# **1. Two Sum**

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

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

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

You can return the answer in any order.

**Example 1:**

Input: nums = [2,7,11,15], target = 9  
Output: [0,1]  
Explanation: Because nums[0] + nums[1] == 9, we return [0, 1].

**Example 2:**

Input: nums = [3,2,4], target = 6  
Output: [1,2]

**Example 3:**

Input: nums = [3,3], target = 6  
Output: [0,1]

**Constraints:**

* 2 <= nums.length <= 104
* -109 <= nums[i] <= 109
* -109 <= target <= 109
* **Only one valid answer exists.**

**Follow-up:**Can you come up with an algorithm that is less than O(n2) time complexity?

# **2667. Create Hello World Function**

https://leetcode.com/problems/create-hello-world-function/description/ Write a function createHelloWorld. It should return a new function that always returns "Hello World".

**Example 1:**

Input: args = []  
Output: "Hello World"  
Explanation:  
const f = createHelloWorld();  
f(); // "Hello World"  
  
The function returned by createHelloWorld should always return "Hello World".

**Example 2:**

Input: args = [{},null,42]  
Output: "Hello World"  
Explanation:  
const f = createHelloWorld();  
f({}, null, 42); // "Hello World"  
  
Any arguments could be passed to the function but it should still always return "Hello World".

**Constraints:**

* 0 <= args.length <= 10

# **88. Merge Sorted Array**

https://leetcode.com/problems/merge-sorted-array/description/

You are given two integer arrays nums1 and nums2, sorted in **non-decreasing order**, and two integers m and n, representing the number of elements in nums1 and nums2 respectively.

**Merge** nums1 and nums2 into a single array sorted in **non-decreasing order**.

The final sorted array should not be returned by the function, but instead be *stored inside the array* nums1. To accommodate this, nums1 has a length of m + n, where the first m elements denote the elements that should be merged, and the last n elements are set to 0 and should be ignored. nums2 has a length of n.

**Example 1:**

Input: nums1 = [1,2,3,0,0,0], m = 3, nums2 = [2,5,6], n = 3  
Output: [1,2,2,3,5,6]  
Explanation: The arrays we are merging are [1,2,3] and [2,5,6].  
The result of the merge is [1,2,2,3,5,6] with the underlined elements coming from nums1.

**Example 2:**

Input: nums1 = [1], m = 1, nums2 = [], n = 0  
Output: [1]  
Explanation: The arrays we are merging are [1] and [].  
The result of the merge is [1].

**Example 3:**

Input: nums1 = [0], m = 0, nums2 = [1], n = 1  
Output: [1]  
Explanation: The arrays we are merging are [] and [1].  
The result of the merge is [1].  
Note that because m = 0, there are no elements in nums1. The 0 is only there to ensure the merge result can fit in nums1.

**Constraints:**

* nums1.length == m + n
* nums2.length == n
* 0 <= m, n <= 200
* 1 <= m + n <= 200
* -109 <= nums1[i], nums2[j] <= 109

**Follow up:** Can you come up with an algorithm that runs in O(m + n) time?

# **20. Valid Parentheses**

https://leetcode.com/problems/valid-parentheses/description/

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

An input string is valid if:

1. Open brackets must be closed by the same type of brackets.
2. Open brackets must be closed in the correct order.
3. Every close bracket has a corresponding open bracket of the same type.

**Example 1:**

Input: s = "()"  
Output: true

**Example 2:**

Input: s = "()[]{}"  
Output: true

**Example 3:**

Input: s = "(]"  
Output: false

**Constraints:**

* 1 <= s.length <= 104
* s consists of parentheses only '()[]{}'.

# **146. LRU Cache**

https://leetcode.com/problems/lru-cache/description/

Design a data structure that follows the constraints of a [**Least Recently Used (LRU) cache**](https://en.wikipedia.org/wiki/Cache_replacement_policies#LRU).

Implement the LRUCache class:

* LRUCache(int capacity) Initialize the LRU cache with **positive** size capacity.
* int get(int key) Return the value of the key if the key exists, otherwise return -1.
* void put(int key, int value) Update the value of the key if the key exists. Otherwise, add the key-value pair to the cache. If the number of keys exceeds the capacity from this operation, **evict** the least recently used key.

The functions get and put must each run in O(1) average time complexity.

**Example 1:**

Input  
["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"]  
[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]  
Output  
[null, null, null, 1, null, -1, null, -1, 3, 4]  
  
Explanation  
LRUCache lRUCache = new LRUCache(2);  
lRUCache.put(1, 1); // cache is {1=1}  
lRUCache.put(2, 2); // cache is {1=1, 2=2}  
lRUCache.get(1); // return 1  
lRUCache.put(3, 3); // LRU key was 2, evicts key 2, cache is {1=1, 3=3}  
lRUCache.get(2); // returns -1 (not found)  
lRUCache.put(4, 4); // LRU key was 1, evicts key 1, cache is {4=4, 3=3}  
lRUCache.get(1); // return -1 (not found)  
lRUCache.get(3); // return 3  
lRUCache.get(4); // return 4

**Constraints:**

* 1 <= capacity <= 3000
* 0 <= key <= 104
* 0 <= value <= 105
* At most 2 \* 105 calls will be made to get and put.

# **9. Palindrome Number**

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

Given an integer x, return true *if* x *is a*

***palindrome***

*, and* false *otherwise*.

**Example 1:**

Input: x = 121  
Output: true  
Explanation: 121 reads as 121 from left to right and from right to left.

**Example 2:**

Input: x = -121  
Output: false  
Explanation: From left to right, it reads -121. From right to left, it becomes 121-. Therefore it is not a palindrome.

**Example 3:**

Input: x = 10  
Output: false  
Explanation: Reads 01 from right to left. Therefore it is not a palindrome.

**Constraints:**

* -231 <= x <= 231 - 1

**Follow up:** Could you solve it without converting the integer to a string?

# **5. Longest Palindromic Substring**

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

Given a string s, return *the longest*

*palindromic*

*substring*

in s.

**Example 1:**

Input: s = "babad"  
Output: "bab"  
Explanation: "aba" is also a valid answer.

**Example 2:**

Input: s = "cbbd"  
Output: "bb"

**Constraints:**

* 1 <= s.length <= 1000
* s consist of only digits and English letters.

# **121. Best Time to Buy and Sell Stock**

https://leetcode.com/problems/best-time-to-buy-and-sell-stock/description/

You are given an array prices where prices[i] is the price of a given stock on the ith day.

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return 0.

**Example 1:**

Input: prices = [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.  
Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

**Example 2:**

Input: prices = [7,6,4,3,1]  
Output: 0  
Explanation: In this case, no transactions are done and the max profit = 0.

**Constraints:**

* 1 <= prices.length <= 105
* 0 <= prices[i] <= 104

# **3. Longest Substring Without Repeating Characters**

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

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

**substring**

without repeating characters.

**Example 1:**

Input: s = "abcabcbb"  
Output: 3  
Explanation: The answer is "abc", with the length of 3.

**Example 2:**

Input: s = "bbbbb"  
Output: 1  
Explanation: The answer is "b", with the length of 1.

**Example 3:**

Input: s = "pwwkew"  
Output: 3  
Explanation: The answer is "wke", with the length of 3.  
Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.

**Constraints:**

* 0 <= s.length <= 5 \* 104
* s consists of English letters, digits, symbols and spaces.

# **42. Trapping Rain Water**

https://leetcode.com/problems/trapping-rain-water/description/

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

**Example 1:**



Input: height = [0,1,0,2,1,0,1,3,2,1,2,1]  
Output: 6  
Explanation: The above elevation map (black section) 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.

**Example 2:**

Input: height = [4,2,0,3,2,5]  
Output: 9

**Constraints:**

* n == height.length
* 1 <= n <= 2 \* 104
* 0 <= height[i] <= 105

# **56. Merge Intervals**

https://leetcode.com/problems/merge-intervals/description/

Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return *an array of the non-overlapping intervals that cover all the intervals in the input*.

**Example 1:**

Input: intervals = [[1,3],[2,6],[8,10],[15,18]]  
Output: [[1,6],[8,10],[15,18]]  
Explanation: Since intervals [1,3] and [2,6] overlap, merge them into [1,6].

**Example 2:**

Input: intervals = [[1,4],[4,5]]  
Output: [[1,5]]  
Explanation: Intervals [1,4] and [4,5] are considered overlapping.

**Constraints:**

* 1 <= intervals.length <= 104
* intervals[i].length == 2
* 0 <= starti <= endi <= 104

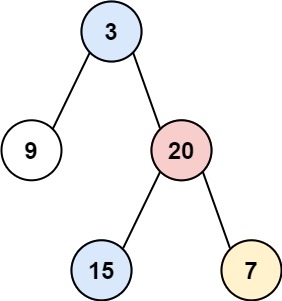
# **314. Binary Tree Vertical Order Traversal**

https://leetcode.com/problems/binary-tree-vertical-order-traversal/description/

Given the root of a binary tree, return ***the vertical order traversal*** *of its nodes' values*. (i.e., from top to bottom, column by column).

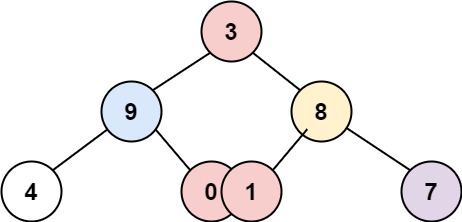
If two nodes are in the same row and column, the order should be from **left to right**.

**Example 1:**



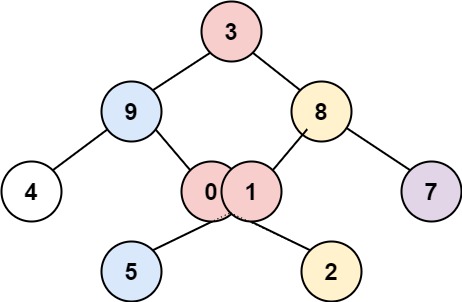
Input: root = [3,9,20,null,null,15,7]  
Output: [[9],[3,15],[20],[7]]

**Example 2:**



Input: root = [3,9,8,4,0,1,7]  
Output: [[4],[9],[3,0,1],[8],[7]]

**Example 3:**



Input: root = [3,9,8,4,0,1,7,null,null,null,2,5]  
Output: [[4],[9,5],[3,0,1],[8,2],[7]]

**Constraints:**

* The number of nodes in the tree is in the range [0, 100].
* -100 <= Node.val <= 100

# **200. Number of Islands**

https://leetcode.com/problems/number-of-islands/description/

Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return *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: grid = [  
 ["1","1","1","1","0"],  
 ["1","1","0","1","0"],  
 ["1","1","0","0","0"],  
 ["0","0","0","0","0"]  
]  
Output: 1

**Example 2:**

Input: grid = [  
 ["1","1","0","0","0"],  
 ["1","1","0","0","0"],  
 ["0","0","1","0","0"],  
 ["0","0","0","1","1"]  
]  
Output: 3

**Constraints:**

* m == grid.length
* n == grid[i].length
* 1 <= m, n <= 300
* grid[i][j] is '0' or '1'.

# **1249. Minimum Remove to Make Valid Parentheses**

https://leetcode.com/problems/minimum-remove-to-make-valid-parentheses/description/

Given a string s of '(' , ')' and lowercase English characters.

Your task is to remove the minimum number of parentheses ( '(' or ')', in any positions ) so that the resulting *parentheses string* is valid and return **any** valid string.

Formally, a *parentheses string* is valid if and only if:

* It is the empty string, contains only lowercase characters, or
* It can be written as AB (A concatenated with B), where A and B are valid strings, or
* It can be written as (A), where A is a valid string.

**Example 1:**

Input: s = "lee(t(c)o)de)"  
Output: "lee(t(c)o)de"  
Explanation: "lee(t(co)de)" , "lee(t(c)ode)" would also be accepted.

**Example 2:**

Input: s = "a)b(c)d"  
Output: "ab(c)d"

**Example 3:**

Input: s = "))(("  
Output: ""  
Explanation: An empty string is also valid.

**Constraints:**

* 1 <= s.length <= 105
* s[i] is either'(' , ')', or lowercase English letter.

# **2. Add Two Numbers**

https://leetcode.com/problems/add-two-numbers/description/

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 contains a single digit. Add the two numbers and return the sum as a linked list.

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

**Example 1:**



Input: l1 = [2,4,3], l2 = [5,6,4]  
Output: [7,0,8]  
Explanation: 342 + 465 = 807.

**Example 2:**

Input: l1 = [0], l2 = [0]  
Output: [0]

**Example 3:**

Input: l1 = [9,9,9,9,9,9,9], l2 = [9,9,9,9]  
Output: [8,9,9,9,0,0,0,1]

**Constraints:**

* The number of nodes in each linked list is in the range [1, 100].
* 0 <= Node.val <= 9
* It is guaranteed that the list represents a number that does not have leading zeros.

# **14. Longest Common Prefix**

https://leetcode.com/problems/longest-common-prefix/description/

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

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

**Example 1:**

Input: strs = ["flower","flow","flight"]  
Output: "fl"

**Example 2:**

Input: strs = ["dog","racecar","car"]  
Output: ""  
Explanation: There is no common prefix among the input strings.

**Constraints:**

* 1 <= strs.length <= 200
* 0 <= strs[i].length <= 200
* strs[i] consists of only lowercase English letters.

# **1757. Recyclable and Low Fat Products**

https://leetcode.com/problems/recyclable-and-low-fat-products/description/

Table: Products

+-------------+---------+  
| Column Name | Type |  
+-------------+---------+  
| product\_id | int |  
| low\_fats | enum |  
| recyclable | enum |  
+-------------+---------+  
product\_id is the primary key (column with unique values) for this table.  
low\_fats is an ENUM (category) of type ('Y', 'N') where 'Y' means this product is low fat and 'N' means it is not.  
recyclable is an ENUM (category) of types ('Y', 'N') where 'Y' means this product is recyclable and 'N' means it is not.

Write a solution to find the ids of products that are both low fat and recyclable.

Return the result table in **any order**.

The result format is in the following example.

**Example 1:**

Input:   
Products table:  
+-------------+----------+------------+  
| product\_id | low\_fats | recyclable |  
+-------------+----------+------------+  
| 0 | Y | N |  
| 1 | Y | Y |  
| 2 | N | Y |  
| 3 | Y | Y |  
| 4 | N | N |  
+-------------+----------+------------+  
Output:   
+-------------+  
| product\_id |  
+-------------+  
| 1 |  
| 3 |  
+-------------+  
Explanation: Only products 1 and 3 are both low fat and recyclable.

# **4. Median of Two Sorted Arrays**

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

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

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

**Example 1:**

Input: nums1 = [1,3], nums2 = [2]  
Output: 2.00000  
Explanation: merged array = [1,2,3] and median is 2.

**Example 2:**

Input: nums1 = [1,2], nums2 = [3,4]  
Output: 2.50000  
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.

**Constraints:**

* nums1.length == m
* nums2.length == n
* 0 <= m <= 1000
* 0 <= n <= 1000
* 1 <= m + n <= 2000
* -106 <= nums1[i], nums2[i] <= 106

# **408. Valid Word Abbreviation**

https://leetcode.com/problems/valid-word-abbreviation/description/

A string can be **abbreviated** by replacing any number of **non-adjacent**, **non-empty** substrings with their lengths. The lengths **should not** have leading zeros.

For example, a string such as "substitution" could be abbreviated as (but not limited to):

* "s10n" ("s ubstitutio n")
* "sub4u4" ("sub stit u tion")
* "12" ("substitution")
* "su3i1u2on" ("su bst i t u ti on")
* "substitution" (no substrings replaced)

The following are **not valid** abbreviations:

* "s55n" ("s ubsti tutio n", the replaced substrings are adjacent)
* "s010n" (has leading zeros)
* "s0ubstitution" (replaces an empty substring)

Given a string word and an abbreviation abbr, return *whether the string* ***matches*** *the given abbreviation*.

A **substring** is a contiguous **non-empty** sequence of characters within a string.

**Example 1:**

Input: word = "internationalization", abbr = "i12iz4n"  
Output: true  
Explanation: The word "internationalization" can be abbreviated as "i12iz4n" ("i nternational iz atio n").

**Example 2:**

Input: word = "apple", abbr = "a2e"  
Output: false  
Explanation: The word "apple" cannot be abbreviated as "a2e".

**Constraints:**

* 1 <= word.length <= 20
* word consists of only lowercase English letters.
* 1 <= abbr.length <= 10
* abbr consists of lowercase English letters and digits.
* All the integers in abbr will fit in a 32-bit integer.

# **15. 3Sum**

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

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

Notice that the solution set must not contain duplicate triplets.

**Example 1:**

Input: nums = [-1,0,1,2,-1,-4]  
Output: [[-1,-1,2],[-1,0,1]]  
Explanation:   
nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0.  
nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0.  
nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0.  
The distinct triplets are [-1,0,1] and [-1,-1,2].  
Notice that the order of the output and the order of the triplets does not matter.

**Example 2:**

Input: nums = [0,1,1]  
Output: []  
Explanation: The only possible triplet does not sum up to 0.

**Example 3:**

Input: nums = [0,0,0]  
Output: [[0,0,0]]  
Explanation: The only possible triplet sums up to 0.

**Constraints:**

* 3 <= nums.length <= 3000
* -105 <= nums[i] <= 105

# **70. Climbing Stairs**

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

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

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

**Example 1:**

Input: n = 2  
Output: 2  
Explanation: There are two ways to climb to the top.  
1. 1 step + 1 step  
2. 2 steps

**Example 2:**

Input: n = 3  
Output: 3  
Explanation: There are three ways to climb to the top.  
1. 1 step + 1 step + 1 step  
2. 1 step + 2 steps  
3. 2 steps + 1 step

**Constraints:**

* 1 <= n <= 45

# **1768. Merge Strings Alternately**

https://leetcode.com/problems/merge-strings-alternately/description/

You are given two strings word1 and word2. Merge the strings by adding letters in alternating order, starting with word1. If a string is longer than the other, append the additional letters onto the end of the merged string.

Return *the merged string.*

**Example 1:**

Input: word1 = "abc", word2 = "pqr"  
Output: "apbqcr"  
Explanation: The merged string will be merged as so:  
word1: a b c  
word2: p q r  
merged: a p b q c r

**Example 2:**

Input: word1 = "ab", word2 = "pqrs"  
Output: "apbqrs"  
Explanation: Notice that as word2 is longer, "rs" is appended to the end.  
word1: a b   
word2: p q r s  
merged: a p b q r s

**Example 3:**

Input: word1 = "abcd", word2 = "pq"  
Output: "apbqcd"  
Explanation: Notice that as word1 is longer, "cd" is appended to the end.  
word1: a b c d  
word2: p q   
merged: a p b q c d

**Constraints:**

* 1 <= word1.length, word2.length <= 100
* word1 and word2 consist of lowercase English letters.

# **49. Group Anagrams**

https://leetcode.com/problems/group-anagrams/description/

Given an array of strings strs, group **the anagrams** together. You can return the answer in **any order**.

An **Anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

**Example 1:**

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

**Example 2:**

Input: strs = [""]  
Output: [[""]]

**Example 3:**

Input: strs = ["a"]  
Output: [["a"]]

**Constraints:**

* 1 <= strs.length <= 104
* 0 <= strs[i].length <= 100
* strs[i] consists of lowercase English letters.

# **2781. Length of the Longest Valid Substring**

https://leetcode.com/problems/length-of-the-longest-valid-substring/description/

You are given a string word and an array of strings forbidden.

A string is called **valid** if none of its substrings are present in forbidden.

Return *the length of the* ***longest valid substring*** *of the string* word.

A **substring** is a contiguous sequence of characters in a string, possibly empty.

**Example 1:**

Input: word = "cbaaaabc", forbidden = ["aaa","cb"]  
Output: 4  
Explanation: There are 11 valid substrings in word: "c", "b", "a", "ba", "aa", "bc", "baa", "aab", "ab", "abc" and "aabc". The length of the longest valid substring is 4.   
It can be shown that all other substrings contain either "aaa" or "cb" as a substring.

**Example 2:**

Input: word = "leetcode", forbidden = ["de","le","e"]  
Output: 4  
Explanation: There are 11 valid substrings in word: "l", "t", "c", "o", "d", "tc", "co", "od", "tco", "cod", and "tcod". The length of the longest valid substring is 4.  
It can be shown that all other substrings contain either "de", "le", or "e" as a substring.

**Constraints:**

* 1 <= word.length <= 105
* word consists only of lowercase English letters.
* 1 <= forbidden.length <= 105
* 1 <= forbidden[i].length <= 10
* forbidden[i] consists only of lowercase English letters.

# **215. Kth Largest Element in an Array**

https://leetcode.com/problems/kth-largest-element-in-an-array/description/

Given an integer array nums and an integer k, return *the* kth *largest element in the array*.

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

Can you solve it without sorting?

**Example 1:**

Input: nums = [3,2,1,5,6,4], k = 2  
Output: 5

**Example 2:**

Input: nums = [3,2,3,1,2,4,5,5,6], k = 4  
Output: 4

**Constraints:**

* 1 <= k <= nums.length <= 105
* -104 <= nums[i] <= 104

# **227. Basic Calculator II**

https://leetcode.com/problems/basic-calculator-ii/description/

Given a string s which represents an expression, *evaluate this expression and return its value*.

The integer division should truncate toward zero.

You may assume that the given expression is always valid. All intermediate results will be in the range of [-231, 231 - 1].

**Note:** You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as eval().

**Example 1:**

Input: s = "3+2\*2"  
Output: 7

**Example 2:**

Input: s = " 3/2 "  
Output: 1

**Example 3:**

Input: s = " 3+5 / 2 "  
Output: 5

**Constraints:**

* 1 <= s.length <= 3 \* 105
* s consists of integers and operators ('+', '-', '\*', '/') separated by some number of spaces.
* s represents **a valid expression**.
* All the integers in the expression are non-negative integers in the range [0, 231 - 1].
* The answer is **guaranteed** to fit in a **32-bit integer**.

# **528. Random Pick with Weight**

https://leetcode.com/problems/random-pick-with-weight/description/

You are given a **0-indexed** array of positive integers w where w[i] describes the **weight** of the ith index.

You need to implement the function pickIndex(), which **randomly** picks an index in the range [0, w.length - 1] (**inclusive**) and returns it. The **probability** of picking an index i is w[i] / sum(w).

* For example, if w = [1, 3], the probability of picking index 0 is 1 / (1 + 3) = 0.25 (i.e., 25%), and the probability of picking index 1 is 3 / (1 + 3) = 0.75 (i.e., 75%).

**Example 1:**

Input  
["Solution","pickIndex"]  
[[[1]],[]]  
Output  
[null,0]  
  
Explanation  
Solution solution = new Solution([1]);  
solution.pickIndex(); // return 0. The only option is to return 0 since there is only one element in w.

**Example 2:**

Input  
["Solution","pickIndex","pickIndex","pickIndex","pickIndex","pickIndex"]  
[[[1,3]],[],[],[],[],[]]  
Output  
[null,1,1,1,1,0]  
  
Explanation  
Solution solution = new Solution([1, 3]);  
solution.pickIndex(); // return 1. It is returning the second element (index = 1) that has a probability of 3/4.  
solution.pickIndex(); // return 1  
solution.pickIndex(); // return 1  
solution.pickIndex(); // return 1  
solution.pickIndex(); // return 0. It is returning the first element (index = 0) that has a probability of 1/4.  
  
Since this is a randomization problem, multiple answers are allowed.  
All of the following outputs can be considered correct:  
[null,1,1,1,1,0]  
[null,1,1,1,1,1]  
[null,1,1,1,0,0]  
[null,1,1,1,0,1]  
[null,1,0,1,0,0]  
......  
and so on.

**Constraints:**

* 1 <= w.length <= 104
* 1 <= w[i] <= 105
* pickIndex will be called at most 104 times.

# **13. Roman to Integer**

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

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

Symbol Value  
I 1  
V 5  
X 10  
L 50  
C 100  
D 500  
M 1000

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

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

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

Given a roman numeral, convert it to an integer.

**Example 1:**

Input: s = "III"  
Output: 3  
Explanation: III = 3.

**Example 2:**

Input: s = "LVIII"  
Output: 58  
Explanation: L = 50, V= 5, III = 3.

**Example 3:**

Input: s = "MCMXCIV"  
Output: 1994  
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

**Constraints:**

* 1 <= s.length <= 15
* s contains only the characters ('I', 'V', 'X', 'L', 'C', 'D', 'M').
* It is **guaranteed** that s is a valid roman numeral in the range [1, 3999].

# **1291. Sequential Digits**

https://leetcode.com/problems/sequential-digits/description/

An integer has *sequential digits* if and only if each digit in the number is one more than the previous digit.

Return a **sorted** list of all the integers in the range [low, high] inclusive that have sequential digits.

**Example 1:**

Input: low = 100, high = 300  
Output: [123,234]

**Example 2:**

Input: low = 1000, high = 13000  
Output: [1234,2345,3456,4567,5678,6789,12345]

**Constraints:**

* 10 <= low <= high <= 10^9

# **11. Container With Most Water**

https://leetcode.com/problems/container-with-most-water/description/

You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]).

Find two lines that together with the x-axis form a container, such that the container contains the most water.

Return *the maximum amount of water a container can store*.

**Notice** that you may not slant the container.

**Example 1:**



Input: height = [1,8,6,2,5,4,8,3,7]  
Output: 49  
Explanation: 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 2:**

Input: height = [1,1]  
Output: 1

**Constraints:**

* n == height.length
* 2 <= n <= 105
* 0 <= height[i] <= 104

# **7. Reverse Integer**

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

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

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

**Example 1:**

Input: x = 123  
Output: 321

**Example 2:**

Input: x = -123  
Output: -321

**Example 3:**

Input: x = 120  
Output: 21

**Constraints:**

* -231 <= x <= 231 - 1

# **1235. Maximum Profit in Job Scheduling**

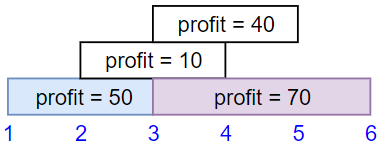
https://leetcode.com/problems/maximum-profit-in-job-scheduling/description/

We have n jobs, where every job is scheduled to be done from startTime[i] to endTime[i], obtaining a profit of profit[i].

You're given the startTime, endTime and profit arrays, return the maximum profit you can take such that there are no two jobs in the subset with overlapping time range.

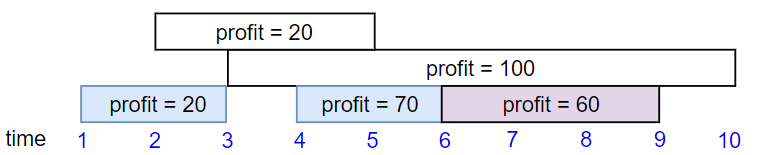
If you choose a job that ends at time X you will be able to start another job that starts at time X.

**Example 1:**



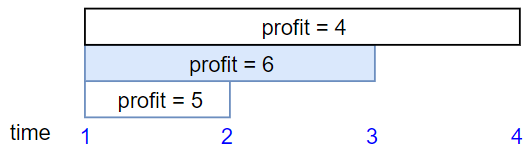
Input: startTime = [1,2,3,3], endTime = [3,4,5,6], profit = [50,10,40,70]  
Output: 120  
Explanation: The subset chosen is the first and fourth job.   
Time range [1-3]+[3-6] , we get profit of 120 = 50 + 70.

**Example 2:**



Input: startTime = [1,2,3,4,6], endTime = [3,5,10,6,9], profit = [20,20,100,70,60]  
Output: 150  
Explanation: The subset chosen is the first, fourth and fifth job.   
Profit obtained 150 = 20 + 70 + 60.

**Example 3:**



Input: startTime = [1,1,1], endTime = [2,3,4], profit = [5,6,4]  
Output: 6

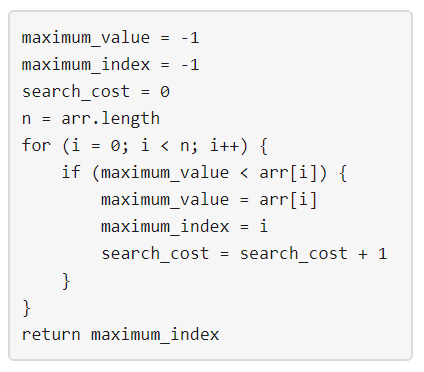
**Constraints:**

* 1 <= startTime.length == endTime.length == profit.length <= 5 \* 104
* 1 <= startTime[i] < endTime[i] <= 109
* 1 <= profit[i] <= 104

# **1420. Build Array Where You Can Find The Maximum Exactly K Comparisons**

https://leetcode.com/problems/build-array-where-you-can-find-the-maximum-exactly-k-comparisons/description/

You are given three integers n, m and k. Consider the following algorithm to find the maximum element of an array of positive integers:



You should build the array arr which has the following properties:

* arr has exactly n integers.
* 1 <= arr[i] <= m where (0 <= i < n).
* After applying the mentioned algorithm to arr, the value search\_cost is equal to k.

Return *the number of ways* to build the array arr under the mentioned conditions. As the answer may grow large, the answer **must be** computed modulo 109 + 7.

**Example 1:**

Input: n = 2, m = 3, k = 1  
Output: 6  
Explanation: The possible arrays are [1, 1], [2, 1], [2, 2], [3, 1], [3, 2] [3, 3]

**Example 2:**

Input: n = 5, m = 2, k = 3  
Output: 0  
Explanation: There are no possible arrays that satisfy the mentioned conditions.

**Example 3:**

Input: n = 9, m = 1, k = 1  
Output: 1  
Explanation: The only possible array is [1, 1, 1, 1, 1, 1, 1, 1, 1]

**Constraints:**

* 1 <= n <= 50
* 1 <= m <= 100
* 0 <= k <= n

# **339. Nested List Weight Sum**

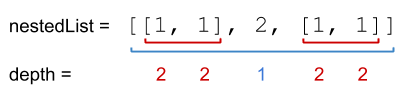
https://leetcode.com/problems/nested-list-weight-sum/description/

You are given a nested list of integers nestedList. Each element is either an integer or a list whose elements may also be integers or other lists.

The **depth** of an integer is the number of lists that it is inside of. For example, the nested list [1,[2,2],[[3],2],1] has each integer's value set to its **depth**.

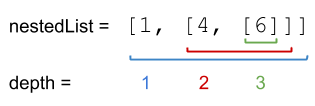
Return *the sum of each integer in* nestedList *multiplied by its* ***depth***.

**Example 1:**



Input: nestedList = [[1,1],2,[1,1]]  
Output: 10  
Explanation: Four 1's at depth 2, one 2 at depth 1. 1\*2 + 1\*2 + 2\*1 + 1\*2 + 1\*2 = 10.

**Example 2:**



Input: nestedList = [1,[4,[6]]]  
Output: 27  
Explanation: One 1 at depth 1, one 4 at depth 2, and one 6 at depth 3. 1\*1 + 4\*2 + 6\*3 = 27.

**Example 3:**

Input: nestedList = [0]  
Output: 0

**Constraints:**

* 1 <= nestedList.length <= 50
* The values of the integers in the nested list is in the range [-100, 100].
* The maximum **depth** of any integer is less than or equal to 50.

# **412. Fizz Buzz**

https://leetcode.com/problems/fizz-buzz/description/

Given an integer n, return *a string array* answer *(****1-indexed****) where*:

* answer[i] == "FizzBuzz" if i is divisible by 3 and 5.
* answer[i] == "Fizz" if i is divisible by 3.
* answer[i] == "Buzz" if i is divisible by 5.
* answer[i] == i (as a string) if none of the above conditions are true.

**Example 1:**

Input: n = 3  
Output: ["1","2","Fizz"]

**Example 2:**

Input: n = 5  
Output: ["1","2","Fizz","4","Buzz"]

**Example 3:**

Input: n = 15  
Output: ["1","2","Fizz","4","Buzz","Fizz","7","8","Fizz","Buzz","11","Fizz","13","14","FizzBuzz"]

**Constraints:**

* 1 <= n <= 104

# **560. Subarray Sum Equals K**

https://leetcode.com/problems/subarray-sum-equals-k/description/

Given an array of integers nums and an integer k, return *the total number of subarrays whose sum equals to* k.

A subarray is a contiguous **non-empty** sequence of elements within an array.

**Example 1:**

Input: nums = [1,1,1], k = 2  
Output: 2

**Example 2:**

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

**Constraints:**

* 1 <= nums.length <= 2 \* 104
* -1000 <= nums[i] <= 1000
* -107 <= k <= 107

# **1650. Lowest Common Ancestor of a Binary Tree III**

https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree-iii/description/

Given two nodes of a binary tree p and q, return *their lowest common ancestor (LCA)*.

Each node will have a reference to its parent node. The definition for Node is below:

class Node {  
 public int val;  
 public Node left;  
 public Node right;  
 public Node parent;  
}

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

**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.

**Example 3:**

Input: root = [1,2], p = 1, q = 2  
Output: 1

**Constraints:**

* The number of nodes in the tree is in the range [2, 105].
* -109 <= Node.val <= 109
* All Node.val are **unique**.
* p != q
* p and q exist in the tree.

# **629. K Inverse Pairs Array**

https://leetcode.com/problems/k-inverse-pairs-array/description/

For an integer array nums, an **inverse pair** is a pair of integers [i, j] where 0 <= i < j < nums.length and nums[i] > nums[j].

Given two integers n and k, return the number of different arrays consisting of numbers from 1 to n such that there are exactly k **inverse pairs**. Since the answer can be huge, return it **modulo** 109 + 7.

**Example 1:**

Input: n = 3, k = 0  
Output: 1  
Explanation: Only the array [1,2,3] which consists of numbers from 1 to 3 has exactly 0 inverse pairs.

**Example 2:**

Input: n = 3, k = 1  
Output: 2  
Explanation: The array [1,3,2] and [2,1,3] have exactly 1 inverse pair.

**Constraints:**

* 1 <= n <= 1000
* 0 <= k <= 1000

# **1570. Dot Product of Two Sparse Vectors**

https://leetcode.com/problems/dot-product-of-two-sparse-vectors/description/

Given two sparse vectors, compute their dot product.

Implement class SparseVector:

* SparseVector(nums) Initializes the object with the vector nums
* dotProduct(vec) Compute the dot product between the instance of *SparseVector* and vec

A **sparse vector** is a vector that has mostly zero values, you should store the sparse vector **efficiently** and compute the dot product between two *SparseVector*.

**Follow up:**What if only one of the vectors is sparse?

**Example 1:**

Input: nums1 = [1,0,0,2,3], nums2 = [0,3,0,4,0]  
Output: 8  
Explanation: v1 = SparseVector(nums1) , v2 = SparseVector(nums2)  
v1.dotProduct(v2) = 1\*0 + 0\*3 + 0\*0 + 2\*4 + 3\*0 = 8

**Example 2:**

Input: nums1 = [0,1,0,0,0], nums2 = [0,0,0,0,2]  
Output: 0  
Explanation: v1 = SparseVector(nums1) , v2 = SparseVector(nums2)  
v1.dotProduct(v2) = 0\*0 + 1\*0 + 0\*0 + 0\*0 + 0\*2 = 0

**Example 3:**

Input: nums1 = [0,1,0,0,2,0,0], nums2 = [1,0,0,0,3,0,4]  
Output: 6

**Constraints:**

* n == nums1.length == nums2.length
* 1 <= n <= 10^5
* 0 <= nums1[i], nums2[i] <= 100

# **880. Decoded String at Index**

https://leetcode.com/problems/decoded-string-at-index/description/

You are given an encoded string s. To decode the string to a tape, the encoded string is read one character at a time and the following steps are taken:

* If the character read is a letter, that letter is written onto the tape.
* If the character read is a digit d, the entire current tape is repeatedly written d - 1 more times in total.

Given an integer k, return *the* kth *letter (****1-indexed)*** *in the decoded string*.

**Example 1:**

Input: s = "leet2code3", k = 10  
Output: "o"  
Explanation: The decoded string is "leetleetcodeleetleetcodeleetleetcode".  
The 10th letter in the string is "o".

**Example 2:**

Input: s = "ha22", k = 5  
Output: "h"  
Explanation: The decoded string is "hahahaha".  
The 5th letter is "h".

**Example 3:**

Input: s = "a2345678999999999999999", k = 1  
Output: "a"  
Explanation: The decoded string is "a" repeated 8301530446056247680 times.  
The 1st letter is "a".

**Constraints:**

* 2 <= s.length <= 100
* s consists of lowercase English letters and digits 2 through 9.
* s starts with a letter.
* 1 <= k <= 109
* It is guaranteed that k is less than or equal to the length of the decoded string.
* The decoded string is guaranteed to have less than 263 letters.

# **680. Valid Palindrome II**

https://leetcode.com/problems/valid-palindrome-ii/description/

Given a string s, return true *if the* s *can be palindrome after deleting* ***at most one*** *character from it*.

**Example 1:**

Input: s = "aba"  
Output: true

**Example 2:**

Input: s = "abca"  
Output: true  
Explanation: You could delete the character 'c'.

**Example 3:**

Input: s = "abc"  
Output: false

**Constraints:**

* 1 <= s.length <= 105
* s consists of lowercase English letters.

# **22. Generate Parentheses**

https://leetcode.com/problems/generate-parentheses/description/

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

**Example 1:**

Input: n = 3  
Output: ["((()))","(()())","(())()","()(())","()()()"]

**Example 2:**

Input: n = 1  
Output: ["()"]

**Constraints:**

* 1 <= n <= 8

# **2468. Split Message Based on Limit**

https://leetcode.com/problems/split-message-based-on-limit/description/

You are given a string, message, and a positive integer, limit.

You must **split** message into one or more **parts** based on limit. Each resulting part should have the suffix "<a/b>", where "b" is to be **replaced** with the total number of parts and "a" is to be **replaced** with the index of the part, starting from 1 and going up to b. Additionally, the length of each resulting part (including its suffix) should be **equal** to limit, except for the last part whose length can be **at most** limit.

The resulting parts should be formed such that when their suffixes are removed and they are all concatenated **in order**, they should be equal to message. Also, the result should contain as few parts as possible.

Return *the parts* message *would be split into as an array of strings*. If it is impossible to split message as required, return *an empty array*.

**Example 1:**

Input: message = "this is really a very awesome message", limit = 9  
Output: ["thi<1/14>","s i<2/14>","s r<3/14>","eal<4/14>","ly <5/14>","a v<6/14>","ery<7/14>"," aw<8/14>","eso<9/14>","me<10/14>"," m<11/14>","es<12/14>","sa<13/14>","ge<14/14>"]  
Explanation:  
The first 9 parts take 3 characters each from the beginning of message.  
The next 5 parts take 2 characters each to finish splitting message.   
In this example, each part, including the last, has length 9.   
It can be shown it is not possible to split message into less than 14 parts.

**Example 2:**

Input: message = "short message", limit = 15  
Output: ["short mess<1/2>","age<2/2>"]  
Explanation:  
Under the given constraints, the string can be split into two parts:   
- The first part comprises of the first 10 characters, and has a length 15.  
- The next part comprises of the last 3 characters, and has a length 8.

**Constraints:**

* 1 <= message.length <= 104
* message consists only of lowercase English letters and ' '.
* 1 <= limit <= 104

# **2038. Remove Colored Pieces if Both Neighbors are the Same Color**

https://leetcode.com/problems/remove-colored-pieces-if-both-neighbors-are-the-same-color/description/

There are n pieces arranged in a line, and each piece is colored either by 'A' or by 'B'. You are given a string colors of length n where colors[i] is the color of the ith piece.

Alice and Bob are playing a game where they take **alternating turns** removing pieces from the line. In this game, Alice moves **first**.

* Alice is only allowed to remove a piece colored 'A' if **both its neighbors** are also colored 'A'. She is **not allowed** to remove pieces that are colored 'B'.
* Bob is only allowed to remove a piece colored 'B' if **both its neighbors** are also colored 'B'. He is **not allowed** to remove pieces that are colored 'A'.
* Alice and Bob **cannot** remove pieces from the edge of the line.
* If a player cannot make a move on their turn, that player **loses** and the other player **wins**.

Assuming Alice and Bob play optimally, return true *if Alice wins, or return* false *if Bob wins*.

**Example 1:**

Input: colors = "AAABABB"  
Output: true  
Explanation:  
AAABABB -> AABABB  
Alice moves first.  
She removes the second 'A' from the left since that is the only 'A' whose neighbors are both 'A'.  
  
Now it's Bob's turn.  
Bob cannot make a move on his turn since there are no 'B's whose neighbors are both 'B'.  
Thus, Alice wins, so return true.

**Example 2:**

Input: colors = "AA"  
Output: false  
Explanation:  
Alice has her turn first.  
There are only two 'A's and both are on the edge of the line, so she cannot move on her turn.  
Thus, Bob wins, so return false.

**Example 3:**

Input: colors = "ABBBBBBBAAA"  
Output: false  
Explanation:  
ABBBBBBBAAA -> ABBBBBBBAA  
Alice moves first.  
Her only option is to remove the second to last 'A' from the right.  
  
ABBBBBBBAA -> ABBBBBBAA  
Next is Bob's turn.  
He has many options for which 'B' piece to remove. He can pick any.  
  
On Alice's second turn, she has no more pieces that she can remove.  
Thus, Bob wins, so return false.

**Constraints:**

* 1 <= colors.length <= 105
* colors consists of only the letters 'A' and 'B'

# **2009. Minimum Number of Operations to Make Array Continuous**

https://leetcode.com/problems/minimum-number-of-operations-to-make-array-continuous/description/

You are given an integer array nums. In one operation, you can replace **any** element in nums with **any** integer.

nums is considered **continuous** if both of the following conditions are fulfilled:

* All elements in nums are **unique**.
* The difference between the **maximum** element and the **minimum** element in nums equals nums.length - 1.

For example, nums = [4, 2, 5, 3] is **continuous**, but nums = [1, 2, 3, 5, 6] is **not continuous**.

Return *the* ***minimum*** *number of operations to make* nums ***continuous***.

**Example 1:**

Input: nums = [4,2,5,3]  
Output: 0  
Explanation: nums is already continuous.

**Example 2:**

Input: nums = [1,2,3,5,6]  
Output: 1  
Explanation: One possible solution is to change the last element to 4.  
The resulting array is [1,2,3,5,4], which is continuous.

**Example 3:**

Input: nums = [1,10,100,1000]  
Output: 3  
Explanation: One possible solution is to:  
- Change the second element to 2.  
- Change the third element to 3.  
- Change the fourth element to 4.  
The resulting array is [1,2,3,4], which is continuous.

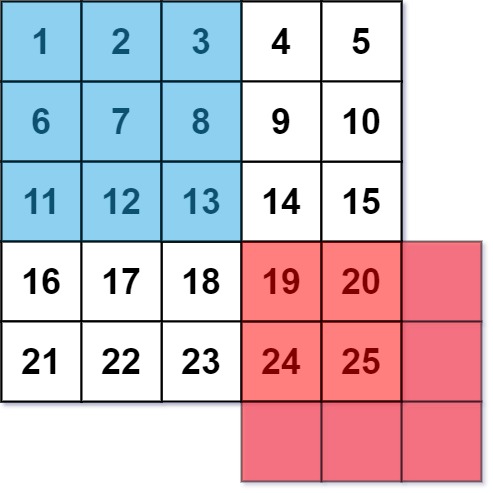
**Constraints:**

* 1 <= nums.length <= 105
* 1 <= nums[i] <= 109

# **661. Image Smoother**

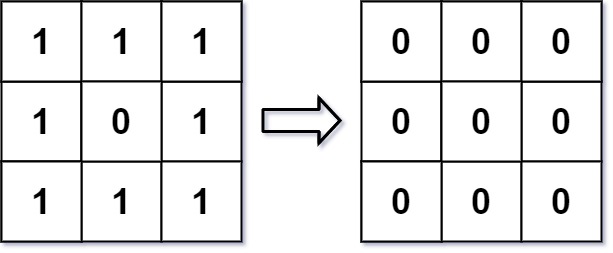
https://leetcode.com/problems/image-smoother/description/

An **image smoother** is a filter of the size 3 x 3 that can be applied to each cell of an image by rounding down the average of the cell and the eight surrounding cells (i.e., the average of the nine cells in the blue smoother). If one or more of the surrounding cells of a cell is not present, we do not consider it in the average (i.e., the average of the four cells in the red smoother).



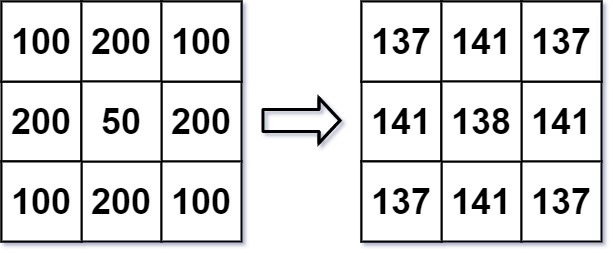
Given an m x n integer matrix img representing the grayscale of an image, return *the image after applying the smoother on each cell of it*.

**Example 1:**



Input: img = [[1,1,1],[1,0,1],[1,1,1]]  
Output: [[0,0,0],[0,0,0],[0,0,0]]  
Explanation:  
For the points (0,0), (0,2), (2,0), (2,2): floor(3/4) = floor(0.75) = 0  
For the points (0,1), (1,0), (1,2), (2,1): floor(5/6) = floor(0.83333333) = 0  
For the point (1,1): floor(8/9) = floor(0.88888889) = 0

**Example 2:**



Input: img = [[100,200,100],[200,50,200],[100,200,100]]  
Output: [[137,141,137],[141,138,141],[137,141,137]]  
Explanation:  
For the points (0,0), (0,2), (2,0), (2,2): floor((100+200+200+50)/4) = floor(137.5) = 137  
For the points (0,1), (1,0), (1,2), (2,1): floor((200+200+50+200+100+100)/6) = floor(141.666667) = 141  
For the point (1,1): floor((50+200+200+200+200+100+100+100+100)/9) = floor(138.888889) = 138

**Constraints:**

* m == img.length
* n == img[i].length
* 1 <= m, n <= 200
* 0 <= img[i][j] <= 255

# **380. Insert Delete GetRandom O(1)**

https://leetcode.com/problems/insert-delete-getrandom-o1/description/

Implement the RandomizedSet class:

* RandomizedSet() Initializes the RandomizedSet object.
* bool insert(int val) Inserts an item val into the set if not present. Returns true if the item was not present, false otherwise.
* bool remove(int val) Removes an item val from the set if present. Returns true if the item was present, false otherwise.
* int getRandom() Returns a random element from the current set of elements (it's guaranteed that at least one element exists when this method is called). Each element must have the **same probability** of being returned.

You must implement the functions of the class such that each function works in **average** O(1) time complexity.

**Example 1:**

Input  
["RandomizedSet", "insert", "remove", "insert", "getRandom", "remove", "insert", "getRandom"]  
[[], [1], [2], [2], [], [1], [2], []]  
Output  
[null, true, false, true, 2, true, false, 2]  
  
Explanation  
RandomizedSet randomizedSet = new RandomizedSet();  
randomizedSet.insert(1); // Inserts 1 to the set. Returns true as 1 was inserted successfully.  
randomizedSet.remove(2); // Returns false as 2 does not exist in the set.  
randomizedSet.insert(2); // Inserts 2 to the set, returns true. Set now contains [1,2].  
randomizedSet.getRandom(); // getRandom() should return either 1 or 2 randomly.  
randomizedSet.remove(1); // Removes 1 from the set, returns true. Set now contains [2].  
randomizedSet.insert(2); // 2 was already in the set, so return false.  
randomizedSet.getRandom(); // Since 2 is the only number in the set, getRandom() will always return 2.

**Constraints:**

* -231 <= val <= 231 - 1
* At most 2 \* 105 calls will be made to insert, remove, and getRandom.
* There will be **at least one** element in the data structure when getRandom is called.

# **169. Majority Element**

https://leetcode.com/problems/majority-element/description/

Given an array nums of size n, return *the majority element*.

The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

**Example 1:**

Input: nums = [3,2,3]  
Output: 3

**Example 2:**

Input: nums = [2,2,1,1,1,2,2]  
Output: 2

**Constraints:**

* n == nums.length
* 1 <= n <= 5 \* 104
* -109 <= nums[i] <= 109

**Follow-up:** Could you solve the problem in linear time and in O(1) space?

# **2235. Add Two Integers**

https://leetcode.com/problems/add-two-integers/description/ Given two integers num1 and num2, return *the* ***sum*** *of the two integers*.

**Example 1:**

Input: num1 = 12, num2 = 5  
Output: 17  
Explanation: num1 is 12, num2 is 5, and their sum is 12 + 5 = 17, so 17 is returned.

**Example 2:**

Input: num1 = -10, num2 = 4  
Output: -6  
Explanation: num1 + num2 = -6, so -6 is returned.

**Constraints:**

* -100 <= num1, num2 <= 100

# **343. Integer Break**

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

Given an integer n, break it into the sum of k **positive integers**, where k >= 2, and maximize the product of those integers.

Return *the maximum product you can get*.

**Example 1:**

Input: n = 2  
Output: 1  
Explanation: 2 = 1 + 1, 1 × 1 = 1.

**Example 2:**

Input: n = 10  
Output: 36  
Explanation: 10 = 3 + 3 + 4, 3 × 3 × 4 = 36.

**Constraints:**

* 2 <= n <= 58

# **176. Second Highest Salary**

https://leetcode.com/problems/second-highest-salary/description/

Table: Employee

+-------------+------+  
| Column Name | Type |  
+-------------+------+  
| id | int |  
| salary | int |  
+-------------+------+  
id is the primary key (column with unique values) for this table.  
Each row of this table contains information about the salary of an employee.

Write a solution to find the second highest salary from the Employee table. If there is no second highest salary, return null (return None in Pandas).

The result format is in the following example.

**Example 1:**

Input:   
Employee table:  
+----+--------+  
| id | salary |  
+----+--------+  
| 1 | 100 |  
| 2 | 200 |  
| 3 | 300 |  
+----+--------+  
Output:   
+---------------------+  
| SecondHighestSalary |  
+---------------------+  
| 200 |  
+---------------------+

**Example 2:**

Input:   
Employee table:  
+----+--------+  
| id | salary |  
+----+--------+  
| 1 | 100 |  
+----+--------+  
Output:   
+---------------------+  
| SecondHighestSalary |  
+---------------------+  
| null |  
+---------------------+

# **53. Maximum Subarray**

https://leetcode.com/problems/maximum-subarray/description/

Given an integer array nums, find the

subarray

with the largest sum, and return *its sum*.

**Example 1:**

Input: nums = [-2,1,-3,4,-1,2,1,-5,4]  
Output: 6  
Explanation: The subarray [4,-1,2,1] has the largest sum 6.

**Example 2:**

Input: nums = [1]  
Output: 1  
Explanation: The subarray [1] has the largest sum 1.

**Example 3:**

Input: nums = [5,4,-1,7,8]  
Output: 23  
Explanation: The subarray [5,4,-1,7,8] has the largest sum 23.

**Constraints:**

* 1 <= nums.length <= 105
* -104 <= nums[i] <= 104

**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.

# **54. Spiral Matrix**

https://leetcode.com/problems/spiral-matrix/description/

Given an m x n matrix, return *all elements of the* matrix *in spiral order*.

**Example 1:**



Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]  
Output: [1,2,3,6,9,8,7,4,5]

**Example 2:**



Input: matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]  
Output: [1,2,3,4,8,12,11,10,9,5,6,7]

**Constraints:**

* m == matrix.length
* n == matrix[i].length
* 1 <= m, n <= 10
* -100 <= matrix[i][j] <= 100

# **253. Meeting Rooms II**

https://leetcode.com/problems/meeting-rooms-ii/description/

Given an array of meeting time intervals intervals where intervals[i] = [starti, endi], return *the minimum number of conference rooms required*.

**Example 1:**

Input: intervals = [[0,30],[5,10],[15,20]]  
Output: 2

**Example 2:**

Input: intervals = [[7,10],[2,4]]  
Output: 1

**Constraints:**

* 1 <= intervals.length <= 104
* 0 <= starti < endi <= 106

# **31. Next Permutation**

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

A **permutation** of an array of integers is an arrangement of its members into a sequence or linear order.

* For example, for arr = [1,2,3], the following are all the permutations of arr: [1,2,3], [1,3,2], [2, 1, 3], [2, 3, 1], [3,1,2], [3,2,1].

The **next permutation** of an array of integers is the next lexicographically greater permutation of its integer. More formally, if all the permutations of the array are sorted in one container according to their lexicographical order, then the **next permutation** of that array is the permutation that follows it in the sorted container. If such arrangement is not possible, the array must be rearranged as the lowest possible order (i.e., sorted in ascending order).

* For example, the next permutation of arr = [1,2,3] is [1,3,2].
* Similarly, the next permutation of arr = [2,3,1] is [3,1,2].
* While the next permutation of arr = [3,2,1] is [1,2,3] because [3,2,1] does not have a lexicographical larger rearrangement.

Given an array of integers nums, *find the next permutation of* nums.

The replacement must be [**in place**](http://en.wikipedia.org/wiki/In-place_algorithm) and use only constant extra memory.

**Example 1:**

Input: nums = [1,2,3]  
Output: [1,3,2]

**Example 2:**

Input: nums = [3,2,1]  
Output: [1,2,3]

**Example 3:**

Input: nums = [1,1,5]  
Output: [1,5,1]

**Constraints:**

* 1 <= nums.length <= 100
* 0 <= nums[i] <= 100

# **48. Rotate Image**

https://leetcode.com/problems/rotate-image/description/

You are given an n x n 2D matrix representing an image, rotate the image by **90** degrees (clockwise).

You have to rotate the image [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm), which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

**Example 1:**



Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]  
Output: [[7,4,1],[8,5,2],[9,6,3]]

**Example 2:**



Input: matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,14,12,16]]  
Output: [[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]

**Constraints:**

* n == matrix.length == matrix[i].length
* 1 <= n <= 20
* -1000 <= matrix[i][j] <= 1000

# **273. Integer to English Words**

https://leetcode.com/problems/integer-to-english-words/description/

Convert a non-negative integer num to its English words representation.

**Example 1:**

Input: num = 123  
Output: "One Hundred Twenty Three"

**Example 2:**

Input: num = 12345  
Output: "Twelve Thousand Three Hundred Forty Five"

**Example 3:**

Input: num = 1234567  
Output: "One Million Two Hundred Thirty Four Thousand Five Hundred Sixty Seven"

**Constraints:**

* 0 <= num <= 231 - 1

# **1347. Minimum Number of Steps to Make Two Strings Anagram**

https://leetcode.com/problems/minimum-number-of-steps-to-make-two-strings-anagram/description/

You are given two strings of the same length s and t. In one step you can choose **any character** of t and replace it with **another character**.

Return *the minimum number of steps* to make t an anagram of s.

An **Anagram** of a string is a string that contains the same characters with a different (or the same) ordering.

**Example 1:**

Input: s = "bab", t = "aba"  
Output: 1  
Explanation: Replace the first 'a' in t with b, t = "bba" which is anagram of s.

**Example 2:**

Input: s = "leetcode", t = "practice"  
Output: 5  
Explanation: Replace 'p', 'r', 'a', 'i' and 'c' from t with proper characters to make t anagram of s.

**Example 3:**

Input: s = "anagram", t = "mangaar"  
Output: 0  
Explanation: "anagram" and "mangaar" are anagrams.

**Constraints:**

* 1 <= s.length <= 5 \* 104
* s.length == t.length
* s and t consist of lowercase English letters only.

# **1041. Robot Bounded In Circle**

https://leetcode.com/problems/robot-bounded-in-circle/description/

On an infinite plane, a robot initially stands at (0, 0) and faces north. Note that:

* The **north direction** is the positive direction of the y-axis.
* The **south direction** is the negative direction of the y-axis.
* The **east direction** is the positive direction of the x-axis.
* The **west direction** is the negative direction of the x-axis.

The robot can receive one of three instructions:

* "G": go straight 1 unit.
* "L": turn 90 degrees to the left (i.e., anti-clockwise direction).
* "R": turn 90 degrees to the right (i.e., clockwise direction).

The robot performs the instructions given in order, and repeats them forever.

Return true if and only if there exists a circle in the plane such that the robot never leaves the circle.

**Example 1:**

Input: instructions = "GGLLGG"  
Output: true  
Explanation: The robot is initially at (0, 0) facing the north direction.  
"G": move one step. Position: (0, 1). Direction: North.  
"G": move one step. Position: (0, 2). Direction: North.  
"L": turn 90 degrees anti-clockwise. Position: (0, 2). Direction: West.  
"L": turn 90 degrees anti-clockwise. Position: (0, 2). Direction: South.  
"G": move one step. Position: (0, 1). Direction: South.  
"G": move one step. Position: (0, 0). Direction: South.  
Repeating the instructions, the robot goes into the cycle: (0, 0) --> (0, 1) --> (0, 2) --> (0, 1) --> (0, 0).  
Based on that, we return true.

**Example 2:**

Input: instructions = "GG"  
Output: false  
Explanation: The robot is initially at (0, 0) facing the north direction.  
"G": move one step. Position: (0, 1). Direction: North.  
"G": move one step. Position: (0, 2). Direction: North.  
Repeating the instructions, keeps advancing in the north direction and does not go into cycles.  
Based on that, we return false.

**Example 3:**

Input: instructions = "GL"  
Output: true  
Explanation: The robot is initially at (0, 0) facing the north direction.  
"G": move one step. Position: (0, 1). Direction: North.  
"L": turn 90 degrees anti-clockwise. Position: (0, 1). Direction: West.  
"G": move one step. Position: (-1, 1). Direction: West.  
"L": turn 90 degrees anti-clockwise. Position: (-1, 1). Direction: South.  
"G": move one step. Position: (-1, 0). Direction: South.  
"L": turn 90 degrees anti-clockwise. Position: (-1, 0). Direction: East.  
"G": move one step. Position: (0, 0). Direction: East.  
"L": turn 90 degrees anti-clockwise. Position: (0, 0). Direction: North.  
Repeating the instructions, the robot goes into the cycle: (0, 0) --> (0, 1) --> (-1, 1) --> (-1, 0) --> (0, 0).  
Based on that, we return true.

**Constraints:**

* 1 <= instructions.length <= 100
* instructions[i] is 'G', 'L' or, 'R'.

# **1762. Buildings With an Ocean View**

https://leetcode.com/problems/buildings-with-an-ocean-view/description/

There are n buildings in a line. You are given an integer array heights of size n that represents the heights of the buildings in the line.

The ocean is to the right of the buildings. A building has an ocean view if the building can see the ocean without obstructions. Formally, a building has an ocean view if all the buildings to its right have a **smaller** height.

Return a list of indices **(0-indexed)** of buildings that have an ocean view, sorted in increasing order.

**Example 1:**

Input: heights = [4,2,3,1]  
Output: [0,2,3]  
Explanation: Building 1 (0-indexed) does not have an ocean view because building 2 is taller.

**Example 2:**

Input: heights = [4,3,2,1]  
Output: [0,1,2,3]  
Explanation: All the buildings have an ocean view.

**Example 3:**

Input: heights = [1,3,2,4]  
Output: [3]  
Explanation: Only building 3 has an ocean view.

**Constraints:**

* 1 <= heights.length <= 105
* 1 <= heights[i] <= 109

# **50. Pow(x, n)**

https://leetcode.com/problems/powx-n/description/

Implement [pow(x, n)](http://www.cplusplus.com/reference/valarray/pow/), which calculates x raised to the power n (i.e., xn).

**Example 1:**

Input: x = 2.00000, n = 10  
Output: 1024.00000

**Example 2:**

Input: x = 2.10000, n = 3  
Output: 9.26100

**Example 3:**

Input: x = 2.00000, n = -2  
Output: 0.25000  
Explanation: 2-2 = 1/22 = 1/4 = 0.25

**Constraints:**

* -100.0 < x < 100.0
* -231 <= n <= 231-1
* n is an integer.
* Either x is not zero or n > 0.
* -104 <= xn <= 104

# **1207. Unique Number of Occurrences**

https://leetcode.com/problems/unique-number-of-occurrences/description/

Given an array of integers arr, return true *if the number of occurrences of each value in the array is* ***unique*** *or* false *otherwise*.

**Example 1:**

Input: arr = [1,2,2,1,1,3]  
Output: true  
Explanation: The value 1 has 3 occurrences, 2 has 2 and 3 has 1. No two values have the same number of occurrences.

**Example 2:**

Input: arr = [1,2]  
Output: false

**Example 3:**

Input: arr = [-3,0,1,-3,1,1,1,-3,10,0]  
Output: true

**Constraints:**

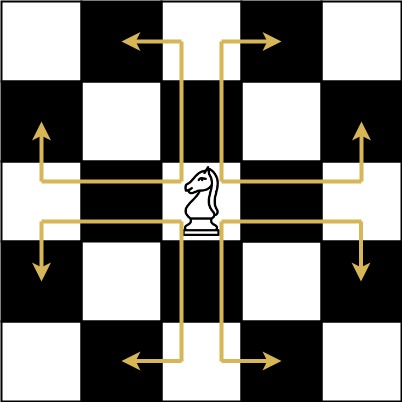
* 1 <= arr.length <= 1000
* -1000 <= arr[i] <= 1000

# **935. Knight Dialer**

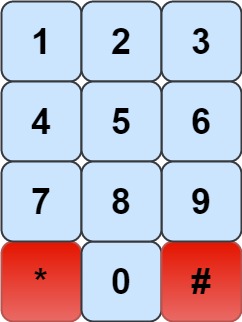
https://leetcode.com/problems/knight-dialer/description/

The chess knight has a **unique movement**, it may move two squares vertically and one square horizontally, or two squares horizontally and one square vertically (with both forming the shape of an **L**). The possible movements of chess knight are shown in this diagram:

A chess knight can move as indicated in the chess diagram below:



We have a chess knight and a phone pad as shown below, the knight **can only stand on a numeric cell** (i.e. blue cell).



Given an integer n, return how many distinct phone numbers of length n we can dial.

You are allowed to place the knight **on any numeric cell** initially and then you should perform n - 1 jumps to dial a number of length n. All jumps should be **valid** knight jumps.

As the answer may be very large, **return the answer modulo** 109 + 7.

**Example 1:**

Input: n = 1  
Output: 10  
Explanation: We need to dial a number of length 1, so placing the knight over any numeric cell of the 10 cells is sufficient.

**Example 2:**

Input: n = 2  
Output: 20  
Explanation: All the valid number we can dial are [04, 06, 16, 18, 27, 29, 34, 38, 40, 43, 49, 60, 61, 67, 72, 76, 81, 83, 92, 94]

**Example 3:**

Input: n = 3131  
Output: 136006598  
Explanation: Please take care of the mod.

**Constraints:**

* 1 <= n <= 5000

# **217. Contains Duplicate**

https://leetcode.com/problems/contains-duplicate/description/

Given an integer array nums, return true if any value appears **at least twice** in the array, and return false if every element is distinct.

**Example 1:**

Input: nums = [1,2,3,1]  
Output: true

**Example 2:**

Input: nums = [1,2,3,4]  
Output: false

**Example 3:**

Input: nums = [1,1,1,3,3,4,3,2,4,2]  
Output: true

**Constraints:**

* 1 <= nums.length <= 105
* -109 <= nums[i] <= 109

# **767. Reorganize String**

https://leetcode.com/problems/reorganize-string/description/

Given a string s, rearrange the characters of s so that any two adjacent characters are not the same.

Return *any possible rearrangement of* s *or return* "" *if not possible*.

**Example 1:**

Input: s = "aab"  
Output: "aba"

**Example 2:**

Input: s = "aaab"  
Output: ""

**Constraints:**

* 1 <= s.length <= 500
* s consists of lowercase English letters.

# **162. Find Peak Element**

https://leetcode.com/problems/find-peak-element/description/

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

Given a **0-indexed** integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to **any of the peaks**.

You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array.

You must write an algorithm that runs in O(log n) time.

**Example 1:**

Input: nums = [1,2,3,1]  
Output: 2  
Explanation: 3 is a peak element and your function should return the index number 2.

**Example 2:**

Input: nums = [1,2,1,3,5,6,4]  
Output: 5  
Explanation: Your function can return either index number 1 where the peak element is 2, or index number 5 where the peak element is 6.

**Constraints:**

* 1 <= nums.length <= 1000
* -231 <= nums[i] <= 231 - 1
* nums[i] != nums[i + 1] for all valid i.

# **21. Merge Two Sorted Lists**

https://leetcode.com/problems/merge-two-sorted-lists/description/

You are given the heads of two sorted linked lists list1 and list2.

Merge the two lists into one **sorted** list. The list should be made by splicing together the nodes of the first two lists.

Return *the head of the merged linked list*.

**Example 1:**



Input: list1 = [1,2,4], list2 = [1,3,4]  
Output: [1,1,2,3,4,4]

**Example 2:**

Input: list1 = [], list2 = []  
Output: []

**Example 3:**

Input: list1 = [], list2 = [0]  
Output: [0]

**Constraints:**

* The number of nodes in both lists is in the range [0, 50].
* -100 <= Node.val <= 100
* Both list1 and list2 are sorted in **non-decreasing** order.

# **368. Largest Divisible Subset**

https://leetcode.com/problems/largest-divisible-subset/description/

Given a set of **distinct** positive integers nums, return the largest subset answer such that every pair (answer[i], answer[j]) of elements in this subset satisfies:

* answer[i] % answer[j] == 0, or
* answer[j] % answer[i] == 0

If there are multiple solutions, return any of them.

**Example 1:**

Input: nums = [1,2,3]  
Output: [1,2]  
Explanation: [1,3] is also accepted.

**Example 2:**

Input: nums = [1,2,4,8]  
Output: [1,2,4,8]

**Constraints:**

* 1 <= nums.length <= 1000
* 1 <= nums[i] <= 2 \* 109
* All the integers in nums are **unique**.

# **231. Power of Two**

https://leetcode.com/problems/power-of-two/description/

Given an integer n, return *true if it is a power of two. Otherwise, return false*.

An integer n is a power of two, if there exists an integer x such that n == 2x.

**Example 1:**

Input: n = 1  
Output: true  
Explanation: 20 = 1

**Example 2:**

Input: n = 16  
Output: true  
Explanation: 24 = 16

**Example 3:**

Input: n = 3  
Output: false

**Constraints:**

* -231 <= n <= 231 - 1

**Follow up:** Could you solve it without loops/recursion?

# **2402. Meeting Rooms III**

https://leetcode.com/problems/meeting-rooms-iii/description/

You are given an integer n. There are n rooms numbered from 0 to n - 1.

You are given a 2D integer array meetings where meetings[i] = [starti, endi] means that a meeting will be held during the **half-closed** time interval [starti, endi). All the values of starti are **unique**.

Meetings are allocated to rooms in the following manner:

1. Each meeting will take place in the unused room with the **lowest** number.
2. If there are no available rooms, the meeting will be delayed until a room becomes free. The delayed meeting should have the **same** duration as the original meeting.
3. When a room becomes unused, meetings that have an earlier original **start** time should be given the room.

Return *the* ***number*** *of the room that held the most meetings.* If there are multiple rooms, return *the room with the* ***lowest*** *number.*

A **half-closed interval** [a, b) is the interval between a and b **including** a and **not including** b.

**Example 1:**

Input: n = 2, meetings = [[0,10],[1,5],[2,7],[3,4]]  
Output: 0  
Explanation:  
- At time 0, both rooms are not being used. The first meeting starts in room 0.  
- At time 1, only room 1 is not being used. The second meeting starts in room 1.  
- At time 2, both rooms are being used. The third meeting is delayed.  
- At time 3, both rooms are being used. The fourth meeting is delayed.  
- At time 5, the meeting in room 1 finishes. The third meeting starts in room 1 for the time period [5,10).  
- At time 10, the meetings in both rooms finish. The fourth meeting starts in room 0 for the time period [10,11).  
Both rooms 0 and 1 held 2 meetings, so we return 0.

**Example 2:**

Input: n = 3, meetings = [[1,20],[2,10],[3,5],[4,9],[6,8]]  
Output: 1  
Explanation:  
- At time 1, all three rooms are not being used. The first meeting starts in room 0.  
- At time 2, rooms 1 and 2 are not being used. The second meeting starts in room 1.  
- At time 3, only room 2 is not being used. The third meeting starts in room 2.  
- At time 4, all three rooms are being used. The fourth meeting is delayed.  
- At time 5, the meeting in room 2 finishes. The fourth meeting starts in room 2 for the time period [5,10).  
- At time 6, all three rooms are being used. The fifth meeting is delayed.  
- At time 10, the meetings in rooms 1 and 2 finish. The fifth meeting starts in room 1 for the time period [10,12).  
Room 0 held 1 meeting while rooms 1 and 2 each held 2 meetings, so we return 1.

**Constraints:**

* 1 <= n <= 100
* 1 <= meetings.length <= 105
* meetings[i].length == 2
* 0 <= starti < endi <= 5 \* 105
* All the values of starti are **unique**.

# **279. Perfect Squares**

https://leetcode.com/problems/perfect-squares/description/

Given an integer n, return *the least number of perfect square numbers that sum to* n.

A **perfect square** is an integer that is the square of an integer; in other words, it is the product of some integer with itself. For example, 1, 4, 9, and 16 are perfect squares while 3 and 11 are not.

**Example 1:**

Input: n = 12  
Output: 3  
Explanation: 12 = 4 + 4 + 4.

**Example 2:**

Input: n = 13  
Output: 2  
Explanation: 13 = 4 + 9.

**Constraints:**

* 1 <= n <= 104

# **2877. Create a DataFrame from List**

https://leetcode.com/problems/create-a-dataframe-from-list/description/

Write a solution to **create** a DataFrame from a 2D list called student\_data. This 2D list contains the IDs and ages of some students.

The DataFrame should have two columns, student\_id and age, and be in the same order as the original 2D list.

The result format is in the following example.

**Example 1:**

Input:  
student\_data:  
[  
 [1, 15],  
 [2, 11],  
 [3, 11],  
 [4, 20]  
]  
Output:  
+------------+-----+  
| student\_id | age |  
+------------+-----+  
| 1 | 15 |  
| 2 | 11 |  
| 3 | 11 |  
| 4 | 20 |  
+------------+-----+  
Explanation:  
A DataFrame was created on top of student\_data, with two columns named student\_id and age.

# **1169. Invalid Transactions**

https://leetcode.com/problems/invalid-transactions/description/

A transaction is possibly invalid if:

* the amount exceeds $1000, or;
* if it occurs within (and including) 60 minutes of another transaction with the **same name** in a **different city**.

You are given an array of strings transaction where transactions[i] consists of comma-separated values representing the name, time (in minutes), amount, and city of the transaction.

Return a list of transactions that are possibly invalid. You may return the answer in **any order**.

**Example 1:**

Input: transactions = ["alice,20,800,mtv","alice,50,100,beijing"]  
Output: ["alice,20,800,mtv","alice,50,100,beijing"]  
Explanation: The first transaction is invalid because the second transaction occurs within a difference of 60 minutes, have the same name and is in a different city. Similarly the second one is invalid too.

**Example 2:**

Input: transactions = ["alice,20,800,mtv","alice,50,1200,mtv"]  
Output: ["alice,50,1200,mtv"]

**Example 3:**

Input: transactions = ["alice,20,800,mtv","bob,50,1200,mtv"]  
Output: ["bob,50,1200,mtv"]

**Constraints:**

* transactions.length <= 1000
* Each transactions[i] takes the form "{name},{time},{amount},{city}"
* Each {name} and {city} consist of lowercase English letters, and have lengths between 1 and 10.
* Each {time} consist of digits, and represent an integer between 0 and 1000.
* Each {amount} consist of digits, and represent an integer between 0 and 2000.

# **2742. Painting the Walls**

https://leetcode.com/problems/painting-the-walls/description/

You are given two **0-indexed** integer arrays, cost and time, of size n representing the costs and the time taken to paint n different walls respectively. There are two painters available:

* A**paid painter** that paints the ith wall in time[i] units of time and takes cost[i] units of money.
* A**free painter** that paints **any** wall in 1 unit of time at a cost of 0. But the free painter can only be used if the paid painter is already **occupied**.

Return *the minimum amount of money required to paint the* n*walls.*

**Example 1:**

Input: cost = [1,2,3,2], time = [1,2,3,2]  
Output: 3  
Explanation: The walls at index 0 and 1 will be painted by the paid painter, and it will take 3 units of time; meanwhile, the free painter will paint the walls at index 2 and 3, free of cost in 2 units of time. Thus, the total cost is 1 + 2 = 3.

**Example 2:**

Input: cost = [2,3,4,2], time = [1,1,1,1]  
Output: 4  
Explanation: The walls at index 0 and 3 will be painted by the paid painter, and it will take 2 units of time; meanwhile, the free painter will paint the walls at index 1 and 2, free of cost in 2 units of time. Thus, the total cost is 2 + 2 = 4.

**Constraints:**

* 1 <= cost.length <= 500
* cost.length == time.length
* 1 <= cost[i] <= 106
* 1 <= time[i] <= 500

# **236. Lowest Common Ancestor of a Binary Tree**

https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/description/

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**).”

**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.

**Example 3:**

Input: root = [1,2], p = 1, q = 2  
Output: 1

**Constraints:**

* The number of nodes in the tree is in the range [2, 105].
* -109 <= Node.val <= 109
* All Node.val are **unique**.
* p != q
* p and q will exist in the tree.

# **1287. Element Appearing More Than 25% In Sorted Array**

https://leetcode.com/problems/element-appearing-more-than-25-in-sorted-array/description/

Given an integer array **sorted** in non-decreasing order, there is exactly one integer in the array that occurs more than 25% of the time, return that integer.

**Example 1:**

Input: arr = [1,2,2,6,6,6,6,7,10]  
Output: 6

**Example 2:**

Input: arr = [1,1]  
Output: 1

**Constraints:**

* 1 <= arr.length <= 104
* 0 <= arr[i] <= 105

# **446. Arithmetic Slices II - Subsequence**

https://leetcode.com/problems/arithmetic-slices-ii-subsequence/description/

Given an integer array nums, return *the number of all the* ***arithmetic subsequences*** *of* nums.

A sequence of numbers is called arithmetic if it consists of **at least three elements** and if the difference between any two consecutive elements is the same.

* For example, [1, 3, 5, 7, 9], [7, 7, 7, 7], and [3, -1, -5, -9] are arithmetic sequences.
* For example, [1, 1, 2, 5, 7] is not an arithmetic sequence.

A **subsequence** of an array is a sequence that can be formed by removing some elements (possibly none) of the array.

* For example, [2,5,10] is a subsequence of [1,2,1,**2**,4,1,**5**,**10**].

The test cases are generated so that the answer fits in **32-bit** integer.

**Example 1:**

Input: nums = [2,4,6,8,10]  
Output: 7  
Explanation: All arithmetic subsequence slices are:  
[2,4,6]  
[4,6,8]  
[6,8,10]  
[2,4,6,8]  
[4,6,8,10]  
[2,4,6,8,10]  
[2,6,10]

**Example 2:**

Input: nums = [7,7,7,7,7]  
Output: 16  
Explanation: Any subsequence of this array is arithmetic.

**Constraints:**

* 1  <= nums.length <= 1000
* -231 <= nums[i] <= 231 - 1

# **867. Transpose Matrix**

https://leetcode.com/problems/transpose-matrix/description/

Given a 2D integer array matrix, return *the* ***transpose*** *of* matrix.

The **transpose** of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.



**Example 1:**

Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]  
Output: [[1,4,7],[2,5,8],[3,6,9]]

**Example 2:**

Input: matrix = [[1,2,3],[4,5,6]]  
Output: [[1,4],[2,5],[3,6]]

**Constraints:**

* m == matrix.length
* n == matrix[i].length
* 1 <= m, n <= 1000
* 1 <= m \* n <= 105
* -109 <= matrix[i][j] <= 109

# **347. Top K Frequent Elements**

https://leetcode.com/problems/top-k-frequent-elements/description/

Given an integer array nums and an integer k, return *the* k *most frequent elements*. You may return the answer in **any order**.

**Example 1:**

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

**Example 2:**

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

**Constraints:**

* 1 <= nums.length <= 105
* -104 <= nums[i] <= 104
* k is in the range [1, the number of unique elements in the array].
* It is **guaranteed** that the answer is **unique**.

**Follow up:** Your algorithm's time complexity must be better than O(n log n), where n is the array's size.

# **1611. Minimum One Bit Operations to Make Integers Zero**

https://leetcode.com/problems/minimum-one-bit-operations-to-make-integers-zero/description/

Given an integer n, you must transform it into 0 using the following operations any number of times:

* Change the rightmost (0th) bit in the binary representation of n.
* Change the ith bit in the binary representation of n if the (i-1)th bit is set to 1 and the (i-2)th through 0th bits are set to 0.

Return *the minimum number of operations to transform* n *into* 0*.*

**Example 1:**

Input: n = 3  
Output: 2  
Explanation: The binary representation of 3 is "11".  
"11" -> "01" with the 2nd operation since the 0th bit is 1.  
"01" -> "00" with the 1st operation.

**Example 2:**

Input: n = 6  
Output: 4  
Explanation: The binary representation of 6 is "110".  
"110" -> "010" with the 2nd operation since the 1st bit is 1 and 0th through 0th bits are 0.  
"010" -> "011" with the 1st operation.  
"011" -> "001" with the 2nd operation since the 0th bit is 1.  
"001" -> "000" with the 1st operation.

**Constraints:**

* 0 <= n <= 109

# **238. Product of Array Except Self**

https://leetcode.com/problems/product-of-array-except-self/description/

Given an integer array nums, return *an array* answer *such that* answer[i] *is equal to the product of all the elements of* nums *except* nums[i].

The product of any prefix or suffix of nums is **guaranteed** to fit in a **32-bit** integer.

You must write an algorithm that runs in O(n) time and without using the division operation.

**Example 1:**

Input: nums = [1,2,3,4]  
Output: [24,12,8,6]

**Example 2:**

Input: nums = [-1,1,0,-3,3]  
Output: [0,0,9,0,0]

**Constraints:**

* 2 <= nums.length <= 105
* -30 <= nums[i] <= 30
* The product of any prefix or suffix of nums is **guaranteed** to fit in a **32-bit** integer.

**Follow up:** Can you solve the problem in O(1) extra space complexity? (The output array **does not** count as extra space for space complexity analysis.)

# **23. Merge k Sorted Lists**

https://leetcode.com/problems/merge-k-sorted-lists/description/

You are given an array of k linked-lists lists, each linked-list is sorted in ascending order.

*Merge all the linked-lists into one sorted linked-list and return it.*

**Example 1:**

Input: lists = [[1,4,5],[1,3,4],[2,6]]  
Output: [1,1,2,3,4,4,5,6]  
Explanation: The linked-lists are:  
[  
 1->4->5,  
 1->3->4,  
 2->6  
]  
merging them into one sorted list:  
1->1->2->3->4->4->5->6

**Example 2:**

Input: lists = []  
Output: []

**Example 3:**

Input: lists = [[]]  
Output: []

**Constraints:**

* k == lists.length
* 0 <= k <= 104
* 0 <= lists[i].length <= 500
* -104 <= lists[i][j] <= 104
* lists[i] is sorted in **ascending order**.
* The sum of lists[i].length will not exceed 104.

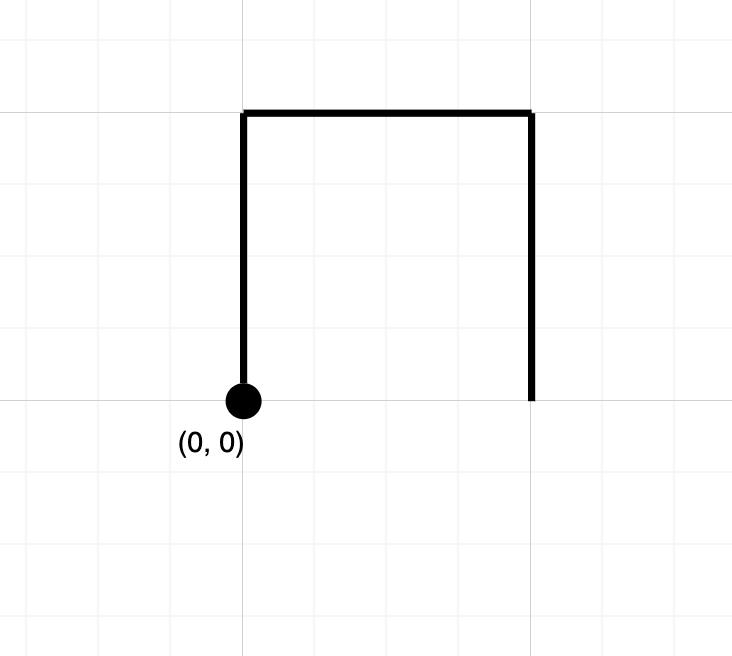
# **1496. Path Crossing**

https://leetcode.com/problems/path-crossing/description/

Given a string path, where path[i] = 'N', 'S', 'E' or 'W', each representing moving one unit north, south, east, or west, respectively. You start at the origin (0, 0) on a 2D plane and walk on the path specified by path.

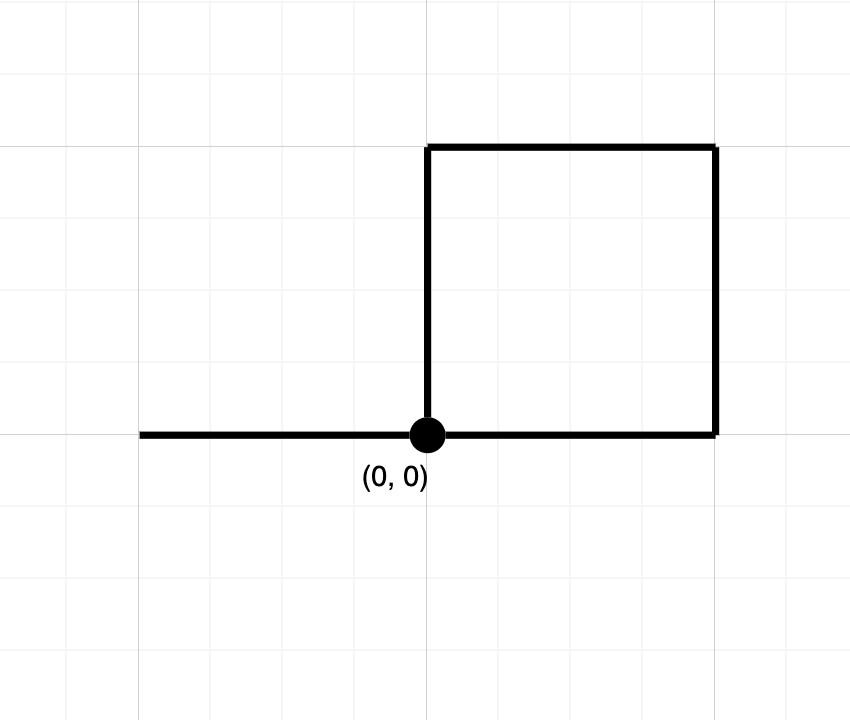
Return true *if the path crosses itself at any point, that is, if at any time you are on a location you have previously visited*. Return false otherwise.

**Example 1:**



Input: path = "NES"  
Output: false   
Explanation: Notice that the path doesn't cross any point more than once.

**Example 2:**



Input: path = "NESWW"  
Output: true  
Explanation: Notice that the path visits the origin twice.

**Constraints:**

* 1 <= path.length <= 104
* path[i] is either 'N', 'S', 'E', or 'W'.

# **283. Move Zeroes**

https://leetcode.com/problems/move-zeroes/description/

Given an integer array nums, move all 0's to the end of it while maintaining the relative order of the non-zero elements.

**Note** that you must do this in-place without making a copy of the array.

**Example 1:**

Input: nums = [0,1,0,3,12]  
Output: [1,3,12,0,0]

**Example 2:**

Input: nums = [0]  
Output: [0]

**Constraints:**

* 1 <= nums.length <= 104
* -231 <= nums[i] <= 231 - 1

**Follow up:** Could you minimize the total number of operations done?

# **128. Longest Consecutive Sequence**

https://leetcode.com/problems/longest-consecutive-sequence/description/

Given an unsorted array of integers nums, return *the length of the longest consecutive elements sequence.*

You must write an algorithm that runs in O(n) time.

**Example 1:**

Input: nums = [100,4,200,1,3,2]  
Output: 4  
Explanation: The longest consecutive elements sequence is [1, 2, 3, 4]. Therefore its length is 4.

**Example 2:**

Input: nums = [0,3,7,2,5,8,4,6,0,1]  
Output: 9

**Constraints:**

* 0 <= nums.length <= 105
* -109 <= nums[i] <= 109

# **907. Sum of Subarray Minimums**

https://leetcode.com/problems/sum-of-subarray-minimums/description/

Given an array of integers arr, find the sum of min(b), where b ranges over every (contiguous) subarray of arr. Since the answer may be large, return the answer **modulo** 109 + 7.

**Example 1:**

Input: arr = [3,1,2,4]  
Output: 17  
Explanation:   
Subarrays are [3], [1], [2], [4], [3,1], [1,2], [2,4], [3,1,2], [1,2,4], [3,1,2,4].   
Minimums are 3, 1, 2, 4, 1, 1, 2, 1, 1, 1.  
Sum is 17.

**Example 2:**

Input: arr = [11,81,94,43,3]  
Output: 444

**Constraints:**

* 1 <= arr.length <= 3 \* 104
* 1 <= arr[i] <= 3 \* 104

# **2030. Smallest K-Length Subsequence With Occurrences of a Letter**

https://leetcode.com/problems/smallest-k-length-subsequence-with-occurrences-of-a-letter/description/

You are given a string s, an integer k, a letter letter, and an integer repetition.

Return *the* ***lexicographically smallest*** *subsequence of* s *of length* k *that has the letter* letter *appear* ***at least*** repetition *times*. The test cases are generated so that the letter appears in s **at least** repetition times.

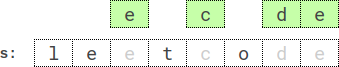
A **subsequence** is a string that can be derived from another string by deleting some or no characters without changing the order of the remaining characters.

A string a is **lexicographically smaller** than a string b if in the first position where a and b differ, string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

**Example 1:**

Input: s = "leet", k = 3, letter = "e", repetition = 1  
Output: "eet"  
Explanation: There are four subsequences of length 3 that have the letter 'e' appear at least 1 time:  
- "lee" (from "leet")  
- "let" (from "leet")  
- "let" (from "leet")  
- "eet" (from "leet")  
The lexicographically smallest subsequence among them is "eet".

**Example 2:**



Input: s = "leetcode", k = 4, letter = "e", repetition = 2  
Output: "ecde"  
Explanation: "ecde" is the lexicographically smallest subsequence of length 4 that has the letter "e" appear at least 2 times.

**Example 3:**

Input: s = "bb", k = 2, letter = "b", repetition = 2  
Output: "bb"  
Explanation: "bb" is the only subsequence of length 2 that has the letter "b" appear at least 2 times.

**Constraints:**

* 1 <= repetition <= k <= s.length <= 5 \* 104
* s consists of lowercase English letters.
* letter is a lowercase English letter, and appears in s at least repetition times.

# **2610. Convert an Array Into a 2D Array With Conditions**

https://leetcode.com/problems/convert-an-array-into-a-2d-array-with-conditions/description/

You are given an integer array nums. You need to create a 2D array from nums satisfying the following conditions:

* The 2D array should contain **only** the elements of the array nums.
* Each row in the 2D array contains **distinct** integers.
* The number of rows in the 2D array should be **minimal**.

Return *the resulting array*. If there are multiple answers, return any of them.

**Note** that the 2D array can have a different number of elements on each row.

**Example 1:**

Input: nums = [1,3,4,1,2,3,1]  
Output: [[1,3,4,2],[1,3],[1]]  
Explanation: We can create a 2D array that contains the following rows:  
- 1,3,4,2  
- 1,3  
- 1  
All elements of nums were used, and each row of the 2D array contains distinct integers, so it is a valid answer.  
It can be shown that we cannot have less than 3 rows in a valid array.

**Example 2:**

Input: nums = [1,2,3,4]  
Output: [[4,3,2,1]]  
Explanation: All elements of the array are distinct, so we can keep all of them in the first row of the 2D array.

**Constraints:**

* 1 <= nums.length <= 200
* 1 <= nums[i] <= nums.length

# **198. House Robber**

https://leetcode.com/problems/house-robber/description/

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 systems connected and **it will automatically contact the police if two adjacent houses were broken into on the same night**.

Given an integer array nums representing the amount of money of each house, return *the maximum amount of money you can rob tonight* ***without alerting the police***.

**Example 1:**

Input: nums = [1,2,3,1]  
Output: 4  
Explanation: Rob house 1 (money = 1) and then rob house 3 (money = 3).  
Total amount you can rob = 1 + 3 = 4.

**Example 2:**

Input: nums = [2,7,9,3,1]  
Output: 12  
Explanation: Rob house 1 (money = 2), rob house 3 (money = 9) and rob house 5 (money = 1).  
Total amount you can rob = 2 + 9 + 1 = 12.

**Constraints:**

* 1 <= nums.length <= 100
* 0 <= nums[i] <= 400

# **17. Letter Combinations of a Phone Number**

https://leetcode.com/problems/letter-combinations-of-a-phone-number/description/

Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in **any order**.

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



**Example 1:**

Input: digits = "23"  
Output: ["ad","ae","af","bd","be","bf","cd","ce","cf"]

**Example 2:**

Input: digits = ""  
Output: []

**Example 3:**

Input: digits = "2"  
Output: ["a","b","c"]

**Constraints:**

* 0 <= digits.length <= 4
* digits[i] is a digit in the range ['2', '9'].

# **68. Text Justification**

https://leetcode.com/problems/text-justification/description/

Given an array of strings words and a width maxWidth, format the text such that each line has exactly maxWidth characters and is fully (left and right) justified.

You should pack your words in a greedy approach; that is, pack as many words as you can in each line. Pad extra spaces ' ' when necessary so that each line has exactly maxWidth characters.

Extra spaces between words should be distributed as evenly as possible. If the number of spaces on a line does not divide evenly between words, the empty slots on the left will be assigned more spaces than the slots on the right.

For the last line of text, it should be left-justified, and no extra space is inserted between words.

**Note:**

* A word is defined as a character sequence consisting of non-space characters only.
* Each word's length is guaranteed to be greater than 0 and not exceed maxWidth.
* The input array words contains at least one word.

**Example 1:**

Input: words = ["This", "is", "an", "example", "of", "text", "justification."], maxWidth = 16  
Output:  
[  
   "This    is    an",  
   "example  of text",  
   "justification.  "  
]

**Example 2:**

Input: words = ["What","must","be","acknowledgment","shall","be"], maxWidth = 16  
Output:  
[  
  "What   must   be",  
  "acknowledgment  ",  
  "shall be        "  
]  
Explanation: Note that the last line is "shall be " instead of "shall be", because the last line must be left-justified instead of fully-justified.  
Note that the second line is also left-justified because it contains only one word.

**Example 3:**

Input: words = ["Science","is","what","we","understand","well","enough","to","explain","to","a","computer.","Art","is","everything","else","we","do"], maxWidth = 20  
Output:  
[  
  "Science  is  what we",  
 "understand      well",  
  "enough to explain to",  
  "a  computer.  Art is",  
  "everything  else  we",  
  "do                  "  
]

**Constraints:**

* 1 <= words.length <= 300
* 1 <= words[i].length <= 20
* words[i] consists of only English letters and symbols.
* 1 <= maxWidth <= 100
* words[i].length <= maxWidth

# **387. First Unique Character in a String**

https://leetcode.com/problems/first-unique-character-in-a-string/description/

Given a string s, *find the first non-repeating character in it and return its index*. If it does not exist, return -1.

**Example 1:**

Input: s = "leetcode"  
Output: 0

**Example 2:**

Input: s = "loveleetcode"  
Output: 2

**Example 3:**

Input: s = "aabb"  
Output: -1

**Constraints:**

* 1 <= s.length <= 105
* s consists of only lowercase English letters.

# **1531. String Compression II**

https://leetcode.com/problems/string-compression-ii/description/

[Run-length encoding](http://en.wikipedia.org/wiki/Run-length_encoding) is a string compression method that works by replacing consecutive identical characters (repeated 2 or more times) with the concatenation of the character and the number marking the count of the characters (length of the run). For example, to compress the string "aabccc" we replace "aa" by "a2" and replace "ccc" by "c3". Thus the compressed string becomes "a2bc3".

Notice that in this problem, we are not adding '1' after single characters.

Given a string s and an integer k. You need to delete **at most** k characters from s such that the run-length encoded version of s has minimum length.

Find the *minimum length of the run-length encoded version of* s *after deleting at most* k *characters*.

**Example 1:**

Input: s = "aaabcccd", k = 2  
Output: 4  
Explanation: Compressing s without deleting anything will give us "a3bc3d" of length 6. Deleting any of the characters 'a' or 'c' would at most decrease the length of the compressed string to 5, for instance delete 2 'a' then we will have s = "abcccd" which compressed is abc3d. Therefore, the optimal way is to delete 'b' and 'd', then the compressed version of s will be "a3c3" of length 4.

**Example 2:**

Input: s = "aabbaa", k = 2  
Output: 2  
Explanation: If we delete both 'b' characters, the resulting compressed string would be "a4" of length 2.

**Example 3:**

Input: s = "aaaaaaaaaaa", k = 0  
Output: 3  
Explanation: Since k is zero, we cannot delete anything. The compressed string is "a11" of length 3.

**Constraints:**

* 1 <= s.length <= 100
* 0 <= k <= s.length
* s contains only lowercase English letters.

# **2483. Minimum Penalty for a Shop**

https://leetcode.com/problems/minimum-penalty-for-a-shop/description/

You are given the customer visit log of a shop represented by a **0-indexed** string customers consisting only of characters 'N' and 'Y':

* if the ith character is 'Y', it means that customers come at the ith hour
* whereas 'N' indicates that no customers come at the ith hour.

If the shop closes at the jth hour (0 <= j <= n), the **penalty** is calculated as follows:

* For every hour when the shop is open and no customers come, the penalty increases by 1.
* For every hour when the shop is closed and customers come, the penalty increases by 1.

Return *the* ***earliest*** *hour at which the shop must be closed to incur a* ***minimum*** *penalty.*

**Note** that if a shop closes at the jth hour, it means the shop is closed at the hour j.

**Example 1:**

Input: customers = "YYNY"  
Output: 2  
Explanation:   
- Closing the shop at the 0th hour incurs in 1+1+0+1 = 3 penalty.  
- Closing the shop at the 1st hour incurs in 0+1+0+1 = 2 penalty.  
- Closing the shop at the 2nd hour incurs in 0+0+0+1 = 1 penalty.  
- Closing the shop at the 3rd hour incurs in 0+0+1+1 = 2 penalty.  
- Closing the shop at the 4th hour incurs in 0+0+1+0 = 1 penalty.  
Closing the shop at 2nd or 4th hour gives a minimum penalty. Since 2 is earlier, the optimal closing time is 2.

**Example 2:**

Input: customers = "NNNNN"  
Output: 0  
Explanation: It is best to close the shop at the 0th hour as no customers arrive.

**Example 3:**

Input: customers = "YYYY"  
Output: 4  
Explanation: It is best to close the shop at the 4th hour as customers arrive at each hour.

**Constraints:**

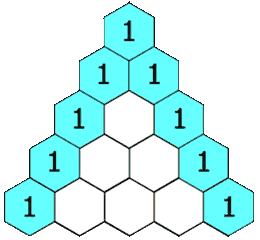
* 1 <= customers.length <= 105
* customers consists only of characters 'Y' and 'N'.

# **118. Pascal's Triangle**

https://leetcode.com/problems/pascals-triangle/description/

Given an integer numRows, return the first numRows of **Pascal's triangle**.

In **Pascal's triangle**, each number is the sum of the two numbers directly above it as shown:



**Example 1:**

Input: numRows = 5  
Output: [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]

**Example 2:**

Input: numRows = 1  
Output: [[1]]

**Constraints:**

* 1 <= numRows <= 30

# **2870. Minimum Number of Operations to Make Array Empty**

https://leetcode.com/problems/minimum-number-of-operations-to-make-array-empty/description/

You are given a **0-indexed** array nums consisting of positive integers.

There are two types of operations that you can apply on the array **any** number of times:

* Choose **two** elements with **equal** values and **delete** them from the array.
* Choose **three** elements with **equal** values and **delete** them from the array.

Return *the* ***minimum*** *number of operations required to make the array empty, or* -1 *if it is not possible*.

**Example 1:**

Input: nums = [2,3,3,2,2,4,2,3,4]  
Output: 4  
Explanation: We can apply the following operations to make the array empty:  
- Apply the first operation on the elements at indices 0 and 3. The resulting array is nums = [3,3,2,4,2,3,4].  
- Apply the first operation on the elements at indices 2 and 4. The resulting array is nums = [3,3,4,3,4].  
- Apply the second operation on the elements at indices 0, 1, and 3. The resulting array is nums = [4,4].  
- Apply the first operation on the elements at indices 0 and 1. The resulting array is nums = [].  
It can be shown that we cannot make the array empty in less than 4 operations.

**Example 2:**

Input: nums = [2,1,2,2,3,3]  
Output: -1  
Explanation: It is impossible to empty the array.

**Constraints:**

* 2 <= nums.length <= 105
* 1 <= nums[i] <= 106

# **175. Combine Two Tables**

https://leetcode.com/problems/combine-two-tables/description/

Table: Person

+-------------+---------+  
| Column Name | Type |  
+-------------+---------+  
| personId | int |  
| lastName | varchar |  
| firstName | varchar |  
+-------------+---------+  
personId is the primary key (column with unique values) for this table.  
This table contains information about the ID of some persons and their first and last names.

Table: Address

+-------------+---------+  
| Column Name | Type |  
+-------------+---------+  
| addressId | int |  
| personId | int |  
| city | varchar |  
| state | varchar |  
+-------------+---------+  
addressId is the primary key (column with unique values) for this table.  
Each row of this table contains information about the city and state of one person with ID = PersonId.

Write a solution to report the first name, last name, city, and state of each person in the Person table. If the address of a personId is not present in the Address table, report null instead.

Return the result table in **any order**.

The result format is in the following example.

**Example 1:**

Input:   
Person table:  
+----------+----------+-----------+  
| personId | lastName | firstName |  
+----------+----------+-----------+  
| 1 | Wang | Allen |  
| 2 | Alice | Bob |  
+----------+----------+-----------+  
Address table:  
+-----------+----------+---------------+------------+  
| addressId | personId | city | state |  
+-----------+----------+---------------+------------+  
| 1 | 2 | New York City | New York |  
| 2 | 3 | Leetcode | California |  
+-----------+----------+---------------+------------+  
Output:   
+-----------+----------+---------------+----------+  
| firstName | lastName | city | state |  
+-----------+----------+---------------+----------+  
| Allen | Wang | Null | Null |  
| Bob | Alice | New York City | New York |  
+-----------+----------+---------------+----------+  
Explanation:   
There is no address in the address table for the personId = 1 so we return null in their city and state.  
addressId = 1 contains information about the address of personId = 2.

# **300. Longest Increasing Subsequence**

https://leetcode.com/problems/longest-increasing-subsequence/description/

Given an integer array nums, return *the length of the longest* ***strictly increasing***

***subsequence***

.

**Example 1:**

Input: nums = [10,9,2,5,3,7,101,18]  
Output: 4  
Explanation: The longest increasing subsequence is [2,3,7,101], therefore the length is 4.

**Example 2:**

Input: nums = [0,1,0,3,2,3]  
Output: 4

**Example 3:**

Input: nums = [7,7,7,7,7,7,7]  
Output: 1

**Constraints:**

* 1 <= nums.length <= 2500
* -104 <= nums[i] <= 104

**Follow up:** Can you come up with an algorithm that runs in O(n log(n)) time complexity?

# **362. Design Hit Counter**

https://leetcode.com/problems/design-hit-counter/description/

Design a hit counter which counts the number of hits received in the past 5 minutes (i.e., the past 300 seconds).

Your system should accept a timestamp parameter (**in seconds** granularity), and you may assume that calls are being made to the system in chronological order (i.e., timestamp is monotonically increasing). Several hits may arrive roughly at the same time.

Implement the HitCounter class:

* HitCounter() Initializes the object of the hit counter system.
* void hit(int timestamp) Records a hit that happened at timestamp (**in seconds**). Several hits may happen at the same timestamp.
* int getHits(int timestamp) Returns the number of hits in the past 5 minutes from timestamp (i.e., the past 300 seconds).

**Example 1:**

Input  
["HitCounter", "hit", "hit", "hit", "getHits", "hit", "getHits", "getHits"]  
[[], [1], [2], [3], [4], [300], [300], [301]]  
Output  
[null, null, null, null, 3, null, 4, 3]  
  
Explanation  
HitCounter hitCounter = new HitCounter();  
hitCounter.hit(1); // hit at timestamp 1.  
hitCounter.hit(2); // hit at timestamp 2.  
hitCounter.hit(3); // hit at timestamp 3.  
hitCounter.getHits(4); // get hits at timestamp 4, return 3.  
hitCounter.hit(300); // hit at timestamp 300.  
hitCounter.getHits(300); // get hits at timestamp 300, return 4.  
hitCounter.getHits(301); // get hits at timestamp 301, return 3.

**Constraints:**

* 1 <= timestamp <= 2 \* 109
* All the calls are being made to the system in chronological order (i.e., timestamp is monotonically increasing).
* At most 300 calls will be made to hit and getHits.

**Follow up:** What if the number of hits per second could be huge? Does your design scale?

# **201. Bitwise AND of Numbers Range**

https://leetcode.com/problems/bitwise-and-of-numbers-range/description/

Given two integers left and right that represent the range [left, right], return *the bitwise AND of all numbers in this range, inclusive*.

**Example 1:**

Input: left = 5, right = 7  
Output: 4

**Example 2:**

Input: left = 0, right = 0  
Output: 0

**Example 3:**

Input: left = 1, right = 2147483647  
Output: 0

**Constraints:**

* 0 <= left <= right <= 231 - 1