

SOLUTION

- https://github.com/Brady31027/leetcode/tree/master/457_Circular_Array_Loop