BITTIGER

Top 100 算法题

1. Two Sum

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].
```

2. 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.
```

3. Longest Substring Without Repeating Characters

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

Examples:

Given `"abcabcbb"`, the answer is `"abc"`, which the length is 3.

Given `"bbbbb"`, the answer is `"b"`, with the length of 1.

Given `"pwwkew"`, 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.

4. 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)).

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
```

5. 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:

```
Input: "babad"

Output: "bab"

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

Example:

```
Input: "cbbd"

Output: "bb"
```

11. Container With Most Water

Given n non-negative integers a_1, a_2, ..., a_n, where each represents a point at coordinate (i, a_i). n vertical lines are drawn such that the two endpoints of line i is at (i, a_i) 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.

14. Longest Common Prefix

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

15. 3Sum

Given an array S of n integers, are there elements a, b, c in S 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.

```
For example, given array S = [-1, 0, 1, 2, -1, -4],

A solution set is:
[
  [-1, 0, 1],
  [-1, -1, 2]
]
```

17. Letter Combinations of a Phone Number

Given a digit string, 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.


```
Input:Digit string "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.

20. Valid Parentheses

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

The brackets must close in the correct order, `"()"` and `"()[]{}"` are all valid but `"(]"` and `"([)]"` are not.

21. 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
```

22. 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:

```
[
  "((()))",
  "(()())",
  "(())()",
  "()(())",
  "()()()"
]
```

23. Merge k Sorted Lists

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

29. Divide Two Integers

Divide two integers without using multiplication, division and mod operator.

If it is overflow, return MAX_INT.

31. Next Permutation

Implement next permutation, which rearranges numbers into the lexicographically next greater permutation of numbers.

If such arrangement is not possible, it must rearrange it as the lowest possible order (ie, sorted in ascending order).

The replacement must be in-place, do not allocate extra memory.

Here are some examples. Inputs are in the left-hand column and its corresponding outputs are in the right-hand column.

`1,2,3` → `1,3,2`

`3,2,1` → `1,2,3`

`1,1,5` → `1,5,1`

33. 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.

38. Count and Say

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*, 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"
```

39. Combination Sum

Given a **set** of candidate numbers (***C***) **(without duplicates)** and a target number (***T***), find all unique combinations in ***C*** where the candidate numbers sums to ***T***.

The **same** repeated number may be chosen from ***C*** unlimited number of times.

Note:

- All numbers (including target) will be positive integers.
- The solution set must not contain duplicate combinations.

For example, given candidate set `[2, 3, 6, 7]` and target `7`,
A solution set is:

```
[
  [7],
  [2, 2, 3]
]
```

41. First Missing Positive

Given an unsorted integer array, find the first missing positive integer.

For example,
Given `[1,2,0]` return `3`,
and `[3,4,-1,1]` return `2`.

Your algorithm should run in *O*(*n*) time and uses constant space.

43. Multiply Strings

Given two non-negative integers `num1` and `num2` represented as strings, return the product of `num1` and `num2`.

Note:

1. The length of both `num1` and `num2` is < 110.
2. Both `num1` and `num2` contains only digits `0-9`.
3. Both `num1` and `num2` does not contain any leading zero.
4. You **must not use any built-in BigInteger library** or **convert the inputs to integer** directly.

49. Group Anagrams

Given an array of strings, group anagrams together.

For example, given: `["eat", "tea", "tan", "ate", "nat", "bat"]`,
Return:

```
[
  ["ate", "eat","tea"],
  ["nat","tan"],
  ["bat"]
]
```

Note: All inputs will be in lower-case.

53. Maximum Subarray

Find the contiguous subarray within an array (containing at least one number) which has the largest sum.

For example, given the array `[-2,1,-3,4,-1,2,1,-5,4]`,
the contiguous subarray `[4,-1,2,1]` has the largest sum = `6`.

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

54. Spiral Matrix

Given a matrix of *m* x *n* elements (*m* rows, *n* columns), return all elements of the matrix in spiral order.

For example,
Given the following matrix:

```
[
 [ 1, 2, 3 ],
 [ 4, 5, 6 ],
 [ 7, 8, 9 ]
]
```

You should return `[1,2,3,6,9,8,7,4,5]`.

55. Jump Game

Given an array of non-negative integers, you are initially positioned at the first index of the array.

Each element in the array represents your maximum jump length at that position.

Determine if you are able to reach the last index.

For example:
A = `[2,3,1,1,4]`, return `true`.

A = `[3,2,1,0,4]`, return `false`.

56. Merge Intervals

Given a collection of intervals, merge all overlapping intervals.

For example,
Given `[1,3],[2,6],[8,10],[15,18]`,
return `[1,6],[8,10],[15,18]`.

62. Unique Paths

A robot is located at the top-left corner of a *m* x *n* grid (marked 'Start' in the diagram below).

The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corner of the grid (marked 'Finish' in the diagram below).

How many possible unique paths are there?

Above is a 3 x 7 grid. How many possible unique paths are there?

Note: *m* and *n* will be at most 100.

70. Climbing Stairs

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
```

72. Edit Distance

Given two words *word1* and *word2*, find the minimum number of steps required to convert *word1* to *word2*. (each operation is counted as 1 step.)

You have the following 3 operations permitted on a word:

a) Insert a character
b) Delete a character
c) Replace a character

76. 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).

For example,

S = `"ADOBECODEBANC"`

T = `"ABC"`

Minimum window is `"BANC"`.

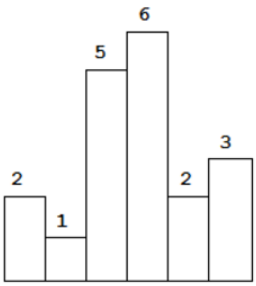
Note:

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

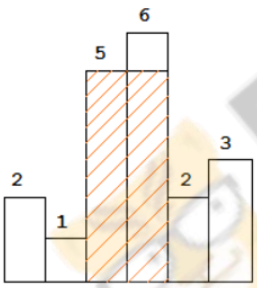
If there are multiple such windows, you are guaranteed that there will always be only one unique minimum window in S.

84. Largest Rectangle in Histogram

Given *n* non-negative integers representing the histogram's bar height where the width of each bar is 1, find the area of largest rectangle in the histogram.



Above is a histogram where width of each bar is 1, given height = `[2,1,5,6,2,3]`.



The largest rectangle is shown in the shaded area, which has area = `10` unit.

For example,

Given heights = `[2,1,5,6,2,3]`,

return `10`.

88. Merge Sorted Array

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

Note:

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

91. Decode Ways

A message containing letters from `A-Z` is being encoded to numbers using the following mapping:

```
'A' -> 1
'B' -> 2
...
'Z' -> 26
```

Given an encoded message containing digits, determine the total number of ways to decode it.

For example,
Given encoded message `"12"`, it could be decoded as `"AB"` (1 2) or `"L"` (12).

The number of ways decoding `"12"` is 2.

94. Binary Tree Inorder Traversal

Given a binary tree, return the *inorder* traversal of its nodes' values.

For example:
Given binary tree `[1,null,2,3]`,

```
   1
    \
     2
    /
   3
```

return `[1,3,2]`.

Note: Recursive solution is trivial, could you do it iteratively?

102. 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]
]
```

103. 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]
]
```

104. Maximum Depth of Binary Tree

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.

105. Construct Binary Tree from Preorder and Inorder Traversal

Given preorder and inorder traversal of a tree, construct the binary tree.

Note:
You may assume that duplicates do not exist in the tree.

108. Convert Sorted Array to Binary Search Tree

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
```

114. Flatten Binary Tree to Linked List

Given a binary tree, flatten it to a linked list in-place.

For example,
Given

```
        1
       / \
      2   5
     / \   \
    3   4   6
```

The flattened tree should look like:

```
   1
    \
     2
      \
       3
        \
         4
          \
           5
            \
             6
```

121. 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 (ie, buy one and sell one share of the stock), design an algorithm to find the maximum profit.

Example 1:

```
Input: [7, 1, 5, 3, 6, 4]
Output: 5

max. difference = 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

In this case, no transaction is done, i.e. max profit = 0.
```

122. Best Time to Buy and Sell Stock II

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 (ie, buy one and sell one share of the stock multiple times). However, you may not engage in multiple transactions at the same time (ie, you must sell the stock before you buy again).

124. Binary Tree Maximum Path Sum

Given a 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.

For example:
Given the below binary tree,

```
  1
 / \
2   3
```

Return `6`.

128. Longest Consecutive Sequence

Given an unsorted array of integers, find the length of the longest consecutive elements sequence.

For example,
Given `[100, 4, 200, 1, 3, 2]`,
The longest consecutive elements sequence is `[1, 2, 3, 4]`. Return its length: `4`.

Your algorithm should run in O(*n*) complexity.

133. Clone Graph

Clone an undirected graph. Each node in the graph contains a `label` and a list of its `neighbors`.

OJ's undirected graph serialization:
Nodes are labeled uniquely.

We use `#` as a separator for each node, and `,` as a separator for node label and each neighbor of the node.

As an example, consider the serialized graph `{0,1,2#1,2#2,2}`.

The graph has a total of three nodes, and therefore contains three parts as separated by `#`.

1. First node is labeled as `0`. Connect node `0` to both nodes `1` and `2`.
2. Second node is labeled as `1`. Connect node `1` to node `2`.
3. Third node is labeled as `2`. Connect node `2` to node `2` (itself), thus forming a self-cycle.

Visually, the graph looks like the following:

```
   1
  / \
 /   \
0 --- 2
     / \
     \_/
```

134. Gas Station

There are *N* gas stations along a circular route, where the amount of gas at station *i* is `gas[i]`.

You have a car with an unlimited gas tank and it costs `cost[i]` of gas to travel from station *i* to its next station (*i*+1). You begin the journey with an empty tank at one of the gas stations.

Return the starting gas station's index if you can travel around the circuit once, otherwise return -1.

Note:
The solution is guaranteed to be unique.

138. 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 of the list.

141. Linked List Cycle

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

Follow up:
Can you solve it without using extra space?

145. Binary Tree Postorder Traversal

Given a binary tree, return the *postorder* traversal of its nodes' values.

For example:
Given binary tree `{1,#,2,3}`,

```
   1
    \
     2
    /
   3
```

return `[3,2,1]`.

Note: Recursive solution is trivial, could you do it iteratively?

146. LRU Cache

Design and implement a data structure for Least Recently Used (LRU) cache. 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.

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
```

149. Max Points on a Line

Given *n* points on a 2D plane, find the maximum number of points that lie on the same straight line.

150. Evaluate Reverse Polish Notation

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are `+`, `-`, `*`, `/`. Each operand may be an integer or another expression.

Some examples:

```
["2", "1", "+", "3", "*"] -> ((2 + 1) * 3) -> 9
["4", "13", "5", "/", "+"] -> (4 + (13 / 5)) -> 6
```

152. Maximum Product Subarray

Find the contiguous subarray within an array (containing at least one number) which has the largest product.

For example, given the array `[2,3,-2,4]`,
the contiguous subarray `[2,3]` has the largest product = `6`.

155. 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.
```

160. Intersection of Two Linked Lists

Write a program to find the node at which the intersection of two singly linked lists begins.

For example, the following two linked lists:

```
A:          a1 → a2
                   ↘
                     c1 → c2 → c3
                   ↗
B:     b1 → b2 → b3
```

begin to intersect at node c1.

Notes:

- If the two linked lists have no intersection at all, return `null`.
- The linked lists must retain their original structure after the function returns.
- You may assume there are no cycles anywhere in the entire linked structure.
- Your code should preferably run in O(n) time and use only O(1) memory.

Credits:

Special thanks to @stellari for adding this problem and creating all test cases.

162. Find Peak Element

A peak element is an element that is greater than its neighbors.

Given an input array where `num[i] ≠ num[i+1]`, find a peak element and return its index.

The array may contain multiple peaks, in that case return the index to any one of the peaks is fine.

You may imagine that `num[-1] = num[n] = -∞`.

For example, in array `[1, 2, 3, 1]`, 3 is a peak element and your function should return the index number 2.

click to show spoilers.

Credits:
Special thanks to @ts for adding this problem and creating all test cases.

166. Fraction to Recurring Decimal

Given two integers representing the numerator and denominator of a fraction, return the fraction in string format.

If the fractional part is repeating, enclose the repeating part in parentheses.

For example,

- Given numerator = 1, denominator = 2, return "0.5".
- Given numerator = 2, denominator = 1, return "2".
- Given numerator = 2, denominator = 3, return "0.(6)".

Credits:
Special thanks to @Shangrila for adding this problem and creating all test cases.

169. Majority Element

Given an array of size *n*, find the majority element. The majority element is the element that appears **more than** `⌊ n/2 ⌋` times.

You may assume that the array is non-empty and the majority element always exist in the array.

Credits:
Special thanks to @ts for adding this problem and creating all test cases.

171. Excel Sheet Column Number

Related to question Excel Sheet Column Title

Given a column title as appear in an Excel sheet, return its corresponding column number.

For example:

```
A -> 1
B -> 2
C -> 3
...
Z -> 26
AA -> 27
AB -> 28
```

Credits:
Special thanks to @ts for adding this problem and creating all test cases.

172. Factorial Trailing Zeroes

Given an integer *n*, return the number of trailing zeroes in *n*!.

Note: Your solution should be in logarithmic time complexity.

Credits:
Special thanks to @ts for adding this problem and creating all test cases.

174. Dungeon Game

The demons had captured the princess (**P**) and imprisoned her in the bottom-right corner of a dungeon. The dungeon consists of M x N rooms laid out in a 2D grid. Our valiant knight (**K**) was initially positioned in the top-left room and must fight his way through the dungeon to rescue the princess.

The knight has an initial health point represented by a positive integer. If at any point his health point drops to 0 or below, he dies immediately.

Some of the rooms are guarded by demons, so the knight loses health (*negative* integers) upon entering these rooms; other rooms are either empty (*0*'s) or contain magic orbs that increase the knight's health (*positive* integers).

In order to reach the princess as quickly as possible, the knight decides to move only rightward or downward in each step.

Write a function to determine the knight's minimum initial health so that he is able to rescue the princess.

For example, given the dungeon below, the initial health of the knight must be at least **7** if he follows the optimal path `RIGHT-> RIGHT -> DOWN -> DOWN`.

-2 (K)	-3	3
-5	-10	1
10	30	-5 (P)

Notes:

- The knight's health has no upper bound.
- Any room can contain threats or power-ups, even the first room the knight enters and the bottom-right room where the princess is imprisoned.

Credits:
Special thanks to @stellari for adding this problem and creating all test cases.

179. Largest Number

Given a list of non negative integers, arrange them such that they form the largest number.

For example, given `[3, 30, 34, 5, 9]`, the largest formed number is `9534330`.

Note: The result may be very large, so you need to return a string instead of an integer.

Credits:
Special thanks to @ts for adding this problem and creating all test cases.

198. House Robber

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stashed, the only constraint stopping you from robbing each of them is that adjacent houses have security system connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight **without alerting the police**.

Credits:
Special thanks to @ifanchu for adding this problem and creating all test cases. Also thanks to @ts for adding additional test cases.

200. 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:

```
11110
11010
11000
00000
```

Answer: 1

Example 2:

```
11000
11000
00100
00011
```

Answer: 3

202. Happy Number

Write an algorithm to determine if a number is "happy".

A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers.

Example: 19 is a happy number

$1^2 + 9^2 = 82$
$8^2 + 2^2 = 68$
$6^2 + 8^2 = 100$
$1^2 + 0^2 + 0^2 = 1$

204. Count Primes

Description:

Count the number of prime numbers less than a non-negative number, ***n***.

Credits:

Special thanks to @mithmatt for adding this problem and creating all test cases.

206. Reverse Linked List

Reverse a singly linked list.

click to show more hints.

207. 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?

For example:

```
2, [[1,0]]
```

There are a total of 2 courses to take. To take course 1 you should have finished course 0. So it is possible.

```
2, [[1,0],[0,1]]
```

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.
2. You may assume that there are no duplicate edges in the input prerequisites.

208. Implement Trie (Prefix Tree)

Implement a trie with `insert`, `search`, and `startsWith` methods.

Note:

You may assume that all inputs are consist of lowercase letters `a-z`.

210. Course Schedule II

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**, return the ordering of courses you should take to finish all courses.

There may be multiple correct orders, you just need to return one of them. If it is impossible to finish all courses, return an empty array.

For example:

```
2, [[1,0]]
```

There are a total of 2 courses to take. To take course 1 you should have finished course 0. So the correct course order is `[0,1]`

```
4, [[1,0],[2,0],[3,1],[3,2]]
```

There are a total of 4 courses to take. To take course 3 you should have finished both courses 1 and 2. Both courses 1 and 2 should be taken after you finished course 0. So one correct course order is `[0,1,2,3]`. Another correct ordering is `[0,2,1,3]`.

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.
2. You may assume that there are no duplicate edges in the input prerequisites.

click to show more hints.

213. House Robber II

Note: This is an extension of House Robber.

After robbing those houses on that street, the thief has found himself a new place for his thievery so that he will not get too much attention. This time, all houses at this place are **arranged in a circle.** That means the first house is the neighbor of the last one. Meanwhile, the security system for these houses remain the same as for those in the previous street.

Given a list of non-negative integers representing the amount of money of each house, determine the maximum amount of money you can rob tonight **without alerting the police**.

Credits:
Special thanks to @Freezen for adding this problem and creating all test cases.

215. 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.
For example,
Given `[3,2,1,5,6,4]` and k = 2, return 5.

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

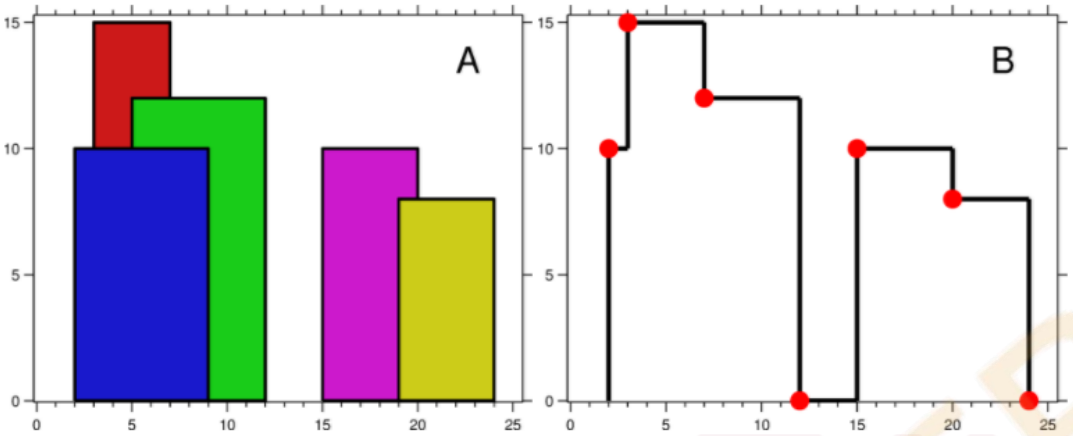
Credits:
Special thanks to @mithmatt for adding this problem and creating all test cases.

217. Contains Duplicate

Given an array of integers, find if the array contains any duplicates. Your function should return true if any value appears at least twice in the array, and it should return false if every element is distinct.

218. The Skyline Problem

A city's skyline is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Now suppose you are **given the locations and height of all the buildings** as shown on a cityscape photo (Figure A), write a program to **output the skyline** formed by these buildings collectively (Figure B).



The geometric information of each building is represented by a triplet of integers `[Li, Ri, Hi]`, where `Li` and `Ri` are the x coordinates of the left and right edge of the ith building, respectively, and `Hi` is its height. It is guaranteed that `0 ≤ Li, Ri ≤ INT_MAX`, `0 < Hi ≤ INT_MAX`, and `Ri - Li > 0`. You may assume all buildings are perfect rectangles grounded on an absolutely flat surface at height 0.

For instance, the dimensions of all buildings in Figure A are recorded as: `[ [2 9 10], [3 7 15], [5 12 12], [15 20 10], [19 24 8] ]` .

The output is a list of "**key points**" (red dots in Figure B) in the format of `[ [x1,y1], [x2, y2], [x3, y3], ... ]` that uniquely defines a skyline. **A key point is the left endpoint of a horizontal line segment**. Note that the last key point, where the rightmost building ends, is merely used to mark the termination of the skyline, and always has zero height. Also, the ground in between any two adjacent buildings should be considered part of the skyline contour.

For instance, the skyline in Figure B should be represented as: `[ [2 10], [3 15], [7 12], [12 0], [15 10], [20 8], [24, 0] ]`.

Notes:

- The number of buildings in any input list is guaranteed to be in the range `[0, 10000]`.
- The input list is already sorted in ascending order by the left x position `Li`.
- The output list must be sorted by the x position.
- There must be no consecutive horizontal lines of equal height in the output skyline. For instance, `[...[2 3], [4 5], [7 5], [11 5], [12 7]...]` is not acceptable; the three lines of height 5 should be merged into one in the final output as such: `[...[2 3], [4 5], [12 7], ...]`

224. Basic Calculator

Implement a basic calculator to evaluate a simple expression string.

The expression string may contain open `(` and closing parentheses `)`, the plus `+` or minus sign `-`, **non-negative** integers and empty spaces ` `.

You may assume that the given expression is always valid.

Some examples:

```
"1 + 1" = 2
" 2-1 + 2 " = 3
"(1+(4+5+2)-3)+(6+8)" = 23
```

Note: Do not use the `eval` built-in library function.

227. Basic Calculator II

Implement a basic calculator to evaluate a simple expression string.

The expression string contains only **non-negative** integers, `+`, `-`, `*`, `/` operators and empty spaces ` `. The integer division should truncate toward zero.

You may assume that the given expression is always valid.

Some examples:

```
"3+2*2" = 7
" 3/2 " = 1
" 3+5 / 2 " = 5
```

Note: Do not use the `eval` built-in library function.

230. Kth Smallest Element in a BST

Given a binary search tree, write a function `kthSmallest` to find the **k**th smallest element in it.

Note:
You may assume k is always valid, 1 ≤ k ≤ BST's total elements.

Follow up:
What if the BST is modified (insert/delete operations) often and you need to find the kth smallest frequently? How would you optimize the kthSmallest routine?

234. Palindrome Linked List

Given a singly linked list, determine if it is a palindrome.

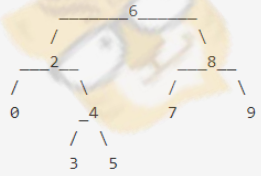
Follow up:

Could you do it in O(n) time and O(1) space?

235. Lowest Common Ancestor of a Binary Search Tree

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

According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes v and w as the lowest node in T that has both v and w as descendants (where we allow **a node to be a descendant of itself**).”

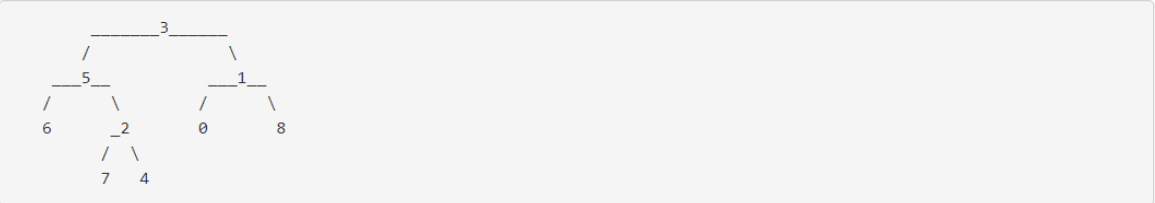


For example, the lowest common ancestor (LCA) of nodes `2` and `8` is `6`. Another example is LCA of nodes `2` and `4` is `2`, since a node can be a descendant of itself according to the LCA definition.

236. 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: “The lowest common ancestor is defined between two nodes v and w as the lowest node in T that has both v and w as descendants (where we allow **a node to be a descendant of itself**).”



For example, the lowest common ancestor (LCA) of nodes `5` and `1` is `3`. Another example is LCA of nodes `5` and `4` is `5`, since a node can be a descendant of itself according to the LCA definition.

237. Delete Node in a Linked List

Write a function to delete a node (except the tail) in a singly linked list, given only access to that node.

Supposed the linked list is `1 -> 2 -> 3 -> 4` and you are given the third node with value `3`, the linked list should become `1 -> 2 -> 4` after calling your function.

239. Sliding Window Maximum

Given an array *nums*, there is a sliding window of size *k* which is moving from the very left of the array to the very right. You can only see the *k* numbers in the window. Each time the sliding window moves right by one position.

For example,
Given *nums* = `[1,3,-1,-3,5,3,6,7]`, and *k* = 3.

```
Window position                Max
---------------               -----
[1  3  -1] -3  5  3  6  7       3
 1 [3  -1  -3] 5  3  6  7       3
 1  3 [-1  -3  5] 3  6  7       5
 1  3  -1 [-3  5  3] 6  7       5
 1  3  -1  -3 [5  3  6] 7       6
 1  3  -1  -3  5 [3  6  7]      7
```

Therefore, return the max sliding window as `[3,3,5,5,6,7]`.

Note:
You may assume *k* is always valid, ie: 1 ≤ k ≤ input array's size for non-empty array.

Follow up:
Could you solve it in linear time?

251. Flatten 2D Vector

Implement an iterator to flatten a 2d vector.

For example,
Given 2d vector =

```
[
  [1,2],
  [3],
  [4,5,6]
]
```

By calling *next* repeatedly until *hasNext* returns false, the order of elements returned by *next* should be: `[1,2,3,4,5,6]`.

Follow up:
As an added challenge, try to code it using only iterators in C++ or iterators in Java.

253. Meeting Rooms II

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

For example,
Given `[[0, 30],[5, 10],[15, 20]]`,
return `2`.

269. Alien Dictionary

There is a new alien language which uses the latin alphabet. However, the order among letters are unknown to you. You receive a list of **non-empty** words from the dictionary, where **words are sorted lexicographically by the rules of this new language**. Derive the order of letters in this language.

Example 1:
Given the following words in dictionary,

```
[
  "wrt",
  "wrf",
  "er",
  "ett",
  "rftt"
]
```

The correct order is: `"wertf"`.

Example 2:
Given the following words in dictionary,

```
[
  "z",
  "x"
]
```

The correct order is: `"zx"`.

Example 3:
Given the following words in dictionary,

```
[
  "z",
  "x",
  "z"
]
```

The order is invalid, so return `""`.

Note:

1. You may assume all letters are in lowercase.
2. You may assume that if a is a prefix of b, then a must appear before b in the given dictionary.
3. If the order is invalid, return an empty string.
4. There may be multiple valid order of letters, return any one of them is fine.

277. Find the Celebrity

Suppose you are at a party with `n` people (labeled from `0` to `n - 1`) and among them, there may exist one celebrity. The definition of a celebrity is that all the other `n - 1` people know him/her but he/she does not know any of them.

Now you want to find out who the celebrity is or verify that there is not one. The only thing you are allowed to do is to ask questions like: "Hi, A. Do you know B?" to get information of whether A knows B. You need to find out the celebrity (or verify there is not one) by asking as few questions as possible (in the asymptotic sense).

You are given a helper function `bool knows(a, b)` which tells you whether A knows B. Implement a function `int findCelebrity(n)`, your function should minimize the number of calls to `knows`.

Note: There will be exactly one celebrity if he/she is in the party. Return the celebrity's label if there is a celebrity in the party. If there is no celebrity, return `-1`.

285. Inorder Successor in BST

Given a binary search tree and a node in it, find the in-order successor of that node in the BST.

Note: If the given node has no in-order successor in the tree, return `null`.

287. Find the Duplicate Number

Given an array *nums* containing *n* + 1 integers where each integer is between 1 and *n* (inclusive), prove that at least one duplicate number must exist. Assume that there is only one duplicate number, find the duplicate one.

Note:

1. You **must not** modify the array (assume the array is read only).
2. You must use only constant, *O*(1) extra space.
3. Your runtime complexity should be less than $O(n^2)$.
4. There is only one duplicate number in the array, but it could be repeated more than once.

289. Game of Life

According to the Wikipedia's article: "The **Game of Life**, also known simply as **Life**, is a cellular automaton devised by the British mathematician John Horton Conway in 1970."

Given a *board* with *m* by *n* cells, each cell has an initial state *live* (1) or *dead* (0). Each cell interacts with its eight neighbors (horizontal, vertical, diagonal) using the following four rules (taken from the above Wikipedia article):

1. Any live cell with fewer than two live neighbors dies, as if caused by under-population.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by over-population..
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

Write a function to compute the next state (after one update) of the board given its current state.

Follow up:

1. Could you solve it in-place? Remember that the board needs to be updated at the same time: You cannot update some cells first and then use their updated values to update other cells.
2. In this question, we represent the board using a 2D array. In principle, the board is infinite, which would cause problems when the active area encroaches the border of the array. How would you address these problems?

295. 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.

Examples:
`[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.

For example:

```
addNum(1)
addNum(2)
findMedian() -> 1.5
addNum(3)
findMedian() -> 2
```

300. Longest Increasing Subsequence

Given an unsorted array of integers, find the length of longest increasing subsequence.

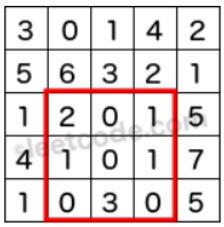
For example,
Given `[10, 9, 2, 5, 3, 7, 101, 18]`,
The longest increasing subsequence is `[2, 3, 7, 101]`, therefore the length is `4`. Note that there may be more than one LIS combination, it is only necessary for you to return the length.

Your algorithm should run in O(n^2) complexity.

Follow up: Could you improve it to O(n log n) time complexity?

308. Range Sum Query 2D - Mutable

Given a 2D matrix *matrix*, find the sum of the elements inside the rectangle defined by its upper left corner (*row*1, *col*1) and lower right corner (*row*2, *col*2).



The above rectangle (with the red border) is defined by (row1, col1) = **(2, 1)** and (row2, col2) = **(4, 3)**, which contains sum = **8**.

Example:

```
Given matrix = [
  [3, 0, 1, 4, 2],
  [5, 6, 3, 2, 1],
  [1, 2, 0, 1, 5],
  [4, 1, 0, 1, 7],
  [1, 0, 3, 0, 5]
]

sumRegion(2, 1, 4, 3) -> 8
update(3, 2, 2)
sumRegion(2, 1, 4, 3) -> 10
```

Note:

1. The matrix is only modifiable by the *update* function.
2. You may assume the number of calls to *update* and *sumRegion* function is distributed evenly.
3. You may assume that *row*1 ≤ *row*2 and *col*1 ≤ *col*2.

315. Count of Smaller Numbers After Self

You are given an integer array *nums* and you have to return a new *counts* array. The *counts* array has the property where `counts[i]` is the number of smaller elements to the right of `nums[i]`.

Example:

```
Given nums = [5, 2, 6, 1]

To the right of 5 there are 2 smaller elements (2 and 1).
To the right of 2 there is only 1 smaller element (1).
To the right of 6 there is 1 smaller element (1).
To the right of 1 there is 0 smaller element.
```

Return the array `[2, 1, 1, 0]`.

322. 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:

coins = `[1, 2, 5]`, amount = `11`

return `3` (11 = 5 + 5 + 1)

Example 2:

coins = `[2]`, amount = `3`

return `-1`.

Note:

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

324. Wiggle Sort II

Given an unsorted array `nums`, reorder it such that `nums[0] < nums[1] > nums[2] < nums[3]...`.

Example:

(1) Given `nums = [1, 5, 1, 1, 6, 4]`, one possible answer is `[1, 4, 1, 5, 1, 6]`.

(2) Given `nums = [1, 3, 2, 2, 3, 1]`, one possible answer is `[2, 3, 1, 3, 1, 2]`.

Note:

You may assume all input has valid answer.

Follow Up:

Can you do it in O(n) time and/or in-place with O(1) extra space?

329. Longest Increasing Path in a Matrix

Given an integer matrix, find the length of the longest increasing path.

From each cell, you can either move to four directions: left, right, up or down. You may NOT move diagonally or move outside of the boundary (i.e. wrap-around is not allowed).

Example 1:

```
nums = [
  [9,9,4],
  [6,6,8],
  [2,1,1]
]
```

Return `4`
The longest increasing path is `[1, 2, 6, 9]`.

Example 2:

```
nums = [
  [3,4,5],
  [3,2,6],
  [2,2,1]
]
```

Return `4`
The longest increasing path is `[3, 4, 5, 6]`. Moving diagonally is not allowed.

334. Increasing Triplet Subsequence

Given an unsorted array return whether an increasing subsequence of length 3 exists or not in the array.

Formally the function should:

> Return true if there exists *i, j, k*
> such that *arr[i] < arr[j] < arr[k]* given 0 ≤ *i* < *j* < *k* ≤ *n*-1 else return false.

Your algorithm should run in O(*n*) time complexity and O(*1*) space complexity.

Examples:
Given `[1, 2, 3, 4, 5]`,
return `true`.

Given `[5, 4, 3, 2, 1]`,
return `false`.

340. Longest Substring with At Most K Distinct Characters

Given a string, find the length of the longest substring T that contains at most *k* distinct characters.

For example, Given s = `“eceba”` and k = 2,

T is "ece" which its length is 3.

341. Flatten Nested List Iterator

Given a nested list of integers, implement an iterator to flatten it.

Each element is either an integer, or a list -- whose elements may also be integers or other lists.

Example 1:
Given the list `[[1,1],2,[1,1]]`,

By calling *next* repeatedly until *hasNext* returns false, the order of elements returned by *next* should be: `[1,1,2,1,1]`.

Example 2:
Given the list `[1,[4,[6]]]`,

By calling *next* repeatedly until *hasNext* returns false, the order of elements returned by *next* should be: `[1,4,6]`.

344. Reverse String

Write a function that takes a string as input and returns the string reversed.

Example:
Given s = "hello", return "olleh".

347. Top K Frequent Elements

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

For example,
Given `[1,1,1,2,2,3]` and k = 2, return `[1,2]`.

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.

348. 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?

350. Intersection of Two Arrays II

Given two arrays, write a function to compute their intersection.

Example:
Given *nums1* = `[1, 2, 2, 1]`, *nums2* = `[2, 2]`, return `[2, 2]`.

Note:

- Each element in the result should appear as many times as it shows in both arrays.
- The result can be in any order.

Follow up:

- What if the given array is already sorted? How would you optimize your algorithm?
- What if *nums1*'s size is small compared to *nums2*'s size? Which algorithm is better?
- What if elements of *nums2* are stored on disk, and the memory is limited such that you cannot load all elements into the memory at once?

371. Sum of Two Integers

Calculate the sum of two integers *a* and *b*, but you are **not allowed** to use the operator `+` and `-`.

Example:

Given *a* = 1 and *b* = 2, return 3.

378. Kth Smallest Element in a Sorted Matrix

Given a *n* x *n* matrix where each of the rows and columns are sorted in ascending order, find the kth smallest element in the matrix.

Note that it is the kth smallest element in the sorted order, not the kth distinct element.

Example:

```
matrix = [
   [ 1,  5,  9],
   [10, 11, 13],
   [12, 13, 15]
],
k = 8,

return 13.
```

Note:
You may assume k is always valid, $1 \le k \le n^2$.

380. Insert Delete GetRandom O(1)

Design a data structure that supports all following operations in *average* **O(1)** time.

1. `insert(val)` : Inserts an item val to the set if not already present.
2. `remove(val)` : Removes an item val from the set if present.
3. `getRandom` : Returns a random element from current set of elements. Each element must have the **same probability** of being returned.

Example:

```
// Init an empty set.
RandomizedSet randomSet = new RandomizedSet();

// Inserts 1 to the set. Returns true as 1 was inserted successfully.
randomSet.insert(1);

// Returns false as 2 does not exist in the set.
randomSet.remove(2);

// Inserts 2 to the set, returns true. Set now contains [1,2].
randomSet.insert(2);

// getRandom should return either 1 or 2 randomly.
randomSet.getRandom();

// Removes 1 from the set, returns true. Set now contains [2].
randomSet.remove(1);

// 2 was already in the set, so return false.
randomSet.insert(2);

// Since 2 is the only number in the set, getRandom always return 2.
randomSet.getRandom();
```

382. Linked List Random Node

Given a singly linked list, return a random node's value from the linked list. Each node must have the **same probability** of being chosen.

Follow up:

What if the linked list is extremely large and its length is unknown to you? Could you solve this efficiently without using extra space?

Example:

```
// Init a singly linked list [1,2,3].
ListNode head = new ListNode(1);
head.next = new ListNode(2);
head.next.next = new ListNode(3);
Solution solution = new Solution(head);

// getRandom() should return either 1, 2, or 3 randomly. Each element should have equal probability of returning.
solution.getRandom();
```

384. Shuffle an Array

Shuffle a set of numbers without duplicates.

Example:

```
// Init an array with set 1, 2, and 3.
int[] nums = {1,2,3};
Solution solution = new Solution(nums);

// Shuffle the array [1,2,3] and return its result. Any permutation of [1,2,3] must equally likely to be returned.
solution.shuffle();

// Resets the array back to its original configuration [1,2,3].
solution.reset();

// Returns the random shuffling of array [1,2,3].
solution.shuffle();
```

387. 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.

395. Longest Substring with At Least K Repeating Characters

Find the length of the longest substring ***T*** of a given string (consists of lowercase letters only) such that every character in ***T*** appears no less than *k* times.

Example 1:

```
Input:
s = "aaabb", k = 3

Output:
3

The longest substring is "aaa", as 'a' is repeated 3 times.
```

Example 2:

```
Input:
s = "ababbc", k = 2

Output:
5

The longest substring is "ababb", as 'a' is repeated 2 times and 'b' is repeated 3 times.
```

398. Random Pick Index

Given an array of integers with possible duplicates, randomly output the index of a given target number. You can assume that the given target number must exist in the array.

Note:
The array size can be very large. Solution that uses too much extra space will not pass the judge.

Example:

```
int[] nums = new int[] {1,2,3,3,3};
Solution solution = new Solution(nums);

// pick(3) should return either index 2, 3, or 4 randomly. Each index should have equal probability of returning.
solution.pick(3);

// pick(1) should return 0. Since in the array only nums[0] is equal to 1.
solution.pick(1);
```

412. Fizz Buzz

Write a program that outputs the string representation of numbers from 1 to *n*.

But for multiples of three it should output “Fizz” instead of the number and for the multiples of five output “Buzz”. For numbers which are multiples of both three and five output “FizzBuzz”.

Example:

```
n = 15,

Return:
[
    "1",
    "2",
    "Fizz",
    "4",
    "Buzz",
    "Fizz",
    "7",
    "8",
    "Fizz",
    "Buzz",
    "11",
    "Fizz",
    "13",
    "14",
    "FizzBuzz"
]
```

454. 4Sum II

Given four lists A, B, C, D of integer values, compute how many tuples `(i, j, k, l)` there are such that `A[i] + B[j] + C[k] + D[l]` is zero.

To make problem a bit easier, all A, B, C, D have same length of N where $0 \le N \le 500$. All integers are in the range of -2^{28} to $2^{28} - 1$ and the result is guaranteed to be at most $2^{31} - 1$.

Example:

```
Input:
A = [ 1, 2]
B = [-2,-1]
C = [-1, 2]
D = [ 0, 2]

Output:
2

Explanation:
The two tuples are:
1. (0, 0, 0, 1) -> A[0] + B[0] + C[0] + D[1] = 1 + (-2) + (-1) + 2 = 0
2. (1, 1, 0, 0) -> A[1] + B[1] + C[0] + D[0] = 2 + (-1) + (-1) + 0 = 0
```

592. Fraction Addition and Subtraction

Given a string representing an expression of fraction addition and subtraction, you need to return the calculation result in string format. The final result should be irreducible fraction. If your final result is an integer, say `2`, you need to change it to the format of fraction that has denominator `1`. So in this case, `2` should be converted to `2/1`.

Example 1:

```
Input:"-1/2+1/2"
Output: "0/1"
```

Example 2:

```
Input:"-1/2+1/2+1/3"
Output: "1/3"
```

Example 3:

```
Input:"1/3-1/2"
Output: "-1/6"
```

Example 4:

```
Input:"5/3+1/3"
Output: "2/1"
```

Note:

1. The input string only contains `'0'` to `'9'`, `'/'`, `'+'` and `'-'`. So does the output.
2. Each fraction (input and output) has format `±numerator/denominator`. If the first input fraction or the output is positive, then `'+'` will be omitted.
3. The input only contains valid **irreducible fractions**, where the **numerator** and **denominator** of each fraction will always be in the range [1,10]. If the denominator is 1, it means this fraction is actually an integer in a fraction format defined above.
4. The number of given fractions will be in the range [1,10].
5. The numerator and denominator of the **final result** are guaranteed to be valid and in the range of 32-bit int.

611. Valid Triangle Number

Given an array consists of non-negative integers, your task is to count the number of triplets chosen from the array that can make triangles if we take them as side lengths of a triangle.

Example 1:

```
Input: [2,2,3,4]
Output: 3
Explanation:
Valid combinations are:
2,3,4 (using the first 2)
2,3,4 (using the second 2)
2,2,3
```

Note:

1. The length of the given array won't exceed 1000.
2. The integers in the given array are in the range of [0, 1000].

697. Degree of an Array

Given a non-empty array of non-negative integers `nums`, the **degree** of this array is defined as the maximum frequency of any one of its elements.

Your task is to find the smallest possible length of a (contiguous) subarray of `nums`, that has the same degree as `nums`.

Example 1:

```
Input: [1, 2, 2, 3, 1]
Output: 2
Explanation:
The input array has a degree of 2 because both elements 1 and 2 appear twice.
Of the subarrays that have the same degree:
[1, 2, 2, 3, 1], [1, 2, 2, 3], [2, 2, 3, 1], [1, 2, 2], [2, 2, 3], [2, 2]
The shortest length is 2. So return 2.
```

Example 2:

```
Input: [1,2,2,3,1,4,2]
Output: 6
```

Note:

- `nums.length` will be between 1 and 50,000.
- `nums[i]` will be an integer between 0 and 49,999.

硅谷精英程序员学员最新 offer 榜

FirstName	LastName	School	Major	Company	Position
J	Yuan	CMU	CS	Morgan Stanley	SDE
M	Ye	GWU	EE	amazon	SDE
M	Ye	GWU	EE	Capital One	SDE
Y	Lou	The Ohio State University	Electrical and Computer Engineering	Microsoft	SDE
J	Yuan	CMU	CS	Goldman Sachs	SDE
S	Sun	Changshu Institute of Technology	Computer Engineering	UBER	SDE
Y	Li	Georgia Institute of Technology	CS	Suning 苏宁	SDE
A	Guan	University of California, San Diego	EE	facebook	SDE
Y	Lou	The Ohio State University	Electrical and Computer Engineering	LinkedIn	SDE
X	Pu	University of Pennsylvania	CS	facebook	SDE
Y	Lou	The Ohio State University	ECE	Microsoft	SDE
Y	Jiang	CMU	CS	Microsoft	SDE
B	Bao	Temple University	CS	QVC	SDE
Y	Jiang	CMU	CS	Microsoft	SDE
J	HU	CMU	MIS	Pinterest	SDE
R	Wang	Cornell Univerisity	Information Science	Adobe	SDE
P	Chan	Georgia Institute of Technology	CS	entytle	Data Analyst
W	Meng	Purdue	CS	Expedia	SDE
Z	Lin	Carnegie Mellon University	Management Information Systems	SONICWALL	Program Manager
J	Lu	Penn State University	Information Science	ebay	SDE
J	Lu	Penn State University	Information Science	Google	AE
Y	Li	State University of New York at Buffalo	CS	SolarCity	SDE
Y	Li	State University of New York at Buffalo	CS	ACELLENT	SDE
Y	Li	State University of New York at Buffalo	CS	Oracle	Full Stack
P	Chan	Georgia Institute of Technology	CS	ennovo	Data Scientist
W	Guan	University of California, San Diego	Electrical Engineering	Bloomberg	SDE
J	Hu	CMU	MIS	LinkedIn	SDE
T	Yang	Case Western Reserve University	EE	Google	SDE
J	Hu	CMU	MIS	UBER	SDE
J	Mou	CMU	CS	UBER	SDE
J	Mou	CMU	CS	Google	SDE
T	Yang	Case Western Reserve University	EE	facebook	SDE
Z	Li	University of Southern California	Electrical Engineering	Microsoft	SDE
J	Mou	CMU	CS	rubrik	SDE
K	Liu	University of Pittsburgh	Materials Science	Google	SDE
W	Zhang	Columbia University	EE	elementum	Supply Chain Management
T	Zhao	Ohio State University	Computer Science	TalkingData	SDE
P	Sun	CMU	Information System Management	UBER	SDE
Y	Zeng	University of New Haven	Computer Science	Google	SDE
X	Xu	University of California, San Diego	Electrical Engineering	PayPal	SDE