TwoSum

-------------------------------------------------

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].

-----------------------------------------------------------------------

Longest Substring Without Repeating Characters

--------------------------------------------------

Given a string, find the length of the longest substring without repeating characters.

Example 1:

Input: "abcabcbb"

Output: 3

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

Example 2:

Input: "bbbbb"

Output: 1

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

Example 3:

Input: "pwwkew"

Output: 3

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

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

-----------------------------------------------------------------------

String to Integer (atoi)

---------------------------------------------------------------------

Implement atoi which converts a string to an integer.

The function first discards as many whitespace characters as necessary until the first non-whitespace character is found. Then, starting from this character, takes an optional initial plus or minus sign followed by as many numerical digits as possible, and interprets them as a numerical value.

The string can contain additional characters after those that form the integral number, which are ignored and have no effect on the behavior of this function.

If the first sequence of non-whitespace characters in str is not a valid integral number, or if no such sequence exists because either str is empty or it contains only whitespace characters, no conversion is performed.

If no valid conversion could be performed, a zero value is returned.

Note:

Only the space character ' ' is considered as whitespace character.

Assume we are dealing with an environment which could only store integers within the 32-bit signed integer range: [−231, 231 − 1]. If the numerical value is out of the range of representable values, INT\_MAX (231 − 1) or INT\_MIN (−231) is returned.

Example 1:

Input: "42"

Output: 42

Example 2:

Input: " -42"

Output: -42

Explanation: The first non-whitespace character is '-', which is the minus sign.

Then take as many numerical digits as possible, which gets 42.

Example 3:

Input: "4193 with words"

Output: 4193

Explanation: Conversion stops at digit '3' as the next character is not a numerical digit.

Example 4:

Input: "words and 987"

Output: 0

Explanation: The first non-whitespace character is 'w', which is not a numerical

digit or a +/- sign. Therefore no valid conversion could be performed.

Example 5:

Input: "-91283472332"

Output: -2147483648

Explanation: The number "-91283472332" is out of the range of a 32-bit signed integer.

Thefore INT\_MIN (−231) is returned.

-----------------------------------------------------------------------

Container With Most Water

-----------------------------------------------------------------------

Given n non-negative integers a1, a2, ..., an , where each represents a point at coordinate (i, ai). n vertical lines are drawn such that the two endpoints of line i is at (i, ai) and (i, 0). Find two lines, which together with x-axis forms a container, such that the container contains the most water.

Note: You may not slant the container and n is at least 2.

The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

Example:

Input: [1,8,6,2,5,4,8,3,7]

Output: 49

-----------------------------------------------------------------------

Integer to Roman

-----------------------------------------------------------------------

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

Example 1:

Input: 3

Output: "III"

Example 2:

Input: 4

Output: "IV"

Example 3:

Input: 9

Output: "IX"

Example 4:

Input: 58

Output: "LVIII"

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

Example 5:

Input: 1994

Output: "MCMXCIV"

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

-----------------------------------------------------------------------

Roman to Integer

-----------------------------------------------------------------------

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.

-----------------------------------------------------------------------

3Sum

-----------------------------------------------------------------------

Given an array nums of n integers, are there elements a, b, c in nums such that a + b + c = 0? Find all unique triplets in the array which gives the sum of zero.

Note:

The solution set must not contain duplicate triplets.

Example:

Given array nums = [-1, 0, 1, 2, -1, -4],

A solution set is:

[

[-1, 0, 1],

[-1, -1, 2]

]

Hints:

So, we essentially need to find three numbers x, y, and z such that they add up to the given value. If we fix one of the numbers say x, we are left with the two-sum problem at hand!

For the two-sum problem, if we fix one of the numbers, say

x

, we have to scan the entire array to find the next number

y

which is

value - x

where value is the input parameter. Can we change our array somehow so that this search becomes faster?

The second train of thought for two-sum is, without changing the array, can we use additional space somehow? Like maybe a hash map to speed up the search?

-----------------------------------------------------------------------

3Sum Closest

-----------------------------------------------------------------------

Given an array nums of n integers and an integer target, find three integers in nums such that the sum is closest to target. Return the sum of the three integers. You may assume that each input would have exactly one solution.

Example:

Given array nums = [-1, 2, 1, -4], and target = 1.

The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).

-----------------------------------------------------------------------

Implement strStr()

-----------------------------------------------------------------------

Implement strStr().

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() and Java's indexOf().

-----------------------------------------------------------------------

Rotate Image

-----------------------------------------------------------------------

You are given an n x n 2D matrix representing an image.

Rotate the image by 90 degrees (clockwise).

Note:

You have to rotate the image in-place, which means you have to modify the input 2D matrix directly. DO NOT allocate another 2D matrix and do the rotation.

Example 1:

Given input matrix =

[

[1,2,3],

[4,5,6],

[7,8,9]

],

rotate the input matrix in-place such that it becomes:

[

[7,4,1],

[8,5,2],

[9,6,3]

]

Example 2:

Given input matrix =

[

[ 5, 1, 9,11],

[ 2, 4, 8,10],

[13, 3, 6, 7],

[15,14,12,16]

],

rotate the input matrix in-place such that it becomes:

[

[15,13, 2, 5],

[14, 3, 4, 1],

[12, 6, 8, 9],

[16, 7,10,11]

]

-----------------------------------------------------------------------

Group Anagrams

-----------------------------------------------------------------------

Given an array of strings, group anagrams together.

Example:

Input: ["eat", "tea", "tan", "ate", "nat", "bat"],

Output:

[

["ate","eat","tea"],

["nat","tan"],

["bat"]

]

Note:

All inputs will be in lowercase.

The order of your output does not matter.

-----------------------------------------------------------------------

Minimum Window Substring

-----------------------------------------------------------------------

Given a string S and a string T, find the minimum window in S which will contain all the characters in T in complexity O(n).

Example:

Input: S = "ADOBECODEBANC", T = "ABC"

Output: "BANC"

Note:

If there is no such window in S that covers all characters in T, return the empty string "".

If there is such window, you are guaranteed that there will always be only one unique minimum window in S.

Hide Hint #1

Use two pointers to create a window of letters in S, which would have all the characters from T.

Hide Hint #2

Since you have to find the minimum window in S which has all the characters from T, you need to expand and contract the window using the two pointers and keep checking the window for all the characters. This approach is also called Sliding Window Approach.

L ------------------------ R , Suppose this is the window that contains all characters of T

L----------------- R , this is the contracted window. We found a smaller window that still contains all the characters in T

When the window is no longer valid, start expanding again using the right pointer.

-----------------------------------------------------------------------

Compare Version Numbers

-----------------------------------------------------------------------

Compare two version numbers version1 and version2.

If version1 > version2 return 1; if version1 < version2 return -1;otherwise return 0.

You may assume that the version strings are non-empty and contain only digits and the . character.

The . character does not represent a decimal point and is used to separate number sequences.

For instance, 2.5 is not "two and a half" or "half way to version three", it is the fifth second-level revision of the second first-level revision.

You may assume the default revision number for each level of a version number to be 0. For example, version number 3.4 has a revision number of 3 and 4 for its first and second level revision number. Its third and fourth level revision number are both 0.

Example 1:

Input: version1 = "0.1", version2 = "1.1"

Output: -1

Example 2:

Input: version1 = "1.0.1", version2 = "1"

Output: 1

Example 3:

Input: version1 = "7.5.2.4", version2 = "7.5.3"

Output: -1

Example 4:

Input: version1 = "1.01", version2 = "1.001"

Output: 0

Explanation: Ignoring leading zeroes, both “01” and “001" represent the same number “1”

Example 5:

Input: version1 = "1.0", version2 = "1.0.0"

Output: 0

Explanation: The first version number does not have a third level revision number, which means its third level revision number is default to "0"

Note:

Version strings are composed of numeric strings separated by dots . and this numeric strings may have leading zeroes.

Version strings do not start or end with dots, and they will not be two consecutive dots.

-----------------------------------------------------------------------

Product of Array Except Self

-----------------------------------------------------------------------

Given an array nums of n integers where n > 1, return an array output such that output[i] is equal to the product of all the elements of nums except nums[i].

Example:

Input: [1,2,3,4]

Output: [24,12,8,6]

Note: Please solve it without division and in O(n).

Follow up:

Could you solve it with constant space complexity? (The output array does not count as extra space for the purpose of space complexity analysis.)

-----------------------------------------------------------------------

Missing Number

-----------------------------------------------------------------------

Given an array containing n distinct numbers taken from 0, 1, 2, ..., n, find the one that is missing from the array.

Example 1:

Input: [3,0,1]

Output: 2

Example 2:

Input: [9,6,4,2,3,5,7,0,1]

Output: 8

Note:

Your algorithm should run in linear runtime complexity. Could you implement it using only constant extra space complexity?

-----------------------------------------------------------------------

Integer to English Words

-----------------------------------------------------------------------

Convert a non-negative integer to its english words representation. Given input is guaranteed to be less than 231 - 1.

Example 1:

Input: 123

Output: "One Hundred Twenty Three"

Example 2:

Input: 12345

Output: "Twelve Thousand Three Hundred Forty Five"

Example 3:

Input: 1234567

Output: "One Million Two Hundred Thirty Four Thousand Five Hundred Sixty Seven"

Example 4:

Input: 1234567891

Output: "One Billion Two Hundred Thirty Four Million Five Hundred Sixty Seven Thousand Eight Hundred Ninety One"

Hide Hint #1

Did you see a pattern in dividing the number into chunk of words? For example, 123 and 123000.

Hide Hint #2

Group the number by thousands (3 digits). You can write a helper function that takes a number less than 1000 and convert just that chunk to words.

Hide Hint #3

There are many edge cases. What are some good test cases? Does your code work with input such as 0? Or 1000010? (middle chunk is zero and should not be printed out)

-----------------------------------------------------------------------

First Unique Character in a String

-----------------------------------------------------------------------

Given a string, find the first non-repeating character in it and return it's index. If it doesn't exist, return -1.

Examples:

s = "leetcode"

return 0.

s = "loveleetcode",

return 2.

Note: You may assume the string contain only lowercase letters.

-----------------------------------------------------------------------

Valid Parentheses

-----------------------------------------------------------------------

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

An input string is valid if:

Open brackets must be closed by the same type of brackets.

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

Hide Hint #1

An interesting property about a valid parenthesis expression is that a sub-expression of a valid expression should also be a valid expression. (Not every sub-expression) e.g.

{ { } [ ] [ [ [ ] ] ] } is VALID expression

[ [ [ ] ] ] is VALID sub-expression

{ } [ ] is VALID sub-expression

Can we exploit this recursive structure somehow?

Hide Hint #2

What if whenever we encounter a matching pair of parenthesis in the expression, we simply remove it from the expression? This would keep on shortening the expression. e.g.

{ { ( { } ) } }

|\_|

{ { ( ) } }

|\_\_\_\_\_\_|

{ { } }

|\_\_\_\_\_\_\_\_\_\_|

{ }

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

VALID EXPRESSION!

Hide Hint #3

The stack data structure can come in handy here in representing this recursive structure of the problem. We can't really process this from the inside out because we don't have an idea about the overall structure. But, the stack can help us process this recursively i.e. from outside to inwards.

-----------------------------------------------------------------------

Most Common Word

-----------------------------------------------------------------------

Given a paragraph and a list of banned words, return the most frequent word that is not in the list of banned words. It is guaranteed there is at least one word that isn't banned, and that the answer is unique.

Words in the list of banned words are given in lowercase, and free of punctuation. Words in the paragraph are not case sensitive. The answer is in lowercase.

Example:

Input:

paragraph = "Bob hit a ball, the hit BALL flew far after it was hit."

banned = ["hit"]

Output: "ball"

Explanation:

"hit" occurs 3 times, but it is a banned word.

"ball" occurs twice (and no other word does), so it is the most frequent non-banned word in the paragraph.

Note that words in the paragraph are not case sensitive,

that punctuation is ignored (even if adjacent to words, such as "ball,"),

and that "hit" isn't the answer even though it occurs more because it is banned.

Note:

1 <= paragraph.length <= 1000.

0 <= banned.length <= 100.

1 <= banned[i].length <= 10.

The answer is unique, and written in lowercase (even if its occurrences in paragraph may have uppercase symbols, and even if it is a proper noun.)

paragraph only consists of letters, spaces, or the punctuation symbols !?',;.

There are no hyphens or hyphenated words.

Words only consist of letters, never apostrophes or other punctuation symbols.

-----------------------------------------------------------------------

Reorder Log Files

-----------------------------------------------------------------------

You have an array of logs. Each log is a space delimited string of words.

For each log, the first word in each log is an alphanumeric identifier. Then, either:

Each word after the identifier will consist only of lowercase letters, or;

Each word after the identifier will consist only of digits.

We will call these two varieties of logs letter-logs and digit-logs. It is guaranteed that each log has at least one word after its identifier.

Reorder the logs so that all of the letter-logs come before any digit-log. The letter-logs are ordered lexicographically ignoring identifier, with the identifier used in case of ties. The digit-logs should be put in their original order.

Return the final order of the logs.

Example 1:

Input: logs = ["dig1 8 1 5 1","let1 art can","dig2 3 6","let2 own kit dig","let3 art zero"]

Output: ["let1 art can","let3 art zero","let2 own kit dig","dig1 8 1 5 1","dig2 3 6"]

Constraints:

0 <= logs.length <= 100

3 <= logs[i].length <= 100

logs[i] is guaranteed to have an identifier, and a word after the identifier.

-----------------------------------------------------------------------

Trapping Rain Water

-----------------------------------------------------------------------

Given *n* non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

  
The above elevation map is represented by array [0,1,0,2,1,0,1,3,2,1,2,1]. In this case, 6 units of rain water (blue section) are being trapped. **Thanks Marcos** for contributing this image!

**Example:**

**Input:** [0,1,0,2,1,0,1,3,2,1,2,1]

**Output:** 6

-----------------------------------------------------------------------

Linked List

-----------------------------------------------------------------------

Add Two Numbers

-----------------------------------------------------------------------

You are given two **non-empty** linked lists representing two non-negative integers. The digits are stored in **reverse order** and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

**Example:**

**Input:** (2 -> 4 -> 3) + (5 -> 6 -> 4)

**Output:** 7 -> 0 -> 8

**Explanation:** 342 + 465 = 807.

-----------------------------------------------------------------------

Merge Two Sorted Lists

-----------------------------------------------------------------------

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

-----------------------------------------------------------------------

Reverse Nodes in k-Group

-----------------------------------------------------------------------

Given a linked list, reverse the nodes of a linked list *k* at a time and return its modified list.

*k* is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of *k* then left-out nodes in the end should remain as it is.

**Example:**

Given this linked list: 1->2->3->4->5

For *k* = 2, you should return: 2->1->4->3->5

For *k* = 3, you should return: 3->2->1->4->5

**Note:**

* Only constant extra memory is allowed.
* You may not alter the values in the list's nodes, only nodes itself may be changed.

-----------------------------------------------------------------------

Copy List with Random Pointer

-----------------------------------------------------------------------

A linked list is given such that each node contains an additional random pointer which could point to any node in the list or null.

Return a [**deep copy**](https://en.wikipedia.org/wiki/Object_copying#Deep_copy) of the list.

The Linked List is represented in the input/output as a list of n nodes. Each node is represented as a pair of [val, random\_index] where:

* val: an integer representing Node.val
* random\_index: the index of the node (range from 0 to n-1) where random pointer points to, or null if it does not point to any node.

**Example 1:**



**Input:** head = [[7,null],[13,0],[11,4],[10,2],[1,0]]

**Output:** [[7,null],[13,0],[11,4],[10,2],[1,0]]

**Example 2:**



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

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

**Example 3:**

****

**Input:** head = [[3,null],[3,0],[3,null]]

**Output:** [[3,null],[3,0],[3,null]]

**Example 4:**

**Input:** head = []

**Output:** []

**Explanation:** Given linked list is empty (null pointer), so return null.

**Constraints:**

* -10000 <= Node.val <= 10000
* Node.random is null or pointing to a node in the linked list.
* Number of Nodes will not exceed 1000.

   Hide Hint #1

Just iterate the linked list and create copies of the nodes on the go. Since a node can be referenced from multiple nodes due to the random pointers, make sure you are not making multiple copies of the same node.

   Hide Hint #2

You may want to use extra space to keep **old node ---> new node** mapping to prevent creating multiples copies of same node.

   Hide Hint #3

We can avoid using extra space for old node ---> new node mapping, by tweaking the original linked list. Simply interweave the nodes of the old and copied list. For e.g.

Old List: A --> B --> C --> D

InterWeaved List: A --> A' --> B --> B' --> C --> C' --> D --> D'

   Hide Hint #4

The interweaving is done using **next** pointers and we can make use of interweaved structure to get the correct reference nodes for **random** pointers.

-----------------------------------------------------------------------

Reverse Linked List

-----------------------------------------------------------------------

Reverse a singly linked list.

**Example:**

**Input:** 1->2->3->4->5->NULL

**Output:** 5->4->3->2->1->NULL

**Follow up:**

A linked list can be reversed either iteratively or recursively. Could you implement both?

-----------------------------------------------------------------------

Merge k Sorted Lists

-----------------------------------------------------------------------

Merge *k* sorted linked lists and return it as one sorted list. Analyze and describe its complexity.

**Example:**

**Input:**

[

  1->4->5,

  1->3->4,

  2->6

]

**Output:** 1->1->2->3->4->4->5->6

-----------------------------------------------------------------------

Trees and Graphs

-----------------------------------------------------------------------

Validate Binary Search Tree

-----------------------------------------------------------------------

Given a binary tree, determine if it is a valid binary search tree (BST).

Assume a BST is defined as follows:

* The left subtree of a node contains only nodes with keys **less than** the node's key.
* The right subtree of a node contains only nodes with keys **greater than** the node's key.
* Both the left and right subtrees must also be binary search trees.

**Example 1:**

2

/ \

1 3

**Input:** [2,1,3]

**Output:** true

**Example 2:**

5

/ \

1 4

  / \

  3 6

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

**Output:** false

**Explanation:** The root node's value is 5 but its right child's value is 4.

-----------------------------------------------------------------------

Symmetric Tree

-----------------------------------------------------------------------

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.

-----------------------------------------------------------------------

Binary Tree Level Order Traversal

-----------------------------------------------------------------------

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

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

3

/ \

9 20

/ \

15 7

return its level order traversal as:

[

[3],

[9,20],

[15,7]

]

-----------------------------------------------------------------------

Binary Tree Zigzag Level Order Traversal

-----------------------------------------------------------------------

Given a binary tree, return the *zigzag level order* traversal of its nodes' values. (ie, from left to right, then right to left for the next level and alternate between).

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

3

/ \

9 20

/ \

15 7

return its zigzag level order traversal as:

[

[3],

[20,9],

[15,7]

]

-----------------------------------------------------------------------

Binary Tree Maximum Path Sum

-----------------------------------------------------------------------

Given a **non-empty** binary tree, find the maximum path sum.

For this problem, a path is defined as any sequence of nodes from some starting node to any node in the tree along the parent-child connections. The path must contain **at least one node** and does not need to go through the root.

**Example 1:**

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

**1**

**/ \**

**2** **3**

**Output:** 6

**Example 2:**

**Input:** [-10,9,20,null,null,15,7]

  -10

   / \

  9  **20**

**/  \**

**15   7**

**Output:** 42

-----------------------------------------------------------------------

Word Ladder II

-----------------------------------------------------------------------

Given two words (*beginWord* and *endWord*), and a dictionary's word list, find all shortest transformation sequence(s) from *beginWord* to *endWord*, such that:

1. Only one letter can be changed at a time
2. Each transformed word must exist in the word list. Note that *beginWord* is *not* a transformed word.

**Note:**

* Return an empty list if there is no such transformation sequence.
* All words have the same length.
* All words contain only lowercase alphabetic characters.
* You may assume no duplicates in the word list.
* You may assume *beginWord* and *endWord* are non-empty and are not the same.

**Example 1:**

**Input:**

beginWord = "hit",

endWord = "cog",

wordList = ["hot","dot","dog","lot","log","cog"]

**Output:**

[

["hit","hot","dot","dog","cog"],

  ["hit","hot","lot","log","cog"]

]

**Example 2:**

**Input:**

beginWord = "hit"

endWord = "cog"

wordList = ["hot","dot","dog","lot","log"]

**Output:** []

**Explanation:** The endWord "cog" is not in wordList, therefore no possibletransformation.

-----------------------------------------------------------------------

Word Ladder

-----------------------------------------------------------------------

Given two words (*beginWord* and *endWord*), and a dictionary's word list, find the length of shortest transformation sequence from *beginWord* to *endWord*, such that:

1. Only one letter can be changed at a time.
2. Each transformed word must exist in the word list. Note that *beginWord* is *not* a transformed word.

**Note:**

* Return 0 if there is no such transformation sequence.
* All words have the same length.
* All words contain only lowercase alphabetic characters.
* You may assume no duplicates in the word list.
* You may assume *beginWord* and *endWord* are non-empty and are not the same.

**Example 1:**

**Input:**

beginWord = "hit",

endWord = "cog",

wordList = ["hot","dot","dog","lot","log","cog"]

**Output:** 5

**Explanation:** As one shortest transformation is "hit" -> "hot" -> "dot" -> "dog" -> "cog",

return its length 5.

**Example 2:**

**Input:**

beginWord = "hit"

endWord = "cog"

wordList = ["hot","dot","dog","lot","log"]

**Output:** 0

**Explanation:** The endWord "cog" is not in wordList, therefore no possibletransformation.

-----------------------------------------------------------------------

Number of Islands

-----------------------------------------------------------------------

Given a 2d grid map of '1's (land) and '0's (water), count the number of islands. An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

**Example 1:**

**Input:**

11110

11010

11000

00000

**Output:** 1

**Example 2:**

**Input:**

11000

11000

00100

00011

**Output:** 3

-----------------------------------------------------------------------

Course Schedule

-----------------------------------------------------------------------

There are a total of *n* courses you have to take, labeled from 0 to n-1.

Some courses may have prerequisites, for example to take course 0 you have to first take course 1, which is expressed as a pair: [0,1]

Given the total number of courses and a list of prerequisite **pairs**, is it possible for you to finish all courses?

**Example 1:**

**Input:** 2, [[1,0]]

**Output:** true

**Explanation:** There are a total of 2 courses to take.

  To take course 1 you should have finished course 0. So it is possible.

**Example 2:**

**Input:** 2, [[1,0],[0,1]]

**Output:** false

**Explanation:** There are a total of 2 courses to take.

  To take course 1 you should have finished course 0, and to take course 0 you should

  also have finished course 1. So it is impossible.

**Note:**

1. The input prerequisites is a graph represented by **a list of edges**, not adjacency matrices. Read more about [how a graph is represented](https://www.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs).
2. You may assume that there are no duplicate edges in the input prerequisites.

   Hide Hint #1

This problem is equivalent to finding if a cycle exists in a directed graph. If a cycle exists, no topological ordering exists and therefore it will be impossible to take all courses.

   Hide Hint #2

[Topological Sort via DFS](https://class.coursera.org/algo-003/lecture/52) - A great video tutorial (21 minutes) on Coursera explaining the basic concepts of Topological Sort.

   Hide Hint #3

Topological sort could also be done via [BFS](http://en.wikipedia.org/wiki/Topological_sorting#Algorithms).

-----------------------------------------------------------------------

Lowest Common Ancestor of a Binary Tree

-----------------------------------------------------------------------

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.

According to the [definition of LCA on Wikipedia](https://en.wikipedia.org/wiki/Lowest_common_ancestor): “The lowest common ancestor is defined between two nodes p and q as the lowest node in T that has both p and q as descendants (where we allow **a node to be a descendant of itself**).”

Given the following binary tree:  root = [3,5,1,6,2,0,8,null,null,7,4]



**Example 1:**

**Input:** root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 1

**Output:** 3

**Explanation:** The LCA of nodes 5 and 1 is 3.

**Example 2:**

**Input:** root = [3,5,1,6,2,0,8,null,null,7,4], p = 5, q = 4

**Output:** 5

**Explanation:** The LCA of nodes 5 and 4 is 5, since a node can be a descendant of itself according to the LCA definition.

**Note:**

* All of the nodes' values will be unique.
* p and q are different and both values will exist in the binary tree.

-----------------------------------------------------------------------

Diameter of Binary Tree

-----------------------------------------------------------------------

Given a binary tree, you need to compute the length of the diameter of the tree. The diameter of a binary tree is the length of the **longest** path between any two nodes in a tree. This path may or may not pass through the root.

**Example:**  
Given a binary tree

1

/ \

2 3

/ \

4 5

Return **3**, which is the length of the path [4,2,1,3] or [5,2,1,3].

**Note:** The length of path between two nodes is represented by the number of edges between them.

-----------------------------------------------------------------------

Cut Off Trees for Golf Event

-----------------------------------------------------------------------

You are asked to cut off trees in a forest for a golf event. The forest is represented as a non-negative 2D map, in this map:

1. 0 represents the obstacle can't be reached.
2. 1 represents the ground can be walked through.
3. The place with number bigger than 1 represents a tree can be walked through, and this positive number represents the tree's height.

You are asked to cut off **all** the trees in this forest in the order of tree's height - always cut off the tree with lowest height first. And after cutting, the original place has the tree will become a grass (value 1).

You will start from the point (0, 0) and you should output the minimum steps **you need to walk** to cut off all the trees. If you can't cut off all the trees, output -1 in that situation.

You are guaranteed that no two trees have the same height and there is at least one tree needs to be cut off.

**Example 1:**

**Input:**

[

[1,2,3],

[0,0,4],

[7,6,5]

]

**Output:** 6

**Example 2:**

**Input:**

[

[1,2,3],

[0,0,0],

[7,6,5]

]

**Output:** -1

**Example 3:**

**Input:**

[

[2,3,4],

[0,0,5],

[8,7,6]

]

**Output:** 6

**Explanation:** You started from the point (0,0) and you can cut off the tree in (0,0) directly without walking.

**Hint**: size of the given matrix will not exceed 50x50.

-----------------------------------------------------------------------

Flood Fill

-----------------------------------------------------------------------

An image is represented by a 2-D array of integers, each integer representing the pixel value of the image (from 0 to 65535).

Given a coordinate (sr, sc) representing the starting pixel (row and column) of the flood fill, and a pixel value newColor, "flood fill" the image.

To perform a "flood fill", consider the starting pixel, plus any pixels connected 4-directionally to the starting pixel of the same color as the starting pixel, plus any pixels connected 4-directionally to those pixels (also with the same color as the starting pixel), and so on. Replace the color of all of the aforementioned pixels with the newColor.

At the end, return the modified image.

**Example 1:**

**Input:**

image = [[1,1,1],[1,1,0],[1,0,1]]

sr = 1, sc = 1, newColor = 2

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

**Explanation:**

From the center of the image (with position (sr, sc) = (1, 1)), all pixels connected

by a path of the same color as the starting pixel are colored with the new color.

Note the bottom corner is not colored 2, because it is not 4-directionally connected

to the starting pixel.

**Note:**

 The length of image and image[0] will be in the range [1, 50].

 The given starting pixel will satisfy 0 <= sr < image.length and 0 <= sc < image[0].length.

 The value of each color in image[i][j] and newColor will be an integer in [0, 65535].

RECURSION

-----------------------------------------------------------------------

Letter Combinations of a Phone Number

-----------------------------------------------------------------------

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent.

A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters.



**Example:**

**Input:** "23"

**Output:** ["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"].

**Note:**

Although the above answer is in lexicographical order, your answer could be in any order you want.

-----------------------------------------------------------------------

Generate Parentheses

-----------------------------------------------------------------------

Given *n* pairs of parentheses, write a function to generate all combinations of well-formed parentheses.

For example, given *n* = 3, a solution set is:

[

"((()))",

"(()())",

"(())()",

"()(())",

"()()()"

]

-----------------------------------------------------------------------

Word Search

-----------------------------------------------------------------------

Given a 2D board and a word, find if the word exists in the grid.

The word can be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The same letter cell may not be used more than once.

**Example:**

board =

[

['A','B','C','E'],

['S','F','C','S'],

['A','D','E','E']

]

Given word = "**ABCCED**", return **true**.

Given word = "**SEE**", return **true**.

Given word = "**ABCB**", return **false**.

-----------------------------------------------------------------------

Word Search II

-----------------------------------------------------------------------

Given a 2D board and a list of words from the dictionary, find all words in the board.

Each word must be constructed from letters of sequentially adjacent cell, where "adjacent" cells are those horizontally or vertically neighboring. The same letter cell may not be used more than once in a word.

**Example:**

**Input:**

**board** = [

['o','a','a','n'],

['e','t','a','e'],

['i','h','k','r'],

['i','f','l','v']

]

**words** = ["oath","pea","eat","rain"]

**Output:**["eat","oath"]

**Note:**

1. All inputs are consist of lowercase letters a-z.
2. The values of words are distinct.

   Hide Hint #1

You would need to optimize your backtracking to pass the larger test. Could you stop backtracking earlier?

   Hide Hint #2

If the current candidate does not exist in all words' prefix, you could stop backtracking immediately. What kind of data structure could answer such query efficiently? Does a hash table work? Why or why not? How about a Trie? If you would like to learn how to implement a basic trie, please work on this problem: [Implement Trie (Prefix Tree)](https://leetcode.com/problems/implement-trie-prefix-tree/) first.

SORTING AND SEARCHING

-----------------------------------------------------------------------

Median of Two Sorted Arrays

-----------------------------------------------------------------------

There are two sorted arrays **nums1** and **nums2** of size m and n respectively.

Find the median of the two sorted arrays. The overall run time complexity should be O(log (m+n)).

You may assume **nums1** and **nums2** cannot be both empty.

**Example 1:**

nums1 = [1, 3]

nums2 = [2]

The median is 2.0

**Example 2:**

nums1 = [1, 2]

nums2 = [3, 4]

The median is (2 + 3)/2 = 2.5

-----------------------------------------------------------------------

Search in Rotated Sorted Array

-----------------------------------------------------------------------

Suppose an array sorted in ascending order is rotated at some pivot unknown to you beforehand.

(i.e., [0,1,2,4,5,6,7] might become [4,5,6,7,0,1,2]).

You are given a target value to search. If found in the array return its index, otherwise return -1.

You may assume no duplicate exists in the array.

Your algorithm's runtime complexity must be in the order of *O*(log *n*).

**Example 1:**

**Input:** nums = [4,5,6,7,0,1,2], target = 0

**Output:** 4

**Example 2:**

**Input:** nums = [4,5,6,7,0,1,2], target = 3

**Output:** -1

-----------------------------------------------------------------------

Merge Intervals

-----------------------------------------------------------------------

Given a collection of intervals, merge all overlapping intervals.

**Example 1:**

**Input:** [[1,3],[2,6],[8,10],[15,18]]

**Output:** [[1,6],[8,10],[15,18]]

**Explanation:** Since intervals [1,3] and [2,6] overlaps, merge them into [1,6].

**Example 2:**

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

**Output:** [[1,5]]

**Explanation:** Intervals [1,4] and [4,5] are considered overlapping.

**NOTE:** input types have been changed on April 15, 2019. Please reset to default code definition to get new method signature.

-----------------------------------------------------------------------

Two Sum II - Input array is sorted

-----------------------------------------------------------------------

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.

The function twoSum should return indices of the two numbers such that they add up to the target, where index1 must be less than index2.

**Note:**

* Your returned answers (both index1 and index2) are not zero-based.
* You may assume that each input would have *exactly* one solution and you may not use the *same* element twice.

**Example:**

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

**Output:** [1,2]

**Explanation:** The sum of 2 and 7 is 9. Therefore index1 = 1, index2 = 2.

-----------------------------------------------------------------------

Kth Largest Element in an Array

-----------------------------------------------------------------------

Find the **k**th largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.

**Example 1:**

**Input:** [3,2,1,5,6,4] and k = 2

**Output:** 5

**Example 2:**

**Input:** [3,2,3,1,2,4,5,5,6] and k = 4

**Output:** 4

**Note:**   
You may assume k is always valid, 1 ≤ k ≤ array's length.

-----------------------------------------------------------------------

Meeting Rooms II

-----------------------------------------------------------------------

Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...] (si < ei), find the minimum number of conference rooms required.

**Example 1:**

**Input:** [[0, 30],[5, 10],[15, 20]]

**Output:** 2

**Example 2:**

**Input:** [[7,10],[2,4]]

**Output:** 1

**NOTE:** input types have been changed on April 15, 2019. Please reset to default code definition to get new method signature.

   Hide Hint #1

Think about how we would approach this problem in a very simplistic way. We will allocate rooms to meetings that occur earlier in the day v/s the ones that occur later on, right?

   Hide Hint #2

If you've figured out that we have to **sort** the meetings by their start time, the next thing to think about is how do we do the allocation?   
There are two scenarios possible here for any meeting. Either there is no meeting room available and a new one has to be allocated, or a meeting room has freed up and this meeting can take place there.

   Hide Hint #3

An important thing to note is that we don't really care **which** room gets freed up while allocating a room for the current meeting. As long as a room is free, our job is done.   
  
We already know the rooms we have allocated till now and we also know when are they due to get free because of the end times of the meetings going on in those rooms. We can simply check the room which is due to get vacated the earliest amongst all the allocated rooms.

   Hide Hint #4

Following up on the previous hint, we can make use of a min-heap to store the end times of the meetings in various rooms.   
  
So, every time we want to check if any room is free or not, simply check the topmost element of the min heap as that would be the room that would get free the earliest out of all the other rooms currently occupied.   
  
If the room we extracted from the top of the min heap isn't free, then no other room is. So, we can save time here and simply allocate a new room.

-----------------------------------------------------------------------

Top K Frequent Elements

-----------------------------------------------------------------------

Given a non-empty array of integers, return the ***k*** most frequent elements.

**Example 1:**

**Input:** nums = [1,1,1,2,2,3], k = 2

**Output:** [1,2]

**Example 2:**

**Input:** nums = [1], k = 1

**Output:** [1]

**Note:**

* You may assume *k* is always valid, 1 ≤ *k* ≤ number of unique elements.
* Your algorithm's time complexity **must be** better than O(*n* log *n*), where *n* is the array's size.

-----------------------------------------------------------------------

K Closest Points to Origin

-----------------------------------------------------------------------

We have a list of points on the plane.  Find the K closest points to the origin (0, 0).

(Here, the distance between two points on a plane is the Euclidean distance.)

You may return the answer in any order.  The answer is guaranteed to be unique (except for the order that it is in.)

**Example 1:**

**Input:** points = [[1,3],[-2,2]], K = 1

**Output:** [[-2,2]]

**Explanation:**

The distance between (1, 3) and the origin is sqrt(10).

The distance between (-2, 2) and the origin is sqrt(8).

Since sqrt(8) < sqrt(10), (-2, 2) is closer to the origin.

We only want the closest K = 1 points from the origin, so the answer is just [[-2,2]].

**Example 2:**

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

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

(The answer [[-2,4],[3,3]] would also be accepted.)

**Note:**

1. 1 <= K <= points.length <= 10000
2. -10000 < points[i][0] < 10000
3. -10000 < points[i][1] < 10000

-----------------------------------------------------------------------

DYNAMIC PROGRAMMING

-----------------------------------------------------------------------

Longest Palindromic Substring

-----------------------------------------------------------------------

Given a string **s**, find the longest palindromic substring in **s**. You may assume that the maximum length of **s** is 1000.

**Example 1:**

**Input:** "babad"

**Output:** "bab"

**Note:** "aba" is also a valid answer.

**Example 2:**

**Input:** "cbbd"

**Output:** "bb"

   Hide Hint #1

How can we reuse a previously computed palindrome to compute a larger palindrome?

   Hide Hint #2

If “aba” is a palindrome, is “xabax” and palindrome? Similarly is “xabay” a palindrome?

   Hide Hint #3

Complexity based hint:  
If we use brute-force and check whether for every start and end position a substring is a palindrome we have O(n^2) start - end pairs and O(n) palindromic checks. Can we reduce the time for palindromic checks to O(1) by reusing some previous computation.

-----------------------------------------------------------------------

Maximum Subarray

-----------------------------------------------------------------------

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.

-----------------------------------------------------------------------

Best Time to Buy and Sell Stock

-----------------------------------------------------------------------

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.

-----------------------------------------------------------------------

Word Break

-----------------------------------------------------------------------

Given a **non-empty** string *s* and a dictionary *wordDict* containing a list of **non-empty** words, determine if *s* can be segmented into a space-separated sequence of one or more dictionary words.

**Note:**

* The same word in the dictionary may be reused multiple times in the segmentation.
* You may assume the dictionary does not contain duplicate words.

**Example 1:**

**Input:** s = "leetcode", wordDict = ["leet", "code"]

**Output:** true

**Explanation:** Return true because "leetcode" can be segmented as "leet code".

**Example 2:**

**Input:** s = "applepenapple", wordDict = ["apple", "pen"]

**Output:** true

**Explanation:** Return true because "applepenapple" can be segmented as "apple pen apple".

  Note that you are allowed to reuse a dictionary word.

**Example 3:**

**Input:** s = "catsandog", wordDict = ["cats", "dog", "sand", "and", "cat"]

**Output:** false

-----------------------------------------------------------------------

Coin Change

-----------------------------------------------------------------------

You are given coins of different denominations and a total amount of money *amount*. Write a function to compute the fewest number of coins that you need to make up that amount. If that amount of money cannot be made up by any combination of the coins, return -1.

**Example 1:**

**Input:** coins = [1, 2, 5], amount = 11

**Output:** 3

**Explanation:** 11 = 5 + 5 + 1

**Example 2:**

**Input:** coins = [2], amount = 3

**Output:** -1

**Note**:  
You may assume that you have an infinite number of each kind of coin.

-----------------------------------------------------------------------

DESIGN

-----------------------------------------------------------------------

LRU Cache

-----------------------------------------------------------------------

Design and implement a data structure for [Least Recently Used (LRU) cache](https://en.wikipedia.org/wiki/Cache_replacement_policies#LRU). It should support the following operations: get and put.

get(key) - Get the value (will always be positive) of the key if the key exists in the cache, otherwise return -1.  
put(key, value) - Set or insert the value if the key is not already present. When the cache reached its capacity, it should invalidate the least recently used item before inserting a new item.

The cache is initialized with a **positive** capacity.

**Follow up:**  
Could you do both operations in **O(1)** time complexity?

**Example:**

LRUCache cache = new LRUCache( 2 /\* capacity \*/ );

cache.put(1, 1);

cache.put(2, 2);

cache.get(1); // returns 1

cache.put(3, 3); // evicts key 2

cache.get(2); // returns -1 (not found)

cache.put(4, 4); // evicts key 1

cache.get(1); // returns -1 (not found)

cache.get(3); // returns 3

cache.get(4); // returns 4

-----------------------------------------------------------------------

Min Stack

-----------------------------------------------------------------------

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.

   Hide Hint #1

Consider each node in the stack having a minimum value. (Credits to @aakarshmadhavan)

-----------------------------------------------------------------------

Find Median from Data Stream

-----------------------------------------------------------------------

Median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value. So the median is the mean of the two middle value.

For example,

[2,3,4], the median is 3

[2,3], the median is (2 + 3) / 2 = 2.5

Design a data structure that supports the following two operations:

* void addNum(int num) - Add a integer number from the data stream to the data structure.
* double findMedian() - Return the median of all elements so far.

**Example:**

addNum(1)

addNum(2)

findMedian() -> 1.5

addNum(3)

findMedian() -> 2

**Follow up:**

1. If all integer numbers from the stream are between 0 and 100, how would you optimize it?
2. If 99% of all integer numbers from the stream are between 0 and 100, how would you optimize it?

-----------------------------------------------------------------------

Serialize and Deserialize Binary Tree

-----------------------------------------------------------------------

Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment.

Design an algorithm to serialize and deserialize a binary tree. There is no restriction on how your serialization/deserialization algorithm should work. You just need to ensure that a binary tree can be serialized to a string and this string can be deserialized to the original tree structure.

**Example:**

You may serialize the following tree:

1

/ \

2 3

/ \

4 5

as "[1,2,3,null,null,4,5]"

**Clarification:** The above format is the same as [how LeetCode serializes a binary tree](https://leetcode.com/faq/#binary-tree). You do not necessarily need to follow this format, so please be creative and come up with different approaches yourself.

**Note:**Do not use class member/global/static variables to store states. Your serialize and deserialize algorithms should be stateless.

-----------------------------------------------------------------------

Design Tic-Tac-Toe

-----------------------------------------------------------------------

Design a Tic-tac-toe game that is played between two players on a *n* x *n* grid.

You may assume the following rules:

1. A move is guaranteed to be valid and is placed on an empty block.
2. Once a winning condition is reached, no more moves is allowed.
3. A player who succeeds in placing *n* of their marks in a horizontal, vertical, or diagonal row wins the game.

**Example:**

Given *n* = 3, assume that player 1 is "X" and player 2 is "O" in the board.

TicTacToe toe = new TicTacToe(3);

toe.move(0, 0, 1); -> Returns 0 (no one wins)

|X| | |

| | | | // Player 1 makes a move at (0, 0).

| | | |

toe.move(0, 2, 2); -> Returns 0 (no one wins)

|X| |O|

| | | | // Player 2 makes a move at (0, 2).

| | | |

toe.move(2, 2, 1); -> Returns 0 (no one wins)

|X| |O|

| | | | // Player 1 makes a move at (2, 2).

| | |X|

toe.move(1, 1, 2); -> Returns 0 (no one wins)

|X| |O|

| |O| | // Player 2 makes a move at (1, 1).

| | |X|

toe.move(2, 0, 1); -> Returns 0 (no one wins)

|X| |O|

| |O| | // Player 1 makes a move at (2, 0).

|X| |X|

toe.move(1, 0, 2); -> Returns 0 (no one wins)

|X| |O|

|O|O| | // Player 2 makes a move at (1, 0).

|X| |X|

toe.move(2, 1, 1); -> Returns 1 (player 1 wins)

|X| |O|

|O|O| | // Player 1 makes a move at (2, 1).

|X|X|X|

**Follow up:**  
Could you do better than O(*n*2) per move() operation?

   Hide Hint #1

Could you trade extra space such that move() operation can be done in O(1)?

   Hide Hint #2

You need two arrays: int rows[n], int cols[n], plus two variables: diagonal, anti\_diagonal.

-----------------------------------------------------------------------

Design Search Autocomplete System

-----------------------------------------------------------------------

Design a search autocomplete system for a search engine. Users may input a sentence (at least one word and end with a special character '#'). For **each character** they type **except '#'**, you need to return the **top 3** historical hot sentences that have prefix the same as the part of sentence already typed. Here are the specific rules:

1. The hot degree for a sentence is defined as the number of times a user typed the exactly same sentence before.
2. The returned top 3 hot sentences should be sorted by hot degree (The first is the hottest one). If several sentences have the same degree of hot, you need to use ASCII-code order (smaller one appears first).
3. If less than 3 hot sentences exist, then just return as many as you can.
4. When the input is a special character, it means the sentence ends, and in this case, you need to return an empty list.

Your job is to implement the following functions:

The constructor function:

AutocompleteSystem(String[] sentences, int[] times): This is the constructor. The input is **historical data**. Sentences is a string array consists of previously typed sentences. Times is the corresponding times a sentence has been typed. Your system should record these historical data.

Now, the user wants to input a new sentence. The following function will provide the next character the user types:

List<String> input(char c): The input c is the next character typed by the user. The character will only be lower-case letters ('a' to 'z'), blank space (' ') or a special character ('#'). Also, the previously typed sentence should be recorded in your system. The output will be the **top 3** historical hot sentences that have prefix the same as the part of sentence already typed.

**Example:**  
**Operation:** AutocompleteSystem(["i love you", "island","ironman", "i love leetcode"], [5,3,2,2])  
The system have already tracked down the following sentences and their corresponding times:  
"i love you" : 5 times  
"island" : 3 times  
"ironman" : 2 times  
"i love leetcode" : 2 times  
Now, the user begins another search:  
  
**Operation:** input('i')  
**Output:** ["i love you", "island","i love leetcode"]  
**Explanation:**  
There are four sentences that have prefix "i". Among them, "ironman" and "i love leetcode" have same hot degree. Since ' ' has ASCII code 32 and 'r' has ASCII code 114, "i love leetcode" should be in front of "ironman". Also we only need to output top 3 hot sentences, so "ironman" will be ignored.  
  
**Operation:** input(' ')  
**Output:** ["i love you","i love leetcode"]  
**Explanation:**  
There are only two sentences that have prefix "i ".  
  
**Operation:** input('a')  
**Output:** []  
**Explanation:**  
There are no sentences that have prefix "i a".  
  
**Operation:** input('#')  
**Output:** []  
**Explanation:**  
The user finished the input, the sentence "i a" should be saved as a historical sentence in system. And the following input will be counted as a new search.

**Note:**

1. The input sentence will always start with a letter and end with '#', and only one blank space will exist between two words.
2. The number of **complete sentences** that to be searched won't exceed 100. The length of each sentence including those in the historical data won't exceed 100.
3. Please use double-quote instead of single-quote when you write test cases even for a character input.
4. Please remember to **RESET** your class variables declared in class AutocompleteSystem, as static/class variables are **persisted across multiple test cases**. Please see [here](https://leetcode.com/faq/#different-output) for more details.

-----------------------------------------------------------------------

Maximum Frequency Stack

-----------------------------------------------------------------------

Implement FreqStack, a class which simulates the operation of a stack-like data structure.

FreqStack has two functions:

* push(int x), which pushes an integer x onto the stack.
* pop(), which **removes** and returns the most frequent element in the stack.
  + If there is a tie for most frequent element, the element closest to the top of the stack is removed and returned.

**Example 1:**

**Input:**

["FreqStack","push","push","push","push","push","push","pop","pop","pop","pop"],

[[],[5],[7],[5],[7],[4],[5],[],[],[],[]]

**Output:** [null,null,null,null,null,null,null,5,7,5,4]

**Explanation**:

After making six .push operations, the stack is [5,7,5,7,4,5] from bottom to top. Then:

pop() -> returns 5, as 5 is the most frequent.

The stack becomes [5,7,5,7,4].

pop() -> returns 7, as 5 and 7 is the most frequent, but 7 is closest to the top.

The stack becomes [5,7,5,4].

pop() -> returns 5.

The stack becomes [5,7,4].

pop() -> returns 4.

The stack becomes [5,7].

**Note:**

* Calls to FreqStack.push(int x) will be such that 0 <= x <= 10^9.
* It is guaranteed that FreqStack.pop() won't be called if the stack has zero elements.
* The total number of FreqStack.push calls will not exceed 10000 in a single test case.
* The total number of FreqStack.pop calls will not exceed 10000 in a single test case.
* The total number of FreqStack.push and FreqStack.pop calls will not exceed 150000 across all test cases.

Others

-----------------------------------------------------------------------

Reverse Integer

-----------------------------------------------------------------------

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 store integers within the 32-bit signed integer range: [−231,  231− 1]. For the purpose of this problem, assume that your function returns 0 when the reversed integer overflows.

-----------------------------------------------------------------------

Second Highest Salary

-----------------------------------------------------------------------

Write a SQL query to get the second highest salary from the Employee table.

+----+--------+

| Id | Salary |

+----+--------+

| 1 | 100 |

| 2 | 200 |

| 3 | 300 |

+----+--------+

For example, given the above Employee table, the query should return 200 as the second highest salary. If there is no second highest salary, then the query should return null.

+---------------------+

| SecondHighestSalary |

+---------------------+

| 200 |

+---------------------+

-----------------------------------------------------------------------

Partition Labels

-----------------------------------------------------------------------

A string S of lowercase letters is given. We want to partition this string into as many parts as possible so that each letter appears in at most one part, and return a list of integers representing the size of these parts.

**Example 1:**

**Input:** S = "ababcbacadefegdehijhklij"

**Output:** [9,7,8]

**Explanation:**

The partition is "ababcbaca", "defegde", "hijhklij".

This is a partition so that each letter appears in at most one part.

A partition like "ababcbacadefegde", "hijhklij" is incorrect, because it splits S into less parts.

**Note:**

1. S will have length in range [1, 500].
2. S will consist of lowercase letters ('a' to 'z') only.

   Hide Hint #1

Try to greedily choose the smallest partition that includes the first letter. If you have something like "abaccbdeffed", then you might need to add b. You can use an map like "last['b'] = 5" to help you expand the width of your partition.

-----------------------------------------------------------------------

Prison Cells After N Days

-----------------------------------------------------------------------

There are 8 prison cells in a row, and each cell is either occupied or vacant.

Each day, whether the cell is occupied or vacant changes according to the following rules:

* If a cell has two adjacent neighbors that are both occupied or both vacant, then the cell becomes occupied.
* Otherwise, it becomes vacant.

(Note that because the prison is a row, the first and the last cells in the row can't have two adjacent neighbors.)

We describe the current state of the prison in the following way: cells[i] == 1 if the i-th cell is occupied, else cells[i] == 0.

Given the initial state of the prison, return the state of the prison after N days (and N such changes described above.)

**Example 1:**

**Input:** cells = [0,1,0,1,1,0,0,1], N = 7

**Output:** [0,0,1,1,0,0,0,0]

**Explanation:**

The following table summarizes the state of the prison on each day:

Day 0: [0, 1, 0, 1, 1, 0, 0, 1]

Day 1: [0, 1, 1, 0, 0, 0, 0, 0]

Day 2: [0, 0, 0, 0, 1, 1, 1, 0]

Day 3: [0, 1, 1, 0, 0, 1, 0, 0]

Day 4: [0, 0, 0, 0, 0, 1, 0, 0]

Day 5: [0, 1, 1, 1, 0, 1, 0, 0]

Day 6: [0, 0, 1, 0, 1, 1, 0, 0]

Day 7: [0, 0, 1, 1, 0, 0, 0, 0]

**Example 2:**

**Input:** cells = [1,0,0,1,0,0,1,0], N = 1000000000

**Output:** [0,0,1,1,1,1,1,0]

**Note:**

1. cells.length == 8
2. cells[i] is in {0, 1}
3. 1 <= N <= 10^9