Array - easy

https://leetcode.com/problems/two-sum/description/

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

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

You can return the answer in any order.

Example 1:

Input: nums = [2,7,11,15], target = 9

Output: [0,1]

Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Example 2:

Input: nums = [3,2,4], target = 6

Output: [1,2]

Example 3:

Input: nums = [3,3], target = 6

Output: [0,1]

Constraints:

2 <= nums.length <= 104

-109 <= nums[i] <= 109

-109 <= target <= 109

Only one valid answer exists.

Array - medium

https://leetcode.com/problems/3sum/description/

Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0.

Notice that the solution set must not contain duplicate triplets.

Example 1:

Input: nums = [-1,0,1,2,-1,-4]

Output: [[-1,-1,2],[-1,0,1]]

Explanation:

nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.

nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.

nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.

The distinct triplets are [-1,0,1] and [-1,-1,2].

Notice that the order of the output and the order of the triplets does not matter.

Example 2:

Input: nums = [0,1,1]

Output: []

Explanation: The only possible triplet does not sum up to 0.

Example 3:

Input: nums = [0,0,0]

Output: [[0,0,0]]

Explanation: The only possible triplet sums up to 0.

Constraints:

3 <= nums.length <= 3000

-105 <= nums[i] <= 105

Array - hard

https://leetcode.com/problems/median-of-two-sorted-arrays/description/

Given two sorted arrays nums1 and nums2 of size m and n respectively, return the median of the two sorted arrays.

The overall run time complexity should be O(log (m+n)).

Example 1:

Input: nums1 = [1,3], nums2 = [2]

Output: 2.00000

Explanation: merged array = [1,2,3] and median is 2.

Example 2:

Input: nums1 = [1,2], nums2 = [3,4]

Output: 2.50000

Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.

Constraints:

nums1.length == m

nums2.length == n

0 <= m <= 1000

0 <= n <= 1000

1 <= m + n <= 2000

-106 <= nums1[i], nums2[i] <= 106

String - easy

https://leetcode.com/problems/roman-to-integer/description/

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, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 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.

Example 1:

Input: s = "III"

Output: 3

Explanation: III = 3.

Example 2:

Input: s = "LVIII"

Output: 58

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

Example 3:

Input: s = "MCMXCIV"

Output: 1994

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

Constraints:

1 <= s.length <= 15

s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M').

It is guaranteed that s is a valid roman numeral in the range [1, 3999].

String - medium

https://leetcode.com/problems/longest-substring-without-repeating-characters/description/

Given a string s, find the length of the longest substring

without repeating characters.

Example 1:

Input: s = "abcabcbb"

Output: 3

Explanation: The answer is "abc", with the length of 3.

Example 2:

Input: s = "bbbbb"

Output: 1

Explanation: The answer is "b", with the length of 1.

Example 3:

Input: s = "pwwkew"

Output: 3

Explanation: The answer is "wke", with the length of 3.

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

Constraints:

0 <= s.length <= 5 * 104

s consists of English letters, digits, symbols and spaces.

String - hard

https://leetcode.com/problems/regular-expression-matching/description/

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '*' where:

'.' Matches any single character.

'*' Matches zero or more of the preceding element.

The matching should cover the entire input string (not partial).

Example 1:

Input: s = "aa", p = "a"

Output: false

Explanation: "a" does not match the entire string "aa".

Example 2:

Input: s = "aa", p = "a*"

Output: true

Explanation: '*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".

Example 3:

Input: s = "ab", p = ".*"

Output: true

Explanation: ".*" means "zero or more (*) of any character (.)".

Constraints:

1 <= s.length <= 20

1 <= p.length <= 20

s contains only lowercase English letters.

p contains only lowercase English letters, '.', and '*'.

It is guaranteed for each appearance of the character '*', there will be a previous valid character to match.

Hash table - easy

https://leetcode.com/problems/two-sum/description/

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to target.

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

You can return the answer in any order.

Example 1:

Input: nums = [2,7,11,15], target = 9

Output: [0,1]

Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

Example 2:

Input: nums = [3,2,4], target = 6

Output: [1,2]

Example 3:

Input: nums = [3,3], target = 6

Output: [0,1]

Constraints:

2 <= nums.length <= 104

-109 <= nums[i] <= 109

-109 <= target <= 109

Only one valid answer exists.

Hash table - medium

https://leetcode.com/problems/longest-substring-without-repeating-characters/description/

Given a string s, find the length of the longest

substring

without repeating characters.

Example 1:

Input: s = "abcabcbb"

Output: 3

Explanation: The answer is "abc", with the length of 3.

Example 2:

Input: s = "bbbbb"

Output: 1

Explanation: The answer is "b", with the length of 1.

Example 3:

Input: s = "pwwkew"

Output: 3

Explanation: The answer is "wke", with the length of 3.

Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

Constraints:

0 <= s.length <= 5 * 104

s consists of English letters, digits, symbols and spaces.

Hash table - hard

https://leetcode.com/problems/substring-with-concatenation-of-all-words/description/

You are given a string s and an array of strings words. All the strings of words are of the same length.

A concatenated string is a string that exactly contains all the strings of any permutation of words concatenated.

For example, if words = ["ab","cd","ef"], then "abcdef", "abefcd", "cdabef", "cdefab", "efabcd", and "efcdab" are all concatenated strings. "acdbef" is not a concatenated string because it is not the concatenation of any permutation of words.

Return an array of the starting indices of all the concatenated substrings in s. You can return the answer in any order.

Example 1:

Input: s = "barfoothefoobarman", words = ["foo","bar"]

Output: [0,9]

Explanation:

The substring starting at 0 is "barfoo". It is the concatenation of ["bar","foo"] which is a permutation of words.

The substring starting at 9 is "foobar". It is the concatenation of ["foo","bar"] which is a permutation of words.

Example 2:

Input: s = "wordgoodgoodgoodbestword", words = ["word","good","best","word"]

Output: []

Explanation:

There is no concatenated substring.

Example 3:

Input: s = "barfoofoobarthefoobarman", words = ["bar","foo","the"]

Output: [6,9,12]

Explanation:

The substring starting at 6 is "foobarthe". It is the concatenation of ["foo","bar","the"].

The substring starting at 9 is "barthefoo". It is the concatenation of ["bar","the","foo"].

The substring starting at 12 is "thefoobar". It is the concatenation of ["the","foo","bar"].

Constraints:

1 <= s.length <= 104

1 <= words.length <= 5000

1 <= words[i].length <= 30

s and words[i] consist of lowercase English letters.

Math - easy

https://leetcode.com/problems/palindrome-number/description/

Given an integer x, return true if x is a

palindrome

, and false otherwise.

Example 1:

Input: x = 121

Output: true

Explanation: 121 reads as 121 from left to right and from right to left.

Example 2:

Input: x = -121

Output: false

Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.

Example 3:

Input: x = 10

Output: false

Explanation: Reads 01 from right to left. Therefore it is not a palindrome.

Constraints:

-231 <= x <= 231 - 1

Math - medium

https://leetcode.com/problems/reverse-integer/description/

Given a signed 32-bit integer x, return x with its digits reversed. If reversing x causes the value to go outside the signed 32-bit integer range [-231, 231 - 1], then return 0.

Assume the environment does not allow you to store 64-bit integers (signed or unsigned).

Example 1:

Input: x = 123

Output: 321

Example 2:

Input: x = -123

Output: -321

Example 3:

Input: x = 120

Output: 21

Constraints:

-231 <= x <= 231 – 1

Math - hard

https://leetcode.com/problems/permutation-sequence/description/

The set [1, 2, 3, ..., n] contains a total of n! unique permutations.

By listing and labeling all of the permutations in order, we get the following sequence for n = 3:

"123"

"132"

"213"

"231"

"312"

"321"

Given n and k, return the kth permutation sequence.

Example 1:

Input: n = 3, k = 3

Output: "213"

Example 2:

Input: n = 4, k = 9

Output: "2314"

Example 3:

Input: n = 3, k = 1

Output: "123"

Constraints:

1 <= n <= 9

1 <= k <= n!

Dynamic programming - easy

https://leetcode.com/problems/climbing-stairs/description/

You are climbing a staircase. It takes n steps to reach the top.

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

Example 1:

Input: n = 2

Output: 2

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

1. 1 step + 1 step

2. 2 steps

Example 2:

Input: n = 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

Constraints:

1 <= n <= 45

Dynamic programming - medium

https://leetcode.com/problems/longest-palindromic-substring/description/

Given a string s, return the longest

palindromic

substring

in s.

Example 1:

Input: s = "babad"

Output: "bab"

Explanation: "aba" is also a valid answer.

Example 2:

Input: s = "cbbd"

Output: "bb"

Constraints:

1 <= s.length <= 1000

s consist of only digits and English letters.

Dynamic programming - hard

https://leetcode.com/problems/regular-expression-matching/description/

Given an input string s and a pattern p, implement regular expression matching with support for '.' and '*' where:

'.' Matches any single character.

'*' Matches zero or more of the preceding element.

The matching should cover the entire input string (not partial).

Example 1:

Input: s = "aa", p = "a"

Output: false

Explanation: "a" does not match the entire string "aa".

Example 2:

Input: s = "aa", p = "a*"

Output: true

Explanation: '*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".

Example 3:

Input: s = "ab", p = ".*"

Output: true

Explanation: ".*" means "zero or more (*) of any character (.)".

Constraints:

1 <= s.length <= 20

1 <= p.length <= 20

s contains only lowercase English letters.

p contains only lowercase English letters, '.', and '*'.

It is guaranteed for each appearance of the character '*', there will be a previous valid character to match.