LeetCode Solutions

Sankalp Sangle

Last updated on May 7, 2020

Contents

Part I

LeetCode Top Interview Questions

Chapter 1

Easy

Link: LeetCode Top Interview Questions: Easy section.

1.1 Arrays

Link: Arrays

1.1.1 26. Remove Duplicates from Sorted Array

Link to question, Link to submission

Concepts Two pointer

Algorithm description

- Maintain a read pointer and a write pointer, both starting from zero.
- Advance the write pointer until you see a new value or reach end of array.
- Write value at write location into read location.

- Return read.

1.1.2 122. Best Time to Buy and Sell Stock II

Link to question, Link to submission

Concepts Greedy

Algorithm description

- Construct a consecutive elements difference array
- Return sum of all positive elements in difference array

1.1.3 189. Rotate Array

Link to question, Link to submission approach 1, Link to submission approach 2

Concepts Cyclic replacements, Implementation

Approach 1 description

- Maintain a visited array and a pointer initialized to 0
- while pointer + k is not visited, replace arr[pointer + k] with arr[pointer]. Update pointer to pointer + k. Set pointer + k to visited, increment a numberOfChanges variable.
- Increment pointer by 1
- Keep doing this while numberOfChanges less than size of array.

Approach 2 description

- Reverse the entire array
- Reverse from start to start + k
- Reverse from start + k to end

1.1.4 217. Contains Duplicate

Link to question, Link to submission

Concepts Hash Table, Set

Algorithm description

- Initialize a Set
- For an element in array, if element in Set, return true
- else add element to Set
- If out of loop, return False

1.1.5 136. Single Number

Link to question, Link to submission

Concepts Bit Manipulation, XOR

Algorithm description

- Initialize an answer variable to 0
- For every element, XOR it to answer. Elements appearing twice get XOR'd out to zero

- Return answer

1.1.6 350. Intersection of Two Arrays II

Link to question, Link to submission approach 1, Link to submission approach 2

Concepts Hash Table, Two Pointers

Approach 1 description

- Form an element:frequency mapping using map for smaller array (to save space)
- Traverse bigger array
- If frequency of element less than 0, add to answer. Decrement frequency

Approach 2 description

- If arrays are sorted, use two pointers p1 and p2
- If nums1[p1] == nums2[p2], add to answer and increment both
- Else if nums1[p1] is smaller, increment p1. Else increment p2
- Keep doing until reach end of either array

1.1.7 66. Plus One

Link to question, Link to submission

Concepts Array

Algorithm description

- Initialize a carry variable to 1
- Traverse array from the end.
  digit[i] = carry + digit mod 10, carry = carry + digit div 10
- Finally, if carry is not zero, insert carry at start of array

1.1.8 283. Move Zeroes

Link to question, Link to submission

Concepts Two Pointers

Algorithm description

- Maintain a read and a write pointer, both initialized to 0
- if read end has zero, increment read end
- else, copy read end to write end and increment both
- After read end reaches end, set all numbers from write end to end as 0

1.1.9 1. Two Sum

Link to question, Link to submission approach 1, Link to submission approach 2

Concepts Hash Table, Two Pointer

Approach 1 description

- Create an element:indices mapping
- Sort the array
- Use two pointers to search for a particular sum
- Once you find the sum, pop index from left pointer, and pop index from right pointer
- Return indices

Approach 2 description

- Create a hashmap of int, int
- Iterate the array with i as looping variable
- If element in hashmap, return (hashmap[element], i)
- Else insert hashmap[target - element] = i

1.1.10 36. Valid Sudoku

Link to question, Link to submission

Concepts Hash Table, Set

Algorithm description

- Create sets to hold numbers for each row, col and square.
- Traverse the sudoku
- If a number is already in the row, col, square, return False
- Else, come out of loop and return true

1.1.11 48. Rotate Image

Link to question, Link to submission

Concepts Array, Circular Permutation

Algorithm description

- Do a counterclockwise circular permutation as mentioned in solution
- Pure implementation problem. No algorithmic skill.

1.2 Strings

Link: Strings

1.2.1 344. Reverse String

Link to question, Link to submission

Concepts Two Pointers

Algorithm description

- Set a left pointer to start of string, right pointer to end
- Swap left and right. Increment left, decrement right
- Do while l less than r

1.2.2 7. Reverse Integer

Link to question, Link to submission

Concepts Two Pointers

Algorithm description

- Reverse the integer by converting to a string
- Store result in long
- If stored result is outside integer limits, return 0
- Else return the reversed number

1.2.3 387. First Unique Character in a String

Link to question, Link to submission

Concepts Hash Map

Algorithm description

- Construct element frequency mapping
- Traverse the string from the start, if frequency of a char is 1, return index
- If reach end of string, return -1

1.2.4 242. Valid Anagram

Link to question, Link to submission

Concepts Hash Map, Counting Sort

Algorithm description

- Traverse through s1, incrementing frequency counts
- Traverse through s2, decrementing frequency counts
- If all counts are zero, return true. Else false.

1.2.5 125. Valid Palindrome

Link to question, Link to submission

Concepts Two Pointers

Algorithm description

- Maintain a left and a right pointer
- Before comparing the two, ensure left and right both are pointing to an alphanumeric character

1.2.6 28. Implement strStr()

Link to question, Link to Approach 1, Link to Approach 2

Concepts Two Pointers, Rabin-Karp Algorithm, Rolling Hash

Approach 1 description

- Traverse haystack until you find a character matching with first character of needle
- Once match is found, keep checking for further characters until either there's a mismatch or you reach end of arrays
- Return index accordingly

Approach 2 description - Rabin-Karp

- Hash the needle using a hash function that is easy to be "rolled", that is it is easy to compute hash for next window if hash for previous window is known
- Traverse the haystack using window of length needle.length(). Hash the window and compare with needle hash. If matched, return the index of start of window
- See implementation carefully, very interesting. Also see LeetCode solution article.

1.2.7 38. Count and Say

Link to question, Link to submission

Concepts Recursion, Two Pointers

Algorithm description

- Base case: n = 1, return "1"
- Get the answer for n-1
- Traverse through answer of n-1
- For each consecutive list of same elements, add the count, followed by the element
- Return answer

1.2.8 14. Longest Common Prefix

Link to question, Link to submission

Concepts Implementation

Algorithm description

- Initialize answer string to ""
- Find length of smallest string
- For i from 0 to min length - 1
- Traverse through all the characters at ith positions
- If different, return answer
- If same, add character to answer

1.3 Linked Lists

Link: Linked Lists

1.3.1 237. Delete Node in a Linked List

Link to question, Link to submission

Concepts Trick

Algorithm description

- Copy value of next node into current node
- Set next ptr of current node to next ptr of next node

1.3.2 19. Remove Nth Node From End of List

Link to question, Link to submission

Concepts Two Pointer

Algorithm description

- To do it in one pass, let a forward pointer advance n steps
- Then, start forwarding a slow pointer as well as the forward pointer one at a time until forward reaches the end
- delete the slow pointer node

1.3.3 206. Reverse Linked List

Link to question, Link to iterative approach, Link to recursive approach

Concepts Implementation

Approach 1 description

- Initialize a prev = NULL, and a curr = head
- While head is not NULL, do a cyclic swap between curr.next, prev, and curr.
- Return prev

Approach 2 description

- If head is NULL or head.next is NULL return head
- l = reversed list for head.next
- head.next.next = head, head.next = NULL. Return l

1.3.4 21. Merge Two Sorted Lists

Link to question, Link to iterative submission, Link to recursive submission

Concepts Two Pointers

Algorithm description Iterative

- Make a dummy node, and let tmp = dummynode
- Keep appending the smaller of the two lists to the dummy node and advance the pointers accordingly
- If one of the lists becomes NULL, append the other list to dummy node
- Return next of tmp

Algorithm description Recursive

- If either of lists is NULL, return the other
- if l1 is smaller, get answer to (l1.next, l2) and set it as l1.next. Return l1
- Else get answer to (l1, l2.next) and set it as l2.next. Return l2

1.3.5 234. Palindrome Linked List

Link to question, Link to submission

Concepts Reverse a linked list, Two Pointers

Algorithm description

- Reverse the second half of the linked list
- Compare nodewise the head of linked list and the head of reversed list to check for palindrome

1.3.6 141. Linked List Cycle

Link to question, Link to submission

Concepts Hare and Tortoise, Two Pointers

Algorithm description

- Initialize a slow and a fast pointer
- Advance slow by 1, fast by 2
- If slow and fast meet, there's a cycle. Else if fast reaches end, there's no cycle.

1.4 Trees

Link: Trees

1.4.1 104. Maximum Depth of Binary Tree

Link to question, Link to recursive submission, Link to iterative submission

Concepts Recursion, Stack

Algorithm description Recursive

- If root is null, return 0
- Else return 1 + max(maxDepth(left), maxDepth(right))

Algorithm description Iterative

- If root is null, return 0
- Initialize stack holding pair of TreeNode and depth
- Push {root, 1}
- While stack is not empty, get top of stack
- If top is leaf, compare with maxDepth
- Push children if any with depth = 1 + parent depth

1.4.2 98. Validate Binary Search Tree

Link to question, Link to iterative submission, Link to recursive submission

Concepts Top-Down

Algorithm description (for recursive/iterative)

- Approach is a top-down one
- At every node, check if node.val is between a range of [small, large]
- If not, return False
- else check left subtree for range[small, node.val] and check right subtree for range[node.val, large]
- Return the AND of the above two

1.4.3 101. Symmetric Tree

Link to question, Link to recursive submission, Link to iterative submission

Concepts Top-Down

Algorithm description (for recursive/iterative)

- Top down approach
- Check if leftTree.val == rightTree.val
- If true, check for leftTree.left, rightTree.right and leftTree.right, rightTree.left
- Else, return False

1.4.4 102. Binary Tree Level Order Traversal

Link to question, Link to submission

Concepts Top-Down, BFS

Algorithm description

- Push root into a queue
- At beginning of an iteration, take size of queue
- Pop out #size items from queue, while adding their children to queue
- Add to level
- Add level to final answer

1.4.5 108. Convert Sorted Array to Binary Search Tree

Link to question, Link to submission

Concepts Recursion, Preorder

Algorithm description

- call procedure with left = 0, right = arr.size() - 1
- if left ¿ right, return NULL
- construct node for middle element
- node.left = procedure(left, middle-1), node.right = procedure(middle+1, right)
- return node

1.5 Sorting and Searching

Link: Sorting and Searching

1.5.1 88. Merge Sorted Array

Link to question, Link to submission

Concepts Two Pointers

Algorithm description

- Create a copy array for nums1
- Maintain write pointer for nums1, p1 for nums1copy, p2 for nums2
- Write smaller of p1, p2 into nums1. Advance smaller and write head.
- Once out of the loop, see which array still has elements remaining. Add them to nums1

1.5.2 278. First Bad Version

Link to question, Link to submission

Concepts Binary Search

Algorithm description

- set left as 0, right as n - 1
- while l ¡= r
- if mid is bad, right = middle - 1

- else left = middle + 1
- Once you come out of loop, return l

1.6 Dynamic Programming

Link: Dynamic Programming

1.6.1 70. Climbing Stairs

Link to question, Link to submission

Concepts Dynamic Programming

Algorithm description

- Ways to reach ith step = ways to reach i-1 th step plus ways to reach i-2 th step

1.6.2 121. Best Time to Buy and Sell Stock

Link to question, Link to submission

Concepts Dynamic Programming

Algorithm description

- Maintain a smallest stock price seen yet variable
- Update maxProfit = max(maxProfit, current price - maxProfit)

1.6.3 53. Maximum Subarray

Link to question, Link to submission

Concepts Dynamic Programming

Algorithm description

- Maintain a current sum variable, denoting the highest sum possible that contains the element at the index
- Maintain a highest sum variable, denoting the highest sum encountered among the current sums

1.6.4 198. House Robber

Link to question, Link to submission, Link to submission (space optimized)

Concepts Dynamic Programming

Algorithm description

- Maintain a dp array with dp[0] = nums[0], dp[1] = max(nums[0], nums[1]). dp[i] denotes maximum amount that can be robbed with first i+1 houses
- dp[i] = max(dp[i-1], dp[i-2] + nums[i])
- Finally return dp[n-1]

1.7 Design

Link: Design