# 1.Two Sum(easy)

Given an array of integers, return **indices** of the two numbers such that they add up to a specific target.

You may assume that each input would have ***exactly*** one solution, and you may not use the *same* element twice.

**Example:**

Given nums = [2, 7, 11, 15], target = 9,

Because nums[**0**] + nums[**1**] = 2 + 7 = 9,

return [**0**, **1**].

输入：

[3,2,4]

6

输出：

[1,2]

# 7. Reverse Integer（easy）

Given a 32-bit signed integer, reverse digits of an integer.

**Example 1:**

**Input:** 123

**Output:** 321

**Example 2:**

**Input:** -123

**Output:** -321

**Example 3:**

**Input:** 120

**Output:** 21

**Note:**  
Assume we are dealing with an environment which could only hold integers within the 32-bit signed integer range. For the purpose of this problem, assume that your function returns 0 when the reversed integer overflows.

# 9. Palindrome Number(easy)

Determine whether an integer is a palindrome. Do this without extra space.

**Some hints:**

Could negative integers be palindromes? (ie, -1)

If you are thinking of converting the integer to string, note the restriction of using extra space.

You could also try reversing an integer. However, if you have solved the problem "Reverse Integer", you know that the reversed integer might overflow. How would you handle such case?

There is a more generic way of solving this problem.

# 13. Roman to Integer(easy)

Easy

10242331FavoriteShare

Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

**Symbol** **Value**

I 1

V 5

X 10

L 50

C 100

D 500

M 1000

For example, two is written as II in Roman numeral, just two one's added together. Twelve is written as, XII, which is simply X + II. The number twenty seven is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

* I can be placed before V (5) and X (10) to make 4 and 9.
* X can be placed before L (50) and C (100) to make 40 and 90.
* C can be placed before D (500) and M (1000) to make 400 and 900.

Given a roman numeral, convert it to an integer. Input is guaranteed to be within the range from 1 to 3999.

**Example 1:**

**Input:** "III"

**Output:** 3

**Example 2:**

**Input:** "IV"

**Output:** 4

**Example 3:**

**Input:** "IX"

**Output:** 9

**Example 4:**

**Input:** "LVIII"

**Output:** 58

**Explanation:** L = 50, V= 5, III = 3.

**Example 5:**

**Input:** "MCMXCIV"

**Output:** 1994

**Explanation:** M = 1000, CM = 900, XC = 90 and IV = 4.

# 14. Longest Common Prefix(easy)

Easy

10391106FavoriteShare

Write a function to find the longest common prefix string amongst an array of strings.

If there is no common prefix, return an empty string "".

**Example 1:**

**Input:** ["flower","flow","flight"]

**Output:** "fl"

**Example 2:**

**Input:** ["dog","racecar","car"]

**Output:** ""

**Explanation:** There is no common prefix among the input strings.

**Note:**

All given inputs are in lowercase letters a-z.

# 20. Valid Parentheses(easy)

Easy

2371117FavoriteShare

Given a string containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.

Note that an empty string is also considered valid.

**Example 1:**

**Input:** "()"

**Output:** true

**Example 2:**

**Input:** "()[]{}"

**Output:** true

**Example 3:**

**Input:** "(]"

**Output:** false

**Example 4:**

**Input:** "([)]"

**Output:** false

**Example 5:**

**Input:** "{[]}"

**Output:** true

# 21. Merge Two Sorted Lists(easy)

Easy

1753240FavoriteShare

Merge two sorted linked lists and return it as a new list. The new list should be made by splicing together the nodes of the first two lists.

**Example:**

**Input:** 1->2->4, 1->3->4

**Output:** 1->1->2->3->4->4

# 26. Remove Duplicates from Sorted Array(easy)

Easy

12682647FavoriteShare

Given a sorted array nums, remove the duplicates [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm) such that each element appear only once and return the new length.

Do not allocate extra space for another array, you must do this by **modifying the input array**[**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm) with O(1) extra memory.

**Example 1:**

Given nums = **[1,1,2]**,

Your function should return length = **2**, with the first two elements of *nums* being **1** and **2** respectively.

It doesn't matter what you leave beyond the returned length.

**Example 2:**

Given nums = **[0,0,1,1,1,2,2,3,3,4]**,

Your function should return length = **5**, with the first five elements of *nums* being modified to **0**, **1**, **2**, **3**, and **4** respectively.

It doesn't matter what values are set beyond the returned length.

**Clarification:**

Confused why the returned value is an integer but your answer is an array?

Note that the input array is passed in by **reference**, which means modification to the input array will be known to the caller as well.

Internally you can think of this:

// **nums** is passed in by reference. (i.e., without making a copy)

int len = removeDuplicates(nums);

// any modification to **nums** in your function would be known by the caller.

// using the length returned by your function, it prints the first **len** elements.

for (int i = 0; i < len; i++) {

    print(nums[i]);

}

# 27. Remove Element(easy)

Easy

6751466FavoriteShare

Given an array nums and a value val, remove all instances of that value [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm) and return the new length.

Do not allocate extra space for another array, you must do this by **modifying the input array**[**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm) with O(1) extra memory.

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

**Example 1:**

Given nums = **[3,2,2,3]**, val = **3**,

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

It doesn't matter what you leave beyond the returned length.

**Example 2:**

Given nums = **[0,1,2,2,3,0,4,2]**, val = **2**,

Your function should return length = **5**, with the first five elements of *nums* containing **0**, **1**, **3**, **0**, and **4**.

Note that the order of those five elements can be arbitrary.

It doesn't matter what values are set beyond the returned length.

**Clarification:**

Confused why the returned value is an integer but your answer is an array?

Note that the input array is passed in by **reference**, which means modification to the input array will be known to the caller as well.

Internally you can think of this:

// **nums** is passed in by reference. (i.e., without making a copy)

int len = removeElement(nums, val);

// any modification to **nums** in your function would be known by the caller.

// using the length returned by your function, it prints the first **len** elements.

for (int i = 0; i < len; i++) {

    print(nums[i]);

}

# 28. Implement strStr()(easy)

Easy

7241117FavoriteShare

Implement [strStr()](http://www.cplusplus.com/reference/cstring/strstr/" \t "_blank).

Return the index of the first occurrence of needle in haystack, or **-1** if needle is not part of haystack.

**Example 1:**

**Input:** haystack = "hello", needle = "ll"

**Output:** 2

**Example 2:**

**Input:** haystack = "aaaaa", needle = "bba"

**Output:** -1

**Clarification:**

What should we return when needle is an empty string? This is a great question to ask during an interview.

For the purpose of this problem, we will return 0 when needle is an empty string. This is consistent to C's [strstr()](http://www.cplusplus.com/reference/cstring/strstr/" \t "_blank) and Java's [indexOf()](https://docs.oracle.com/javase/7/docs/api/java/lang/String.html" \l "indexOf(java.lang.String)" \t "_blank).

Accepted

361,703

Submissions

1,172,841

# 35. Search Insert Position(easy)

Easy

1068154FavoriteShare

Given a sorted array and a target value, return the index if the target is found. If not, return the index where it would be if it were inserted in order.

You may assume no duplicates in the array.

**Example 1:**

**Input:** [1,3,5,6], 5

**Output:** 2

**Example 2:**

**Input:** [1,3,5,6], 2

**Output:** 1

**Example 3:**

**Input:** [1,3,5,6], 7

**Output:** 4

**Example 4:**

**Input:** [1,3,5,6], 0

**Output:** 0

# 38. Count and Say(easy)

Easy

6254420FavoriteShare

The count-and-say sequence is the sequence of integers with the first five terms as following:

1. 1

2. 11

3. 21

4. 1211

5. 111221

1 is read off as "one 1" or 11.  
11 is read off as "two 1s" or 21.  
21 is read off as "one 2, then one 1" or 1211.

Given an integer *n* where 1 ≤ *n* ≤ 30, generate the *n*th term of the count-and-say sequence.

Note: Each term of the sequence of integers will be represented as a string.

**Example 1:**

**Input:** 1

**Output:** "1"

**Example 2:**

**Input:** 4

**Output:** "1211"

# 53. Maximum Subarray(easy)

Easy

3394119FavoriteShare

Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

**Example:**

**Input:** [-2,1,-3,4,-1,2,1,-5,4],

**Output:** 6

**Explanation:** [4,-1,2,1] has the largest sum = 6.

**Follow up:**

If you have figured out the O(*n*) solution, try coding another solution using the divide and conquer approach, which is more subtle.

就是一个最大子列和问题，用动态规划的思路解决~

# 58. Length of Last Word(easy)

Easy

3071237FavoriteShare

Given a string *s* consists of upper/lower-case alphabets and empty space characters ' ', return the length of last word in the string.

If the last word does not exist, return 0.

**Note:** A word is defined as a character sequence consists of non-space characters only.

**Example:**

**Input:** "Hello World"

**Output:** 5

# 66. Plus One(easy)

Easy

7141303FavoriteShare

Given a **non-empty** array of digits representing a non-negative integer, plus one to the integer.

The digits are stored such that the most significant digit is at the head of the list, and each element in the array contain a single digit.

You may assume the integer does not contain any leading zero, except the number 0 itself.

**Example 1:**

**Input:** [1,2,3]

**Output:** [1,2,4]

**Explanation:** The array represents the integer 123.

**Example 2:**

**Input:** [4,3,2,1]

**Output:** [4,3,2,2]

**Explanation:** The array represents the integer 4321.

# 67. Add Binary(easy)

Easy

784163FavoriteShare

Given two binary strings, return their sum (also a binary string).

The input strings are both **non-empty** and contains only characters 1 or 0.

**Example 1:**

**Input:** a = "11", b = "1"

**Output:** "100"

**Example 2:**

**Input:** a = "1010", b = "1011"

**Output:** "10101"

# 69. Sqrt(x)(easy)

Easy

6231151FavoriteShare

Implement int sqrt(int x).

Compute and return the square root of *x*, where *x* is guaranteed to be a non-negative integer.

Since the return type is an integer, the decimal digits are truncated and only the integer part of the result is returned.

**Example 1:**

**Input:** 4

**Output:** 2

**Example 2:**

**Input:** 8

**Output:** 2

**Explanation:** The square root of 8 is 2.82842..., and since

  the decimal part is truncated, 2 is returned.

注意int类型的表示范围可能导致溢出

# 70. Climbing Stairs(easy)

Easy

177167FavoriteShare

You are climbing a stair case. It takes *n* steps to reach to the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

**Note:** Given *n* will be a positive integer.

**Example 1:**

**Input:** 2

**Output:** 2

**Explanation:** There are two ways to climb to the top.

1. 1 step + 1 step

2. 2 steps

**Example 2:**

**Input:** 3

**Output:** 3

**Explanation:** There are three ways to climb to the top.

1. 1 step + 1 step + 1 step

2. 1 step + 2 steps

3. 2 steps + 1 step

本质就是个斐波那契

# 83. Remove Duplicates from Sorted List(easy)

Easy

65772FavoriteShare

Given a sorted linked list, delete all duplicates such that each element appear only *once*.

**Example 1:**

**Input:** 1->1->2

**Output:** 1->2

**Example 2:**

**Input:** 1->1->2->3->3

**Output:** 1->2->3

# 88. Merge Sorted Array(easy)

Easy

9392470FavoriteShare

Given two sorted integer arrays *nums1* and *nums2*, merge *nums2* into *nums1* as one sorted array.

**Note:**

* The number of elements initialized in *nums1* and *nums2* are *m* and *n* respectively.
* You may assume that *nums1* has enough space (size that is greater or equal to *m* + *n*) to hold additional elements from *nums2*.

**Example:**

**Input:**

nums1 = [1,2,3,0,0,0], m = 3

nums2 = [2,5,6], n = 3

**Output:** [1,2,2,3,5,6]

# 100. Same Tree(easy)

Easy

99931FavoriteShare

Given two binary trees, write a function to check if they are the same or not.

Two binary trees are considered the same if they are structurally identical and the nodes have the same value.

**Example 1:**

**Input:** 1 1

/ \ / \

2 3 2 3

[1,2,3], [1,2,3]

**Output:** true

**Example 2:**

**Input:** 1 1

/ \

2 2

[1,2], [1,null,2]

**Output:** false

**Example 3:**

**Input:** 1 1

/ \ / \

2 1 1 2

[1,2,1], [1,1,2]

**Output:** false

# 101. Symmetric Tree(easy)

Easy

190843FavoriteShare

Given a binary tree, check whether it is a mirror of itself (ie, symmetric around its center).

For example, this binary tree [1,2,2,3,4,4,3] is symmetric:

1

/ \

2 2

/ \ / \

3 4 4 3

But the following [1,2,2,null,3,null,3] is not:

1

/ \

2 2

\ \

3 3

**Note:**  
Bonus points if you could solve it both recursively and iteratively.

# 104. Maximum Depth of Binary Tree(easy)

Easy

113247FavoriteShare

Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

**Note:** A leaf is a node with no children.

**Example:**

Given binary tree [3,9,20,null,null,15,7],

3

/ \

9 20

/ \

15 7

return its depth = 3.

# 107. Binary Tree Level Order Traversal II(easy)

Easy

639113FavoriteShare

Given a binary tree, return the *bottom-up level order* traversal of its nodes' values. (ie, from left to right, level by level from leaf to root).

For example:  
Given binary tree [3,9,20,null,null,15,7],

3

/ \

9 20

/ \

15 7

return its bottom-up level order traversal as:

[

[15,7],

[9,20],

[3]

]

# 108. Convert Sorted Array to Binary Search Tree(easy)

Easy

97599FavoriteShare

Given an array where elements are sorted in ascending order, convert it to a height balanced BST.

For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of *every* node never differ by more than 1.

**Example:**

Given the sorted array: [-10,-3,0,5,9],

One possible answer is: [0,-3,9,-10,null,5], which represents the following height balanced BST:

0

/ \

-3 9

/ /

-10 5

# 110. Balanced Binary Tree(easy)

Easy

106492FavoriteShare

Given a binary tree, determine if it is height-balanced.

For this problem, a height-balanced binary tree is defined as:

a binary tree in which the depth of the two subtrees of *every* node never differ by more than 1.

**Example 1:**

Given the following tree [3,9,20,null,null,15,7]:

3

/ \

9 20

/ \

15 7

Return true.  
  
**Example 2:**

Given the following tree [1,2,2,3,3,null,null,4,4]:

1

/ \

2 2

/ \

3 3

/ \

4 4

Return false.

# 111. Minimum Depth of Binary Tree(easy)

Easy

644322FavoriteShare

Given a binary tree, find its minimum depth.

The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node.

**Note:** A leaf is a node with no children.

**Example:**

Given binary tree [3,9,20,null,null,15,7],

3

/ \

9 20

/ \

15 7

return its minimum depth = 2.

# 112. Path Sum(easy)

Easy

806261FavoriteShare

Given a binary tree and a sum, determine if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum.

**Note:** A leaf is a node with no children.

**Example:**

Given the below binary tree and sum = 22,

**5**

**/** \

**4** 8

**/** / \

**11** 13 4

/ **\** \

7 **2** 1

return true, as there exist a root-to-leaf path 5->4->11->2 which sum is 22.

# 118. Pascal's Triangle(easy)

Easy

62072FavoriteShare

Given a non-negative integer *numRows*, generate the first *numRows* of Pascal's triangle.

  
In Pascal's triangle, each number is the sum of the two numbers directly above it.

**Example:**

**Input:** 5

**Output:**

[

[1],

[1,1],

[1,2,1],

[1,3,3,1],

[1,4,6,4,1]

]

# 119. Pascal's Triangle II(easy)

Easy

419153FavoriteShare

Given a non-negative index *k* where *k* ≤ 33, return the *k*th index row of the Pascal's triangle.

Note that the row index starts from 0.

  
In Pascal's triangle, each number is the sum of the two numbers directly above it.

**Example:**

**Input:** 3

**Output:** [1,3,3,1]

**Follow up:**

Could you optimize your algorithm to use only *O*(*k*) extra space?

# 121. Best Time to Buy and Sell Stock(easy)

Easy

2273113FavoriteShare

Say you have an array for which the *i*th element is the price of a given stock on day *i*.

If you were only permitted to complete at most one transaction (i.e., buy one and sell one share of the stock), design an algorithm to find the maximum profit.

Note that you cannot sell a stock before you buy one.

**Example 1:**

**Input:** [7,1,5,3,6,4]

**Output:** 5

**Explanation:** Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

  Not 7-1 = 6, as selling price needs to be larger than buying price.

**Example 2:**

**Input:** [7,6,4,3,1]

**Output:** 0

**Explanation:** In this case, no transaction is done, i.e. max profit = 0.

# 122. Best Time to Buy and Sell Stock II(easy)

Easy

8631211FavoriteShare

Say you have an array for which the *i*th element is the price of a given stock on day *i*.

Design an algorithm to find the maximum profit. You may complete as many transactions as you like (i.e., buy one and sell one share of the stock multiple times).

**Note:** You may not engage in multiple transactions at the same time (i.e., you must sell the stock before you buy again).

**Example 1:**

**Input:** [7,1,5,3,6,4]

**Output:** 7

**Explanation:** Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4.

  Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.

**Example 2:**

**Input:** [1,2,3,4,5]

**Output:** 4

**Explanation:** Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.

  Note that you cannot buy on day 1, buy on day 2 and sell them later, as you are

  engaging multiple transactions at the same time. You must sell before buying again.

**Example 3:**

**Input:** [7,6,4,3,1]

**Output:** 0

**Explanation:** In this case, no transaction is done, i.e. max profit = 0.

# 125. Valid Palindrome(easy)

Easy

5401519FavoriteShare

Given a string, determine if it is a palindrome, considering only alphanumeric characters and ignoring cases.

**Note:** For the purpose of this problem, we define empty string as valid palindrome.

**Example 1:**

**Input:** "A man, a plan, a canal: Panama"

**Output:** true

**Example 2:**

**Input:** "race a car"

**Output:** false

# 136. Single Number(easy)

Easy

221183FavoriteShare

Given a **non-empty** array of integers, every element appears *twice* except for one. Find that single one.

**Note:**

Your algorithm should have a linear runtime complexity. Could you implement it without using extra memory?

**Example 1:**

**Input:** [2,2,1]

**Output:** 1

**Example 2:**

**Input:** [4,1,2,1,2]

**Output:** 4

# 141. Linked List Cycle(easy)

Easy

1410139FavoriteShare

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

To represent a cycle in the given linked list, we use an integer pos which represents the position (0-indexed) in the linked list where tail connects to. If pos is -1, then there is no cycle in the linked list.

**Example 1:**

**Input:** head = [3,2,0,-4], pos = 1

**Output:** true

**Explanation:** There is a cycle in the linked list, where tail connects to the second node.



**Example 2:**

**Input:** head = [1,2], pos = 0

**Output:** true

**Explanation:** There is a cycle in the linked list, where tail connects to the first node.



**Example 3:**

**Input:** head = [1], pos = -1

**Output:** false

**Explanation:** There is no cycle in the linked list.



**Follow up:**

Can you solve it using *O(1)* (i.e. constant) memory?

# 155. Min Stack(easy)

Easy

1647172FavoriteShare

Design a stack that supports push, pop, top, and retrieving the minimum element in constant time.

* push(x) -- Push element x onto stack.
* pop() -- Removes the element on top of the stack.
* top() -- Get the top element.
* getMin() -- Retrieve the minimum element in the stack.

**Example:**

MinStack minStack = new MinStack();

minStack.push(-2);

minStack.push(0);

minStack.push(-3);

minStack.getMin(); --> Returns -3.

minStack.pop();

minStack.top(); --> Returns 0.

minStack.getMin(); --> Returns -2.