1. Two Sum

Given an array of integers nums and an integer target, return indices of the two numbers such that they add up to targ
et.
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]
Output: 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?

---

2. Add Two Numbers

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse or
der, 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.

---

3. Longest Substring Without Repeating Characters

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.

Example 4:

Input: s = ""
Output: 0

Constraints:

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

---

4. Median of Two Sorted Arrays

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.

Example 3:

Input: nums1 = [0,0], nums2 = [0,0]
Output: 0.00000

Example 4:

Input: nums1 = [], nums2 = [1]
Output: 1.00000

Example 5:

Input: nums1 = [2], nums2 = []
Output: 2.00000

Constraints:

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

---

5. Longest Palindromic Substring

Given a string s, return the longest palindromic substring in s.

Example 1:

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

Example 2:

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

Example 3:

Input: s = "a"
Output: "a"

Example 4:

Input: s = "ac"
Output: "a"

Constraints:

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

---

6. Zigzag Conversion

The string "PAYPALISHIRING" is written in a zigzag pattern on a given number of rows like this: (you may want t
o display this pattern in a fixed font for better legibility)

P A H N
A P L S I I G
Y I R

And then read line by line: "PAHNAPLSIIGYIR"

Write the code that will take a string and make this conversion given a number of rows:

string convert(string s, int numRows);

Example 1:

Input: s = "PAYPALISHIRING", numRows = 3
Output: "PAHNAPLSIIGYIR"

Example 2:

Input: s = "PAYPALISHIRING", numRows = 4
Output: "PINALSIGYAHRPI"
Explanation:
P I N
A L S I G
Y A H R
P I

Example 3:

Input: s = "A", numRows = 1
Output: "A"

Constraints:

1 <= s.length <= 1000
s consists of English letters (lower-case and upper-case), ',' and '.'.
1 <= numRows <= 1000

---

7. Reverse Integer

Given a signed 32-bit integer x, return x with its digits reversed. If reversing x causes the value to go outside the sign ed 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

Example 4:
Input: x = 0
Output: 0

Constraints:

-231 <= x <= 231 - 1

---

8. String to Integer (atoi)

Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer (similar to C/C++'s atoi f unction).
The algorithm for myAtoi(string s) is as follows:

Read in and ignore any leading whitespace.
Check if the next character (if not already at the end of the string) is '-' or '+'. Read this character in if it is either. Thi s determines if the final result is negative or positive respectively. Assume the result is positive if neither is present.
Read in next the characters until the next non-digit character or the end of the input is reached. The rest of the string is ignored.
Convert these digits into an integer (i.e. "123" -> 123, "0032" -> 32). If no digits were read, then the integer is 0. Ch ange the sign as necessary (from step 2).
If the integer is out of the 32-bit signed integer range [-231, 231 - 1], then clamp the integer so that it remains in the r ange. Specifically, integers less than -231 should be clamped to -231, and integers greater than 231 - 1 should be cla mped to 231 - 1.
Return the integer as the final result.

Note:

Only the space character ' ' is considered a whitespace character.
Do not ignore any characters other than the leading whitespace or the rest of the string after the digits.

Example 1:

Input: s = "42"
Output: 42
Explanation: The underlined characters are what is read in, the caret is the current reader position.
Step 1: "42" (no characters read because there is no leading whitespace)
^
Step 2: "42" (no characters read because there is neither a '-' nor '+')
^
Step 3: "42" ("42" is read in)
^
The parsed integer is 42.
Since 42 is in the range [-231, 231 - 1], the final result is 42.

Example 2:

Input: s = "   -42"
Output: -42
Explanation:
Step 1: "   -42" (leading whitespace is read and ignored)
         ^
Step 2: "   -42" ('-' is read, so the result should be negative)
          ^
Step 3: "   -42" ("42" is read in)
            ^
The parsed integer is -42.
Since -42 is in the range [-231, 231 - 1], the final result is -42.

Example 3:

Input: s = "4193 with words"
Output: 4193
Explanation:
Step 1: "4193 with words" (no characters read because there is no leading whitespace)
        ^
Step 2: "4193 with words" (no characters read because there is neither a '-' nor '+')
        ^
Step 3: "4193 with words" ("4193" is read in; reading stops because the next character is a non-digit)
            ^
The parsed integer is 4193.
Since 4193 is in the range [-231, 231 - 1], the final result is 4193.

Example 4:

Input: s = "words and 987"
Output: 0
Explanation:
Step 1: "words and 987" (no characters read because there is no leading whitespace)
        ^
Step 2: "words and 987" (no characters read because there is neither a '-' nor '+')
        ^
Step 3: "words and 987" (reading stops immediately because there is a non-digit 'w')
        ^
The parsed integer is 0 because no digits were read.
Since 0 is in the range [-231, 231 - 1], the final result is 0.

Example 5:

Input: s = "-91283472332"
Output: -2147483648
Explanation:
Step 1: "-91283472332" (no characters read because there is no leading whitespace)
        ^
Step 2: "-91283472332" ('-' is read, so the result should be negative)
         ^
Step 3: "-91283472332" ("91283472332" is read in)
                ^
The parsed integer is -91283472332.
Since -91283472332 is less than the lower bound of the range [-231, 231 - 1], the final result is clamped to -231 = -2

147483648.

Constraints:

0 <= s.length <= 200
s consists of English letters (lower-case and upper-case), digits (0-9), ' ', '+', '-', and '.'.

---

9. Palindrome Number

Given an integer x, return true if x is palindrome integer.
An integer is a palindrome when it reads the same backward as forward. For example, 121 is palindrome while 123 i s not.

Example 1:

Input: x = 121
Output: true

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.

Example 4:

Input: x = -101
Output: false

Constraints:

-231 <= x <= 231 - 1

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

---

10. Regular Expression Matching

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

'.' Matches any single character.
'*' Matches zero or more of the preceding element.

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

Example 1:

Input: s = "aa", p = "a"
Output: false
Explanation: "a" does not match the entire string "aa".

Example 2:

Input: s = "aa", p = "a*"
Output: true
Explanation: '*' means zero or more of the preceding element, 'a'. Therefore, by repeating 'a' once, it becomes "aa".

Example 3:

Input: s = "ab", p = ".*"
Output: true
Explanation: ".*" means "zero or more (*) of any character (.)".

Example 4:

Input: s = "aab", p = "c*a*b"
Output: true
Explanation: c can be repeated 0 times, a can be repeated 1 time. Therefore, it matches "aab".

Example 5:

Input: s = "mississippi", p = "mis*is*p*."
Output: false

Constraints:

1 <= s.length <= 20
1 <= p.length <= 30
s contains only lowercase English letters.
p contains only lowercase English letters, '.', and '*'.
It is guaranteed for each appearance of the character '*', there will be a previous valid character to match.

---

11. Container With Most Water

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

Example 3:

Input: height = [4,3,2,1,4]
Output: 16

Example 4:

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

Constraints:

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

---

12. Integer to Roman

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

Symbol	Value
I	1
V	5

X 10
L 50
C 100
D 500
M 1000

For example, 2 is written as II in Roman numeral, just two one's added together. 12 is written as XII, which is simpl y 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. I nstead, the number four is written as IV. Because the one is before the five we subtract it making four. The same pri nciple applies to the number nine, which is written as IX. There are six instances where subtraction is used:

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

Given an integer, convert it to a roman numeral.

Example 1:

Input: num = 3
Output: "III"

Example 2:

Input: num = 4
Output: "IV"

Example 3:

Input: num = 9
Output: "IX"

Example 4:

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

Example 5:

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

Constraints:

1 <= num <= 3999

---

13. Roman to Integer

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

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

For example, 2 is written as II in Roman numeral, just two one's 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

Example 2:

Input: s = "IV"
Output: 4

Example 3:

Input: s = "IX"
Output: 9

Example 4:

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

Example 5:

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].

---

14. Longest Common Prefix

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 lower-case English letters.

---

15. 3Sum

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]]
Example 2:
Input: nums = []

Output: []
Example 3:
Input: nums = [0]
Output: []

Constraints:

0 <= nums.length <= 3000
-105 <= nums[i] <= 105

---

16. 3Sum Closest

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

Example 1:

Input: nums = [-1,2,1,-4], target = 1
Output: 2
Explanation: The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).

Example 2:

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

Constraints:

3 <= nums.length <= 1000
-1000 <= nums[i] <= 1000
-104 <= target <= 104

---

17. Letter Combinations of a Phone Number

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

esent. Return the answer in any order.
A mapping of digit to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any lette rs.

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'].

---

18. 4Sum

Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that:

0 <= a, b, c, d < n
a, b, c, and d are distinct.
nums[a] + nums[b] + nums[c] + nums[d] == target

You may return the answer in any order.

Example 1:

Input: nums = [1,0,-1,0,-2,2], target = 0
Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

Example 2:

Input: nums = [2,2,2,2,2], target = 8
Output: [[2,2,2,2]]

Constraints:

1 <= nums.length <= 200
-109 <= nums[i] <= 109
-109 <= target <= 109

---

19. Remove Nth Node From End of List

Given the head of a linked list, remove the nth node from the end of the list and return its head.

Example 1:

Input: head = [1,2,3,4,5], n = 2
Output: [1,2,3,5]

Example 2:

Input: head = [1], n = 1
Output: []

Example 3:

Input: head = [1,2], n = 1
Output: [1]

Constraints:

The number of nodes in the list is sz.
1 <= sz <= 30
0 <= Node.val <= 100
1 <= n <= sz

Follow up: Could you do this in one pass?

---

20. Valid Parentheses

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

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

Example 1:

Input: s = "()"
Output: true

Example 2:

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

Example 3:

Input: s = "(]"
Output: false

Example 4:

Input: s = "([)]"
Output: false

Example 5:

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

Constraints:

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

---

21. Merge Two Sorted Lists

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

---

22. Generate Parentheses

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

---

23. Merge k Sorted Lists

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 <= 10^4
0 <= lists[i].length <= 500
-10^4 <= lists[i][j] <= 10^4
lists[i] is sorted in ascending order.
The sum of lists[i].length won't exceed 10^4.

---

24. Swap Nodes in Pairs

Given a linked list, swap every two adjacent nodes and return its head. You must solve the problem without modifyi ng the values in the list's nodes (i.e., only nodes themselves may be changed.)

Example 1:

Input: head = [1,2,3,4]
Output: [2,1,4,3]

Example 2:

Input: head = []
Output: []

Example 3:

Input: head = [1]
Output: [1]

Constraints:

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

---

25. Reverse Nodes in k-Group

Given a linked list, reverse the nodes of a linked list k at a time and return its modified list.
k is a positive integer and is less than or equal to the length of the linked list. If the number of nodes is not a multiple of k then left-out nodes, in the end, should remain as it is.
You may not alter the values in the list's nodes, only nodes themselves may be changed.

Example 1:

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

Example 2:

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

Example 3:

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

Example 4:

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

Constraints:

The number of nodes in the list is in the range sz.
1 <= sz <= 5000
0 <= Node.val <= 1000
1 <= k <= sz

Follow-up: Can you solve the problem in O(1) extra memory space?

---

26. Remove Duplicates from Sorted Array

Given an integer array nums sorted in non-decreasing order, remove the duplicates in-place such that each unique ele ment appears only once. The relative order of the elements should be kept the same.
Since it is impossible to change the length of the array in some languages, you must instead have the result be placed in the first part of the array nums. More formally, if there are k elements after removing the duplicates, then the first k elements of nums should hold the final result. It does not matter what you leave beyond the first k elements.
Return k after placing the final result in the first k slots of nums.
Do not allocate extra space for another array. You must do this by modifying the input array in-place with O(1) extra memory.
Custom Judge:
The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length

int k = removeDuplicates(nums); // Calls your implementation

assert k == expectedNums.length;
for (int i = 0; i < k; i++) {
    assert nums[i] == expectedNums[i];
}
```

If all assertions pass, then your solution will be accepted.

Example 1:

Input: nums = [1,1,2]
Output: 2, nums = [1,2,_]
Explanation: Your function should return k = 2, with the first two elements of nums being 1 and 2 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).

Example 2:

Input: nums = [0,0,1,1,1,2,2,3,3,4]
Output: 5, nums = [0,1,2,3,4,_,_,_,_,_]
Explanation: Your function should return k = 5, with the first five elements of nums being 0, 1, 2, 3, and 4 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).

Constraints:

0 <= nums.length <= 3 * 104
-100 <= nums[i] <= 100
nums is sorted in non-decreasing order.

---

27. Remove Element

Given an integer array nums and an integer val, remove all occurrences of val in nums in-place. The relative order of the elements may be changed.
Since it is impossible to change the length of the array in some languages, you must instead have the result be placed in the first part of the array nums. More formally, if there are k elements after removing the duplicates, then the first k elements of nums should hold the final result. It does not matter what you leave beyond the first k elements.
Return k after placing the final result in the first k slots of nums.
Do not allocate extra space for another array. You must do this by modifying the input array in-place with O(1) extra memory.
Custom Judge:
The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int val = ...; // Value to remove
int[] expectedNums = [...]; // The expected answer with correct length.
                            // It is sorted with no values equaling val.

int k = removeElement(nums, val); // Calls your implementation

assert k == expectedNums.length;
sort(nums, 0, k); // Sort the first k elements of nums
for (int i = 0; i < actualLength; i++) {
    assert nums[i] == expectedNums[i];
}
```

If all assertions pass, then your solution will be accepted.

Example 1:

Input: nums = [3,2,2,3], val = 3
Output: 2, nums = [2,2,_,_]
Explanation: Your function should return k = 2, with the first two elements of nums being 2.
It does not matter what you leave beyond the returned k (hence they are underscores).

Example 2:

Input: nums = [0,1,2,2,3,0,4,2], val = 2
Output: 5, nums = [0,1,4,0,3,_,_,_]
Explanation: Your function should return k = 5, with the first five elements of nums containing 0, 0, 1, 3, and 4.
Note that the five elements can be returned in any order.

It does not matter what you leave beyond the returned k (hence they are underscores).

Constraints:

0 <= nums.length <= 100
0 <= nums[i] <= 50
0 <= val <= 100

---

28. Implement strStr()

Implement strStr().
Return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.
Clarification:
What should we return when needle is an empty string? This is a great question to ask during an interview.
For the purpose of this problem, we will return 0 when needle is an empty string. This is consistent to C's strstr() and Java's indexOf().

Example 1:
Input: haystack = "hello", needle = "ll"
Output: 2
Example 2:
Input: haystack = "aaaaa", needle = "bba"
Output: -1
Example 3:
Input: haystack = "", needle = ""
Output: 0

Constraints:

0 <= haystack.length, needle.length <= 5 * 104
haystack and needle consist of only lower-case English characters.

---

29. Divide Two Integers

Given two integers dividend and divisor, divide two integers without using multiplication, division, and mod operato
r.
The integer division should truncate toward zero, which means losing its fractional part. For example, 8.345 would b

e truncated to 8, and -2.7335 would be truncated to -2.
Return the quotient after dividing dividend by divisor.
Note: Assume we are dealing with an environment that could only store integers within the 32-bit signed integer ran ge: [−231, 231 − 1]. For this problem, if the quotient is strictly greater than 231 - 1, then return 231 - 1, and if the qu otient is strictly less than -231, then return -231.

Example 1:

Input: dividend = 10, divisor = 3
Output: 3
Explanation: 10/3 = 3.33333.. which is truncated to 3.

Example 2:

Input: dividend = 7, divisor = -3
Output: -2
Explanation: 7/-3 = -2.33333.. which is truncated to -2.

Example 3:

Input: dividend = 0, divisor = 1
Output: 0

Example 4:

Input: dividend = 1, divisor = 1
Output: 1

Constraints:

-231 <= dividend, divisor <= 231 - 1
divisor != 0

---

30. Substring with Concatenation of All Words

You are given a string s and an array of strings words of the same length. Return all starting indices of substring(s) i n s that is a concatenation of each word in words exactly once, in any order, and without any intervening characters. You can return the answer in any order.

Example 1:

Input: s = "barfoothefoobarman", words = ["foo","bar"]
Output: [0,9]
Explanation: Substrings starting at index 0 and 9 are "barfoo" and "foobar" respectively.
The output order does not matter, returning [9,0] is fine too.

Example 2:

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

Example 3:

Input: s = "barfoofoobarthefoobarman", words = ["bar","foo","the"]
Output: [6,9,12]

Constraints:

1 <= s.length <= 104
s consists of lower-case English letters.
1 <= words.length <= 5000
1 <= words[i].length <= 30
words[i] consists of lower-case English letters.

---

31. Next Permutation

Implement next permutation, which rearranges numbers into the lexicographically next greater permutation of numb
ers.
If such an arrangement is not possible, it must rearrange it as the lowest possible order (i.e., sorted in ascending orde
r).
The replacement must be in place 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]
Example 4:
Input: nums = [1]
Output: [1]

Constraints:

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

---

32. Longest Valid Parentheses

Given a string containing just the characters '(' and ')', find the length of the longest valid (well-formed) parentheses s ubstring.

Example 1:

Input: s = "(()"
Output: 2
Explanation: The longest valid parentheses substring is "()".

Example 2:

Input: s = ")()())"
Output: 4
Explanation: The longest valid parentheses substring is "()()".

Example 3:

Input: s = ""
Output: 0

Constraints:

0 <= s.length <= 3 * 104
s[i] is '(', or ')'.

---

33. Search in Rotated Sorted Array

There is an integer array nums sorted in ascending order (with distinct values).
Prior to being passed to your function, nums is possibly rotated at an unknown pivot index k (1 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). F or example, [0,1,2,4,5,6,7] might be rotated at pivot index 3 and become [4,5,6,7,0,1,2].
Given the array nums after the possible rotation and an integer target, return the index of target if it is in nums, or -1 if it is not in nums.
You must write an algorithm with O(log n) runtime complexity.

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

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

Constraints:

1 <= nums.length <= 5000
-104 <= nums[i] <= 104
All values of nums are unique.
nums is an ascending array that is possibly rotated.
-104 <= target <= 104

---

34. Find First and Last Position of Element in Sorted Array

Given an array of integers nums sorted in non-decreasing order, find the starting and ending position of a given targe t value.
If target is not found in the array, return [-1, -1].
You must write an algorithm with O(log n) runtime complexity.

Example 1:
Input: nums = [5,7,7,8,8,10], target = 8
Output: [3,4]
Example 2:
Input: nums = [5,7,7,8,8,10], target = 6
Output: [-1,-1]
Example 3:
Input: nums = [], target = 0
Output: [-1,-1]

Constraints:

0 <= nums.length <= 105
-109 <= nums[i] <= 109
nums is a non-decreasing array.
-109 <= target <= 109

---

35. Search Insert Position

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the in dex where it would be if it were inserted in order.
You must write an algorithm with O(log n) runtime complexity.

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

Constraints:

1 <= nums.length <= 104
-104 <= nums[i] <= 104
nums contains distinct values sorted in ascending order.
-104 <= target <= 104

---

36. Valid Sudoku

Determine if a 9 x 9 Sudoku board is valid. Only the filled cells need to be validated according to the following rules :

Each row must contain the digits 1-9 without repetition.
Each column must contain the digits 1-9 without repetition.
Each of the nine 3 x 3 sub-boxes of the grid must contain the digits 1-9 without repetition.

Note:

A Sudoku board (partially filled) could be valid but is not necessarily solvable.
Only the filled cells need to be validated according to the mentioned rules.

Example 1:

Input: board =
[["5","3",".",".","7",".",".",".",".","."]
,["6",".",".","1","9","5",".",".",".","."]
,[".","9","8",".",".",".",".",".","6","."]
,["8",".",".",".",".","6",".",".",".","3"]
,["4",".",".","8",".","3",".",".",".","1"]
,["7",".",".",".",".","2",".",".",".","6"]
,[".","6",".",".",".",".",".","2","8","."]
,[".",".",".","4","1","9",".",".",".","5"]
,[".",".",".",".","8",".",".",".","7","9"]]
Output: true

Example 2:

Input: board =
[["8","3",".",".","7",".",".",".",".","."]
,["6",".",".","1","9","5",".",".",".","."]
,[".","9","8",".",".",".",".",".","6","."]
,["8",".",".",".",".","6",".",".",".","3"]
,["4",".",".","8",".","3",".",".",".","1"]
,["7",".",".",".",".","2",".",".",".","6"]
,[".","6",".",".",".",".",".","2","8","."]
,[".",".",".","4","1","9",".",".",".","5"]
,[".",".",".",".",".","8",".",".",".","9"]]
Output: false
Explanation: Same as Example 1, except with the 5 in the top left corner being modified to 8. Since there are two 8's in the top left 3x3 sub-box, it is invalid.

Constraints:

board.length == 9
board[i].length == 9
board[i][j] is a digit 1-9 or '.'.

---

37. Sudoku Solver

Write a program to solve a Sudoku puzzle by filling the empty cells.
A sudoku solution must satisfy all of the following rules:

Each of the digits 1-9 must occur exactly once in each row.
Each of the digits 1-9 must occur exactly once in each column.
Each of the digits 1-9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.

The '.' character indicates empty cells.

Example 1:

Input: board = [["5","3",".",".","7",".",".",".",".","."],["6",".",".","1","9","5",".",".",".","."],[".","9","8",".",".",".",".",".","6","."],["8",".",".",".",".","6",".",".",".",".","3"],["4",".",".",".","8",".","3",".",".",".","1"],["7",".",".",".",".",".","2",".",".",".",".","6"],[".","6",".",".",".",".",".","2","8",".","."],[".",".",".",".",".","4","1","9",".",".",".","5"],[".",".",".",".",".",".","8",".",".",".","7","9"]]
Output: [["5","3","4","6","7","8","9","1","2"],["6","7","2","1","9","5","3","4","8"],["1","9","8","3","4","2","5","6","7"],["8","5","9","7","6","1","4","2","3"],["4","2","6","8","5","3","7","9","1"],["7","1","3","9","2","4","8","5","6"],["9","6","1","5","3","7","2","8","4"],["2","8","7","4","1","9","6","3","5"],["3","4","5","2","8","6","1","7","9"]]
Explanation: The input board is shown above and the only valid solution is shown below:

Constraints:

board.length == 9
board[i].length == 9
board[i][j] is a digit or '.'.
It is guaranteed that the input board has only one solution.

---

38. Count and Say

The count-and-say sequence is a sequence of digit strings defined by the recursive formula:

countAndSay(1) = "1"
countAndSay(n) is the way you would "say" the digit string from countAndSay(n-1), which is then converted into a different digit string.

To determine how you "say" a digit string, split it into the minimal number of groups so that each group is a contiguous section all of the same character. Then for each group, say the number of characters, then say the character. To convert the saying into a digit string, replace the counts with a number and concatenate every saying.
For example, the saying and conversion for digit string "3322251":

Given a positive integer n, return the nth term of the count-and-say sequence.

Example 1:

Input: n = 1
Output: "1"
Explanation: This is the base case.

Example 2:

Input: n = 4
Output: "1211"
Explanation:
countAndSay(1) = "1"
countAndSay(2) = say "1" = one 1 = "11"
countAndSay(3) = say "11" = two 1's = "21"
countAndSay(4) = say "21" = one 2 + one 1 = "12" + "11" = "1211"

Constraints:

1 <= n <= 30

---

39. Combination Sum

Given an array of distinct integers candidates and a target integer target, return a list of all unique combinations of ca ndidates where the chosen numbers sum to target. You may return the combinations in any order.
The same number may be chosen from candidates an unlimited number of times. Two combinations are unique if the frequency of at least one of the chosen numbers is different.
It is guaranteed that the number of unique combinations that sum up to target is less than 150 combinations for the gi ven input.

Example 1:

Input: candidates = [2,3,6,7], target = 7
Output: [[2,2,3],[7]]
Explanation:
2 and 3 are candidates, and 2 + 2 + 3 = 7. Note that 2 can be used multiple times.
7 is a candidate, and 7 = 7.
These are the only two combinations.

Example 2:

Input: candidates = [2,3,5], target = 8
Output: [[2,2,2,2],[2,3,3],[3,5]]

Example 3:

Input: candidates = [2], target = 1
Output: []

Example 4:

Input: candidates = [1], target = 1
Output: [[1]]

Example 5:

Input: candidates = [1], target = 2
Output: [[1,1]]

Constraints:

1 <= candidates.length <= 30
1 <= candidates[i] <= 200
All elements of candidates are distinct.
1 <= target <= 500

---

40. Combination Sum II

Given a collection of candidate numbers (candidates) and a target number (target), find all unique combinations in ca ndidates where the candidate numbers sum to target.
Each number in candidates may only be used once in the combination.
Note: The solution set must not contain duplicate combinations.

Example 1:

Input: candidates = [10,1,2,7,6,1,5], target = 8
Output:
[
[1,1,6],
[1,2,5],
[1,7],
[2,6]
]

Example 2:

Input: candidates = [2,5,2,1,2], target = 5
Output:
[
[1,2,2],
[5]
]

Constraints:

1 <= candidates.length <= 100
1 <= candidates[i] <= 50
1 <= target <= 30

---

41. First Missing Positive

Given an unsorted integer array nums, return the smallest missing positive integer.
You must implement an algorithm that runs in O(n) time and uses constant extra space.

Example 1:
Input: nums = [1,2,0]
Output: 3
Example 2:
Input: nums = [3,4,-1,1]
Output: 2
Example 3:
Input: nums = [7,8,9,11,12]
Output: 1

Constraints:

1 <= nums.length <= 5 * 105
-231 <= nums[i] <= 231 - 1

---

42. Trapping Rain Water

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much w ater 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

---

43. Multiply Strings

Given two non-negative integers num1 and num2 represented as strings, return the product of num1 and num2, also represented as a string.
Note: You must not use any built-in BigInteger library or convert the inputs to integer directly.

Example 1:
Input: num1 = "2", num2 = "3"
Output: "6"
Example 2:
Input: num1 = "123", num2 = "456"
Output: "56088"

Constraints:

1 <= num1.length, num2.length <= 200
num1 and num2 consist of digits only.
Both num1 and num2 do not contain any leading zero, except the number 0 itself.

---

44. Wildcard Matching

Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '*' where:

'?' Matches any single character.
'*' Matches any sequence of characters (including the empty sequence).

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

Example 1:

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

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

Example 2:

Input: s = "aa", p = "*"
Output: true
Explanation: '*' matches any sequence.

Example 3:

Input: s = "cb", p = "?a"
Output: false
Explanation: '?' matches 'c', but the second letter is 'a', which does not match 'b'.

Example 4:

Input: s = "adceb", p = "*a*b"
Output: true
Explanation: The first '*' matches the empty sequence, while the second '*' matches the substring "dce".

Example 5:

Input: s = "acdcb", p = "a*c?b"
Output: false

Constraints:

0 <= s.length, p.length <= 2000
s contains only lowercase English letters.
p contains only lowercase English letters, '?' or '*'.

---

45. Jump Game II

Given an array of non-negative integers nums, you are initially positioned at the first index of the array.
Each element in the array represents your maximum jump length at that position.
Your goal is to reach the last index in the minimum number of jumps.
You can assume that you can always reach the last index.

Example 1:

Input: nums = [2,3,1,1,4]
Output: 2
Explanation: The minimum number of jumps to reach the last index is 2. Jump 1 step from index 0 to 1, then 3 steps to the last index.

Example 2:

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

Constraints:

1 <= nums.length <= 104
0 <= nums[i] <= 1000

---

46. Permutations

Given an array nums of distinct integers, return all the possible permutations. You can return the answer in any order
.

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

Constraints:

1 <= nums.length <= 6
-10 <= nums[i] <= 10
All the integers of nums are unique.

---

47. Permutations II

Given a collection of numbers, nums, that might contain duplicates, return all possible unique permutations in any or
der.

Example 1:

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

Example 2:

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

Constraints:

1 <= nums.length <= 8
-10 <= nums[i] <= 10

---

48. Rotate Image

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, which means you have to modify the input 2D matrix directly. DO NOT alloc ate 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]]

Example 3:

Input: matrix = [[1]]
Output: [[1]]

Example 4:

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

Constraints:

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

---

49. Group Anagrams

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 t he 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.

---

50. Pow(x, n)

Implement pow(x, n), 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
-104 <= xn <= 104

---

51. N-Queens

The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other.
Given an integer n, return all distinct solutions to the n-queens puzzle. You may return the answer in any order.
Each solution contains a distinct board configuration of the n-queens' placement, where 'Q' and '.' both indicate a que en and an empty space, respectively.

Example 1:

Input: n = 4
Output: [[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]
Explanation: There exist two distinct solutions to the 4-queens puzzle as shown above

Example 2:

Input: n = 1
Output: [["Q"]]

Constraints:

1 <= n <= 9

---

52. N-Queens II

The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other.
Given an integer n, return the number of distinct solutions to the n-queens puzzle.

Example 1:

Input: n = 4
Output: 2
Explanation: There are two distinct solutions to the 4-queens puzzle as shown.

Example 2:

Input: n = 1
Output: 1

Constraints:

1 <= n <= 9

---

53. Maximum Subarray

Given an integer array nums, find the contiguous subarray (containing at least one number) which has the largest su m and return its sum.
A subarray is a contiguous part of an array.

Example 1:

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

Example 2:

Input: nums = [1]
Output: 1

Example 3:

Input: nums = [5,4,-1,7,8]
Output: 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 appro ach, which is more subtle.

---

54. Spiral Matrix

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

---

55. Jump Game

You are given an integer array nums. You are initially positioned at the array's first index, and each element in the ar ray represents your maximum jump length at that position.
Return true if you can reach the last index, or false otherwise.

Example 1:

Input: nums = [2,3,1,1,4]
Output: true
Explanation: Jump 1 step from index 0 to 1, then 3 steps to the last index.

Example 2:

Input: nums = [3,2,1,0,4]
Output: false
Explanation: You will always arrive at index 3 no matter what. Its maximum jump length is 0, which makes it impos sible to reach the last index.

Constraints:

1 <= nums.length <= 104
0 <= nums[i] <= 105

---

56. Merge Intervals

Given an array of intervals where intervals[i] = [starti, endi], merge all overlapping intervals, and return an array of t he 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] overlaps, 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

---

57. Insert Interval

You are given an array of non-overlapping intervals intervals where intervals[i] = [starti, endi] represent the start and the end of the ith interval and intervals is sorted in ascending order by starti. You are also given an interval newInterval = [start, end] that represents the start and end of another interval.
Insert newInterval into intervals such that intervals is still sorted in ascending order by starti and intervals still does not have any overlapping intervals (merge overlapping intervals if necessary).
Return intervals after the insertion.

Example 1:

Input: intervals = [[1,3],[6,9]], newInterval = [2,5]
Output: [[1,5],[6,9]]

Example 2:

Input: intervals = [[1,2],[3,5],[6,7],[8,10],[12,16]], newInterval = [4,8]
Output: [[1,2],[3,10],[12,16]]
Explanation: Because the new interval [4,8] overlaps with [3,5],[6,7],[8,10].
Example 3:

Input: intervals = [], newInterval = [5,7]
Output: [[5,7]]

Example 4:

Input: intervals = [[1,5]], newInterval = [2,3]
Output: [[1,5]]

Example 5:

Input: intervals = [[1,5]], newInterval = [2,7]
Output: [[1,7]]

Constraints:

0 <= intervals.length <= 104
intervals[i].length == 2
0 <= starti <= endi <= 105
intervals is sorted by starti in ascending order.
newInterval.length == 2
0 <= start <= end <= 105

---

58. Length of Last Word

Given a string s consisting of some words separated by some number of spaces, return the length of the last word in t he string.
A word is a maximal substring consisting of non-space characters only.

Example 1:

Input: s = "Hello World"
Output: 5
Explanation: The last word is "World" with length 5.

Example 2:

Input: s = "   fly me   to   the moon "
Output: 4
Explanation: The last word is "moon" with length 4.

Example 3:

Input: s = "luffy is still joyboy"
Output: 6
Explanation: The last word is "joyboy" with length 6.

Constraints:

1 <= s.length <= 104
s consists of only English letters and spaces ' '.
There will be at least one word in s.

---

59. Spiral Matrix II

Given a positive integer n, generate an n x n matrix filled with elements from 1 to n2 in spiral order.

Example 1:

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

Example 2:

Input: n = 1
Output: [[1]]

Constraints:

1 <= n <= 20

---

60. Permutation Sequence

The set [1, 2, 3, ..., n] contains a total of n! unique permutations.
By listing and labeling all of the permutations in order, we get the following sequence for n = 3:

"123"
"132"
"213"
"231"
"312"
"321"

Given n and k, return the kth permutation sequence.

Example 1:
Input: n = 3, k = 3
Output: "213"
Example 2:
Input: n = 4, k = 9
Output: "2314"
Example 3:
Input: n = 3, k = 1
Output: "123"

Constraints:

1 <= n <= 9
1 <= k <= n!

---

61. Rotate List

Given the head of a linked list, rotate the list to the right by k places.

Example 1:

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

Example 2:

Input: head = [0,1,2], k = 4
Output: [2,0,1]

Constraints:

The number of nodes in the list is in the range [0, 500].
-100 <= Node.val <= 100
0 <= k <= 2 * 109

---

62. Unique Paths

A robot is located at the top-left corner of a m x n grid (marked 'Start' in the diagram below).
The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corn er of the grid (marked 'Finish' in the diagram below).
How many possible unique paths are there?

Example 1:

Input: m = 3, n = 7
Output: 28

Example 2:

Input: m = 3, n = 2
Output: 3
Explanation:
From the top-left corner, there are a total of 3 ways to reach the bottom-right corner:
1. Right -> Down -> Down
2. Down -> Down -> Right
3. Down -> Right -> Down

Example 3:

Input: m = 7, n = 3
Output: 28

Example 4:

Input: m = 3, n = 3
Output: 6

Constraints:

1 <= m, n <= 100
It's guaranteed that the answer will be less than or equal to 2 * 109.

---

63. Unique Paths II

A robot is located at the top-left corner of a m x n grid (marked 'Start' in the diagram below).
The robot can only move either down or right at any point in time. The robot is trying to reach the bottom-right corn
er of the grid (marked 'Finish' in the diagram below).
Now consider if some obstacles are added to the grids. How many unique paths would there be?
An obstacle and space is marked as 1 and 0 respectively in the grid.

Example 1:

Input: obstacleGrid = [[0,0,0],[0,1,0],[0,0,0]]
Output: 2
Explanation: There is one obstacle in the middle of the 3x3 grid above.
There are two ways to reach the bottom-right corner:
1. Right -> Right -> Down -> Down
2. Down -> Down -> Right -> Right

Example 2:

Input: obstacleGrid = [[0,1],[0,0]]
Output: 1

Constraints:

m == obstacleGrid.length
n == obstacleGrid[i].length
1 <= m, n <= 100
obstacleGrid[i][j] is 0 or 1.

---

64. Minimum Path Sum

Given a m x n grid filled with non-negative numbers, find a path from top left to bottom right, which minimizes the sum of all numbers along its path.
Note: You can only move either down or right at any point in time.

Example 1:

Input: grid = [[1,3,1],[1,5,1],[4,2,1]]
Output: 7
Explanation: Because the path 1 → 3 → 1 → 1 → 1 minimizes the sum.

Example 2:

Input: grid = [[1,2,3],[4,5,6]]
Output: 12

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 200
0 <= grid[i][j] <= 100

---

65. Valid Number

A valid number can be split up into these components (in order):

A decimal number or an integer.
(Optional) An 'e' or 'E', followed by an integer.

A decimal number can be split up into these components (in order):

(Optional) A sign character (either '+' or '-').
One of the following formats:

One or more digits, followed by a dot '.'.
One or more digits, followed by a dot '.', followed by one or more digits.
A dot '.', followed by one or more digits.

An integer can be split up into these components (in order):

(Optional) A sign character (either '+' or '-').
One or more digits.

For example, all the following are valid numbers: ["2", "0089", "-0.1", "+3.14", "4.", "-.9", "2e10", "-90E3", "3e+7", "+6e-1", "53.5e93", "-123.456e789"], while the following are not valid numbers: ["abc", "1a", "1e", "e3", "99e2.5", "--6", "-+3", "95a54e53"].
Given a string s, return true if s is a valid number.

Example 1:

Input: s = "0"
Output: true

Example 2:

Input: s = "e"
Output: false

Example 3:

Input: s = "."
Output: false

Example 4:

Input: s = ".1"
Output: true

Constraints:

1 <= s.length <= 20
s consists of only English letters (both uppercase and lowercase), digits (0-9), plus '+', minus '-', or dot '.'.

---

66. Plus One

You are given a large integer represented as an integer array digits, where each digits[i] is the ith digit of the integer. The digits are ordered from most significant to least significant in left-to-right order. The large integer does not cont ain any leading 0's.
Increment the large integer by one and return the resulting array of digits.

Example 1:

Input: digits = [1,2,3]
Output: [1,2,4]
Explanation: The array represents the integer 123.
Incrementing by one gives 123 + 1 = 124.
Thus, the result should be [1,2,4].

Example 2:

Input: digits = [4,3,2,1]
Output: [4,3,2,2]
Explanation: The array represents the integer 4321.
Incrementing by one gives 4321 + 1 = 4322.
Thus, the result should be [4,3,2,2].

Example 3:

Input: digits = [0]
Output: [1]
Explanation: The array represents the integer 0.
Incrementing by one gives 0 + 1 = 1.
Thus, the result should be [1].

Example 4:

Input: digits = [9]
Output: [1,0]
Explanation: The array represents the integer 9.
Incrementing by one gives 9 + 1 = 10.
Thus, the result should be [1,0].

Constraints:

1 <= digits.length <= 100
0 <= digits[i] <= 9
digits does not contain any leading 0's.

---

67. Add Binary

Given two binary strings a and b, return their sum as a binary string.

Example 1:
Input: a = "11", b = "1"
Output: "100"
Example 2:
Input: a = "1010", b = "1011"
Output: "10101"

Constraints:

1 <= a.length, b.length <= 104
a and b consist only of '0' or '1' characters.
Each string does not contain leading zeros except for the zero itself.

---

68. Text Justification

Given an array of strings words and a width maxWidth, format the text such that each line has exactly maxWidth ch aracters 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 sp aces ' ' 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 di vide 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 becase 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

---

69. Sqrt(x)

Given a non-negative integer x, compute and return the square root of x.
Since the return type is an integer, the decimal digits are truncated, and only the integer part of the result is returned.
Note: You are not allowed to use any built-in exponent function or operator, such as pow(x, 0.5) or x ** 0.5.

Example 1:

Input: x = 4
Output: 2

Example 2:

Input: x = 8
Output: 2
Explanation: The square root of 8 is 2.82842..., and since the decimal part is truncated, 2 is returned.

Constraints:

0 <= x <= 231 - 1

---

70. Climbing Stairs

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

---

71. Simplify Path

Given a string path, which is an absolute path (starting with a slash '/') to a file or directory in a Unix-style file syste m, convert it to the simplified canonical path.
In a Unix-style file system, a period '.' refers to the current directory, a double period '..' refers to the directory up a le vel, and any multiple consecutive slashes (i.e. '//') are treated as a single slash '/'. For this problem, any other format o f periods such as '...' are treated as file/directory names.
The canonical path should have the following format:

The path starts with a single slash '/'.
Any two directories are separated by a single slash '/'.
The path does not end with a trailing '/'.
The path only contains the directories on the path from the root directory to the target file or directory (i.e., no period '.' or double period '..')

Return the simplified canonical path.

Example 1:

Input: path = "/home/"
Output: "/home"
Explanation: Note that there is no trailing slash after the last directory name.

Example 2:

Input: path = "/../"
Output: "/"
Explanation: Going one level up from the root directory is a no-op, as the root level is the highest level you can go.

Example 3:

Input: path = "/home//foo/"
Output: "/home/foo"
Explanation: In the canonical path, multiple consecutive slashes are replaced by a single one.

Example 4:

Input: path = "/a/./b/../../c/"
Output: "/c"

Constraints:

1 <= path.length <= 3000
path consists of English letters, digits, period '.', slash '/' or '_'.
path is a valid absolute Unix path.

---

72. Edit Distance

Given two strings word1 and word2, return the minimum number of operations required to convert word1 to word2.
You have the following three operations permitted on a word:

Insert a character
Delete a character
Replace a character

Example 1:

Input: word1 = "horse", word2 = "ros"
Output: 3
Explanation:
horse -> rorse (replace 'h' with 'r')
rorse -> rose (remove 'r')
rose -> ros (remove 'e')

Example 2:

Input: word1 = "intention", word2 = "execution"
Output: 5
Explanation:
intention -> inention (remove 't')
inention -> enention (replace 'i' with 'e')
enention -> exention (replace 'n' with 'x')
exention -> exection (replace 'n' with 'c')
exection -> execution (insert 'u')

Constraints:

0 <= word1.length, word2.length <= 500
word1 and word2 consist of lowercase English letters.

---

73. Set Matrix Zeroes

Given an m x n integer matrix matrix, if an element is 0, set its entire row and column to 0's, and return the matrix. You must do it in place.

Example 1:

Input: matrix = [[1,1,1],[1,0,1],[1,1,1]]
Output: [[1,0,1],[0,0,0],[1,0,1]]

Example 2:

Input: matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]
Output: [[0,0,0,0],[0,4,5,0],[0,3,1,0]]

Constraints:

m == matrix.length
n == matrix[0].length
1 <= m, n <= 200
-231 <= matrix[i][j] <= 231 - 1

Follow up:

A straightforward solution using O(mn) space is probably a bad idea.

A simple improvement uses O(m + n) space, but still not the best solution.
Could you devise a constant space solution?

---

74. Search a 2D Matrix

Write an efficient algorithm that searches for a value in an m x n matrix. This matrix has the following properties:

Integers in each row are sorted from left to right.
The first integer of each row is greater than the last integer of the previous row.

Example 1:

Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 3
Output: true

Example 2:

Input: matrix = [[1,3,5,7],[10,11,16,20],[23,30,34,60]], target = 13
Output: false

Constraints:

m == matrix.length
n == matrix[i].length
1 <= m, n <= 100
-104 <= matrix[i][j], target <= 104

---

75. Sort Colors

Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color a re adjacent, with the colors in the order red, white, and blue.
We will use the integers 0, 1, and 2 to represent the color red, white, and blue, respectively.
You must solve this problem without using the library's sort function.

Example 1:

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

Constraints:

n == nums.length
1 <= n <= 300
nums[i] is 0, 1, or 2.

Follow up: Could you come up with a one-pass algorithm using only constant extra space?

---

76. Minimum Window Substring

Given two strings s and t of lengths m and n respectively, return the minimum window substring of s such that every character in t (including duplicates) is included in the window. If there is no such substring, return the empty string "".
The testcases will be generated such that the answer is unique.
A substring is a contiguous sequence of characters within the string.

Example 1:

Input: s = "ADOBECODEBANC", t = "ABC"
Output: "BANC"
Explanation: The minimum window substring "BANC" includes 'A', 'B', and 'C' from string t.

Example 2:

Input: s = "a", t = "a"
Output: "a"
Explanation: The entire string s is the minimum window.

Example 3:

Input: s = "a", t = "aa"
Output: ""
Explanation: Both 'a's from t must be included in the window.
Since the largest window of s only has one 'a', return empty string.

Constraints:

m == s.length
n == t.length
1 <= m, n <= 105
s and t consist of uppercase and lowercase English letters.

Follow up: Could you find an algorithm that runs in O(m + n) time?

---

77. Combinations

Given two integers n and k, return all possible combinations of k numbers out of the range [1, n].
You may return the answer in any order.

Example 1:

Input: n = 4, k = 2
Output:
[
  [2,4],
  [3,4],
  [2,3],
  [1,2],
  [1,3],
  [1,4],
]

Example 2:

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

Constraints:

1 <= n <= 20
1 <= k <= n

---

78. Subsets

Given an integer array nums of unique elements, return all possible subsets (the power set).
The solution set must not contain duplicate subsets. Return the solution in any order.

Example 1:

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

Example 2:

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

Constraints:

1 <= nums.length <= 10
-10 <= nums[i] <= 10
All the numbers of nums are unique.

---

79. Word Search

Given an m x n grid of characters board and a string word, return true if word exists in the grid.
The word can be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or verti cally neighboring. The same letter cell may not be used more than once.

Example 1:

Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCCED"
Output: true

Example 2:

Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "SEE"
Output: true

Example 3:

Input: board = [["A","B","C","E"],["S","F","C","S"],["A","D","E","E"]], word = "ABCB"
Output: false

Constraints:

m == board.length
n = board[i].length
1 <= m, n <= 6
1 <= word.length <= 15
board and word consists of only lowercase and uppercase English letters.

Follow up: Could you use search pruning to make your solution faster with a larger board?

---

80. Remove Duplicates from Sorted Array II

Given an integer array nums sorted in non-decreasing order, remove some duplicates in-place such that each unique element appears at most twice. The relative order of the elements should be kept the same.
Since it is impossible to change the length of the array in some languages, you must instead have the result be placed in the first part of the array nums. More formally, if there are k elements after removing the duplicates, then the first k elements of nums should hold the final result. It does not matter what you leave beyond the first k elements.
Return k after placing the final result in the first k slots of nums.
Do not allocate extra space for another array. You must do this by modifying the input array in-place with O(1) extra memory.
Custom Judge:
The judge will test your solution with the following code:

```
int[] nums = [...]; // Input array
int[] expectedNums = [...]; // The expected answer with correct length

int k = removeDuplicates(nums); // Calls your implementation

assert k == expectedNums.length;
for (int i = 0; i < k; i++) {
    assert nums[i] == expectedNums[i];
}
```

If all assertions pass, then your solution will be accepted.

Example 1:

Input: nums = [1,1,1,2,2,3]
Output: 5, nums = [1,1,2,2,3,_]
Explanation: Your function should return k = 5, with the first five elements of nums being 1, 1, 2, 2 and 3 respectively.
It does not matter what you leave beyond the returned k (hence they are underscores).

Example 2:

Input: nums = [0,0,1,1,1,1,2,3,3]
Output: 7, nums = [0,0,1,1,2,3,3,_,_]

Explanation: Your function should return k = 7, with the first seven elements of nums being 0, 0, 1, 1, 2, 3 and 3 res pectively.
It does not matter what you leave beyond the returned k (hence they are underscores).

Constraints:

1 <= nums.length <= 3 * 104
-104 <= nums[i] <= 104
nums is sorted in non-decreasing order.

---

81. Search in Rotated Sorted Array II

There is an integer array nums sorted in non-decreasing order (not necessarily with distinct values).
Before being passed to your function, nums is rotated at an unknown pivot index k (0 <= k < nums.length) such that the resulting array is [nums[k], nums[k+1], ..., nums[n-1], nums[0], nums[1], ..., nums[k-1]] (0-indexed). For exampl e, [0,1,2,4,4,4,5,6,6,7] might be rotated at pivot index 5 and become [4,5,6,6,7,0,1,2,4,4].
Given the array nums after the rotation and an integer target, return true if target is in nums, or false if it is not in nu ms.
You must decrease the overall operation steps as much as possible.

Example 1:
Input: nums = [2,5,6,0,0,1,2], target = 0
Output: true
Example 2:
Input: nums = [2,5,6,0,0,1,2], target = 3
Output: false

Constraints:

1 <= nums.length <= 5000
-104 <= nums[i] <= 104
nums is guaranteed to be rotated at some pivot.
-104 <= target <= 104

Follow up: This problem is similar to Search in Rotated Sorted Array, but nums may contain duplicates. Would this affect the runtime complexity? How and why?

---

82. Remove Duplicates from Sorted List II

Given the head of a sorted linked list, delete all nodes that have duplicate numbers, leaving only distinct numbers fro m the original list. Return the linked list sorted as well.

Example 1:

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

Example 2:

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

Constraints:

The number of nodes in the list is in the range [0, 300].
-100 <= Node.val <= 100
The list is guaranteed to be sorted in ascending order.

---

83. Remove Duplicates from Sorted List

Given the head of a sorted linked list, delete all duplicates such that each element appears only once. Return the link ed list sorted as well.

Example 1:

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

Example 2:

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

Constraints:

The number of nodes in the list is in the range [0, 300].
-100 <= Node.val <= 100
The list is guaranteed to be sorted in ascending order.

---

84. Largest Rectangle in Histogram

Given an array of integers heights representing the histogram's bar height where the width of each bar is 1, return the area of the largest rectangle in the histogram.

Example 1:

Input: heights = [2,1,5,6,2,3]
Output: 10
Explanation: The above is a histogram where width of each bar is 1.
The largest rectangle is shown in the red area, which has an area = 10 units.

Example 2:

Input: heights = [2,4]
Output: 4

Constraints:

1 <= heights.length <= 105
0 <= heights[i] <= 104

---

85. Maximal Rectangle

Given a rows x cols binary matrix filled with 0's and 1's, find the largest rectangle containing only 1's and return its a rea.

Example 1:

Input: matrix = [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0","1","0"]]
Output: 6
Explanation: The maximal rectangle is shown in the above picture.

Example 2:

Input: matrix = []
Output: 0

Example 3:

Input: matrix = [["0"]]
Output: 0

Example 4:

Input: matrix = [["1"]]
Output: 1

Example 5:

Input: matrix = [["0","0"]]
Output: 0

Constraints:

rows == matrix.length
cols == matrix[i].length
0 <= row, cols <= 200
matrix[i][j] is '0' or '1'.

---

86. Partition List

Given the head of a linked list and a value x, partition it such that all nodes less than x come before nodes greater tha n or equal to x.
You should preserve the original relative order of the nodes in each of the two partitions.

Example 1:

Input: head = [1,4,3,2,5,2], x = 3
Output: [1,2,2,4,3,5]

Example 2:

Input: head = [2,1], x = 2
Output: [1,2]

Constraints:

The number of nodes in the list is in the range [0, 200].
-100 <= Node.val <= 100
-200 <= x <= 200

---

87. Scramble String

We can scramble a string s to get a string t using the following algorithm:

If the length of the string is 1, stop.
If the length of the string is > 1, do the following:

Split the string into two non-empty substrings at a random index, i.e., if the string is s, divide it to x and y where s = x + y.
Randomly decide to swap the two substrings or to keep them in the same order. i.e., after this step, s may become s = x + y or s = y + x.
Apply step 1 recursively on each of the two substrings x and y.

Given two strings s1 and s2 of the same length, return true if s2 is a scrambled string of s1, otherwise, return false.

Example 1:

Input: s1 = "great", s2 = "rgeat"
Output: true
Explanation: One possible scenario applied on s1 is:
"great" --> "gr/eat" // divide at random index.
"gr/eat" --> "gr/eat" // random decision is not to swap the two substrings and keep them in order.
"gr/eat" --> "g/r / e/at" // apply the same algorithm recursively on both substrings. divide at ranom index each of the m.
"g/r / e/at" --> "r/g / e/at" // random decision was to swap the first substring and to keep the second substring in the s ame order.
"r/g / e/at" --> "r/g / e/ a/t" // again apply the algorithm recursively, divide "at" to "a/t".
"r/g / e/ a/t" --> "r/g / e/ a/t" // random decision is to keep both substrings in the same order.
The algorithm stops now and the result string is "rgeat" which is s2.
As there is one possible scenario that led s1 to be scrambled to s2, we return true.

Example 2:

Input: s1 = "abcde", s2 = "caebd"
Output: false

Example 3:

Input: s1 = "a", s2 = "a"
Output: true

Constraints:

s1.length == s2.length
1 <= s1.length <= 30
s1 and s2 consist of lower-case English letters.

---

88. Merge Sorted Array

You are given two integer arrays nums1 and nums2, sorted in non-decreasing order, and two integers m and n, repre senting 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 accom modate 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 nu ms1.

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?

---

89. Gray Code

An n-bit gray code sequence is a sequence of 2n integers where:

Every integer is in the inclusive range [0, 2n - 1],
The first integer is 0,
An integer appears no more than once in the sequence,
The binary representation of every pair of adjacent integers differs by exactly one bit, and
The binary representation of the first and last integers differs by exactly one bit.

Given an integer n, return any valid n-bit gray code sequence.

Example 1:

Input: n = 2
Output: [0,1,3,2]
Explanation:
The binary representation of [0,1,3,2] is [00,01,11,10].
- 00 and 01 differ by one bit
- 01 and 11 differ by one bit
- 11 and 10 differ by one bit
- 10 and 00 differ by one bit
[0,2,3,1] is also a valid gray code sequence, whose binary representation is [00,10,11,01].
- 00 and 10 differ by one bit
- 10 and 11 differ by one bit
- 11 and 01 differ by one bit
- 01 and 00 differ by one bit

Example 2:

Input: n = 1
Output: [0,1]

Constraints:

1 <= n <= 16

---

90. Subsets II

Given an integer array nums that may contain duplicates, return all possible subsets (the power set).
The solution set must not contain duplicate subsets. Return the solution in any order.

Example 1:
Input: nums = [1,2,2]
Output: [[],[1],[1,2],[1,2,2],[2],[2,2]]
Example 2:
Input: nums = [0]
Output: [[],[0]]

Constraints:

1 <= nums.length <= 10
-10 <= nums[i] <= 10

---

91. Decode Ways

A message containing letters from A-Z can be encoded into numbers using the following mapping:

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

To decode an encoded message, all the digits must be grouped then mapped back into letters using the reverse of the mapping above (there may be multiple ways). For example, "11106" can be mapped into:

"AAJF" with the grouping (1 1 10 6)
"KJF" with the grouping (11 10 6)

Note that the grouping (1 11 06) is invalid because "06" cannot be mapped into 'F' since "6" is different from "06".
Given a string s containing only digits, return the number of ways to decode it.
The answer is guaranteed to fit in a 32-bit integer.

Example 1:

Input: s = "12"
Output: 2
Explanation: "12" could be decoded as "AB" (1 2) or "L" (12).

Example 2:

Input: s = "226"
Output: 3
Explanation: "226" could be decoded as "BZ" (2 26), "VF" (22 6), or "BBF" (2 2 6).

Example 3:

Input: s = "0"
Output: 0
Explanation: There is no character that is mapped to a number starting with 0.
The only valid mappings with 0 are 'J' -> "10" and 'T' -> "20", neither of which start with 0.
Hence, there are no valid ways to decode this since all digits need to be mapped.

Example 4:

Input: s = "06"
Output: 0
Explanation: "06" cannot be mapped to "F" because of the leading zero ("6" is different from "06").

Constraints:

1 <= s.length <= 100
s contains only digits and may contain leading zero(s).

---

92. Reverse Linked List II

Given the head of a singly linked list and two integers left and right where left <= right, reverse the nodes of the list f rom position left to position right, and return the reversed list.

Example 1:

Input: head = [1,2,3,4,5], left = 2, right = 4
Output: [1,4,3,2,5]

Example 2:

Input: head = [5], left = 1, right = 1
Output: [5]

Constraints:

The number of nodes in the list is n.
1 <= n <= 500
-500 <= Node.val <= 500
1 <= left <= right <= n

Follow up: Could you do it in one pass?

---

93. Restore IP Addresses

A valid IP address consists of exactly four integers separated by single dots. Each integer is between 0 and 255 (incl usive) and cannot have leading zeros.

For example, "0.1.2.201" and "192.168.1.1" are valid IP addresses, but "0.011.255.245", "192.168.1.312" and "192.1 68@1.1" are invalid IP addresses.

Given a string s containing only digits, return all possible valid IP addresses that can be formed by inserting dots into s. You are not allowed to reorder or remove any digits in s. You may return the valid IP addresses in any order.

Example 1:
Input: s = "25525511135"
Output: ["255.255.11.135","255.255.111.35"]
Example 2:
Input: s = "0000"
Output: ["0.0.0.0"]
Example 3:
Input: s = "1111"
Output: ["1.1.1.1"]
Example 4:
Input: s = "010010"
Output: ["0.10.0.10","0.100.1.0"]
Example 5:
Input: s = "101023"
Output: ["1.0.10.23","1.0.102.3","10.1.0.23","10.10.2.3","101.0.2.3"]

Constraints:

0 <= s.length <= 20
s consists of digits only.

---

94. Binary Tree Inorder Traversal

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

Example 1:

Input: root = [1,null,2,3]
Output: [1,3,2]

Example 2:

Input: root = []
Output: []

Example 3:

Input: root = [1]
Output: [1]

Example 4:

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

Example 5:

Input: root = [1,null,2]
Output: [1,2]

Constraints:

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

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

---

95. Unique Binary Search Trees II

Given an integer n, return all the structurally unique BST's (binary search trees), which has exactly n nodes of uniqu e values from 1 to n. Return the answer in any order.

Example 1:

Input: n = 3
Output: [[1,null,2,null,3],[1,null,3,2],[2,1,3],[3,1,null,null,2],[3,2,null,1]]

Example 2:

Input: n = 1
Output: [[1]]

Constraints:

1 <= n <= 8

---

96. Unique Binary Search Trees

Given an integer n, return the number of structurally unique BST's (binary search trees) which has exactly n nodes of unique values from 1 to n.

Example 1:

Input: n = 3
Output: 5

Example 2:

Input: n = 1
Output: 1

Constraints:

1 <= n <= 19

---

97. Interleaving String

Given strings s1, s2, and s3, find whether s3 is formed by an interleaving of s1 and s2.
An interleaving of two strings s and t is a configuration where they are divided into non-empty substrings such that:

s = s1 + s2 + ... + sn
t = t1 + t2 + ... + tm
|n - m| <= 1
The interleaving is s1 + t1 + s2 + t2 + s3 + t3 + ... or t1 + s1 + t2 + s2 + t3 + s3 + ...

Note: a + b is the concatenation of strings a and b.

Example 1:

Input: s1 = "aabcc", s2 = "dbbca", s3 = "aadbbcbcac"
Output: true

Example 2:

Input: s1 = "aabcc", s2 = "dbbca", s3 = "aadbbbaccc"
Output: false

Example 3:

Input: s1 = "", s2 = "", s3 = ""
Output: true

Constraints:

0 <= s1.length, s2.length <= 100
0 <= s3.length <= 200
s1, s2, and s3 consist of lowercase English letters.

Follow up: Could you solve it using only O(s2.length) additional memory space?

---

98. Validate Binary Search Tree

Given the root of a binary tree, determine if it is a valid binary search tree (BST).
A valid BST is defined as follows:

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

Example 1:

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

Example 2:

Input: root = [5,1,4,null,null,3,6]
Output: false
Explanation: The root node's value is 5 but its right child's value is 4.

Constraints:

The number of nodes in the tree is in the range [1, 104].
-231 <= Node.val <= 231 - 1

---

99. Recover Binary Search Tree

You are given the root of a binary search tree (BST), where the values of exactly two nodes of the tree were swappe d by mistake. Recover the tree without changing its structure.

Example 1:

Input: root = [1,3,null,null,2]
Output: [3,1,null,null,2]
Explanation: 3 cannot be a left child of 1 because 3 > 1. Swapping 1 and 3 makes the BST valid.

Example 2:

Input: root = [3,1,4,null,null,2]
Output: [2,1,4,null,null,3]
Explanation: 2 cannot be in the right subtree of 3 because 2 < 3. Swapping 2 and 3 makes the BST valid.

Constraints:

The number of nodes in the tree is in the range [2, 1000].
-231 <= Node.val <= 231 - 1

Follow up: A solution using O(n) space is pretty straight-forward. Could you devise a constant O(1) space solution?

---

100. Same Tree

Given the roots of two binary trees p and q, write a function to check if they are the same or not.

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

Example 1:

Input: p = [1,2,3], q = [1,2,3]
Output: true

Example 2:

Input: p = [1,2], q = [1,null,2]
Output: false

Example 3:

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

Constraints:

The number of nodes in both trees is in the range [0, 100].
-104 <= Node.val <= 104

---

101. Symmetric Tree

Given the root of a binary tree, check whether it is a mirror of itself (i.e., symmetric around its center).

Example 1:

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

Example 2:

Input: root = [1,2,2,null,3,null,3]
Output: false

Constraints:

The number of nodes in the tree is in the range [1, 1000].
-100 <= Node.val <= 100

Follow up: Could you solve it both recursively and iteratively?

---

102. Binary Tree Level Order Traversal

Given the root of a binary tree, return the level order traversal of its nodes' values. (i.e., from left to right, level by le vel).

Example 1:

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

Example 2:

Input: root = [1]
Output: [[1]]

Example 3:

Input: root = []
Output: []

Constraints:

The number of nodes in the tree is in the range [0, 2000].
-1000 <= Node.val <= 1000

---

103. Binary Tree Zigzag Level Order Traversal

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

Example 1:

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

Example 2:

Input: root = [1]
Output: [[1]]

Example 3:

Input: root = []
Output: []

Constraints:

The number of nodes in the tree is in the range [0, 2000].
-100 <= Node.val <= 100

---

104. Maximum Depth of Binary Tree

Given the root of a binary tree, return its maximum depth.
A binary tree's maximum depth is the number of nodes along the longest path from the root node down to the farthes t leaf node.

Example 1:

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

Example 2:

Input: root = [1,null,2]
Output: 2

Example 3:

Input: root = []
Output: 0

Example 4:

Input: root = [0]
Output: 1

Constraints:

The number of nodes in the tree is in the range [0, 104].
-100 <= Node.val <= 100

---

105. Construct Binary Tree from Preorder and Inorder Traversal

Given two integer arrays preorder and inorder where preorder is the preorder traversal of a binary tree and inorder is the inorder traversal of the same tree, construct and return the binary tree.

Example 1:

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

Example 2:

Input: preorder = [-1], inorder = [-1]
Output: [-1]

Constraints:

1 <= preorder.length <= 3000
inorder.length == preorder.length
-3000 <= preorder[i], inorder[i] <= 3000
preorder and inorder consist of unique values.
Each value of inorder also appears in preorder.
preorder is guaranteed to be the preorder traversal of the tree.
inorder is guaranteed to be the inorder traversal of the tree.

---

106. Construct Binary Tree from Inorder and Postorder Traversal

Given two integer arrays inorder and postorder where inorder is the inorder traversal of a binary tree and postorder is the postorder traversal of the same tree, construct and return the binary tree.

Example 1:

Input: inorder = [9,3,15,20,7], postorder = [9,15,7,20,3]

Output: [3,9,20,null,null,15,7]

Example 2:

Input: inorder = [-1], postorder = [-1]
Output: [-1]

Constraints:

1 <= inorder.length <= 3000
postorder.length == inorder.length
-3000 <= inorder[i], postorder[i] <= 3000
inorder and postorder consist of unique values.
Each value of postorder also appears in inorder.
inorder is guaranteed to be the inorder traversal of the tree.
postorder is guaranteed to be the postorder traversal of the tree.

---

107. Binary Tree Level Order Traversal II

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

Example 1:

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

Example 2:

Input: root = [1]
Output: [[1]]

Example 3:

Input: root = []
Output: []

Constraints:

The number of nodes in the tree is in the range [0, 2000].
-1000 <= Node.val <= 1000

---

108. Convert Sorted Array to Binary Search Tree

Given an integer array nums where the elements are sorted in ascending order, convert it to a height-balanced binary search tree.
A height-balanced binary tree is a binary tree in which the depth of the two subtrees of every node never differs by more than one.

Example 1:

Input: nums = [-10,-3,0,5,9]
Output: [0,-3,9,-10,null,5]
Explanation: [0,-10,5,null,-3,null,9] is also accepted:

Example 2:

Input: nums = [1,3]
Output: [3,1]
Explanation: [1,3] and [3,1] are both a height-balanced BSTs.

Constraints:

1 <= nums.length <= 104
-104 <= nums[i] <= 104
nums is sorted in a strictly increasing order.

---

109. Convert Sorted List to Binary Search Tree

Given the head of a singly linked list where elements are sorted in ascending order, convert it to a height balanced B ST.
For this problem, a height-balanced binary tree is defined as a binary tree in which the depth of the two subtrees of e very node never differ by more than 1.

Example 1:

Input: head = [-10,-3,0,5,9]
Output: [0,-3,9,-10,null,5]

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

Example 2:

Input: head = []
Output: []

Example 3:

Input: head = [0]
Output: [0]

Example 4:

Input: head = [1,3]
Output: [3,1]

Constraints:

The number of nodes in head is in the range [0, 2 * 104].
-105 <= Node.val <= 105

---

110. Balanced Binary Tree

Given a binary tree, determine if it is height-balanced.
For this problem, a height-balanced binary tree is defined as:

a binary tree in which the left and right subtrees of every node differ in height by no more than 1.

Example 1:

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

Example 2:

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

Example 3:

Input: root = []
Output: true

Constraints:

The number of nodes in the tree is in the range [0, 5000].
-104 <= Node.val <= 104

---

111. Minimum Depth of Binary Tree

Given a binary tree, find its minimum depth.
The minimum depth is the number of nodes along the shortest path from the root node down to the nearest leaf node.
Note: A leaf is a node with no children.

Example 1:

Input: root = [3,9,20,null,null,15,7]
Output: 2

Example 2:

Input: root = [2,null,3,null,4,null,5,null,6]
Output: 5

Constraints:

The number of nodes in the tree is in the range [0, 105].
-1000 <= Node.val <= 1000

---

112. Path Sum

Given the root of a binary tree and an integer targetSum, return true if the tree has a root-to-leaf path such that addin g up all the values along the path equals targetSum.
A leaf is a node with no children.

Example 1:

Input: root = [5,4,8,11,null,13,4,7,2,null,null,null,1], targetSum = 22
Output: true
Explanation: The root-to-leaf path with the target sum is shown.

Example 2:

Input: root = [1,2,3], targetSum = 5
Output: false
Explanation: There two root-to-leaf paths in the tree:
(1 --> 2): The sum is 3.
(1 --> 3): The sum is 4.
There is no root-to-leaf path with sum = 5.

Example 3:

Input: root = [], targetSum = 0
Output: false
Explanation: Since the tree is empty, there are no root-to-leaf paths.

Constraints:

The number of nodes in the tree is in the range [0, 5000].
-1000 <= Node.val <= 1000
-1000 <= targetSum <= 1000

---

113. Path Sum II

Given the root of a binary tree and an integer targetSum, return all root-to-leaf paths where the sum of the node valu es in the path equals targetSum. Each path should be returned as a list of the node values, not node references.
A root-to-leaf path is a path starting from the root and ending at any leaf node. A leaf is a node with no children.

Example 1:

Input: root = [5,4,8,11,null,13,4,7,2,null,null,5,1], targetSum = 22
Output: [[5,4,11,2],[5,8,4,5]]
Explanation: There are two paths whose sum equals targetSum:
5 + 4 + 11 + 2 = 22
5 + 8 + 4 + 5 = 22

Example 2:

Input: root = [1,2,3], targetSum = 5
Output: []

Example 3:

Input: root = [1,2], targetSum = 0
Output: []

Constraints:

The number of nodes in the tree is in the range [0, 5000].
-1000 <= Node.val <= 1000
-1000 <= targetSum <= 1000

---

114. Flatten Binary Tree to Linked List

Given the root of a binary tree, flatten the tree into a "linked list":

The "linked list" should use the same TreeNode class where the right child pointer points to the next node in the list and the left child pointer is always null.
The "linked list" should be in the same order as a pre-order traversal of the binary tree.

Example 1:

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

Example 2:

Input: root = []
Output: []

Example 3:

Input: root = [0]
Output: [0]

Constraints:

The number of nodes in the tree is in the range [0, 2000].
-100 <= Node.val <= 100

Follow up: Can you flatten the tree in-place (with O(1) extra space)?

---

115. Distinct Subsequences

Given two strings s and t, return the number of distinct subsequences of s which equals t.
A string's subsequence is a new string formed from the original string by deleting some (can be none) of the characte rs without disturbing the remaining characters' relative positions. (i.e., "ACE" is a subsequence of "ABCDE" while " AEC" is not).
The test cases are generated so that the answer fits on a 32-bit signed integer.

Example 1:

Input: s = "rabbbit", t = "rabbit"
Output: 3
Explanation:
As shown below, there are 3 ways you can generate "rabbit" from S.
rabbbit
rabbbit
rabbbit

Example 2:

Input: s = "babgbag", t = "bag"
Output: 5
Explanation:
As shown below, there are 5 ways you can generate "bag" from S.
babgbag
babgbag
babgbag
babgbag
babgbag

Constraints:

1 <= s.length, t.length <= 1000
s and t consist of English letters.

---

116. Populating Next Right Pointers in Each Node

You are given a perfect binary tree where all leaves are on the same level, and every parent has two children. The bi nary tree has the following definition:

struct Node {

```
  int val;
  Node *left;
  Node *right;
  Node *next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.
Initially, all next pointers are set to NULL.

Example 1:

Input: root = [1,2,3,4,5,6,7]
Output: [1,#,2,3,#,4,5,6,7,#]
Explanation: Given the above perfect binary tree (Figure A), your function should populate each next pointer to poin t to its next right node, just like in Figure B. The serialized output is in level order as connected by the next pointers, with '#' signifying the end of each level.

Example 2:

Input: root = []
Output: []

Constraints:

The number of nodes in the tree is in the range [0, 212 - 1].
-1000 <= Node.val <= 1000

Follow-up:

You may only use constant extra space.
The recursive approach is fine. You may assume implicit stack space does not count as extra space for this problem.

---

117. Populating Next Right Pointers in Each Node II

Given a binary tree

```
struct Node {
  int val;
  Node *left;
  Node *right;
  Node *next;
}
```

Populate each next pointer to point to its next right node. If there is no next right node, the next pointer should be set to NULL.
Initially, all next pointers are set to NULL.

Example 1:

Input: root = [1,2,3,4,5,null,7]
Output: [1,#,2,3,#,4,5,7,#]
Explanation: Given the above binary tree (Figure A), your function should populate each next pointer to point to its next right node, just like in Figure B. The serialized output is in level order as connected by the next pointers, with '#' signifying the end of each level.

Example 2:

Input: root = []
Output: []

Constraints:

The number of nodes in the tree is in the range [0, 6000].
-100 <= Node.val <= 100

Follow-up:

You may only use constant extra space.
The recursive approach is fine. You may assume implicit stack space does not count as extra space for this problem.

---

118. Pascal's Triangle

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

---

119. Pascal's Triangle II

Given an integer rowIndex, return the rowIndexth (0-indexed) row of the Pascal's triangle.
In Pascal's triangle, each number is the sum of the two numbers directly above it as shown:

Example 1:
Input: rowIndex = 3
Output: [1,3,3,1]
Example 2:
Input: rowIndex = 0
Output: [1]
Example 3:
Input: rowIndex = 1
Output: [1,1]

Constraints:

0 <= rowIndex <= 33

Follow up: Could you optimize your algorithm to use only O(rowIndex) extra space?

---

120. Triangle

Given a triangle array, return the minimum path sum from top to bottom.
For each step, you may move to an adjacent number of the row below. More formally, if you are on index i on the cu rrent row, you may move to either index i or index i + 1 on the next row.

Example 1:

Input: triangle = [[2],[3,4],[6,5,7],[4,1,8,3]]
Output: 11
Explanation: The triangle looks like:
   2
  3 4
 6 5 7
4 1 8 3

The minimum path sum from top to bottom is 2 + 3 + 5 + 1 = 11 (underlined above).

Example 2:

Input: triangle = [[-10]]
Output: -10

Constraints:

1 <= triangle.length <= 200
triangle[0].length == 1
triangle[i].length == triangle[i - 1].length + 1
-104 <= triangle[i][j] <= 104

Follow up: Could you do this using only O(n) extra space, where n is the total number of rows in the triangle?

---

121. Best Time to Buy and Sell Stock

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 futu re 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

---

122. Best Time to Buy and Sell Stock II

You are given an integer array prices where prices[i] is the price of a given stock on the ith day.
On each day, you may decide to buy and/or sell the stock. You can only hold at most one share of the stock at any ti me. However, you can buy it then immediately sell it on the same day.
Find and return the maximum profit you can achieve.

Example 1:

Input: prices = [7,1,5,3,6,4]
Output: 7
Explanation: Buy on day 2 (price = 1) and sell on day 3 (price = 5), profit = 5-1 = 4.
Then buy on day 4 (price = 3) and sell on day 5 (price = 6), profit = 6-3 = 3.
Total profit is 4 + 3 = 7.

Example 2:

Input: prices = [1,2,3,4,5]
Output: 4
Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.
Total profit is 4.

Example 3:

Input: prices = [7,6,4,3,1]
Output: 0
Explanation: There is no way to make a positive profit, so we never buy the stock to achieve the maximum profit of 0.

Constraints:

1 <= prices.length <= 3 * 104
0 <= prices[i] <= 104

---

123. Best Time to Buy and Sell Stock III

You are given an array prices where prices[i] is the price of a given stock on the ith day.
Find the maximum profit you can achieve. You may complete at most two transactions.
Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy agai n).

Example 1:

Input: prices = [3,3,5,0,0,3,1,4]
Output: 6
Explanation: Buy on day 4 (price = 0) and sell on day 6 (price = 3), profit = 3-0 = 3.
Then buy on day 7 (price = 1) and sell on day 8 (price = 4), profit = 4-1 = 3.
Example 2:

Input: prices = [1,2,3,4,5]
Output: 4
Explanation: Buy on day 1 (price = 1) and sell on day 5 (price = 5), profit = 5-1 = 4.
Note that you cannot buy on day 1, buy on day 2 and sell them later, as you are engaging multiple transactions at the same time. You must sell before buying again.

Example 3:

Input: prices = [7,6,4,3,1]
Output: 0
Explanation: In this case, no transaction is done, i.e. max profit = 0.

Example 4:

Input: prices = [1]
Output: 0

Constraints:

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

---

124. Binary Tree Maximum Path Sum

A path in a binary tree is a sequence of nodes where each pair of adjacent nodes in the sequence has an edge connect ing them. A node can only appear in the sequence at most once. Note that the path does not need to pass through the root.
The path sum of a path is the sum of the node's values in the path.
Given the root of a binary tree, return the maximum path sum of any non-empty path.

Example 1:

Input: root = [1,2,3]
Output: 6
Explanation: The optimal path is 2 -> 1 -> 3 with a path sum of 2 + 1 + 3 = 6.

Example 2:

Input: root = [-10,9,20,null,null,15,7]
Output: 42
Explanation: The optimal path is 15 -> 20 -> 7 with a path sum of 15 + 20 + 7 = 42.

Constraints:

The number of nodes in the tree is in the range [1, 3 * 104].
-1000 <= Node.val <= 1000

---

125. Valid Palindrome

A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and removing all non-alphan umeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers. Given a string s, return true if it is a palindrome, or false otherwise.

Example 1:

Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.

Example 2:

Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.

Example 3:

Input: s = " "
Output: true
Explanation: s is an empty string "" after removing non-alphanumeric characters.
Since an empty string reads the same forward and backward, it is a palindrome.

Constraints:

1 <= s.length <= 2 * 105
s consists only of printable ASCII characters.

---

126. Word Ladder II

A transformation sequence from word beginWord to word endWord using a dictionary wordList is a sequence of wo rds beginWord -> s1 -> s2 -> ... -> sk such that:

Every adjacent pair of words differs by a single letter.
Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.
sk == endWord

Given two words, beginWord and endWord, and a dictionary wordList, return all the shortest transformation sequen ces from beginWord to endWord, or an empty list if no such sequence exists. Each sequence should be returned as a list of the words [beginWord, s1, s2, ..., sk].

Example 1:

Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]
Output: [["hit","hot","dot","dog","cog"],["hit","hot","lot","log","cog"]]
Explanation: There are 2 shortest transformation sequences:
"hit" -> "hot" -> "dot" -> "dog" -> "cog"
"hit" -> "hot" -> "lot" -> "log" -> "cog"

Example 2:

Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log"]
Output: []
Explanation: The endWord "cog" is not in wordList, therefore there is no valid transformation sequence.

Constraints:

1 <= beginWord.length <= 5
endWord.length == beginWord.length
1 <= wordList.length <= 1000
wordList[i].length == beginWord.length
beginWord, endWord, and wordList[i] consist of lowercase English letters.
beginWord != endWord
All the words in wordList are unique.

---

127. Word Ladder

A transformation sequence from word beginWord to word endWord using a dictionary wordList is a sequence of wo rds beginWord -> s1 -> s2 -> ... -> sk such that:

Every adjacent pair of words differs by a single letter.
Every si for 1 <= i <= k is in wordList. Note that beginWord does not need to be in wordList.

sk == endWord

Given two words, beginWord and endWord, and a dictionary wordList, return the number of words in the shortest tr
ansformation sequence from beginWord to endWord, or 0 if no such sequence exists.

Example 1:

Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log","cog"]
Output: 5
Explanation: One shortest transformation sequence is "hit" -> "hot" -> "dot" -> "dog" -> cog", which is 5 words long
.

Example 2:

Input: beginWord = "hit", endWord = "cog", wordList = ["hot","dot","dog","lot","log"]
Output: 0
Explanation: The endWord "cog" is not in wordList, therefore there is no valid transformation sequence.

Constraints:

1 <= beginWord.length <= 10
endWord.length == beginWord.length
1 <= wordList.length <= 5000
wordList[i].length == beginWord.length
beginWord, endWord, and wordList[i] consist of lowercase English letters.
beginWord != endWord
All the words in wordList are unique.

---

128. Longest Consecutive Sequence

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

---

129. Sum Root to Leaf Numbers

You are given the root of a binary tree containing digits from 0 to 9 only.
Each root-to-leaf path in the tree represents a number.

For example, the root-to-leaf path 1 -> 2 -> 3 represents the number 123.

Return the total sum of all root-to-leaf numbers. Test cases are generated so that the answer will fit in a 32-bit intege
r.
A leaf node is a node with no children.

Example 1:

Input: root = [1,2,3]
Output: 25
Explanation:
The root-to-leaf path 1->2 represents the number 12.
The root-to-leaf path 1->3 represents the number 13.
Therefore, sum = 12 + 13 = 25.

Example 2:

Input: root = [4,9,0,5,1]
Output: 1026
Explanation:
The root-to-leaf path 4->9->5 represents the number 495.
The root-to-leaf path 4->9->1 represents the number 491.
The root-to-leaf path 4->0 represents the number 40.
Therefore, sum = 495 + 491 + 40 = 1026.

Constraints:

The number of nodes in the tree is in the range [1, 1000].
0 <= Node.val <= 9
The depth of the tree will not exceed 10.

---

130. Surrounded Regions

Given an m x n matrix board containing 'X' and 'O', capture all regions that are 4-directionally surrounded by 'X'.
A region is captured by flipping all 'O's into 'X's in that surrounded region.

Example 1:

Input: board = [["X","X","X","X"],["X","O","O","X"],["X","X","O","X"],["X","O","X","X"]]
Output: [["X","X","X","X"],["X","X","X","X"],["X","X","X","X"],["X","O","X","X"]]
Explanation: Surrounded regions should not be on the border, which means that any 'O' on the border of the board ar e not flipped to 'X'. Any 'O' that is not on the border and it is not connected to an 'O' on the border will be flipped to ' X'. Two cells are connected if they are adjacent cells connected horizontally or vertically.

Example 2:

Input: board = [["X"]]
Output: [["X"]]

Constraints:

m == board.length
n == board[i].length
1 <= m, n <= 200
board[i][j] is 'X' or 'O'.

---

131. Palindrome Partitioning

Given a string s, partition s such that every substring of the partition is a palindrome. Return all possible palindrome partitioning of s.
A palindrome string is a string that reads the same backward as forward.

Example 1:
Input: s = "aab"
Output: [["a","a","b"],["aa","b"]]
Example 2:
Input: s = "a"
Output: [["a"]]

Constraints:

1 <= s.length <= 16
s contains only lowercase English letters.

---

132. Palindrome Partitioning II

Given a string s, partition s such that every substring of the partition is a palindrome.
Return the minimum cuts needed for a palindrome partitioning of s.

Example 1:

Input: s = "aab"
Output: 1
Explanation: The palindrome partitioning ["aa","b"] could be produced using 1 cut.

Example 2:

Input: s = "a"
Output: 0

Example 3:

Input: s = "ab"
Output: 1

Constraints:

1 <= s.length <= 2000
s consists of lower-case English letters only.

---

133. Clone Graph

Given a reference of a node in a connected undirected graph.
Return a deep copy (clone) of the graph.
Each node in the graph contains a value (int) and a list (List[Node]) of its neighbors.

```
class Node {
    public int val;
    public List<Node> neighbors;
```

}

Test case format:
For simplicity, each node's value is the same as the node's index (1-indexed). For example, the first node with val == 1, the second node with val == 2, and so on. The graph is represented in the test case using an adjacency list.
An adjacency list is a collection of unordered lists used to represent a finite graph. Each list describes the set of neighbors of a node in the graph.
The given node will always be the first node with val = 1. You must return the copy of the given node as a reference to the cloned graph.

Example 1:

Input: adjList = [[2,4],[1,3],[2,4],[1,3]]
Output: [[2,4],[1,3],[2,4],[1,3]]
Explanation: There are 4 nodes in the graph.
1st node (val = 1)'s neighbors are 2nd node (val = 2) and 4th node (val = 4).
2nd node (val = 2)'s neighbors are 1st node (val = 1) and 3rd node (val = 3).
3rd node (val = 3)'s neighbors are 2nd node (val = 2) and 4th node (val = 4).
4th node (val = 4)'s neighbors are 1st node (val = 1) and 3rd node (val = 3).

Example 2:

Input: adjList = [[]]
Output: [[]]
Explanation: Note that the input contains one empty list. The graph consists of only one node with val = 1 and it does not have any neighbors.

Example 3:

Input: adjList = []
Output: []
Explanation: This an empty graph, it does not have any nodes.

Example 4:

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

Constraints:

The number of nodes in the graph is in the range [0, 100].
1 <= Node.val <= 100
Node.val is unique for each node.
There are no repeated edges and no self-loops in the graph.
The Graph is connected and all nodes can be visited starting from the given node.

---

134. Gas Station

There are n gas stations along a circular route, where the amount of gas at the ith station is gas[i].
You have a car with an unlimited gas tank and it costs cost[i] of gas to travel from the ith station to its next (i + 1)th station. You begin the journey with an empty tank at one of the gas stations.
Given two integer arrays gas and cost, return the starting gas station's index if you can travel around the circuit once in the clockwise direction, otherwise return -1. If there exists a solution, it is guaranteed to be unique

Example 1:

Input: gas = [1,2,3,4,5], cost = [3,4,5,1,2]
Output: 3
Explanation:
Start at station 3 (index 3) and fill up with 4 unit of gas. Your tank = 0 + 4 = 4
Travel to station 4. Your tank = 4 - 1 + 5 = 8
Travel to station 0. Your tank = 8 - 2 + 1 = 7
Travel to station 1. Your tank = 7 - 3 + 2 = 6
Travel to station 2. Your tank = 6 - 4 + 3 = 5
Travel to station 3. The cost is 5. Your gas is just enough to travel back to station 3.
Therefore, return 3 as the starting index.

Example 2:

Input: gas = [2,3,4], cost = [3,4,3]
Output: -1
Explanation:
You can't start at station 0 or 1, as there is not enough gas to travel to the next station.
Let's start at station 2 and fill up with 4 unit of gas. Your tank = 0 + 4 = 4
Travel to station 0. Your tank = 4 - 3 + 2 = 3
Travel to station 1. Your tank = 3 - 3 + 3 = 3
You cannot travel back to station 2, as it requires 4 unit of gas but you only have 3.
Therefore, you can't travel around the circuit once no matter where you start.

Constraints:

gas.length == n
cost.length == n
1 <= n <= 105
0 <= gas[i], cost[i] <= 104

---

135. Candy

There are n children standing in a line. Each child is assigned a rating value given in the integer array ratings.
You are giving candies to these children subjected to the following requirements:

Each child must have at least one candy.
Children with a higher rating get more candies than their neighbors.

Return the minimum number of candies you need to have to distribute the candies to the children.

Example 1:

Input: ratings = [1,0,2]
Output: 5
Explanation: You can allocate to the first, second and third child with 2, 1, 2 candies respectively.

Example 2:

Input: ratings = [1,2,2]
Output: 4
Explanation: You can allocate to the first, second and third child with 1, 2, 1 candies respectively.
The third child gets 1 candy because it satisfies the above two conditions.

Constraints:

n == ratings.length
1 <= n <= 2 * 104
0 <= ratings[i] <= 2 * 104

---

136. Single Number

Given a non-empty array of integers nums, every element appears twice except for one. Find that single one.
You must implement a solution with a linear runtime complexity and use only constant extra space.

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

Constraints:

1 <= nums.length <= 3 * 104

-3 * 104 <= nums[i] <= 3 * 104
Each element in the array appears twice except for one element which appears only once.

---

137. Single Number II

Given an integer array nums where every element appears three times except for one, which appears exactly once. Fi nd the single element and return it.
You must implement a solution with a linear runtime complexity and use only constant extra space.

Example 1:
Input: nums = [2,2,3,2]
Output: 3
Example 2:
Input: nums = [0,1,0,1,0,1,99]
Output: 99

Constraints:

1 <= nums.length <= 3 * 104
-231 <= nums[i] <= 231 - 1
Each element in nums appears exactly three times except for one element which appears once.

---

138. Copy List with Random Pointer

A linked list of length n is given such that each node contains an additional random pointer, which could point to an y node in the list, or null.
Construct a deep copy of the list. The deep copy should consist of exactly n brand new nodes, where each new node has its value set to the value of its corresponding original node. Both the next and random pointer of the new nodes s hould point to new nodes in the copied list such that the pointers in the original list and copied list represent the same list state. None of the pointers in the new list should point to nodes in the original list.
For example, if there are two nodes X and Y in the original list, where X.random --> Y, then for the corresponding t wo nodes x and y in the copied list, x.random --> y.
Return the head of the copied linked list.
The linked list is represented in the input/output as a list of n nodes. Each node is represented as a pair of [val, rando m_index] where:

val: an integer representing Node.val
random_index: the index of the node (range from 0 to n-1) that the random pointer points to, or null if it does not poi

nt to any node.

Your code will only be given the head of the original linked list.

Example 1:

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

Example 2:

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

Example 3:

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

Example 4:

Input: head = []
Output: []
Explanation: The given linked list is empty (null pointer), so return null.

Constraints:

0 <= n <= 1000
-10000 <= Node.val <= 10000
Node.random is null or is pointing to some node in the linked list.

---

139. Word Break

Given a string s and a dictionary of strings wordDict, return true if s can be segmented into a space-separated sequen ce of one or more dictionary words.
Note that the same word in the dictionary may be reused multiple times in the segmentation.

Example 1:

Input: s = "leetcode", wordDict = ["leet","code"]
Output: true
Explanation: Return true because "leetcode" can be segmented as "leet code".

Example 2:

Input: s = "applepenapple", wordDict = ["apple","pen"]
Output: true
Explanation: Return true because "applepenapple" can be segmented as "apple pen apple".
Note that you are allowed to reuse a dictionary word.

Example 3:

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

Constraints:

1 <= s.length <= 300
1 <= wordDict.length <= 1000
1 <= wordDict[i].length <= 20
s and wordDict[i] consist of only lowercase English letters.
All the strings of wordDict are unique.

---

140. Word Break II

Given a string s and a dictionary of strings wordDict, add spaces in s to construct a sentence where each word is a va lid dictionary word. Return all such possible sentences in any order.
Note that the same word in the dictionary may be reused multiple times in the segmentation.

Example 1:

Input: s = "catsanddog", wordDict = ["cat","cats","and","sand","dog"]
Output: ["cats and dog","cat sand dog"]

Example 2:

Input: s = "pineapplepenapple", wordDict = ["apple","pen","applepen","pine","pineapple"]
Output: ["pine apple pen apple","pineapple pen apple","pine applepen apple"]
Explanation: Note that you are allowed to reuse a dictionary word.

Example 3:

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

Constraints:

1 <= s.length <= 20

1 <= wordDict.length <= 1000
1 <= wordDict[i].length <= 10
s and wordDict[i] consist of only lowercase English letters.
All the strings of wordDict are unique.

---

141. Linked List Cycle

Given head, the head of a linked list, determine if the linked list has a cycle in it.
There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following t he next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to. Note th at pos is not passed as a parameter.
Return true if there is a cycle in the linked list. Otherwise, return false.

Example 1:

Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).

Example 2:

Input: head = [1,2], pos = 0
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.

Example 3:

Input: head = [1], pos = -1
Output: false
Explanation: There is no cycle in the linked list.

Constraints:

The number of the nodes in the list is in the range [0, 104].
-105 <= Node.val <= 105
pos is -1 or a valid index in the linked-list.

Follow up: Can you solve it using O(1) (i.e. constant) memory?

---

142. Linked List Cycle II

Given the head of a linked list, return the node where the cycle begins. If there is no cycle, return null.
There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following t he next pointer. Internally, pos is used to denote the index of the node that tail's next pointer is connected to (0-index ed). It is -1 if there is no cycle. Note that pos is not passed as a parameter.
Do not modify the linked list.

Example 1:

Input: head = [3,2,0,-4], pos = 1
Output: tail connects to node index 1
Explanation: There is a cycle in the linked list, where tail connects to the second node.

Example 2:

Input: head = [1,2], pos = 0
Output: tail connects to node index 0
Explanation: There is a cycle in the linked list, where tail connects to the first node.

Example 3:

Input: head = [1], pos = -1
Output: no cycle
Explanation: There is no cycle in the linked list.

Constraints:

The number of the nodes in the list is in the range [0, 104].
-105 <= Node.val <= 105
pos is -1 or a valid index in the linked-list.

Follow up: Can you solve it using O(1) (i.e. constant) memory?

---

143. Reorder List

You are given the head of a singly linked-list. The list can be represented as:

L0 → L1 → … → Ln - 1 → Ln

Reorder the list to be on the following form:

L0 → Ln → L1 → Ln - 1 → L2 → Ln - 2 → …

You may not modify the values in the list's nodes. Only nodes themselves may be changed.

Example 1:

Input: head = [1,2,3,4]
Output: [1,4,2,3]

Example 2:

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

Constraints:

The number of nodes in the list is in the range [1, 5 * 104].
1 <= Node.val <= 1000

---

144. Binary Tree Preorder Traversal

Given the root of a binary tree, return the preorder traversal of its nodes' values.

Example 1:

Input: root = [1,null,2,3]
Output: [1,2,3]

Example 2:

Input: root = []
Output: []

Example 3:

Input: root = [1]
Output: [1]

Example 4:

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

Example 5:

Input: root = [1,null,2]
Output: [1,2]

Constraints:

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

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

---

145. Binary Tree Postorder Traversal

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

Example 1:

Input: root = [1,null,2,3]
Output: [3,2,1]

Example 2:

Input: root = []
Output: []

Example 3:

Input: root = [1]
Output: [1]

Example 4:

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

Example 5:

Input: root = [1,null,2]
Output: [2,1]

Constraints:

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

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

---

146. LRU Cache

Design a data structure that follows the constraints of a Least Recently Used (LRU) cache.
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 cac
he. 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.

---

147. Insertion Sort List

Given the head of a singly linked list, sort the list using insertion sort, and return the sorted list's head.
The steps of the insertion sort algorithm:

Insertion sort iterates, consuming one input element each repetition and growing a sorted output list.
At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sort ed list and inserts it there.
It repeats until no input elements remain.

The following is a graphical example of the insertion sort algorithm. The partially sorted list (black) initially contain s only the first element in the list. One element (red) is removed from the input data and inserted in-place into the sor ted list with each iteration.

Example 1:

Input: head = [4,2,1,3]
Output: [1,2,3,4]

Example 2:

Input: head = [-1,5,3,4,0]
Output: [-1,0,3,4,5]

Constraints:

The number of nodes in the list is in the range [1, 5000].
-5000 <= Node.val <= 5000

---

148. Sort List

Given the head of a linked list, return the list after sorting it in ascending order.

Example 1:

Input: head = [4,2,1,3]
Output: [1,2,3,4]

Example 2:

Input: head = [-1,5,3,4,0]
Output: [-1,0,3,4,5]

Example 3:

Input: head = []
Output: []

Constraints:

The number of nodes in the list is in the range [0, 5 * 104].
-105 <= Node.val <= 105

Follow up: Can you sort the linked list in O(n logn) time and O(1) memory (i.e. constant space)?

---

149. Max Points on a Line

Given an array of points where points[i] = [xi, yi] represents a point on the X-Y plane, return the maximum number of points that lie on the same straight line.

Example 1:

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

Example 2:

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

Constraints:

1 <= points.length <= 300

points[i].length == 2
-104 <= xi, yi <= 104
All the points are unique.

---

150. Evaluate Reverse Polish Notation

Evaluate the value of an arithmetic expression in Reverse Polish Notation.
Valid operators are +, -, *, and /. Each operand may be an integer or another expression.
Note that division between two integers should truncate toward zero.
It is guaranteed that the given RPN expression is always valid. That means the expression would always evaluate to a result, and there will not be any division by zero operation.

Example 1:

Input: tokens = ["2","1","+","3","*"]
Output: 9
Explanation: ((2 + 1) * 3) = 9

Example 2:

Input: tokens = ["4","13","5","/","+"]
Output: 6
Explanation: (4 + (13 / 5)) = 6

Example 3:

Input: tokens = ["10","6","9","3","+","-11","*","/","*","17","+","5","+"]
Output: 22
Explanation: ((10 * (6 / ((9 + 3) * -11))) + 17) + 5
= ((10 * (6 / (12 * -11))) + 17) + 5
= ((10 * (6 / -132)) + 17) + 5
= ((10 * 0) + 17) + 5
= (0 + 17) + 5
= 17 + 5
= 22

Constraints:

1 <= tokens.length <= 104
tokens[i] is either an operator: "+", "-", "*", or "/", or an integer in the range [-200, 200].

---

151. Reverse Words in a String

Given an input string s, reverse the order of the words.
A word is defined as a sequence of non-space characters. The words in s will be separated by at least one space.
Return a string of the words in reverse order concatenated by a single space.
Note that s may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

Example 1:

Input: s = "the sky is blue"
Output: "blue is sky the"

Example 2:

Input: s = " hello world "
Output: "world hello"
Explanation: Your reversed string should not contain leading or trailing spaces.

Example 3:

Input: s = "a good   example"
Output: "example good a"
Explanation: You need to reduce multiple spaces between two words to a single space in the reversed string.

Example 4:

Input: s = " Bob   Loves Alice   "
Output: "Alice Loves Bob"

Example 5:

Input: s = "Alice does not even like bob"
Output: "bob like even not does Alice"

Constraints:

1 <= s.length <= 104
s contains English letters (upper-case and lower-case), digits, and spaces ' '.
There is at least one word in s.

Follow-up: If the string data type is mutable in your language, can you solve it in-place with O(1) extra space?

---

152. Maximum Product Subarray

Given an integer array nums, find a contiguous non-empty subarray within the array that has the largest product, and return the product.
It is guaranteed that the answer will fit in a 32-bit integer.
A subarray is a contiguous subsequence of the array.

Example 1:

Input: nums = [2,3,-2,4]
Output: 6
Explanation: [2,3] has the largest product 6.

Example 2:

Input: nums = [-2,0,-1]
Output: 0
Explanation: The result cannot be 2, because [-2,-1] is not a subarray.

Constraints:

1 <= nums.length <= 2 * 104
-10 <= nums[i] <= 10
The product of any prefix or suffix of nums is guaranteed to fit in a 32-bit integer.

---

153. Find Minimum in Rotated Sorted Array

Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array num s = [0,1,2,4,5,6,7] might become:

[4,5,6,7,0,1,2] if it was rotated 4 times.
[0,1,2,4,5,6,7] if it was rotated 7 times.

Notice that rotating an array [a[0], a[1], a[2], ..., a[n-1]] 1 time results in the array [a[n-1], a[0], a[1], a[2], ..., a[n-2]].
Given the sorted rotated array nums of unique elements, return the minimum element of this array.
You must write an algorithm that runs in O(log n) time.

Example 1:

Input: nums = [3,4,5,1,2]
Output: 1
Explanation: The original array was [1,2,3,4,5] rotated 3 times.

Example 2:

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

Output: 0
Explanation: The original array was [0,1,2,4,5,6,7] and it was rotated 4 times.

Example 3:

Input: nums = [11,13,15,17]
Output: 11
Explanation: The original array was [11,13,15,17] and it was rotated 4 times.

Constraints:

n == nums.length
1 <= n <= 5000
-5000 <= nums[i] <= 5000
All the integers of nums are unique.
nums is sorted and rotated between 1 and n times.

---

154. Find Minimum in Rotated Sorted Array II

Suppose an array of length n sorted in ascending order is rotated between 1 and n times. For example, the array num s = [0,1,4,4,5,6,7] might become:

[4,5,6,7,0,1,4] if it was rotated 4 times.
[0,1,4,4,5,6,7] if it was rotated 7 times.

Notice that rotating an array [a[0], a[1], a[2], ..., a[n-1]] 1 time results in the array [a[n-1], a[0], a[1], a[2], ..., a[n-2]].
Given the sorted rotated array nums that may contain duplicates, return the minimum element of this array.
You must decrease the overall operation steps as much as possible.

Example 1:
Input: nums = [1,3,5]
Output: 1
Example 2:
Input: nums = [2,2,2,0,1]
Output: 0

Constraints:

n == nums.length
1 <= n <= 5000
-5000 <= nums[i] <= 5000
nums is sorted and rotated between 1 and n times.

Follow up: This problem is similar to Find Minimum in Rotated Sorted Array, but nums may contain duplicates. Wo

uld this affect the runtime complexity? How and why?

---

155. Min Stack

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

MinStack() initializes the stack object.
void push(int val) pushes the element val onto the stack.
void pop() removes the element on the top of the stack.
int top() gets the top element of the stack.
int getMin() retrieves the minimum element in the stack.

Example 1:

Input
["MinStack","push","push","push","getMin","pop","top","getMin"]
[[],[-2],[0],[-3],[],[],[],[]]

Output
[null,null,null,null,-3,null,0,-2]

Explanation
MinStack minStack = new MinStack();
minStack.push(-2);
minStack.push(0);
minStack.push(-3);
minStack.getMin(); // return -3
minStack.pop();
minStack.top();   // return 0
minStack.getMin(); // return -2

Constraints:

-231 <= val <= 231 - 1
Methods pop, top and getMin operations will always be called on non-empty stacks.
At most 3 * 104 calls will be made to push, pop, top, and getMin.

---

160. Intersection of Two Linked Lists

Given the heads of two singly linked-lists headA and headB, return the node at which the two lists intersect. If the tw o linked lists have no intersection at all, return null.
For example, the following two linked lists begin to intersect at node c1:

The test cases are generated such that there are no cycles anywhere in the entire linked structure.
Note that the linked lists must retain their original structure after the function returns.
Custom Judge:
The inputs to the judge are given as follows (your program is not given these inputs):

intersectVal - The value of the node where the intersection occurs. This is 0 if there is no intersected node.
listA - The first linked list.
listB - The second linked list.
skipA - The number of nodes to skip ahead in listA (starting from the head) to get to the intersected node.
skipB - The number of nodes to skip ahead in listB (starting from the head) to get to the intersected node.

The judge will then create the linked structure based on these inputs and pass the two heads, headA and headB to yo ur program. If you correctly return the intersected node, then your solution will be accepted.

Example 1:

Input: intersectVal = 8, listA = [4,1,8,4,5], listB = [5,6,1,8,4,5], skipA = 2, skipB = 3
Output: Intersected at '8'
Explanation: The intersected node's value is 8 (note that this must not be 0 if the two lists intersect).
From the head of A, it reads as [4,1,8,4,5]. From the head of B, it reads as [5,6,1,8,4,5]. There are 2 nodes before the intersected node in A; There are 3 nodes before the intersected node in B.

Example 2:

Input: intersectVal = 2, listA = [1,9,1,2,4], listB = [3,2,4], skipA = 3, skipB = 1
Output: Intersected at '2'
Explanation: The intersected node's value is 2 (note that this must not be 0 if the two lists intersect).
From the head of A, it reads as [1,9,1,2,4]. From the head of B, it reads as [3,2,4]. There are 3 nodes before the inters ected node in A; There are 1 node before the intersected node in B.

Example 3:

Input: intersectVal = 0, listA = [2,6,4], listB = [1,5], skipA = 3, skipB = 2
Output: No intersection
Explanation: From the head of A, it reads as [2,6,4]. From the head of B, it reads as [1,5]. Since the two lists do not i ntersect, intersectVal must be 0, while skipA and skipB can be arbitrary values.
Explanation: The two lists do not intersect, so return null.

Constraints:

The number of nodes of listA is in the m.
The number of nodes of listB is in the n.
0 <= m, n <= 3 * 104
1 <= Node.val <= 105

0 <= skipA <= m
0 <= skipB <= n
intersectVal is 0 if listA and listB do not intersect.
intersectVal == listA[skipA] == listB[skipB] if listA and listB intersect.

Follow up: Could you write a solution that runs in O(n) time and use only O(1) memory?

---

162. Find Peak Element

A peak element is an element that is strictly greater than its neighbors.
Given an integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return th e index to any of the peaks.
You may imagine that nums[-1] = nums[n] = -∞.
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 t he peak element is 6.

Constraints:

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

---

164. Maximum Gap

Given an integer array nums, return the maximum difference between two successive elements in its sorted form. If t he array contains less than two elements, return 0.
You must write an algorithm that runs in linear time and uses linear extra space.

Example 1:

Input: nums = [3,6,9,1]
Output: 3
Explanation: The sorted form of the array is [1,3,6,9], either (3,6) or (6,9) has the maximum difference 3.

Example 2:

Input: nums = [10]
Output: 0
Explanation: The array contains less than 2 elements, therefore return 0.

Constraints:

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

---

165. Compare Version Numbers

Given two version numbers, version1 and version2, compare them.

Version numbers consist of one or more revisions joined by a dot '.'. Each revision consists of digits and may contain leading zeros. Every revision contains at least one character. Revisions are 0-indexed from left to right, with the left most revision being revision 0, the next revision being revision 1, and so on. For example 2.5.33 and 0.1 are valid ve rsion numbers.
To compare version numbers, compare their revisions in left-to-right order. Revisions are compared using their integ er value ignoring any leading zeros. This means that revisions 1 and 001 are considered equal. If a version number d oes not specify a revision at an index, then treat the revision as 0. For example, version 1.0 is less than version 1.1 be cause their revision 0s are the same, but their revision 1s are 0 and 1 respectively, and 0 < 1.
Return the following:

If version1 < version2, return -1.
If version1 > version2, return 1.
Otherwise, return 0.

Example 1:

Input: version1 = "1.01", version2 = "1.001"
Output: 0
Explanation: Ignoring leading zeroes, both "01" and "001" represent the same integer "1".

Example 2:

Input: version1 = "1.0", version2 = "1.0.0"
Output: 0

Explanation: version1 does not specify revision 2, which means it is treated as "0".

Example 3:

Input: version1 = "0.1", version2 = "1.1"
Output: -1
Explanation: version1's revision 0 is "0", while version2's revision 0 is "1". 0 < 1, so version1 < version2.

Example 4:

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

Example 5:

Input: version1 = "7.5.2.4", version2 = "7.5.3"
Output: -1

Constraints:

1 <= version1.length, version2.length <= 500
version1 and version2 only contain digits and '.'.
version1 and version2 are valid version numbers.
All the given revisions in version1 and version2 can be stored in a 32-bit integer.

---

166. Fraction to Recurring Decimal

Given two integers representing the numerator and denominator of a fraction, return the fraction in string format.
If the fractional part is repeating, enclose the repeating part in parentheses.
If multiple answers are possible, return any of them.
It is guaranteed that the length of the answer string is less than 104 for all the given inputs.

Example 1:
Input: numerator = 1, denominator = 2
Output: "0.5"
Example 2:
Input: numerator = 2, denominator = 1
Output: "2"
Example 3:
Input: numerator = 2, denominator = 3
Output: "0.(6)"
Example 4:
Input: numerator = 4, denominator = 333
Output: "0.(012)"
Example 5:
Input: numerator = 1, denominator = 5

Output: "0.2"

Constraints:

-231 <= numerator, denominator <= 231 - 1
denominator != 0

---

167. Two Sum II - Input Array Is Sorted

Given a 1-indexed array of integers numbers that is already sorted in non-decreasing order, find two numbers such th at they add up to a specific target number. Let these two numbers be numbers[index1] and numbers[index2] where 1 <= index1 < index2 <= numbers.length.
Return the indices of the two numbers, index1 and index2, added by one as an integer array [index1, index2] of lengt h 2.
The tests are generated such that there is exactly one solution. You may not use the same element twice.

Example 1:

Input: numbers = [2,7,11,15], target = 9
Output: [1,2]
Explanation: The sum of 2 and 7 is 9. Therefore, index1 = 1, index2 = 2. We return [1, 2].

Example 2:

Input: numbers = [2,3,4], target = 6
Output: [1,3]
Explanation: The sum of 2 and 4 is 6. Therefore index1 = 1, index2 = 3. We return [1, 3].

Example 3:

Input: numbers = [-1,0], target = -1
Output: [1,2]
Explanation: The sum of -1 and 0 is -1. Therefore index1 = 1, index2 = 2. We return [1, 2].

Constraints:

2 <= numbers.length <= 3 * 104
-1000 <= numbers[i] <= 1000
numbers is sorted in non-decreasing order.
-1000 <= target <= 1000
The tests are generated such that there is exactly one solution.

---

168. Excel Sheet Column Title

Given an integer columnNumber, return its corresponding column title as it appears in an Excel sheet.
For example:

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

Example 1:

Input: columnNumber = 1
Output: "A"

Example 2:

Input: columnNumber = 28
Output: "AB"

Example 3:

Input: columnNumber = 701
Output: "ZY"

Example 4:

Input: columnNumber = 2147483647
Output: "FXSHRXW"

Constraints:

1 <= columnNumber <= 231 - 1

---

169. Majority Element

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 elem ent 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
-231 <= nums[i] <= 231 - 1

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

---

171. Excel Sheet Column Number

Given a string columnTitle that represents the column title as appear in an Excel sheet, return its corresponding colu mn number.
For example:

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

Example 1:

Input: columnTitle = "A"
Output: 1

Example 2:

Input: columnTitle = "AB"
Output: 28

Example 3:

Input: columnTitle = "ZY"
Output: 701

Example 4:

Input: columnTitle = "FXSHRXW"
Output: 2147483647

Constraints:

1 <= columnTitle.length <= 7
columnTitle consists only of uppercase English letters.
columnTitle is in the range ["A", "FXSHRXW"].

---

172. Factorial Trailing Zeroes

Given an integer n, return the number of trailing zeroes in n!.
Note that n! = n * (n - 1) * (n - 2) * ... * 3 * 2 * 1.

Example 1:

Input: n = 3
Output: 0
Explanation: 3! = 6, no trailing zero.

Example 2:

Input: n = 5
Output: 1
Explanation: 5! = 120, one trailing zero.

Example 3:

Input: n = 0
Output: 0

Constraints:

0 <= n <= 104

Follow up: Could you write a solution that works in logarithmic time complexity?

---

173. Binary Search Tree Iterator

Implement the BSTIterator class that represents an iterator over the in-order traversal of a binary search tree (BST):

BSTIterator(TreeNode root) Initializes an object of the BSTIterator class. The root of the BST is given as part of the constructor. The pointer should be initialized to a non-existent number smaller than any element in the BST.
boolean hasNext() Returns true if there exists a number in the traversal to the right of the pointer, otherwise returns f alse.
int next() Moves the pointer to the right, then returns the number at the pointer.

Notice that by initializing the pointer to a non-existent smallest number, the first call to next() will return the smalles t element in the BST.
You may assume that next() calls will always be valid. That is, there will be at least a next number in the in-order tra versal when next() is called.

Example 1:

Input
["BSTIterator", "next", "next", "hasNext", "next", "hasNext", "next", "hasNext", "next", "hasNext"]
[[[7, 3, 15, null, null, 9, 20]], [], [], [], [], [], [], [], [], []]
Output
[null, 3, 7, true, 9, true, 15, true, 20, false]

Explanation
BSTIterator bSTIterator = new BSTIterator([7, 3, 15, null, null, 9, 20]);
bSTIterator.next();   // return 3
bSTIterator.next();   // return 7
bSTIterator.hasNext(); // return True
bSTIterator.next();   // return 9
bSTIterator.hasNext(); // return True
bSTIterator.next();   // return 15
bSTIterator.hasNext(); // return True
bSTIterator.next();   // return 20
bSTIterator.hasNext(); // return False

Constraints:

The number of nodes in the tree is in the range [1, 105].
0 <= Node.val <= 106
At most 105 calls will be made to hasNext, and next.

Follow up:

Could you implement next() and hasNext() to run in average O(1) time and use O(h) memory, where h is the height of the tree?

---

174. Dungeon Game

The demons had captured the princess and imprisoned her in the bottom-right corner of a dungeon. The dungeon consists of m x n rooms laid out in a 2D grid. Our valiant knight was initially positioned in the top-left room and must fight his way through dungeon to rescue the princess.
The knight has an initial health point represented by a positive integer. If at any point his health point drops to 0 or below, he dies immediately.
Some of the rooms are guarded by demons (represented by negative integers), so the knight loses health upon entering these rooms; other rooms are either empty (represented as 0) or contain magic orbs that increase the knight's health (represented by positive integers).
To reach the princess as quickly as possible, the knight decides to move only rightward or downward in each step.
Return the knight's minimum initial health so that he can rescue the princess.
Note that any room can contain threats or power-ups, even the first room the knight enters and the bottom-right room where the princess is imprisoned.

Example 1:

Input: dungeon = [[-2,-3,3],[-5,-10,1],[10,30,-5]]
Output: 7
Explanation: The initial health of the knight must be at least 7 if he follows the optimal path: RIGHT-> RIGHT -> DOWN -> DOWN.

Example 2:

Input: dungeon = [[0]]
Output: 1

Constraints:

m == dungeon.length
n == dungeon[i].length
1 <= m, n <= 200
-1000 <= dungeon[i][j] <= 1000

---

175. Combine Two Tables

Table: Person

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

Write an SQL query to report the first name, last name, city, and state of each person in the Person table. If the addre ss of a personId is not present in the Address table, report null instead.
Return the result table in any order.
The query 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.

---

176. Second Highest Salary

Table: Employee

```
+-------------+------+
| Column Name | Type |
+-------------+------+
| id          | int  |
| salary      | int  |
+-------------+------+
```

id is the primary key column for this table.
Each row of this table contains information about the salary of an employee.

Write an SQL query to report the second highest salary from the Employee table. If there is no second highest salary, the query should report null.
The query 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 |
+---------------------+

---

177. Nth Highest Salary

Table: Employee

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

Write an SQL query to report the nth highest salary from the Employee table. If there is no nth highest salary, the qu ery should report null.
The query result format is in the following example.

Example 1:

Input:
Employee table:
+----+--------+
| id | salary |
+----+--------+
1	100
2	200
3	300
+----+--------+
n = 2
Output:
+------------------------+
| getNthHighestSalary(2) |
+------------------------+
| 200 |
+------------------------+

Example 2:

Input:
Employee table:
+----+--------+
| id | salary |
+----+--------+
| 1 | 100 |
+----+--------+
n = 2
Output:
+------------------------+
| getNthHighestSalary(2) |
+------------------------+
| null |
+------------------------+

---

178. Rank Scores

Table: Scores

+-------------+---------+
| Column Name | Type |
+-------------+---------+
| id | int |
| score | decimal |
+-------------+---------+
id is the primary key for this table.
Each row of this table contains the score of a game. Score is a floating point value with two decimal places.

Write an SQL query to rank the scores. The ranking should be calculated according to the following rules:

The scores should be ranked from the highest to the lowest.
If there is a tie between two scores, both should have the same ranking.
After a tie, the next ranking number should be the next consecutive integer value. In other words, there should be no holes between ranks.

Return the result table ordered by score in descending order.
The query result format is in the following example.

Example 1:

Input:
Scores table:
+----+-------+
| id | score |
+----+-------+
| 1 | 3.50 |

2	3.65
3	4.00
4	3.85
5	4.00
6	3.65
+----+-------+
Output:
+-------+------+
| score | rank |
+-------+------+
4.00	1
4.00	1
3.85	2
3.65	3
3.65	3
3.50	4
+-------+------+

---

179. Largest Number

Given a list of non-negative integers nums, arrange them such that they form the largest number.
Note: The result may be very large, so you need to return a string instead of an integer.

Example 1:

Input: nums = [10,2]
Output: "210"

Example 2:

Input: nums = [3,30,34,5,9]
Output: "9534330"

Example 3:

Input: nums = [1]
Output: "1"

Example 4:

Input: nums = [10]
Output: "10"

Constraints:

1 <= nums.length <= 100

0 <= nums[i] <= 109

---

180. Consecutive Numbers

Table: Logs

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| num         | varchar |
+-------------+---------+
```
id is the primary key for this table.

Write an SQL query to find all numbers that appear at least three times consecutively.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Logs table:
```
+----+-----+
| id | num |
+----+-----+
| 1  | 1   |
| 2  | 1   |
| 3  | 1   |
| 4  | 2   |
| 5  | 1   |
| 6  | 2   |
| 7  | 2   |
+----+-----+
```
Output:
```
+-----------------+
| ConsecutiveNums |
+-----------------+
| 1               |
+-----------------+
```
Explanation: 1 is the only number that appears consecutively for at least three times.

---

181. Employees Earning More Than Their Managers

Table: Employee

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
| salary      | int     |
| managerId   | int     |
+-------------+---------+
```

id is the primary key column for this table.
Each row of this table indicates the ID of an employee, their name, salary, and the ID of their manager.

Write an SQL query to find the employees who earn more than their managers.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Employee table:

```
+----+-------+--------+-----------+
| id | name  | salary | managerId |
+----+-------+--------+-----------+
| 1  | Joe   | 70000  | 3         |
| 2  | Henry | 80000  | 4         |
| 3  | Sam   | 60000  | Null      |
| 4  | Max   | 90000  | Null      |
+----+-------+--------+-----------+
```

Output:

```
+----------+
| Employee |
+----------+
| Joe      |
+----------+
```

Explanation: Joe is the only employee who earns more than his manager.

---

182. Duplicate Emails

Table: Person

+-------------+---------+
| Column Name | Type |
+-------------+---------+
| id | int |
| email | varchar |
+-------------+---------+
id is the primary key column for this table.
Each row of this table contains an email. The emails will not contain uppercase letters.

Write an SQL query to report all the duplicate emails.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Person table:
+----+---------+
| id | email |
+----+---------+
1	a@b.com
2	c@d.com
3	a@b.com
+----+---------+
Output:
+---------+
| Email |
+---------+
| a@b.com |
+---------+
Explanation: a@b.com is repeated two times.

---

183. Customers Who Never Order

Table: Customers

+-------------+---------+
| Column Name | Type |
+-------------+---------+
| id | int |
| name | varchar |
+-------------+---------+
id is the primary key column for this table.
Each row of this table indicates the ID and name of a customer.

Table: Orders

+-------------+------+
| Column Name | Type |
+-------------+------+
| id | int |
| customerId | int |
+-------------+------+
id is the primary key column for this table.
customerId is a foreign key of the ID from the Customers table.
Each row of this table indicates the ID of an order and the ID of the customer who ordered it.

Write an SQL query to report all customers who never order anything.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Customers table:
+----+-------+
| id | name |
+----+-------+
1	Joe
2	Henry
3	Sam
4	Max
+----+-------+
Orders table:
+----+------------+
| id | customerId |
+----+------------+
| 1 | 3 |
| 2 | 1 |
+----+------------+
Output:
+-----------+
| Customers |
+-----------+
| Henry |
| Max |
+-----------+

---

184. Department Highest Salary

Table: Employee

+--------------+---------+
| Column Name | Type |
+--------------+---------+
id	int
name	varchar
salary	int
departmentId	int
+--------------+---------+
id is the primary key column for this table.
departmentId is a foreign key of the ID from the Department table.
Each row of this table indicates the ID, name, and salary of an employee. It also contains the ID of their department.

Table: Department

+-------------+---------+
| Column Name | Type |
+-------------+---------+
| id | int |
| name | varchar |
+-------------+---------+
id is the primary key column for this table.
Each row of this table indicates the ID of a department and its name.

Write an SQL query to find employees who have the highest salary in each of the departments.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Employee table:
+----+-------+--------+--------------+
| id | name | salary | departmentId |
+----+-------+--------+--------------+
1	Joe	70000	1
2	Jim	90000	1
3	Henry	80000	2
4	Sam	60000	2
5	Max	90000	1
+----+-------+--------+--------------+
Department table:
+----+-------+
| id | name |
+----+-------+
| 1 | IT |
| 2 | Sales |
+----+-------+
Output:
+------------+----------+--------+
| Department | Employee | Salary |
+------------+----------+--------+
| IT | Jim | 90000 |

| Sales | Henry | 80000 |
| IT | Max | 90000 |
+------------+----------+--------+
Explanation: Max and Jim both have the highest salary in the IT department and Henry has the highest salary in the Sales department.

---

185. Department Top Three Salaries

Table: Employee

+--------------+---------+
| Column Name | Type |
+--------------+---------+
id	int
name	varchar
salary	int
departmentId	int
+--------------+---------+
id is the primary key column for this table.
departmentId is a foreign key of the ID from the Department table.
Each row of this table indicates the ID, name, and salary of an employee. It also contains the ID of their department.

Table: Department

+-------------+---------+
| Column Name | Type |
+-------------+---------+
| id | int |
| name | varchar |
+-------------+---------+
id is the primary key column for this table.
Each row of this table indicates the ID of a department and its name.

A company's executives are interested in seeing who earns the most money in each of the company's departments. A high earner in a department is an employee who has a salary in the top three unique salaries for that department.
Write an SQL query to find the employees who are high earners in each of the departments.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Employee table:
+----+-------+--------+--------------+
| id | name | salary | departmentId |

```
+----+-------+--------+--------------+
| 1  | Joe   | 85000  | 1            |
| 2  | Henry | 80000  | 2            |
| 3  | Sam   | 60000  | 2            |
| 4  | Max   | 90000  | 1            |
| 5  | Janet | 69000  | 1            |
| 6  | Randy | 85000  | 1            |
| 7  | Will  | 70000  | 1            |
+----+-------+--------+--------------+
```
Department table:
```
+----+-------+
| id | name  |
+----+-------+
| 1  | IT    |
| 2  | Sales |
+----+-------+
```
Output:
```
+------------+----------+--------+
| Department | Employee | Salary |
+------------+----------+--------+
| IT         | Max      | 90000  |
| IT         | Joe      | 85000  |
| IT         | Randy    | 85000  |
| IT         | Will     | 70000  |
| Sales      | Henry    | 80000  |
| Sales      | Sam      | 60000  |
+------------+----------+--------+
```
Explanation:
In the IT department:
- Max earns the highest unique salary
- Both Randy and Joe earn the second-highest unique salary
- Will earns the third-highest unique salary

In the Sales department:
- Henry earns the highest salary
- Sam earns the second-highest salary
- There is no third-highest salary as there are only two employees

---

187. Repeated DNA Sequences

The DNA sequence is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'.

For example, "ACGAATTCCG" is a DNA sequence.

When studying DNA, it is useful to identify repeated sequences within the DNA.
Given a string s that represents a DNA sequence, return all the 10-letter-long sequences (substrings) that occur more than once in a DNA molecule. You may return the answer in any order.

Example 1:
Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"
Output: ["AAAAACCCCC","CCCCCAAAAA"]
Example 2:
Input: s = "AAAAAAAAAAAAA"
Output: ["AAAAAAAAAA"]

Constraints:

1 <= s.length <= 105
s[i] is either 'A', 'C', 'G', or 'T'.

---

188. Best Time to Buy and Sell Stock IV

You are given an integer array prices where prices[i] is the price of a given stock on the ith day, and an integer k.
Find the maximum profit you can achieve. You may complete at most k transactions.
Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy agai n).

Example 1:

Input: k = 2, prices = [2,4,1]
Output: 2
Explanation: Buy on day 1 (price = 2) and sell on day 2 (price = 4), profit = 4-2 = 2.

Example 2:

Input: k = 2, prices = [3,2,6,5,0,3]
Output: 7
Explanation: Buy on day 2 (price = 2) and sell on day 3 (price = 6), profit = 6-2 = 4. Then buy on day 5 (price = 0) a nd sell on day 6 (price = 3), profit = 3-0 = 3.

Constraints:

0 <= k <= 100
0 <= prices.length <= 1000
0 <= prices[i] <= 1000

---

189. Rotate Array

Given an array, rotate the array to the right by k steps, where k is non-negative.

Example 1:

Input: nums = [1,2,3,4,5,6,7], k = 3
Output: [5,6,7,1,2,3,4]
Explanation:
rotate 1 steps to the right: [7,1,2,3,4,5,6]
rotate 2 steps to the right: [6,7,1,2,3,4,5]
rotate 3 steps to the right: [5,6,7,1,2,3,4]

Example 2:

Input: nums = [-1,-100,3,99], k = 2
Output: [3,99,-1,-100]
Explanation:
rotate 1 steps to the right: [99,-1,-100,3]
rotate 2 steps to the right: [3,99,-1,-100]

Constraints:

1 <= nums.length <= 105
-231 <= nums[i] <= 231 - 1
0 <= k <= 105

Follow up:

Try to come up with as many solutions as you can. There are at least three different ways to solve this problem.
Could you do it in-place with O(1) extra space?

---

190. Reverse Bits

Reverse bits of a given 32 bits unsigned integer.
Note:

Note that in some languages, such as Java, there is no unsigned integer type. In this case, both input and output will be given as a signed integer type. They should not affect your implementation, as the integer's internal binary represe ntation is the same, whether it is signed or unsigned.
In Java, the compiler represents the signed integers using 2's complement notation. Therefore, in Example 2 above, t he input represents the signed integer -3 and the output represents the signed integer -1073741825.

Example 1:

Input: n = 00000010100101000001111010011100
Output: 964176192 (00111001011110000010100101000000)
Explanation: The input binary string 00000010100101000001111010011100 represents the unsigned integer 432615 96, so return 964176192 which its binary representation is 00111001011110000010100101000000.

Example 2:

Input: n = 11111111111111111111111111111101
Output: 3221225471 (10111111111111111111111111111111)
Explanation: The input binary string 11111111111111111111111111111101 represents the unsigned integer 429496 7293, so return 3221225471 which its binary representation is 10111111111111111111111111111111.

Constraints:

The input must be a binary string of length 32

Follow up: If this function is called many times, how would you optimize it?

---

191. Number of 1 Bits

Write a function that takes an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight).
Note:

Note that in some languages, such as Java, there is no unsigned integer type. In this case, the input will be given as a signed integer type. It should not affect your implementation, as the integer's internal binary representation is the sa me, whether it is signed or unsigned.
In Java, the compiler represents the signed integers using 2's complement notation. Therefore, in Example 3, the inp ut represents the signed integer. -3.

Example 1:

Input: n = 00000000000000000000000000001011
Output: 3
Explanation: The input binary string 00000000000000000000000000001011 has a total of three '1' bits.

Example 2:

Input: n = 00000000000000000000000010000000
Output: 1
Explanation: The input binary string 00000000000000000000000010000000 has a total of one '1' bit.

Example 3:

Input: n = 11111111111111111111111111111101
Output: 31
Explanation: The input binary string 11111111111111111111111111111101 has a total of thirty one '1' bits.

Constraints:

The input must be a binary string of length 32.

Follow up: If this function is called many times, how would you optimize it?

---

192. Word Frequency

Write a bash script to calculate the frequency of each word in a text file words.txt.
For simplicity sake, you may assume:

words.txt contains only lowercase characters and space ' ' characters.
Each word must consist of lowercase characters only.
Words are separated by one or more whitespace characters.

Example:
Assume that words.txt has the following content:

the day is sunny the the
the sunny is is

Your script should output the following, sorted by descending frequency:

the 4
is 3
sunny 2
day 1

Note:

Don't worry about handling ties, it is guaranteed that each word's frequency count is unique.
Could you write it in one-line using Unix pipes?

---

193. Valid Phone Numbers

Given a text file file.txt that contains a list of phone numbers (one per line), write a one-liner bash script to print all v
alid phone numbers.
You may assume that a valid phone number must appear in one of the following two formats: (xxx) xxx-xxxx or xxx
-xxx-xxxx. (x means a digit)
You may also assume each line in the text file must not contain leading or trailing white spaces.
Example:
Assume that file.txt has the following content:

987-123-4567
123 456 7890
(123) 456-7890

Your script should output the following valid phone numbers:

987-123-4567
(123) 456-7890

---

194. Transpose File

Given a text file file.txt, transpose its content.
You may assume that each row has the same number of columns, and each field is separated by the ' ' character.
Example:
If file.txt has the following content:

name age
alice 21
ryan 30

Output the following:

name alice ryan
age 21 30

---

195. Tenth Line

Given a text file file.txt, print just the 10th line of the file.
Example:

Assume that file.txt has the following content:

Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10

Your script should output the tenth line, which is:

Line 10

Note:
1. If the file contains less than 10 lines, what should you output?
2. There's at least three different solutions. Try to explore all possibilities.

---

196. Delete Duplicate Emails

Table: Person

+-------------+---------+
| Column Name | Type |
+-------------+---------+
| id | int |
| email | varchar |
+-------------+---------+
id is the primary key column for this table.
Each row of this table contains an email. The emails will not contain uppercase letters.

Write an SQL query to delete all the duplicate emails, keeping only one unique email with the smallest id.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Person table:
+----+------------------+
| id | email |
+----+------------------+
| 1 | john@example.com |
| 2 | bob@example.com |

| 3 | john@example.com |
+----+------------------+
Output:
+----+------------------+
| id | email |
+----+------------------+
| 1 | john@example.com |
| 2 | bob@example.com |
+----+------------------+
Explanation: john@example.com is repeated two times. We keep the row with the smallest Id = 1.

---

197. Rising Temperature

Table: Weather

+---------------+---------+
| Column Name | Type |
+---------------+---------+
id	int
recordDate	date
temperature	int
+---------------+---------+
id is the primary key for this table.
This table contains information about the temperature on a certain day.

Write an SQL query to find all dates' Id with higher temperatures compared to its previous dates (yesterday).
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Weather table:
+----+------------+-------------+
| id | recordDate | temperature |
+----+------------+-------------+
1	2015-01-01	10
2	2015-01-02	25
3	2015-01-03	20
4	2015-01-04	30
+----+------------+-------------+
Output:
+----+
| id |
+----+
| 2 |

| 4 |
+----+
Explanation:
In 2015-01-02, the temperature was higher than the previous day (10 -> 25).
In 2015-01-04, the temperature was higher than the previous day (20 -> 30).

---

198. House Robber

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stash ed, the only constraint stopping you from robbing each of them is that adjacent houses have security systems connect ed 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 mone y 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

---

199. Binary Tree Right Side View

Given the root of a binary tree, imagine yourself standing on the right side of it, return the values of the nodes you ca n see ordered from top to bottom.

Example 1:

Input: root = [1,2,3,null,5,null,4]
Output: [1,3,4]

Example 2:

Input: root = [1,null,3]
Output: [1,3]

Example 3:

Input: root = []
Output: []

Constraints:

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

---

200. Number of Islands

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 island s.
An island is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may ass ume 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'.

---

201. Bitwise AND of Numbers Range

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

---

202. Happy Number

Write an algorithm to determine if a number n is happy.
A happy number is a number defined by the following process:

Starting with any positive integer, replace the number by the sum of the squares of its digits.
Repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not incl ude 1.
Those numbers for which this process ends in 1 are happy.

Return true if n is a happy number, and false if not.

Example 1:

Input: n = 19
Output: true
Explanation:
12 + 92 = 82
82 + 22 = 68
62 + 82 = 100
12 + 02 + 02 = 1

Example 2:

Input: n = 2
Output: false

Constraints:

1 <= n <= 231 - 1

---

203. Remove Linked List Elements

Given the head of a linked list and an integer val, remove all the nodes of the linked list that has Node.val == val, an d return the new head.

Example 1:

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

Example 2:

Input: head = [], val = 1
Output: []

Example 3:

Input: head = [7,7,7,7], val = 7

Output: []

Constraints:

The number of nodes in the list is in the range [0, 104].
1 <= Node.val <= 50
0 <= val <= 50

---

204. Count Primes

Given an integer n, return the number of prime numbers that are strictly less than n.

Example 1:

Input: n = 10
Output: 4
Explanation: There are 4 prime numbers less than 10, they are 2, 3, 5, 7.

Example 2:

Input: n = 0
Output: 0

Example 3:

Input: n = 1
Output: 0

Constraints:

0 <= n <= 5 * 106

---

205. Isomorphic Strings

Given two strings s and t, determine if they are isomorphic.
Two strings s and t are isomorphic if the characters in s can be replaced to get t.
All occurrences of a character must be replaced with another character while preserving the order of characters. No t

wo characters may map to the same character, but a character may map to itself.

Example 1:
Input: s = "egg", t = "add"
Output: true
Example 2:
Input: s = "foo", t = "bar"
Output: false
Example 3:
Input: s = "paper", t = "title"
Output: true

Constraints:

1 <= s.length <= 5 * 104
t.length == s.length
s and t consist of any valid ascii character.

---

206. Reverse Linked List

Given the head of a singly linked list, reverse the list, and return the reversed list.

Example 1:

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

Example 2:

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

Example 3:

Input: head = []
Output: []

Constraints:

The number of nodes in the list is the range [0, 5000].
-5000 <= Node.val <= 5000

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

---

207. Course Schedule

There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai.

For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1.

Return true if you can finish all courses. Otherwise, return false.

Example 1:

Input: numCourses = 2, prerequisites = [[1,0]]
Output: true
Explanation: There are a total of 2 courses to take.
To take course 1 you should have finished course 0. So it is possible.

Example 2:

Input: numCourses = 2, prerequisites = [[1,0],[0,1]]
Output: false
Explanation: There are a total of 2 courses to take.
To take course 1 you should have finished course 0, and to take course 0 you should also have finished course 1. So i t is impossible.

Constraints:

1 <= numCourses <= 105
0 <= prerequisites.length <= 5000
prerequisites[i].length == 2
0 <= ai, bi < numCourses
All the pairs prerequisites[i] are unique.

---

208. Implement Trie (Prefix Tree)

A trie (pronounced as "try") or prefix tree is a tree data structure used to efficiently store and retrieve keys in a datas et of strings. There are various applications of this data structure, such as autocomplete and spellchecker.
Implement the Trie class:

Trie() Initializes the trie object.
void insert(String word) Inserts the string word into the trie.
boolean search(String word) Returns true if the string word is in the trie (i.e., was inserted before), and false otherwi
se.
boolean startsWith(String prefix) Returns true if there is a previously inserted string word that has the prefix prefix,
and false otherwise.

Example 1:

Input
["Trie", "insert", "search", "search", "startsWith", "insert", "search"]
[[], ["apple"], ["apple"], ["app"], ["app"], ["app"], ["app"]]
Output
[null, null, true, false, true, null, true]

Explanation
Trie trie = new Trie();
trie.insert("apple");
trie.search("apple");   // return True
trie.search("app");   // return False
trie.startsWith("app"); // return True
trie.insert("app");
trie.search("app");   // return True

Constraints:

1 <= word.length, prefix.length <= 2000
word and prefix consist only of lowercase English letters.
At most 3 * 104 calls in total will be made to insert, search, and startsWith.

---

209. Minimum Size Subarray Sum

Given an array of positive integers nums and a positive integer target, return the minimal length of a contiguous suba
rray [numsl, numsl+1, ..., numsr-1, numsr] of which the sum is greater than or equal to target. If there is no such sub
array, return 0 instead.

Example 1:

Input: target = 7, nums = [2,3,1,2,4,3]
Output: 2
Explanation: The subarray [4,3] has the minimal length under the problem constraint.

Example 2:

Input: target = 4, nums = [1,4,4]
Output: 1

Example 3:

Input: target = 11, nums = [1,1,1,1,1,1,1,1]
Output: 0

Constraints:

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

Follow up: If you have figured out the O(n) solution, try coding another solution of which the time complexity is O(n log(n)).

---

210. Course Schedule II

There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course bi first if you want to take course ai.

For example, the pair [0, 1], indicates that to take course 0 you have to first take course 1.

Return the ordering of courses you should take to finish all courses. If there are many valid answers, return any of th em. If it is impossible to finish all courses, return an empty array.

Example 1:

Input: numCourses = 2, prerequisites = [[1,0]]
Output: [0,1]
Explanation: There are a total of 2 courses to take. To take course 1 you should have finished course 0. So the correc t course order is [0,1].

Example 2:

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

Example 3:

Input: numCourses = 1, prerequisites = []
Output: [0]

Constraints:

1 <= numCourses <= 2000
0 <= prerequisites.length <= numCourses * (numCourses - 1)
prerequisites[i].length == 2
0 <= ai, bi < numCourses
ai != bi
All the pairs [ai, bi] are distinct.

---

211. Design Add and Search Words Data Structure

Design a data structure that supports adding new words and finding if a string matches any previously added string. Implement the WordDictionary class:

WordDictionary() Initializes the object.
void addWord(word) Adds word to the data structure, it can be matched later.
bool search(word) Returns true if there is any string in the data structure that matches word or false otherwise. word may contain dots '.' where dots can be matched with any letter.

Example:

Input
["WordDictionary","addWord","addWord","addWord","search","search","search","search"]
[[],["bad"],["dad"],["mad"],["pad"],["bad"],[".ad"],["b.."]]
Output
[null,null,null,null,false,true,true,true]

Explanation
WordDictionary wordDictionary = new WordDictionary();
wordDictionary.addWord("bad");
wordDictionary.addWord("dad");
wordDictionary.addWord("mad");
wordDictionary.search("pad"); // return False
wordDictionary.search("bad"); // return True
wordDictionary.search(".ad"); // return True
wordDictionary.search("b.."); // return True

Constraints:

1 <= word.length <= 500
word in addWord consists lower-case English letters.
word in search consist of '.' or lower-case English letters.
At most 50000 calls will be made to addWord and search.

---

212. Word Search II

Given an m x n board of characters and a list of strings words, return all words on the board.
Each word must be constructed from letters of sequentially adjacent cells, where adjacent cells are horizontally or ve rtically neighboring. The same letter cell may not be used more than once in a word.

Example 1:

Input: board = [["o","a","a","n"],["e","t","a","e"],["i","h","k","r"],["i","f","l","v"]], words = ["oath","pea","eat","rain"]
Output: ["eat","oath"]

Example 2:

Input: board = [["a","b"],["c","d"]], words = ["abcb"]
Output: []

Constraints:

m == board.length
n == board[i].length
1 <= m, n <= 12
board[i][j] is a lowercase English letter.
1 <= words.length <= 3 * 104
1 <= words[i].length <= 10
words[i] consists of lowercase English letters.
All the strings of words are unique.

---

213. House Robber II

You are a professional robber planning to rob houses along a street. Each house has a certain amount of money stash ed. All houses at this place are arranged in a circle. That means the first house is the neighbor of the last one. Meanw hile, adjacent houses have a security system connected, and it will automatically contact the police if two adjacent ho uses 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 mone y you can rob tonight without alerting the police.

Example 1:

Input: nums = [2,3,2]
Output: 3
Explanation: You cannot rob house 1 (money = 2) and then rob house 3 (money = 2), because they are adjacent hous
es.

Example 2:

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 3:

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

Constraints:

1 <= nums.length <= 100
0 <= nums[i] <= 1000

---

214. Shortest Palindrome

You are given a string s. You can convert s to a palindrome by adding characters in front of it.
Return the shortest palindrome you can find by performing this transformation.

Example 1:
Input: s = "aacecaaa"
Output: "aaacecaaa"
Example 2:
Input: s = "abcd"
Output: "dcbabcd"

Constraints:

0 <= s.length <= 5 * 104
s consists of lowercase English letters only.

---

215. Kth Largest Element in an Array

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.

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 <= 104
-104 <= nums[i] <= 104

---

216. Combination Sum III

Find all valid combinations of k numbers that sum up to n such that the following conditions are true:

Only numbers 1 through 9 are used.
Each number is used at most once.

Return a list of all possible valid combinations. The list must not contain the same combination twice, and the combi nations may be returned in any order.

Example 1:

Input: k = 3, n = 7
Output: [[1,2,4]]
Explanation:
1 + 2 + 4 = 7
There are no other valid combinations.
Example 2:

Input: k = 3, n = 9
Output: [[1,2,6],[1,3,5],[2,3,4]]
Explanation:
1 + 2 + 6 = 9
1 + 3 + 5 = 9

2 + 3 + 4 = 9
There are no other valid combinations.

Example 3:

Input: k = 4, n = 1
Output: []
Explanation: There are no valid combinations.
Using 4 different numbers in the range [1,9], the smallest sum we can get is 1+2+3+4 = 10 and since 10 > 1, there ar e no valid combination.

Example 4:

Input: k = 3, n = 2
Output: []
Explanation: There are no valid combinations.

Example 5:

Input: k = 9, n = 45
Output: [[1,2,3,4,5,6,7,8,9]]
Explanation:
1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 = 45
There are no other valid combinations.

Constraints:

2 <= k <= 9
1 <= n <= 60

---

217. Contains Duplicate

Given an integer array nums, return true if any value appears at least twice in the array, and return false if every elem ent 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

---

218. The Skyline Problem

A city's skyline is the outer contour of the silhouette formed by all the buildings in that city when viewed from a dist ance. Given the locations and heights of all the buildings, return the skyline formed by these buildings collectively. The geometric information of each building is given in the array buildings where buildings[i] = [lefti, righti, heighti]:

lefti is the x coordinate of the left edge of the ith building.
righti is the x coordinate of the right edge of the ith building.
heighti is the height of the ith building.

You may assume all buildings are perfect rectangles grounded on an absolutely flat surface at height 0.
The skyline should be represented as a list of "key points" sorted by their x-coordinate in the form [[x1,y1],[x2,y2],.. .]. Each key point is the left endpoint of some horizontal segment in the skyline except the last point in the list, whic h always has a y-coordinate 0 and is used to mark the skyline's termination where the rightmost building ends. Any g round between the leftmost and rightmost buildings should be part of the skyline's contour.
Note: There must be no consecutive horizontal lines of equal height in the output skyline. For instance, [...,[2 3],[4 5],[7 5],[11 5],[12 7],...] is not acceptable; the three lines of height 5 should be merged into one in the final output as s uch: [...,[2 3],[4 5],[12 7],...]

Example 1:

Input: buildings = [[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]
Output: [[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
Explanation:
Figure A shows the buildings of the input.
Figure B shows the skyline formed by those buildings. The red points in figure B represent the key points in the outp ut list.

Example 2:

Input: buildings = [[0,2,3],[2,5,3]]
Output: [[0,3],[5,0]]

Constraints:

1 <= buildings.length <= 104
0 <= lefti < righti <= 231 - 1
1 <= heighti <= 231 - 1
buildings is sorted by lefti in non-decreasing order.

---

219. Contains Duplicate II

Given an integer array nums and an integer k, return true if there are two distinct indices i and j in the array such that nums[i] == nums[j] and abs(i - j) <= k.

Example 1:

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

Example 2:

Input: nums = [1,0,1,1], k = 1
Output: true

Example 3:

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

Constraints:

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

---

220. Contains Duplicate III

Given an integer array nums and two integers k and t, return true if there are two distinct indices i and j in the array s uch that abs(nums[i] - nums[j]) <= t and abs(i - j) <= k.

Example 1:
Input: nums = [1,2,3,1], k = 3, t = 0
Output: true
Example 2:
Input: nums = [1,0,1,1], k = 1, t = 2
Output: true
Example 3:

Input: nums = [1,5,9,1,5,9], k = 2, t = 3
Output: false

Constraints:

1 <= nums.length <= 2 * 104
-231 <= nums[i] <= 231 - 1
0 <= k <= 104
0 <= t <= 231 - 1

---

221. Maximal Square

Given an m x n binary matrix filled with 0's and 1's, find the largest square containing only 1's and return its area.

Example 1:

Input: matrix = [["1","0","1","0","0"],["1","0","1","1","1"],["1","1","1","1","1"],["1","0","0","1","0"]]
Output: 4

Example 2:

Input: matrix = [["0","1"],["1","0"]]
Output: 1

Example 3:

Input: matrix = [["0"]]
Output: 0

Constraints:

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

---

222. Count Complete Tree Nodes

Given the root of a complete binary tree, return the number of the nodes in the tree.
According to Wikipedia, every level, except possibly the last, is completely filled in a complete binary tree, and all n odes in the last level are as far left as possible. It can have between 1 and 2h nodes inclusive at the last level h.
Design an algorithm that runs in less than O(n) time complexity.

Example 1:

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

Example 2:

Input: root = []
Output: 0

Example 3:

Input: root = [1]
Output: 1

Constraints:

The number of nodes in the tree is in the range [0, 5 * 104].
0 <= Node.val <= 5 * 104
The tree is guaranteed to be complete.

---

223. Rectangle Area

Given the coordinates of two rectilinear rectangles in a 2D plane, return the total area covered by the two rectangles.
The first rectangle is defined by its bottom-left corner (ax1, ay1) and its top-right corner (ax2, ay2).
The second rectangle is defined by its bottom-left corner (bx1, by1) and its top-right corner (bx2, by2).

Example 1:

Input: ax1 = -3, ay1 = 0, ax2 = 3, ay2 = 4, bx1 = 0, by1 = -1, bx2 = 9, by2 = 2
Output: 45

Example 2:

Input: ax1 = -2, ay1 = -2, ax2 = 2, ay2 = 2, bx1 = -2, by1 = -2, bx2 = 2, by2 = 2
Output: 16

Constraints:

-104 <= ax1, ay1, ax2, ay2, bx1, by1, bx2, by2 <= 104

---

224. Basic Calculator

Given a string s representing a valid expression, implement a basic calculator to evaluate it, and return the result of t he evaluation.
Note: You are not allowed to use any built-in function which evaluates strings as mathematical expressions, such as eval().

Example 1:

Input: s = "1 + 1"
Output: 2

Example 2:

Input: s = " 2-1 + 2 "
Output: 3

Example 3:

Input: s = "(1+(4+5+2)-3)+(6+8)"
Output: 23

Constraints:

1 <= s.length <= 3 * 105
s consists of digits, '+', '-', '(', ')', and ' '.
s represents a valid expression.
'+' is not used as a unary operation (i.e., "+1" and "+(2 + 3)" is invalid).
'-' could be used as a unary operation (i.e., "-1" and "-(2 + 3)" is valid).
There will be no two consecutive operators in the input.
Every number and running calculation will fit in a signed 32-bit integer.

---

225. Implement Stack using Queues

Implement a last-in-first-out (LIFO) stack using only two queues. The implemented stack should support all the func tions of a normal stack (push, top, pop, and empty).
Implement the MyStack class:

void push(int x) Pushes element x to the top of the stack.
int pop() Removes the element on the top of the stack and returns it.
int top() Returns the element on the top of the stack.
boolean empty() Returns true if the stack is empty, false otherwise.

Notes:

You must use only standard operations of a queue, which means that only push to back, peek/pop from front, size an d is empty operations are valid.
Depending on your language, the queue may not be supported natively. You may simulate a queue using a list or deq ue (double-ended queue) as long as you use only a queue's standard operations.

Example 1:

Input
["MyStack", "push", "push", "top", "pop", "empty"]
[[], [1], [2], [], [], []]
Output
[null, null, null, 2, 2, false]

Explanation
MyStack myStack = new MyStack();
myStack.push(1);
myStack.push(2);
myStack.top(); // return 2
myStack.pop(); // return 2
myStack.empty(); // return False

Constraints:

1 <= x <= 9
At most 100 calls will be made to push, pop, top, and empty.
All the calls to pop and top are valid.

Follow-up: Can you implement the stack using only one queue?

---

226. Invert Binary Tree

Given the root of a binary tree, invert the tree, and return its root.

Example 1:

Input: root = [4,2,7,1,3,6,9]
Output: [4,7,2,9,6,3,1]

Example 2:

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

Example 3:

Input: root = []
Output: []

Constraints:

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

---

227. Basic Calculator II

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.

---

228. Summary Ranges

You are given a sorted unique integer array nums.
Return the smallest sorted list of ranges that cover all the numbers in the array exactly. That is, each element of num s is covered by exactly one of the ranges, and there is no integer x such that x is in one of the ranges but not in nums.
Each range [a,b] in the list should be output as:

"a->b" if a != b
"a" if a == b

Example 1:

Input: nums = [0,1,2,4,5,7]
Output: ["0->2","4->5","7"]
Explanation: The ranges are:
[0,2] --> "0->2"
[4,5] --> "4->5"
[7,7] --> "7"

Example 2:

Input: nums = [0,2,3,4,6,8,9]
Output: ["0","2->4","6","8->9"]
Explanation: The ranges are:
[0,0] --> "0"
[2,4] --> "2->4"
[6,6] --> "6"
[8,9] --> "8->9"

Example 3:

Input: nums = []
Output: []

Example 4:

Input: nums = [-1]
Output: ["-1"]

Example 5:

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

Constraints:

0 <= nums.length <= 20
-231 <= nums[i] <= 231 - 1
All the values of nums are unique.
nums is sorted in ascending order.

---

229. Majority Element II

Given an integer array of size n, find all elements that appear more than ⌊ n/3 ⌋ times.

Example 1:

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

Example 2:

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

Example 3:

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

Constraints:

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

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

---

230. Kth Smallest Element in a BST

Given the root of a binary search tree, and an integer k, return the kth smallest value (1-indexed) of all the values of t he nodes in the tree.

Example 1:

Input: root = [3,1,4,null,2], k = 1
Output: 1

Example 2:

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

Constraints:

The number of nodes in the tree is n.
1 <= k <= n <= 104
0 <= Node.val <= 104

Follow up: If the BST is modified often (i.e., we can do insert and delete operations) and you need to find the kth sm allest frequently, how would you optimize?

---

231. Power of Two

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

Example 4:

Input: n = 4
Output: true

Example 5:

Input: n = 5
Output: false

Constraints:

-231 <= n <= 231 - 1

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

---

232. Implement Queue using Stacks

Implement a first in first out (FIFO) queue using only two stacks. The implemented queue should support all the fun ctions of a normal queue (push, peek, pop, and empty).
Implement the MyQueue class:

void push(int x) Pushes element x to the back of the queue.
int pop() Removes the element from the front of the queue and returns it.
int peek() Returns the element at the front of the queue.
boolean empty() Returns true if the queue is empty, false otherwise.

Notes:

You must use only standard operations of a stack, which means only push to top, peek/pop from top, size, and is em pty operations are valid.
Depending on your language, the stack may not be supported natively. You may simulate a stack using a list or dequ e (double-ended queue) as long as you use only a stack's standard operations.

Example 1:

Input
["MyQueue", "push", "push", "peek", "pop", "empty"]
[[], [1], [2], [], [], []]
Output
[null, null, null, 1, 1, false]

Explanation
MyQueue myQueue = new MyQueue();
myQueue.push(1); // queue is: [1]

myQueue.push(2); // queue is: [1, 2] (leftmost is front of the queue)
myQueue.peek(); // return 1
myQueue.pop(); // return 1, queue is [2]
myQueue.empty(); // return false

Constraints:

1 <= x <= 9
At most 100 calls will be made to push, pop, peek, and empty.
All the calls to pop and peek are valid.

Follow-up: Can you implement the queue such that each operation is amortized O(1) time complexity? In other word s, performing n operations will take overall O(n) time even if one of those operations may take longer.

---

233. Number of Digit One

Given an integer n, count the total number of digit 1 appearing in all non-negative integers less than or equal to n.

Example 1:

Input: n = 13
Output: 6

Example 2:

Input: n = 0
Output: 0

Constraints:

0 <= n <= 109

---

234. Palindrome Linked List

Given the head of a singly linked list, return true if it is a palindrome.

Example 1:

Input: head = [1,2,2,1]
Output: true

Example 2:

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

Constraints:

The number of nodes in the list is in the range [1, 105].
0 <= Node.val <= 9

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

---

235. Lowest Common Ancestor of a Binary Search Tree

Given a binary search tree (BST), find the lowest common ancestor (LCA) of two given nodes in the BST.
According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes p an d 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 = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 8
Output: 6
Explanation: The LCA of nodes 2 and 8 is 6.

Example 2:

Input: root = [6,2,8,0,4,7,9,null,null,3,5], p = 2, q = 4
Output: 2
Explanation: The LCA of nodes 2 and 4 is 2, since a node can be a descendant of itself according to the LCA definiti on.

Example 3:

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

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 BST.

---

236. Lowest Common Ancestor of a Binary Tree

Given a binary tree, find the lowest common ancestor (LCA) of two given nodes in the tree.
According to the definition of LCA on Wikipedia: “The lowest common ancestor is defined between two nodes p an d 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 definiti on.

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.

---

237. Delete Node in a Linked List

Write a function to delete a node in a singly-linked list. You will not be given access to the head of the list, instead y ou will be given access to the node to be deleted directly.
It is guaranteed that the node to be deleted is not a tail node in the list.

Example 1:

Input: head = [4,5,1,9], node = 5
Output: [4,1,9]
Explanation: You are given the second node with value 5, the linked list should become 4 -> 1 -> 9 after calling your function.

Example 2:

Input: head = [4,5,1,9], node = 1
Output: [4,5,9]
Explanation: You are given the third node with value 1, the linked list should become 4 -> 5 -> 9 after calling your f unction.

Example 3:

Input: head = [1,2,3,4], node = 3
Output: [1,2,4]

Example 4:

Input: head = [0,1], node = 0
Output: [1]

Example 5:

Input: head = [-3,5,-99], node = -3
Output: [5,-99]

Constraints:

The number of the nodes in the given list is in the range [2, 1000].
-1000 <= Node.val <= 1000
The value of each node in the list is unique.
The node to be deleted is in the list and is not a tail node

---

238. Product of Array Except Self

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 spa ce for space complexity analysis.)

---

239. Sliding Window Maximum

You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of th e array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.
Return the max sliding window.

Example 1:

Input: nums = [1,3,-1,-3,5,3,6,7], k = 3
Output: [3,3,5,5,6,7]
Explanation:

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

Example 2:

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

Example 3:

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

Example 4:

Input: nums = [9,11], k = 2
Output: [11]

Example 5:

Input: nums = [4,-2], k = 2
Output: [4]

Constraints:

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

---

240. Search a 2D Matrix II

Write an efficient algorithm that searches for a target value in an m x n integer matrix. The matrix has the following properties:

Integers in each row are sorted in ascending from left to right.
Integers in each column are sorted in ascending from top to bottom.

Example 1:

Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 5
Output: true

Example 2:

Input: matrix = [[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]], target = 20
Output: false

Constraints:

m == matrix.length
n == matrix[i].length
1 <= n, m <= 300
-109 <= matrix[i][j] <= 109
All the integers in each row are sorted in ascending order.
All the integers in each column are sorted in ascending order.
-109 <= target <= 109

---

241. Different Ways to Add Parentheses

Given a string expression of numbers and operators, return all possible results from computing all the different possi ble ways to group numbers and operators. You may return the answer in any order.

Example 1:

Input: expression = "2-1-1"
Output: [0,2]
Explanation:
((2-1)-1) = 0
(2-(1-1)) = 2

Example 2:

Input: expression = "2*3-4*5"
Output: [-34,-14,-10,-10,10]
Explanation:
(2*(3-(4*5))) = -34
((2*3)-(4*5)) = -14
((2*(3-4))*5) = -10
(2*((3-4)*5)) = -10
(((2*3)-4)*5) = 10

Constraints:

1 <= expression.length <= 20
expression consists of digits and the operator '+', '-', and '*'.
All the integer values in the input expression are in the range [0, 99].

---

242. Valid Anagram

Given two strings s and t, return true if t is an anagram of s, and false otherwise.

Example 1:
Input: s = "anagram", t = "nagaram"
Output: true
Example 2:
Input: s = "rat", t = "car"
Output: false

Constraints:

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

Follow up: What if the inputs contain Unicode characters? How would you adapt your solution to such a case?

---

257. Binary Tree Paths

Given the root of a binary tree, return all root-to-leaf paths in any order.
A leaf is a node with no children.

Example 1:

Input: root = [1,2,3,null,5]
Output: ["1->2->5","1->3"]

Example 2:

Input: root = [1]
Output: ["1"]

Constraints:

The number of nodes in the tree is in the range [1, 100].
-100 <= Node.val <= 100

---

258. Add Digits

Given an integer num, repeatedly add all its digits until the result has only one digit, and return it.

Example 1:

Input: num = 38
Output: 2
Explanation: The process is
38 --> 3 + 8 --> 11
11 --> 1 + 1 --> 2
Since 2 has only one digit, return it.

Example 2:

Input: num = 0
Output: 0

Constraints:

0 <= num <= 231 - 1

Follow up: Could you do it without any loop/recursion in O(1) runtime?

---

260. Single Number III

Given an integer array nums, in which exactly two elements appear only once and all the other elements appear exac tly twice. Find the two elements that appear only once. You can return the answer in any order.
You must write an algorithm that runs in linear runtime complexity and uses only constant extra space.

Example 1:

Input: nums = [1,2,1,3,2,5]
Output: [3,5]
Explanation: [5, 3] is also a valid answer.

Example 2:

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

Example 3:

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

Constraints:

2 <= nums.length <= 3 * 104
-231 <= nums[i] <= 231 - 1
Each integer in nums will appear twice, only two integers will appear once.

---

262. Trips and Users

Table: Trips

+-------------+----------+
| Column Name | Type |
+-------------+----------+
id	int
client_id	int
driver_id	int
city_id	int
status	enum
request_at	date
+-------------+----------+
id is the primary key for this table.
The table holds all taxi trips. Each trip has a unique id, while client_id and driver_id are foreign keys to the users_id at the Users table.
Status is an ENUM type of ('completed', 'cancelled_by_driver', 'cancelled_by_client').

Table: Users

+-------------+----------+
| Column Name | Type |
+-------------+----------+
users_id	int
banned	enum
role	enum
+-------------+----------+
users_id is the primary key for this table.
The table holds all users. Each user has a unique users_id, and role is an ENUM type of ('client', 'driver', 'partner').
banned is an ENUM type of ('Yes', 'No').

The cancellation rate is computed by dividing the number of canceled (by client or driver) requests with unbanned us ers by the total number of requests with unbanned users on that day.
Write a SQL query to find the cancellation rate of requests with unbanned users (both client and driver must not be b anned) each day between "2013-10-01" and "2013-10-03". Round Cancellation Rate to two decimal points.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Trips table:

id	client_id	driver_id	city_id	status	request_at
1	1	10	1	completed	2013-10-01
2	2	11	1	cancelled_by_driver	2013-10-01
3	3	12	6	completed	2013-10-01
4	4	13	6	cancelled_by_client	2013-10-01
5	1	10	1	completed	2013-10-02
6	2	11	6	completed	2013-10-02
7	3	12	6	completed	2013-10-02
8	2	12	12	completed	2013-10-03
9	3	10	12	completed	2013-10-03
10	4	13	12	cancelled_by_driver	2013-10-03

Users table:

users_id	banned	role
1	No	client
2	Yes	client
3	No	client
4	No	client
10	No	driver
11	No	driver
12	No	driver
13	No	driver

Output:

Day	Cancellation Rate
2013-10-01	0.33
2013-10-02	0.00
2013-10-03	0.50

Explanation:
On 2013-10-01:
- There were 4 requests in total, 2 of which were canceled.
- However, the request with Id=2 was made by a banned client (User_Id=2), so it is ignored in the calculation.
- Hence there are 3 unbanned requests in total, 1 of which was canceled.
- The Cancellation Rate is (1 / 3) = 0.33

On 2013-10-02:
- There were 3 requests in total, 0 of which were canceled.
- The request with Id=6 was made by a banned client, so it is ignored.

- Hence there are 2 unbanned requests in total, 0 of which were canceled.
- The Cancellation Rate is (0 / 2) = 0.00

On 2013-10-03:
- There were 3 requests in total, 1 of which was canceled.
- The request with Id=8 was made by a banned client, so it is ignored.
- Hence there are 2 unbanned request in total, 1 of which were canceled.
- The Cancellation Rate is (1 / 2) = 0.50

---

263. Ugly Number

An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5.
Given an integer n, return true if n is an ugly number.

Example 1:

Input: n = 6
Output: true
Explanation: 6 = 2 × 3
Example 2:

Input: n = 8
Output: true
Explanation: 8 = 2 × 2 × 2

Example 3:

Input: n = 14
Output: false
Explanation: 14 is not ugly since it includes the prime factor 7.

Example 4:

Input: n = 1
Output: true
Explanation: 1 has no prime factors, therefore all of its prime factors are limited to 2, 3, and 5.

Constraints:

-231 <= n <= 231 - 1

---

264. Ugly Number II

An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5.
Given an integer n, return the nth ugly number.

Example 1:

Input: n = 10
Output: 12
Explanation: [1, 2, 3, 4, 5, 6, 8, 9, 10, 12] is the sequence of the first 10 ugly numbers.

Example 2:

Input: n = 1
Output: 1
Explanation: 1 has no prime factors, therefore all of its prime factors are limited to 2, 3, and 5.

Constraints:

1 <= n <= 1690

---

268. Missing Number

Given an array nums containing n distinct numbers in the range [0, n], return the only number in the range that is mi ssing from the array.

Example 1:

Input: nums = [3,0,1]
Output: 2
Explanation: n = 3 since there are 3 numbers, so all numbers are in the range [0,3]. 2 is the missing number in the ra nge since it does not appear in nums.

Example 2:

Input: nums = [0,1]
Output: 2
Explanation: n = 2 since there are 2 numbers, so all numbers are in the range [0,2]. 2 is the missing number in the ra nge since it does not appear in nums.

Example 3:

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

Explanation: n = 9 since there are 9 numbers, so all numbers are in the range [0,9]. 8 is the missing number in the ra nge since it does not appear in nums.

Example 4:

Input: nums = [0]
Output: 1
Explanation: n = 1 since there is 1 number, so all numbers are in the range [0,1]. 1 is the missing number in the rang e since it does not appear in nums.

Constraints:

n == nums.length
1 <= n <= 104
0 <= nums[i] <= n
All the numbers of nums are unique.

Follow up: Could you implement a solution using only O(1) extra space complexity and O(n) runtime complexity?

---

273. Integer to English Words

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"
Example 4:
Input: num = 1234567891
Output: "One Billion Two Hundred Thirty Four Million Five Hundred Sixty Seven Thousand Eight Hundred Ninety One"

Constraints:

0 <= num <= 231 - 1

---

274. H-Index

Given an array of integers citations where citations[i] is the number of citations a researcher received for their ith pa per, return compute the researcher's h-index.
According to the definition of h-index on Wikipedia: A scientist has an index h if h of their n papers have at least h c itations each, and the other n − h papers have no more than h citations each.
If there are several possible values for h, the maximum one is taken as the h-index.

Example 1:

Input: citations = [3,0,6,1,5]
Output: 3
Explanation: [3,0,6,1,5] means the researcher has 5 papers in total and each of them had received 3, 0, 6, 1, 5 citation s respectively.
Since the researcher has 3 papers with at least 3 citations each and the remaining two with no more than 3 citations e ach, their h-index is 3.

Example 2:

Input: citations = [1,3,1]
Output: 1

Constraints:

n == citations.length
1 <= n <= 5000
0 <= citations[i] <= 1000

---

275. H-Index II

Given an array of integers citations where citations[i] is the number of citations a researcher received for their ith pa per and citations is sorted in an ascending order, return compute the researcher's h-index.
According to the definition of h-index on Wikipedia: A scientist has an index h if h of their n papers have at least h c itations each, and the other n − h papers have no more than h citations each.
If there are several possible values for h, the maximum one is taken as the h-index.
You must write an algorithm that runs in logarithmic time.

Example 1:

Input: citations = [0,1,3,5,6]
Output: 3
Explanation: [0,1,3,5,6] means the researcher has 5 papers in total and each of them had received 0, 1, 3, 5, 6 citation

s respectively.
Since the researcher has 3 papers with at least 3 citations each and the remaining two with no more than 3 citations e ach, their h-index is 3.

Example 2:

Input: citations = [1,2,100]
Output: 2

Constraints:

n == citations.length
1 <= n <= 105
0 <= citations[i] <= 1000
citations is sorted in ascending order.

---

278. First Bad Version

You are a product manager and currently leading a team to develop a new product. Unfortunately, the latest version of your product fails the quality check. Since each version is developed based on the previous version, all the version s after a bad version are also bad.
Suppose you have n versions [1, 2, ..., n] and you want to find out the first bad one, which causes all the following o nes to be bad.
You are given an API bool isBadVersion(version) which returns whether version is bad. Implement a function to fin d the first bad version. You should minimize the number of calls to the API.

Example 1:

Input: n = 5, bad = 4
Output: 4
Explanation:
call isBadVersion(3) -> false
call isBadVersion(5) -> true
call isBadVersion(4) -> true
Then 4 is the first bad version.

Example 2:

Input: n = 1, bad = 1
Output: 1

Constraints:

1 <= bad <= n <= 231 - 1

---

279. Perfect Squares

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 its elf. 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

---

282. Expression Add Operators

Given a string num that contains only digits and an integer target, return all possibilities to insert the binary operators '+', '-', and/or '*' between the digits of num so that the resultant expression evaluates to the target value.
Note that operands in the returned expressions should not contain leading zeros.

Example 1:

Input: num = "123", target = 6
Output: ["1*2*3","1+2+3"]
Explanation: Both "1*2*3" and "1+2+3" evaluate to 6.

Example 2:

Input: num = "232", target = 8
Output: ["2*3+2","2+3*2"]

Explanation: Both "2*3+2" and "2+3*2" evaluate to 8.

Example 3:

Input: num = "105", target = 5
Output: ["1*0+5","10-5"]
Explanation: Both "1*0+5" and "10-5" evaluate to 5.
Note that "1-05" is not a valid expression because the 5 has a leading zero.

Example 4:

Input: num = "00", target = 0
Output: ["0*0","0+0","0-0"]
Explanation: "0*0", "0+0", and "0-0" all evaluate to 0.
Note that "00" is not a valid expression because the 0 has a leading zero.

Example 5:

Input: num = "3456237490", target = 9191
Output: []
Explanation: There are no expressions that can be created from "3456237490" to evaluate to 9191.

Constraints:

1 <= num.length <= 10
num consists of only digits.
-231 <= target <= 231 - 1

---

283. Move Zeroes

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

---

284. Peeking Iterator

Design an iterator that supports the peek operation on an existing iterator in addition to the hasNext and the next operations.
Implement the PeekingIterator class:

PeekingIterator(Iterator<int> nums) Initializes the object with the given integer iterator iterator.
int next() Returns the next element in the array and moves the pointer to the next element.
boolean hasNext() Returns true if there are still elements in the array.
int peek() Returns the next element in the array without moving the pointer.

Note: Each language may have a different implementation of the constructor and Iterator, but they all support the int next() and boolean hasNext() functions.

Example 1:

Input
["PeekingIterator", "next", "peek", "next", "next", "hasNext"]
[[[1, 2, 3]], [], [], [], [], []]
Output
[null, 1, 2, 2, 3, false]

Explanation
PeekingIterator peekingIterator = new PeekingIterator([1, 2, 3]); // [1,2,3]
peekingIterator.next();   // return 1, the pointer moves to the next element [1,2,3].
peekingIterator.peek();   // return 2, the pointer does not move [1,2,3].
peekingIterator.next();   // return 2, the pointer moves to the next element [1,2,3]
peekingIterator.next();   // return 3, the pointer moves to the next element [1,2,3]
peekingIterator.hasNext(); // return False

Constraints:

1 <= nums.length <= 1000
1 <= nums[i] <= 1000
All the calls to next and peek are valid.
At most 1000 calls will be made to next, hasNext, and peek.

Follow up: How would you extend your design to be generic and work with all types, not just integer?

---

287. Find the Duplicate Number

Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n] inclusive.
There is only one repeated number in nums, return this repeated number.
You must solve the problem without modifying the array nums and uses only constant extra space.

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

Constraints:

1 <= n <= 105
nums.length == n + 1
1 <= nums[i] <= n
All the integers in nums appear only once except for precisely one integer which appears two or more times.

Follow up:

How can we prove that at least one duplicate number must exist in nums?
Can you solve the problem in linear runtime complexity?

---

289. Game of Life

According to Wikipedia's article: "The Game of Life, also known simply as Life, is a cellular automaton devised by t he British mathematician John Horton Conway in 1970."
The board is made up of an m x n grid of cells, where each cell has an initial state: live (represented by a 1) or dead (represented by a 0). Each cell interacts with its eight neighbors (horizontal, vertical, diagonal) using the following fo ur rules (taken from the above Wikipedia article):

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

The next state is created by applying the above rules simultaneously to every cell in the current state, where births an d deaths occur simultaneously. Given the current state of the m x n grid board, return the next state.

Example 1:

Input: board = [[0,1,0],[0,0,1],[1,1,1],[0,0,0]]
Output: [[0,0,0],[1,0,1],[0,1,1],[0,1,0]]

Example 2:

Input: board = [[1,1],[1,0]]
Output: [[1,1],[1,1]]

Constraints:

m == board.length
n == board[i].length
1 <= m, n <= 25
board[i][j] is 0 or 1.

Follow up:

Could you solve it in-place? Remember that the board needs to be updated simultaneously: You cannot update some cells first and then use their updated values to update other cells.
In this question, we represent the board using a 2D array. In principle, the board is infinite, which would cause probl ems when the active area encroaches upon the border of the array (i.e., live cells reach the border). How would you a ddress these problems?

---

290. Word Pattern

Given a pattern and a string s, find if s follows the same pattern.
Here follow means a full match, such that there is a bijection between a letter in pattern and a non-empty word in s.

Example 1:

Input: pattern = "abba", s = "dog cat cat dog"
Output: true

Example 2:

Input: pattern = "abba", s = "dog cat cat fish"
Output: false

Example 3:

Input: pattern = "aaaa", s = "dog cat cat dog"
Output: false

Example 4:

Input: pattern = "abba", s = "dog dog dog dog"
Output: false

Constraints:

1 <= pattern.length <= 300
pattern contains only lower-case English letters.
1 <= s.length <= 3000
s contains only lower-case English letters and spaces ' '.
s does not contain any leading or trailing spaces.
All the words in s are separated by a single space.

---

292. Nim Game

You are playing the following Nim Game with your friend:

Initially, there is a heap of stones on the table.
You and your friend will alternate taking turns, and you go first.
On each turn, the person whose turn it is will remove 1 to 3 stones from the heap.
The one who removes the last stone is the winner.

Given n, the number of stones in the heap, return true if you can win the game assuming both you and your friend pl
ay optimally, otherwise return false.

Example 1:

Input: n = 4
Output: false
Explanation: These are the possible outcomes:
1. You remove 1 stone. Your friend removes 3 stones, including the last stone. Your friend wins.
2. You remove 2 stones. Your friend removes 2 stones, including the last stone. Your friend wins.
3. You remove 3 stones. Your friend removes the last stone. Your friend wins.
In all outcomes, your friend wins.

Example 2:

Input: n = 1
Output: true

Example 3:

Input: n = 2
Output: true

Constraints:

1 <= n <= 231 - 1

---

295. Find Median from Data Stream

The median is the middle value in an ordered integer list. If the size of the list is even, there is no middle value and t he median is the mean of the two middle values.

For example, for arr = [2,3,4], the median is 3.
For example, for arr = [2,3], the median is (2 + 3) / 2 = 2.5.

Implement the MedianFinder class:

MedianFinder() initializes the MedianFinder object.
void addNum(int num) adds the integer num from the data stream to the data structure.
double findMedian() returns the median of all elements so far. Answers within 10-5 of the actual answer will be acce pted.

Example 1:

Input
["MedianFinder", "addNum", "addNum", "findMedian", "addNum", "findMedian"]
[[], [1], [2], [], [3], []]
Output
[null, null, null, 1.5, null, 2.0]

Explanation
MedianFinder medianFinder = new MedianFinder();
medianFinder.addNum(1);   // arr = [1]
medianFinder.addNum(2);   // arr = [1, 2]
medianFinder.findMedian(); // return 1.5 (i.e., (1 + 2) / 2)
medianFinder.addNum(3);   // arr[1, 2, 3]
medianFinder.findMedian(); // return 2.0

Constraints:

-105 <= num <= 105

There will be at least one element in the data structure before calling findMedian.
At most 5 * 104 calls will be made to addNum and findMedian.

Follow up:

If all integer numbers from the stream are in the range [0, 100], how would you optimize your solution?
If 99% of all integer numbers from the stream are in the range [0, 100], how would you optimize your solution?

---

297. Serialize and Deserialize Binary Tree

Serialization is the process of converting a data structure or object into a sequence of bits so that it can be stored in a file or memory buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer environment.
Design an algorithm to serialize and deserialize a binary tree. There is no restriction on how your serialization/deserialization algorithm should work. You just need to ensure that a binary tree can be serialized to a string and this string can be deserialized to the original tree structure.
Clarification: The input/output format is the same as how LeetCode serializes a binary tree. You do not necessarily need to follow this format, so please be creative and come up with different approaches yourself.

Example 1:

Input: root = [1,2,3,null,null,4,5]
Output: [1,2,3,null,null,4,5]

Example 2:

Input: root = []
Output: []

Example 3:

Input: root = [1]
Output: [1]

Example 4:

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

Constraints:

The number of nodes in the tree is in the range [0, 104].
-1000 <= Node.val <= 1000

---

299. Bulls and Cows

You are playing the Bulls and Cows game with your friend.
You write down a secret number and ask your friend to guess what the number is. When your friend makes a guess, you provide a hint with the following info:

The number of "bulls", which are digits in the guess that are in the correct position.
The number of "cows", which are digits in the guess that are in your secret number but are located in the wrong posit ion. Specifically, the non-bull digits in the guess that could be rearranged such that they become bulls.

Given the secret number secret and your friend's guess guess, return the hint for your friend's guess.
The hint should be formatted as "xAyB", where x is the number of bulls and y is the number of cows. Note that both secret and guess may contain duplicate digits.

Example 1:

Input: secret = "1807", guess = "7810"
Output: "1A3B"
Explanation: Bulls are connected with a '|' and cows are underlined:
"1807"
  |
"7810"
Example 2:

Input: secret = "1123", guess = "0111"
Output: "1A1B"
Explanation: Bulls are connected with a '|' and cows are underlined:
"1123"   "1123"
  | or |
"0111"   "0111"
Note that only one of the two unmatched 1s is counted as a cow since the non-bull digits can only be rearranged to al low one 1 to be a bull.

Example 3:

Input: secret = "1", guess = "0"
Output: "0A0B"

Example 4:

Input: secret = "1", guess = "1"
Output: "1A0B"

Constraints:

1 <= secret.length, guess.length <= 1000

secret.length == guess.length
secret and guess consist of digits only.

---

300. Longest Increasing Subsequence

Given an integer array nums, return the length of the longest strictly increasing subsequence.
A subsequence is a sequence that can be derived from an array by deleting some or no elements without changing th e order of the remaining elements. For example, [3,6,2,7] is a subsequence of the array [0,3,1,6,2,2,7].

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?

---

301. Remove Invalid Parentheses

Given a string s that contains parentheses and letters, remove the minimum number of invalid parentheses to make th e input string valid.
Return all the possible results. You may return the answer in any order.

Example 1:

Input: s = "()())()"
Output: ["(())()","()()()"]

Example 2:

Input: s = "(a)())()"
Output: ["(a())()","(a)()()"]

Example 3:

Input: s = ")("
Output: [""]

Constraints:

1 <= s.length <= 25
s consists of lowercase English letters and parentheses '(' and ')'.
There will be at most 20 parentheses in s.

---

303. Range Sum Query - Immutable

Given an integer array nums, handle multiple queries of the following type:

Calculate the sum of the elements of nums between indices left and right inclusive where left <= right.

Implement the NumArray class:

NumArray(int[] nums) Initializes the object with the integer array nums.
int sumRange(int left, int right) Returns the sum of the elements of nums between indices left and right inclusive (i.e. nums[left] + nums[left + 1] + ... + nums[right]).

Example 1:

Input
["NumArray", "sumRange", "sumRange", "sumRange"]
[[[-2, 0, 3, -5, 2, -1]], [0, 2], [2, 5], [0, 5]]
Output
[null, 1, -1, -3]

Explanation
NumArray numArray = new NumArray([-2, 0, 3, -5, 2, -1]);
numArray.sumRange(0, 2); // return (-2) + 0 + 3 = 1
numArray.sumRange(2, 5); // return 3 + (-5) + 2 + (-1) = -1

numArray.sumRange(0, 5); // return (-2) + 0 + 3 + (-5) + 2 + (-1) = -3

Constraints:

1 <= nums.length <= 104
-105 <= nums[i] <= 105
0 <= left <= right < nums.length
At most 104 calls will be made to sumRange.

---

304. Range Sum Query 2D - Immutable

Given a 2D matrix matrix, handle multiple queries of the following type:

Calculate the sum of the elements of matrix inside the rectangle defined by its upper left corner (row1, col1) and low er right corner (row2, col2).

Implement the NumMatrix class:

NumMatrix(int[][] matrix) Initializes the object with the integer matrix matrix.
int sumRegion(int row1, int col1, int row2, int col2) Returns the sum of the elements of matrix inside the rectangle d efined by its upper left corner (row1, col1) and lower right corner (row2, col2).

Example 1:

Input
["NumMatrix", "sumRegion", "sumRegion", "sumRegion"]
[[[[3, 0, 1, 4, 2], [5, 6, 3, 2, 1], [1, 2, 0, 1, 5], [4, 1, 0, 1, 7], [1, 0, 3, 0, 5]]], [2, 1, 4, 3], [1, 1, 2, 2], [1, 2, 2, 4]]
Output
[null, 8, 11, 12]

Explanation
NumMatrix numMatrix = new NumMatrix([[3, 0, 1, 4, 2], [5, 6, 3, 2, 1], [1, 2, 0, 1, 5], [4, 1, 0, 1, 7], [1, 0, 3, 0, 5]]);
numMatrix.sumRegion(2, 1, 4, 3); // return 8 (i.e sum of the red rectangle)
numMatrix.sumRegion(1, 1, 2, 2); // return 11 (i.e sum of the green rectangle)
numMatrix.sumRegion(1, 2, 2, 4); // return 12 (i.e sum of the blue rectangle)

Constraints:

m == matrix.length
n == matrix[i].length
1 <= m, n <= 200
-105 <= matrix[i][j] <= 105
0 <= row1 <= row2 < m

0 <= col1 <= col2 < n
At most 104 calls will be made to sumRegion.

---

306. Additive Number

Additive number is a string whose digits can form additive sequence.
A valid additive sequence should contain at least three numbers. Except for the first two numbers, each subsequent n umber in the sequence must be the sum of the preceding two.
Given a string containing only digits '0'-'9', write a function to determine if it's an additive number.
Note: Numbers in the additive sequence cannot have leading zeros, so sequence 1, 2, 03 or 1, 02, 3 is invalid.

Example 1:

Input: "112358"
Output: true
Explanation: The digits can form an additive sequence: 1, 1, 2, 3, 5, 8.
1 + 1 = 2, 1 + 2 = 3, 2 + 3 = 5, 3 + 5 = 8

Example 2:

Input: "199100199"
Output: true
Explanation: The additive sequence is: 1, 99, 100, 199.
1 + 99 = 100, 99 + 100 = 199

Constraints:

num consists only of digits '0'-'9'.
1 <= num.length <= 35

Follow up:
How would you handle overflow for very large input integers?

---

307. Range Sum Query - Mutable

Given an integer array nums, handle multiple queries of the following types:

Update the value of an element in nums.
Calculate the sum of the elements of nums between indices left and right inclusive where left <= right.

Implement the NumArray class:

NumArray(int[] nums) Initializes the object with the integer array nums.
void update(int index, int val) Updates the value of nums[index] to be val.
int sumRange(int left, int right) Returns the sum of the elements of nums between indices left and right inclusive (i.e. nums[left] + nums[left + 1] + ... + nums[right]).

Example 1:

Input
["NumArray", "sumRange", "update", "sumRange"]
[[[1, 3, 5]], [0, 2], [1, 2], [0, 2]]
Output
[null, 9, null, 8]

Explanation
NumArray numArray = new NumArray([1, 3, 5]);
numArray.sumRange(0, 2); // return 1 + 3 + 5 = 9
numArray.update(1, 2);   // nums = [1, 2, 5]
numArray.sumRange(0, 2); // return 1 + 2 + 5 = 8

Constraints:

1 <= nums.length <= 3 * 104
-100 <= nums[i] <= 100
0 <= index < nums.length
-100 <= val <= 100
0 <= left <= right < nums.length
At most 3 * 104 calls will be made to update and sumRange.

---

309. Best Time to Buy and Sell Stock with Cooldown

You are given an array prices where prices[i] is the price of a given stock on the ith day.
Find the maximum profit you can achieve. You may complete as many transactions as you like (i.e., buy one and sel l one share of the stock multiple times) with the following restrictions:

After you sell your stock, you cannot buy stock on the next day (i.e., cooldown one day).

Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy agai n).

Example 1:

Input: prices = [1,2,3,0,2]

Output: 3
Explanation: transactions = [buy, sell, cooldown, buy, sell]

Example 2:

Input: prices = [1]
Output: 0

Constraints:

1 <= prices.length <= 5000
0 <= prices[i] <= 1000

---

310. Minimum Height Trees

A tree is an undirected graph in which any two vertices are connected by exactly one path. In other words, any conne cted graph without simple cycles is a tree.
Given a tree of n nodes labelled from 0 to n - 1, and an array of n - 1 edges where edges[i] = [ai, bi] indicates that the re is an undirected edge between the two nodes ai and bi in the tree, you can choose any node of the tree as the root. When you select a node x as the root, the result tree has height h. Among all possible rooted trees, those with minim um height (i.e. min(h)) are called minimum height trees (MHTs).
Return a list of all MHTs' root labels. You can return the answer in any order.
The height of a rooted tree is the number of edges on the longest downward path between the root and a leaf.

Example 1:

Input: n = 4, edges = [[1,0],[1,2],[1,3]]
Output: [1]
Explanation: As shown, the height of the tree is 1 when the root is the node with label 1 which is the only MHT.

Example 2:

Input: n = 6, edges = [[3,0],[3,1],[3,2],[3,4],[5,4]]
Output: [3,4]

Example 3:

Input: n = 1, edges = []
Output: [0]

Example 4:

Input: n = 2, edges = [[0,1]]
Output: [0,1]

Constraints:

1 <= n <= 2 * 104
edges.length == n - 1
0 <= ai, bi < n
ai != bi
All the pairs (ai, bi) are distinct.
The given input is guaranteed to be a tree and there will be no repeated edges.

---

312. Burst Balloons

You are given n balloons, indexed from 0 to n - 1. Each balloon is painted with a number on it represented by an arra y nums. You are asked to burst all the balloons.
If you burst the ith balloon, you will get nums[i - 1] * nums[i] * nums[i + 1] coins. If i - 1 or i + 1 goes out of bounds of the array, then treat it as if there is a balloon with a 1 painted on it.
Return the maximum coins you can collect by bursting the balloons wisely.

Example 1:

Input: nums = [3,1,5,8]
Output: 167
Explanation:
nums = [3,1,5,8] --> [3,5,8] --> [3,8] --> [8] --> []
coins = 3*1*5 + 3*5*8 + 1*3*8 + 1*8*1 = 167
Example 2:

Input: nums = [1,5]
Output: 10

Constraints:

n == nums.length
1 <= n <= 500
0 <= nums[i] <= 100

---

313. Super Ugly Number

A super ugly number is a positive integer whose prime factors are in the array primes.
Given an integer n and an array of integers primes, return the nth super ugly number.
The nth super ugly number is guaranteed to fit in a 32-bit signed integer.

Example 1:

Input: n = 12, primes = [2,7,13,19]
Output: 32
Explanation: [1,2,4,7,8,13,14,16,19,26,28,32] is the sequence of the first 12 super ugly numbers given primes = [2,7,13,19].

Example 2:

Input: n = 1, primes = [2,3,5]
Output: 1
Explanation: 1 has no prime factors, therefore all of its prime factors are in the array primes = [2,3,5].

Constraints:

1 <= n <= 106
1 <= primes.length <= 100
2 <= primes[i] <= 1000
primes[i] is guaranteed to be a prime number.
All the values of primes are unique and sorted in ascending order.

---

315. Count of Smaller Numbers After Self

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

Example 1:

Input: nums = [5,2,6,1]
Output: [2,1,1,0]
Explanation:
To the right of 5 there are 2 smaller elements (2 and 1).
To the right of 2 there is only 1 smaller element (1).
To the right of 6 there is 1 smaller element (1).
To the right of 1 there is 0 smaller element.

Example 2:

Input: nums = [-1]
Output: [0]

Example 3:

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

Constraints:

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

---

316. Remove Duplicate Letters

Given a string s, remove duplicate letters so that every letter appears once and only once. You must make sure your r esult is the smallest in lexicographical order among all possible results.

Example 1:

Input: s = "bcabc"
Output: "abc"

Example 2:

Input: s = "cbacdcbc"
Output: "acdb"

Constraints:

1 <= s.length <= 104
s consists of lowercase English letters.

Note: This question is the same as 1081: https://leetcode.com/problems/smallest-subsequence-of-distinct-characters/

---

318. Maximum Product of Word Lengths

Given a string array words, return the maximum value of length(word[i]) * length(word[j]) where the two words do not share common letters. If no such two words exist, return 0.

Example 1:

Input: words = ["abcw","baz","foo","bar","xtfn","abcdef"]
Output: 16
Explanation: The two words can be "abcw", "xtfn".

Example 2:

Input: words = ["a","ab","abc","d","cd","bcd","abcd"]
Output: 4
Explanation: The two words can be "ab", "cd".

Example 3:

Input: words = ["a","aa","aaa","aaaa"]
Output: 0
Explanation: No such pair of words.

Constraints:

2 <= words.length <= 1000
1 <= words[i].length <= 1000
words[i] consists only of lowercase English letters.

---

319. Bulb Switcher

There are n bulbs that are initially off. You first turn on all the bulbs, then you turn off every second bulb.
On the third round, you toggle every third bulb (turning on if it's off or turning off if it's on). For the ith round, you t oggle every i bulb. For the nth round, you only toggle the last bulb.
Return the number of bulbs that are on after n rounds.

Example 1:

Input: n = 3
Output: 1
Explanation: At first, the three bulbs are [off, off, off].
After the first round, the three bulbs are [on, on, on].
After the second round, the three bulbs are [on, off, on].
After the third round, the three bulbs are [on, off, off].
So you should return 1 because there is only one bulb is on.
Example 2:

Input: n = 0
Output: 0

Example 3:

Input: n = 1
Output: 1

Constraints:

0 <= n <= 109

---

321. Create Maximum Number

You are given two integer arrays nums1 and nums2 of lengths m and n respectively. nums1 and nums2 represent the digits of two numbers. You are also given an integer k.
Create the maximum number of length k <= m + n from digits of the two numbers. The relative order of the digits fr om the same array must be preserved.
Return an array of the k digits representing the answer.

Example 1:

Input: nums1 = [3,4,6,5], nums2 = [9,1,2,5,8,3], k = 5
Output: [9,8,6,5,3]

Example 2:

Input: nums1 = [6,7], nums2 = [6,0,4], k = 5
Output: [6,7,6,0,4]

Example 3:

Input: nums1 = [3,9], nums2 = [8,9], k = 3
Output: [9,8,9]

Constraints:

m == nums1.length
n == nums2.length
1 <= m, n <= 500
0 <= nums1[i], nums2[i] <= 9
1 <= k <= m + n

---

322. Coin Change

You are given an integer array coins representing coins of different denominations and an integer amount representin g a total amount of money.
Return the fewest number of coins that you need to make up that amount. If that amount of money cannot be made u p by any combination of the coins, return -1.
You may assume that you have an infinite number of each kind of coin.

Example 1:

Input: coins = [1,2,5], amount = 11
Output: 3
Explanation: 11 = 5 + 5 + 1

Example 2:

Input: coins = [2], amount = 3
Output: -1

Example 3:

Input: coins = [1], amount = 0
Output: 0

Example 4:

Input: coins = [1], amount = 1
Output: 1

Example 5:

Input: coins = [1], amount = 2
Output: 2

Constraints:

1 <= coins.length <= 12
1 <= coins[i] <= 231 - 1
0 <= amount <= 104

---

324. Wiggle Sort II

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

You may assume the input array always has a valid answer.

Example 1:

Input: nums = [1,5,1,1,6,4]
Output: [1,6,1,5,1,4]
Explanation: [1,4,1,5,1,6] is also accepted.

Example 2:

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

Constraints:

1 <= nums.length <= 5 * 104
0 <= nums[i] <= 5000
It is guaranteed that there will be an answer for the given input nums.

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

---

326. Power of Three

Given an integer n, return true if it is a power of three. Otherwise, return false.
An integer n is a power of three, if there exists an integer x such that n == 3x.

Example 1:
Input: n = 27
Output: true
Example 2:
Input: n = 0
Output: false
Example 3:
Input: n = 9
Output: true
Example 4:
Input: n = 45
Output: false

Constraints:

-231 <= n <= 231 - 1

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

---

327. Count of Range Sum

Given an integer array nums and two integers lower and upper, return the number of range sums that lie in [lower, u
pper] inclusive.
Range sum S(i, j) is defined as the sum of the elements in nums between indices i and j inclusive, where i <= j.

Example 1:

Input: nums = [-2,5,-1], lower = -2, upper = 2
Output: 3
Explanation: The three ranges are: [0,0], [2,2], and [0,2] and their respective sums are: -2, -1, 2.

Example 2:

Input: nums = [0], lower = 0, upper = 0
Output: 1

Constraints:

1 <= nums.length <= 105
-231 <= nums[i] <= 231 - 1
-105 <= lower <= upper <= 105
The answer is guaranteed to fit in a 32-bit integer.

---

328. Odd Even Linked List

Given the head of a singly linked list, group all the nodes with odd indices together followed by the nodes with even indices, and return the reordered list.
The first node is considered odd, and the second node is even, and so on.
Note that the relative order inside both the even and odd groups should remain as it was in the input.
You must solve the problem in O(1) extra space complexity and O(n) time complexity.

Example 1:

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

Example 2:

Input: head = [2,1,3,5,6,4,7]
Output: [2,3,6,7,1,5,4]

Constraints:

n == number of nodes in the linked list
0 <= n <= 104
-106 <= Node.val <= 106

---

329. Longest Increasing Path in a Matrix

Given an m x n integers matrix, return the length of the longest increasing path in matrix.
From each cell, you can either move in four directions: left, right, up, or down. You may not move diagonally or mo ve outside the boundary (i.e., wrap-around is not allowed).

Example 1:

Input: matrix = [[9,9,4],[6,6,8],[2,1,1]]
Output: 4
Explanation: The longest increasing path is [1, 2, 6, 9].

Example 2:

Input: matrix = [[3,4,5],[3,2,6],[2,2,1]]
Output: 4
Explanation: The longest increasing path is [3, 4, 5, 6]. Moving diagonally is not allowed.

Example 3:

Input: matrix = [[1]]
Output: 1

Constraints:

m == matrix.length
n == matrix[i].length
1 <= m, n <= 200
0 <= matrix[i][j] <= 231 - 1

---

330. Patching Array

Given a sorted integer array nums and an integer n, add/patch elements to the array such that any number in the rang e [1, n] inclusive can be formed by the sum of some elements in the array.
Return the minimum number of patches required.

Example 1:

Input: nums = [1,3], n = 6
Output: 1
Explanation:
Combinations of nums are [1], [3], [1,3], which form possible sums of: 1, 3, 4.
Now if we add/patch 2 to nums, the combinations are: [1], [2], [3], [1,3], [2,3], [1,2,3].
Possible sums are 1, 2, 3, 4, 5, 6, which now covers the range [1, 6].
So we only need 1 patch.

Example 2:

Input: nums = [1,5,10], n = 20
Output: 2
Explanation: The two patches can be [2, 4].

Example 3:

Input: nums = [1,2,2], n = 5
Output: 0

Constraints:

1 <= nums.length <= 1000
1 <= nums[i] <= 104
nums is sorted in ascending order.
1 <= n <= 231 - 1

---

331. Verify Preorder Serialization of a Binary Tree

One way to serialize a binary tree is to use preorder traversal. When we encounter a non-null node, we record the no de's value. If it is a null node, we record using a sentinel value such as '#'.

For example, the above binary tree can be serialized to the string "9,3,4,#,#,1,#,#,2,#,6,#,#", where '#' represents a nu ll node.

Given a string of comma-separated values preorder, return true if it is a correct preorder traversal serialization of a bi
nary tree.
It is guaranteed that each comma-separated value in the string must be either an integer or a character '#' representing
null pointer.
You may assume that the input format is always valid.

For example, it could never contain two consecutive commas, such as "1,,3".

Note: You are not allowed to reconstruct the tree.

Example 1:
Input: preorder = "9,3,4,#,#,1,#,#,2,#,6,#,#"
Output: true
Example 2:
Input: preorder = "1,#"
Output: false
Example 3:
Input: preorder = "9,#,#,1"
Output: false

Constraints:

1 <= preorder.length <= 104
preorder consist of integers in the range [0, 100] and '#' separated by commas ','.

---

332. Reconstruct Itinerary

You are given a list of airline tickets where tickets[i] = [fromi, toi] represent the departure and the arrival airports of
one flight. Reconstruct the itinerary in order and return it.
All of the tickets belong to a man who departs from "JFK", thus, the itinerary must begin with "JFK". If there are mu
ltiple valid itineraries, you should return the itinerary that has the smallest lexical order when read as a single string.

For example, the itinerary ["JFK", "LGA"] has a smaller lexical order than ["JFK", "LGB"].

You may assume all tickets form at least one valid itinerary. You must use all the tickets once and only once.

Example 1:

Input: tickets = [["MUC","LHR"],["JFK","MUC"],["SFO","SJC"],["LHR","SFO"]]
Output: ["JFK","MUC","LHR","SFO","SJC"]

Example 2:

Input: tickets = [["JFK","SFO"],["JFK","ATL"],["SFO","ATL"],["ATL","JFK"],["ATL","SFO"]]

Output: ["JFK","ATL","JFK","SFO","ATL","SFO"]
Explanation: Another possible reconstruction is ["JFK","SFO","ATL","JFK","ATL","SFO"] but it is larger in lexical order.

Constraints:

1 <= tickets.length <= 300
tickets[i].length == 2
fromi.length == 3
toi.length == 3
fromi and toi consist of uppercase English letters.
fromi != toi

---

334. Increasing Triplet Subsequence

Given an integer array nums, return true if there exists a triple of indices (i, j, k) such that i < j < k and nums[i] < nums[j] < nums[k]. If no such indices exists, return false.

Example 1:

Input: nums = [1,2,3,4,5]
Output: true
Explanation: Any triplet where i < j < k is valid.

Example 2:

Input: nums = [5,4,3,2,1]
Output: false
Explanation: No triplet exists.

Example 3:

Input: nums = [2,1,5,0,4,6]
Output: true
Explanation: The triplet (3, 4, 5) is valid because nums[3] == 0 < nums[4] == 4 < nums[5] == 6.

Constraints:

1 <= nums.length <= 5 * 105
-231 <= nums[i] <= 231 - 1

Follow up: Could you implement a solution that runs in O(n) time complexity and O(1) space complexity?

---

335. Self Crossing

You are given an array of integers distance.
You start at point (0,0) on an X-Y plane and you move distance[0] meters to the north, then distance[1] meters to the west, distance[2] meters to the south, distance[3] meters to the east, and so on. In other words, after each move, your direction changes counter-clockwise.
Return true if your path crosses itself, and false if it does not.

Example 1:

Input: distance = [2,1,1,2]
Output: true

Example 2:

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

Example 3:

Input: distance = [1,1,1,1]
Output: true

Constraints:

1 <= distance.length <= 105
1 <= distance[i] <= 105

---

336. Palindrome Pairs

Given a list of unique words, return all the pairs of the distinct indices (i, j) in the given list, so that the concatenation of the two words words[i] + words[j] is a palindrome.

Example 1:

Input: words = ["abcd","dcba","lls","s","sssll"]
Output: [[0,1],[1,0],[3,2],[2,4]]
Explanation: The palindromes are ["dcbaabcd","abcddcba","slls","llssssll"]

Example 2:

Input: words = ["bat","tab","cat"]
Output: [[0,1],[1,0]]
Explanation: The palindromes are ["battab","tabbat"]

Example 3:

Input: words = ["a",""]
Output: [[0,1],[1,0]]

Constraints:

1 <= words.length <= 5000
0 <= words[i].length <= 300
words[i] consists of lower-case English letters.

---

337. House Robber III

The thief has found himself a new place for his thievery again. There is only one entrance to this area, called root. Besides the root, each house has one and only one parent house. After a tour, the smart thief realized that all houses i n this place form a binary tree. It will automatically contact the police if two directly-linked houses were broken into on the same night.
Given the root of the binary tree, return the maximum amount of money the thief can rob without alerting the police.

Example 1:

Input: root = [3,2,3,null,3,null,1]
Output: 7
Explanation: Maximum amount of money the thief can rob = 3 + 3 + 1 = 7.

Example 2:

Input: root = [3,4,5,1,3,null,1]
Output: 9
Explanation: Maximum amount of money the thief can rob = 4 + 5 = 9.

Constraints:

The number of nodes in the tree is in the range [1, 104].
0 <= Node.val <= 104

---

338. Counting Bits

Given an integer n, return an array ans of length n + 1 such that for each i (0 <= i <= n), ans[i] is the number of 1's in the binary representation of i.

Example 1:

Input: n = 2
Output: [0,1,1]
Explanation:
0 --> 0
1 --> 1
2 --> 10

Example 2:

Input: n = 5
Output: [0,1,1,2,1,2]
Explanation:
0 --> 0
1 --> 1
2 --> 10
3 --> 11
4 --> 100
5 --> 101

Constraints:

0 <= n <= 105

Follow up:

It is very easy to come up with a solution with a runtime of O(n log n). Can you do it in linear time O(n) and possibl y in a single pass?
Can you do it without using any built-in function (i.e., like __builtin_popcount in C++)?

---

341. Flatten Nested List Iterator

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. Implement an iterator to flatten it.
Implement the NestedIterator class:

NestedIterator(List<NestedInteger> nestedList) Initializes the iterator with the nested list nestedList.
int next() Returns the next integer in the nested list.
boolean hasNext() Returns true if there are still some integers in the nested list and false otherwise.

Your code will be tested with the following pseudocode:

```
initialize iterator with nestedList
res = []
while iterator.hasNext()
    append iterator.next() to the end of res
return res
```

If res matches the expected flattened list, then your code will be judged as correct.

Example 1:

Input: nestedList = [[1,1],2,[1,1]]
Output: [1,1,2,1,1]
Explanation: By calling next repeatedly until hasNext returns false, the order of elements returned by next should be: [1,1,2,1,1].

Example 2:

Input: nestedList = [1,[4,[6]]]
Output: [1,4,6]
Explanation: By calling next repeatedly until hasNext returns false, the order of elements returned by next should be: [1,4,6].

Constraints:

1 <= nestedList.length <= 500
The values of the integers in the nested list is in the range [-106, 106].

---

342. Power of Four

Given an integer n, return true if it is a power of four. Otherwise, return false.
An integer n is a power of four, if there exists an integer x such that n == 4x.

Example 1:
Input: n = 16
Output: true

Example 2:
Input: n = 5
Output: false
Example 3:
Input: n = 1
Output: true

Constraints:

-231 <= n <= 231 - 1

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

---

343. Integer Break

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

---

344. Reverse String

Write a function that reverses a string. The input string is given as an array of characters s.

You must do this by modifying the input array in-place with O(1) extra memory.

Example 1:
Input: s = ["h","e","l","l","o"]
Output: ["o","l","l","e","h"]
Example 2:
Input: s = ["H","a","n","n","a","h"]
Output: ["h","a","n","n","a","H"]

Constraints:

1 <= s.length <= 105
s[i] is a printable ascii character.

---

345. Reverse Vowels of a String

Given a string s, reverse only all the vowels in the string and return it.
The vowels are 'a', 'e', 'i', 'o', and 'u', and they can appear in both cases.

Example 1:
Input: s = "hello"
Output: "holle"
Example 2:
Input: s = "leetcode"
Output: "leotcede"

Constraints:

1 <= s.length <= 3 * 105
s consist of printable ASCII characters.

---

347. Top K Frequent Elements

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
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.

---

349. Intersection of Two Arrays

Given two integer arrays nums1 and nums2, return an array of their intersection. Each element in the result must be unique and you may return the result in any order.

Example 1:

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

Example 2:

Input: nums1 = [4,9,5], nums2 = [9,4,9,8,4]
Output: [9,4]
Explanation: [4,9] is also accepted.

Constraints:

1 <= nums1.length, nums2.length <= 1000
0 <= nums1[i], nums2[i] <= 1000

---

350. Intersection of Two Arrays II

Given two integer arrays nums1 and nums2, return an array of their intersection. Each element in the result must app ear as many times as it shows in both arrays and you may return the result in any order.

Example 1:

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

Example 2:

Input: nums1 = [4,9,5], nums2 = [9,4,9,8,4]
Output: [4,9]
Explanation: [9,4] is also accepted.

Constraints:

1 <= nums1.length, nums2.length <= 1000
0 <= nums1[i], nums2[i] <= 1000

Follow up:

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

---

352. Data Stream as Disjoint Intervals

Given a data stream input of non-negative integers a1, a2, ..., an, summarize the numbers seen so far as a list of disjo int intervals.
Implement the SummaryRanges class:

SummaryRanges() Initializes the object with an empty stream.
void addNum(int val) Adds the integer val to the stream.
int[][] getIntervals() Returns a summary of the integers in the stream currently as a list of disjoint intervals [starti, en di].

Example 1:

Input
["SummaryRanges", "addNum", "getIntervals", "addNum", "getIntervals", "addNum", "getIntervals", "addNum", "g etIntervals", "addNum", "getIntervals"]
[[], [1], [], [3], [], [7], [], [2], [], [6], []]
Output

[null, null, [[1, 1]], null, [[1, 1], [3, 3]], null, [[1, 1], [3, 3], [7, 7]], null, [[1, 3], [7, 7]], null, [[1, 3], [6, 7]]]

Explanation
SummaryRanges summaryRanges = new SummaryRanges();
summaryRanges.addNum(1);   // arr = [1]
summaryRanges.getIntervals(); // return [[1, 1]]
summaryRanges.addNum(3);   // arr = [1, 3]
summaryRanges.getIntervals(); // return [[1, 1], [3, 3]]
summaryRanges.addNum(7);   // arr = [1, 3, 7]
summaryRanges.getIntervals(); // return [[1, 1], [3, 3], [7, 7]]
summaryRanges.addNum(2);   // arr = [1, 2, 3, 7]
summaryRanges.getIntervals(); // return [[1, 3], [7, 7]]
summaryRanges.addNum(6);   // arr = [1, 2, 3, 6, 7]
summaryRanges.getIntervals(); // return [[1, 3], [6, 7]]

Constraints:

0 <= val <= 104
At most 3 * 104 calls will be made to addNum and getIntervals.

Follow up: What if there are lots of merges and the number of disjoint intervals is small compared to the size of the data stream?

---

354. Russian Doll Envelopes

You are given a 2D array of integers envelopes where envelopes[i] = [wi, hi] represents the width and the height of a n envelope.
One envelope can fit into another if and only if both the width and height of one envelope are greater than the other e nvelope's width and height.
Return the maximum number of envelopes you can Russian doll (i.e., put one inside the other).
Note: You cannot rotate an envelope.

Example 1:

Input: envelopes = [[5,4],[6,4],[6,7],[2,3]]
Output: 3
Explanation: The maximum number of envelopes you can Russian doll is 3 ([2,3] => [5,4] => [6,7]).

Example 2:

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

Constraints:

1 <= envelopes.length <= 5000
envelopes[i].length == 2
1 <= wi, hi <= 104

---

355. Design Twitter

Design a simplified version of Twitter where users can post tweets, follow/unfollow another user, and is able to see t he 10 most recent tweets in the user's news feed.
Implement the Twitter class:

Twitter() Initializes your twitter object.
void postTweet(int userId, int tweetId) Composes a new tweet with ID tweetId by the user userId. Each call to this f unction will be made with a unique tweetId.
List<Integer> getNewsFeed(int userId) Retrieves the 10 most recent tweet IDs in the user's news feed. Each item in t he news feed must be posted by users who the user followed or by the user themself. Tweets must be ordered from most recent to least recent.
void follow(int followerId, int followeeId) The user with ID followerId started following the user with ID followeeId .
void unfollow(int followerId, int followeeId) The user with ID followerId started unfollowing the user with ID follo weeId.

Example 1:

Input
["Twitter", "postTweet", "getNewsFeed", "follow", "postTweet", "getNewsFeed", "unfollow", "getNewsFeed"]
[[], [1, 5], [1], [1, 2], [2, 6], [1], [1, 2], [1]]
Output
[null, null, [5], null, null, [6, 5], null, [5]]

Explanation
Twitter twitter = new Twitter();
twitter.postTweet(1, 5); // User 1 posts a new tweet (id = 5).
twitter.getNewsFeed(1); // User 1's news feed should return a list with 1 tweet id -> [5]. return [5]
twitter.follow(1, 2);   // User 1 follows user 2.
twitter.postTweet(2, 6); // User 2 posts a new tweet (id = 6).
twitter.getNewsFeed(1); // User 1's news feed should return a list with 2 tweet ids -> [6, 5]. Tweet id 6 should prece de tweet id 5 because it is posted after tweet id 5.
twitter.unfollow(1, 2); // User 1 unfollows user 2.
twitter.getNewsFeed(1); // User 1's news feed should return a list with 1 tweet id -> [5], since user 1 is no longer foll owing user 2.

Constraints:

1 <= userId, followerId, followeeId <= 500
0 <= tweetId <= 104

All the tweets have unique IDs.
At most 3 * 104 calls will be made to postTweet, getNewsFeed, follow, and unfollow.

---

357. Count Numbers with Unique Digits

Given an integer n, return the count of all numbers with unique digits, x, where 0 <= x < 10n.

Example 1:

Input: n = 2
Output: 91
Explanation: The answer should be the total numbers in the range of 0 ≤ x < 100, excluding 11,22,33,44,55,66,77,88
,99

Example 2:

Input: n = 0
Output: 1

Constraints:

0 <= n <= 8

---

363. Max Sum of Rectangle No Larger Than K

Given an m x n matrix matrix and an integer k, return the max sum of a rectangle in the matrix such that its sum is n
o larger than k.
It is guaranteed that there will be a rectangle with a sum no larger than k.

Example 1:

Input: matrix = [[1,0,1],[0,-2,3]], k = 2
Output: 2
Explanation: Because the sum of the blue rectangle [[0, 1], [-2, 3]] is 2, and 2 is the max number no larger than k (k
= 2).

Example 2:

Input: matrix = [[2,2,-1]], k = 3
Output: 3

Constraints:

m == matrix.length
n == matrix[i].length
1 <= m, n <= 100
-100 <= matrix[i][j] <= 100
-105 <= k <= 105

Follow up: What if the number of rows is much larger than the number of columns?

---

365. Water and Jug Problem

You are given two jugs with capacities jug1Capacity and jug2Capacity liters. There is an infinite amount of water su pply available. Determine whether it is possible to measure exactly targetCapacity liters using these two jugs.
If targetCapacity liters of water are measurable, you must have targetCapacity liters of water contained within one or both buckets by the end.
Operations allowed:

Fill any of the jugs with water.
Empty any of the jugs.
Pour water from one jug into another till the other jug is completely full, or the first jug itself is empty.

Example 1:

Input: jug1Capacity = 3, jug2Capacity = 5, targetCapacity = 4
Output: true
Explanation: The famous Die Hard example

Example 2:

Input: jug1Capacity = 2, jug2Capacity = 6, targetCapacity = 5
Output: false

Example 3:

Input: jug1Capacity = 1, jug2Capacity = 2, targetCapacity = 3
Output: true

Constraints:

1 <= jug1Capacity, jug2Capacity, targetCapacity <= 106

---

367. Valid Perfect Square

Given a positive integer num, write a function which returns True if num is a perfect square else False.
Follow up: Do not use any built-in library function such as sqrt.

Example 1:
Input: num = 16
Output: true
Example 2:
Input: num = 14
Output: false

Constraints:

1 <= num <= 2^31 - 1

---

368. Largest Divisible Subset

Given a set of distinct positive integers nums, return the largest subset answer such that every pair (answer[i], answe r[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.

---

371. Sum of Two Integers

Given two integers a and b, return the sum of the two integers without using the operators + and -.

Example 1:
Input: a = 1, b = 2
Output: 3
Example 2:
Input: a = 2, b = 3
Output: 5

Constraints:

-1000 <= a, b <= 1000

---

372. Super Pow

Your task is to calculate ab mod 1337 where a is a positive integer and b is an extremely large positive integer given in the form of an array.

Example 1:
Input: a = 2, b = [3]
Output: 8
Example 2:
Input: a = 2, b = [1,0]
Output: 1024
Example 3:
Input: a = 1, b = [4,3,3,8,5,2]
Output: 1
Example 4:

Input: a = 2147483647, b = [2,0,0]
Output: 1198

Constraints:

1 <= a <= 231 - 1
1 <= b.length <= 2000
0 <= b[i] <= 9
b doesn't contain leading zeros.

---

373. Find K Pairs with Smallest Sums

You are given two integer arrays nums1 and nums2 sorted in ascending order and an integer k.
Define a pair (u, v) which consists of one element from the first array and one element from the second array.
Return the k pairs (u1, v1), (u2, v2), ..., (uk, vk) with the smallest sums.

Example 1:

Input: nums1 = [1,7,11], nums2 = [2,4,6], k = 3
Output: [[1,2],[1,4],[1,6]]
Explanation: The first 3 pairs are returned from the sequence: [1,2],[1,4],[1,6],[7,2],[7,4],[11,2],[7,6],[11,4],[11,6]

Example 2:

Input: nums1 = [1,1,2], nums2 = [1,2,3], k = 2
Output: [[1,1],[1,1]]
Explanation: The first 2 pairs are returned from the sequence: [1,1],[1,1],[1,2],[2,1],[1,2],[2,2],[1,3],[1,3],[2,3]

Example 3:

Input: nums1 = [1,2], nums2 = [3], k = 3
Output: [[1,3],[2,3]]
Explanation: All possible pairs are returned from the sequence: [1,3],[2,3]

Constraints:

1 <= nums1.length, nums2.length <= 105
-109 <= nums1[i], nums2[i] <= 109
nums1 and nums2 both are sorted in ascending order.
1 <= k <= 1000

---

374. Guess Number Higher or Lower

We are playing the Guess Game. The game is as follows:
I pick a number from 1 to n. You have to guess which number I picked.
Every time you guess wrong, I will tell you whether the number I picked is higher or lower than your guess.
You call a pre-defined API int guess(int num), which returns 3 possible results:

-1: The number I picked is lower than your guess (i.e. pick < num).
1: The number I picked is higher than your guess (i.e. pick > num).
0: The number I picked is equal to your guess (i.e. pick == num).

Return the number that I picked.

Example 1:
Input: n = 10, pick = 6
Output: 6
Example 2:
Input: n = 1, pick = 1
Output: 1
Example 3:
Input: n = 2, pick = 1
Output: 1
Example 4:
Input: n = 2, pick = 2
Output: 2

Constraints:

1 <= n <= 231 - 1
1 <= pick <= n

---

375. Guess Number Higher or Lower II

We are playing the Guessing Game. The game will work as follows:

I pick a number between 1 and n.
You guess a number.
If you guess the right number, you win the game.
If you guess the wrong number, then I will tell you whether the number I picked is higher or lower, and you will con tinue guessing.
Every time you guess a wrong number x, you will pay x dollars. If you run out of money, you lose the game.

Given a particular n, return the minimum amount of money you need to guarantee a win regardless of what number I pick.

Example 1:

Input: n = 10
Output: 16
Explanation: The winning strategy is as follows:
- The range is [1,10]. Guess 7.
  - If this is my number, your total is $0. Otherwise, you pay $7.
  - If my number is higher, the range is [8,10]. Guess 9.
    - If this is my number, your total is $7. Otherwise, you pay $9.
    - If my number is higher, it must be 10. Guess 10. Your total is $7 + $9 = $16.
    - If my number is lower, it must be 8. Guess 8. Your total is $7 + $9 = $16.
  - If my number is lower, the range is [1,6]. Guess 3.
    - If this is my number, your total is $7. Otherwise, you pay $3.
    - If my number is higher, the range is [4,6]. Guess 5.
      - If this is my number, your total is $7 + $3 = $10. Otherwise, you pay $5.
      - If my number is higher, it must be 6. Guess 6. Your total is $7 + $3 + $5 = $15.
      - If my number is lower, it must be 4. Guess 4. Your total is $7 + $3 + $5 = $15.
    - If my number is lower, the range is [1,2]. Guess 1.
      - If this is my number, your total is $7 + $3 = $10. Otherwise, you pay $1.
      - If my number is higher, it must be 2. Guess 2. Your total is $7 + $3 + $1 = $11.
The worst case in all these scenarios is that you pay $16. Hence, you only need $16 to guarantee a win.

Example 2:

Input: n = 1
Output: 0
Explanation: There is only one possible number, so you can guess 1 and not have to pay anything.

Example 3:

Input: n = 2
Output: 1
Explanation: There are two possible numbers, 1 and 2.
- Guess 1.
  - If this is my number, your total is $0. Otherwise, you pay $1.
  - If my number is higher, it must be 2. Guess 2. Your total is $1.
The worst case is that you pay $1.

Constraints:

1 <= n <= 200

---

376. Wiggle Subsequence

A wiggle sequence is a sequence where the differences between successive numbers strictly alternate between positi ve and negative. The first difference (if one exists) may be either positive or negative. A sequence with one element and a sequence with two non-equal elements are trivially wiggle sequences.

For example, [1, 7, 4, 9, 2, 5] is a wiggle sequence because the differences (6, -3, 5, -7, 3) alternate between positive and negative.
In contrast, [1, 4, 7, 2, 5] and [1, 7, 4, 5, 5] are not wiggle sequences. The first is not because its first two differences are positive, and the second is not because its last difference is zero.

A subsequence is obtained by deleting some elements (possibly zero) from the original sequence, leaving the remain ing elements in their original order.
Given an integer array nums, return the length of the longest wiggle subsequence of nums.

Example 1:

Input: nums = [1,7,4,9,2,5]
Output: 6
Explanation: The entire sequence is a wiggle sequence with differences (6, -3, 5, -7, 3).

Example 2:

Input: nums = [1,17,5,10,13,15,10,5,16,8]
Output: 7
Explanation: There are several subsequences that achieve this length.
One is [1, 17, 10, 13, 10, 16, 8] with differences (16, -7, 3, -3, 6, -8).

Example 3:

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

Constraints:

1 <= nums.length <= 1000
0 <= nums[i] <= 1000

Follow up: Could you solve this in O(n) time?

---

377. Combination Sum IV

Given an array of distinct integers nums and a target integer target, return the number of possible combinations that a dd up to target.
The answer is guaranteed to fit in a 32-bit integer.

Example 1:

Input: nums = [1,2,3], target = 4
Output: 7
Explanation:
The possible combination ways are:
(1, 1, 1, 1)
(1, 1, 2)
(1, 2, 1)
(1, 3)
(2, 1, 1)
(2, 2)
(3, 1)
Note that different sequences are counted as different combinations.

Example 2:

Input: nums = [9], target = 3
Output: 0

Constraints:

1 <= nums.length <= 200
1 <= nums[i] <= 1000
All the elements of nums are unique.
1 <= target <= 1000

Follow up: What if negative numbers are allowed in the given array? How does it change the problem? What limitati on we need to add to the question to allow negative numbers?

---

378. Kth Smallest Element in a Sorted Matrix

Given an n x n matrix where each of the rows and columns is sorted in ascending order, return the kth smallest elem ent in the matrix.
Note that it is the kth smallest element in the sorted order, not the kth distinct element.
You must find a solution with a memory complexity better than O(n2).

Example 1:

Input: matrix = [[1,5,9],[10,11,13],[12,13,15]], k = 8
Output: 13
Explanation: The elements in the matrix are [1,5,9,10,11,12,13,13,15], and the 8th smallest number is 13

Example 2:

Input: matrix = [[-5]], k = 1

Output: -5

Constraints:

n == matrix.length == matrix[i].length
1 <= n <= 300
-109 <= matrix[i][j] <= 109
All the rows and columns of matrix are guaranteed to be sorted in non-decreasing order.
1 <= k <= n2

Follow up:

Could you solve the problem with a constant memory (i.e., O(1) memory complexity)?
Could you solve the problem in O(n) time complexity? The solution may be too advanced for an interview but you may find reading this paper fun.

---

380. Insert Delete GetRandom O(1)

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 otherw ise.
bool remove(int val) Removes an item val from the set if present. Returns true if the item was present, false otherwis e.
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.

---

381. Insert Delete GetRandom O(1) - Duplicates allowed

RandomizedCollection is a data structure that contains a collection of numbers, possibly duplicates (i.e., a multiset). It should support inserting and removing specific elements and also removing a random element.
Implement the RandomizedCollection class:

RandomizedCollection() Initializes the empty RandomizedCollection object.
bool insert(int val) Inserts an item val into the multiset, even if the item is already present. Returns true if the item is not present, false otherwise.
bool remove(int val) Removes an item val from the multiset if present. Returns true if the item is present, false other wise. Note that if val has multiple occurrences in the multiset, we only remove one of them.
int getRandom() Returns a random element from the current multiset of elements. The probability of each element b eing returned is linearly related to the number of same values the multiset contains.

You must implement the functions of the class such that each function works on average O(1) time complexity.
Note: The test cases are generated such that getRandom will only be called if there is at least one item in the Rando mizedCollection.

Example 1:

Input
["RandomizedCollection", "insert", "insert", "insert", "getRandom", "remove", "getRandom"]
[[], [1], [1], [2], [], [1], []]
Output
[null, true, false, true, 2, true, 1]

Explanation
RandomizedCollection randomizedCollection = new RandomizedCollection();
randomizedCollection.insert(1);   // return true since the collection does not contain 1.
                                  // Inserts 1 into the collection.
randomizedCollection.insert(1);   // return false since the collection contains 1.
                                  // Inserts another 1 into the collection. Collection now contains [1,1].
randomizedCollection.insert(2);   // return true since the collection does not contain 2.
                                  // Inserts 2 into the collection. Collection now contains [1,1,2].
randomizedCollection.getRandom(); // getRandom should:
                                  // - return 1 with probability 2/3, or
                                  // - return 2 with probability 1/3.

randomizedCollection.remove(1);   // return true since the collection contains 1.
                                  // Removes 1 from the collection. Collection now contains [1,2].
randomizedCollection.getRandom(); // getRandom should return 1 or 2, both equally likely.

Constraints:

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

---

382. Linked List Random Node

Given a singly linked list, return a random node's value from the linked list. Each node must have the same probabili ty of being chosen.
Implement the Solution class:

Solution(ListNode head) Initializes the object with the integer array nums.
int getRandom() Chooses a node randomly from the list and returns its value. All the nodes of the list should be equa lly likely to be choosen.

Example 1:

Input
["Solution", "getRandom", "getRandom", "getRandom", "getRandom", "getRandom"]
[[[1, 2, 3]], [], [], [], [], []]
Output
[null, 1, 3, 2, 2, 3]

Explanation
Solution solution = new Solution([1, 2, 3]);
solution.getRandom(); // return 1
solution.getRandom(); // return 3
solution.getRandom(); // return 2
solution.getRandom(); // return 2
solution.getRandom(); // return 3
// getRandom() should return either 1, 2, or 3 randomly. Each element should have equal probability of returning.

Constraints:

The number of nodes in the linked list will be in the range [1, 104].
-104 <= Node.val <= 104
At most 104 calls will be made to getRandom.

Follow up:

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

---

383. Ransom Note

Given two stings ransomNote and magazine, return true if ransomNote can be constructed from magazine and false otherwise.
Each letter in magazine can only be used once in ransomNote.

Example 1:
Input: ransomNote = "a", magazine = "b"
Output: false
Example 2:
Input: ransomNote = "aa", magazine = "ab"
Output: false
Example 3:
Input: ransomNote = "aa", magazine = "aab"
Output: true

Constraints:

1 <= ransomNote.length, magazine.length <= 105
ransomNote and magazine consist of lowercase English letters.

---

384. Shuffle an Array

Given an integer array nums, design an algorithm to randomly shuffle the array. All permutations of the array should be equally likely as a result of the shuffling.
Implement the Solution class:

Solution(int[] nums) Initializes the object with the integer array nums.
int[] reset() Resets the array to its original configuration and returns it.
int[] shuffle() Returns a random shuffling of the array.

Example 1:

Input
["Solution", "shuffle", "reset", "shuffle"]
[[[1, 2, 3]], [], [], []]
Output
[null, [3, 1, 2], [1, 2, 3], [1, 3, 2]]

Explanation
Solution solution = new Solution([1, 2, 3]);
solution.shuffle(); // Shuffle the array [1,2,3] and return its result.
                     // Any permutation of [1,2,3] must be equally likely to be returned.
                     // Example: return [3, 1, 2]
solution.reset();   // Resets the array back to its original configuration [1,2,3]. Return [1, 2, 3]
solution.shuffle(); // Returns the random shuffling of array [1,2,3]. Example: return [1, 3, 2]

Constraints:

1 <= nums.length <= 200
-106 <= nums[i] <= 106
All the elements of nums are unique.
At most 5 * 104 calls in total will be made to reset and shuffle.

---

385. Mini Parser

Given a string s represents the serialization of a nested list, implement a parser to deserialize it and return the deseria lized NestedInteger.
Each element is either an integer or a list whose elements may also be integers or other lists.

Example 1:

Input: s = "324"
Output: 324
Explanation: You should return a NestedInteger object which contains a single integer 324.

Example 2:

Input: s = "[123,[456,[789]]]"
Output: [123,[456,[789]]]
Explanation: Return a NestedInteger object containing a nested list with 2 elements:
1. An integer containing value 123.
2. A nested list containing two elements:
   i. An integer containing value 456.
   ii. A nested list with one element:
      a. An integer containing value 789

Constraints:

1 <= s.length <= 5 * 104
s consists of digits, square brackets "[]", negative sign '-', and commas ','.
s is the serialization of valid NestedInteger.
All the values in the input are in the range [-106, 106].

---

386. Lexicographical Numbers

Given an integer n, return all the numbers in the range [1, n] sorted in lexicographical order.
You must write an algorithm that runs in O(n) time and uses O(1) extra space.

Example 1:
Input: n = 13
Output: [1,10,11,12,13,2,3,4,5,6,7,8,9]
Example 2:
Input: n = 2
Output: [1,2]

Constraints:

1 <= n <= 5 * 104

---

387. First Unique Character in a String

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.

---

388. Longest Absolute File Path

Suppose we have a file system that stores both files and directories. An example of one system is represented in the following picture:

Here, we have dir as the only directory in the root. dir contains two subdirectories, subdir1 and subdir2. subdir1 contains a file file1.ext and subdirectory subsubdir1. subdir2 contains a subdirectory subsubdir2, which contains a file file2.ext.
In text form, it looks like this (with □ representing the tab character):

dir
□ subdir1
□ □ file1.ext
□ □ subsubdir1
□ subdir2
□ □ subsubdir2
□ □ □ file2.ext

If we were to write this representation in code, it will look like this: "dir\n\tsubdir1\n\t\tfile1.ext\n\t\tsubsubdir1\n\tsubdir2\n\t\tsubsubdir2\n\t\t\tfile2.ext". Note that the '\n' and '\t' are the new-line and tab characters.
Every file and directory has a unique absolute path in the file system, which is the order of directories that must be opened to reach the file/directory itself, all concatenated by '/'s. Using the above example, the absolute path to file2.ext is "dir/subdir2/subsubdir2/file2.ext". Each directory name consists of letters, digits, and/or spaces. Each file name is of the form name.extension, where name and extension consist of letters, digits, and/or spaces.
Given a string input representing the file system in the explained format, return the length of the longest absolute path to a file in the abstracted file system. If there is no file in the system, return 0.

Example 1:

Input: input = "dir\n\tsubdir1\n\tsubdir2\n\t\tfile.ext"
Output: 20
Explanation: We have only one file, and the absolute path is "dir/subdir2/file.ext" of length 20.

Example 2:

Input: input = "dir\n\tsubdir1\n\t\tfile1.ext\n\t\tsubsubdir1\n\tsubdir2\n\t\tsubsubdir2\n\t\t\tfile2.ext"
Output: 32
Explanation: We have two files:

"dir/subdir1/file1.ext" of length 21
"dir/subdir2/subsubdir2/file2.ext" of length 32.
We return 32 since it is the longest absolute path to a file.

Example 3:

Input: input = "a"
Output: 0
Explanation: We do not have any files, just a single directory named "a".

Example 4:

Input: input = "file1.txt\nfile2.txt\nlongfile.txt"
Output: 12
Explanation: There are 3 files at the root directory.
Since the absolute path for anything at the root directory is just the name itself, the answer is "longfile.txt" with lengt
h 12.

Constraints:

1 <= input.length <= 104
input may contain lowercase or uppercase English letters, a new line character '\n', a tab character '\t', a dot '.', a spac
e ' ', and digits.

---

389. Find the Difference

You are given two strings s and t.
String t is generated by random shuffling string s and then add one more letter at a random position.
Return the letter that was added to t.

Example 1:

Input: s = "abcd", t = "abcde"
Output: "e"
Explanation: 'e' is the letter that was added.

Example 2:

Input: s = "", t = "y"
Output: "y"

Example 3:

Input: s = "a", t = "aa"
Output: "a"

Example 4:

Input: s = "ae", t = "aea"
Output: "a"

Constraints:

0 <= s.length <= 1000
t.length == s.length + 1
s and t consist of lower-case English letters.

---

390. Elimination Game

You have a list arr of all integers in the range [1, n] sorted in a strictly increasing order. Apply the following algorith m on arr:

Starting from left to right, remove the first number and every other number afterward until you reach the end of the li st.
Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers.
Keep repeating the steps again, alternating left to right and right to left, until a single number remains.

Given the integer n, return the last number that remains in arr.

Example 1:

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

Example 2:

Input: n = 1
Output: 1

Constraints:

1 <= n <= 109

---

391. Perfect Rectangle

Given an array rectangles where rectangles[i] = [xi, yi, ai, bi] represents an axis-aligned rectangle. The bottom-left p
oint of the rectangle is (xi, yi) and the top-right point of it is (ai, bi).
Return true if all the rectangles together form an exact cover of a rectangular region.

Example 1:

Input: rectangles = [[1,1,3,3],[3,1,4,2],[3,2,4,4],[1,3,2,4],[2,3,3,4]]
Output: true
Explanation: All 5 rectangles together form an exact cover of a rectangular region.

Example 2:

Input: rectangles = [[1,1,2,3],[1,3,2,4],[3,1,4,2],[3,2,4,4]]
Output: false
Explanation: Because there is a gap between the two rectangular regions.

Example 3:

Input: rectangles = [[1,1,3,3],[3,1,4,2],[1,3,2,4],[3,2,4,4]]
Output: false
Explanation: Because there is a gap in the top center.

Example 4:

Input: rectangles = [[1,1,3,3],[3,1,4,2],[1,3,2,4],[2,2,4,4]]
Output: false
Explanation: Because two of the rectangles overlap with each other.

Constraints:

1 <= rectangles.length <= 2 * 104
rectangles[i].length == 4
-105 <= xi, yi, ai, bi <= 105

---

392. Is Subsequence

Given two strings s and t, return true if s is a subsequence of t, or false otherwise.
A subsequence of a string is a new string that is formed from the original string by deleting some (can be none) of th e characters without disturbing the relative positions of the remaining characters. (i.e., "ace" is a subsequence of "abc de" while "aec" is not).

Example 1:
Input: s = "abc", t = "ahbgdc"
Output: true
Example 2:
Input: s = "axc", t = "ahbgdc"
Output: false

Constraints:

0 <= s.length <= 100
0 <= t.length <= 104
s and t consist only of lowercase English letters.

Follow up: Suppose there are lots of incoming s, say s1, s2, ..., sk where k >= 109, and you want to check one by on e to see if t has its subsequence. In this scenario, how would you change your code?

---

393. UTF-8 Validation

Given an integer array data representing the data, return whether it is a valid UTF-8 encoding.
A character in UTF8 can be from 1 to 4 bytes long, subjected to the following rules:

For a 1-byte character, the first bit is a 0, followed by its Unicode code.
For an n-bytes character, the first n bits are all one's, the n + 1 bit is 0, followed by n - 1 bytes with the most significa nt 2 bits being 10.

This is how the UTF-8 encoding would work:

```
   Char. number range  |        UTF-8 octet sequence
      (hexadecimal)    |              (binary)
   --------------------+---------------------------------------------
   0000 0000-0000 007F | 0xxxxxxx
   0000 0080-0000 07FF | 110xxxxx 10xxxxxx
   0000 0800-0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx
   0001 0000-0010 FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
```

Note: The input is an array of integers. Only the least significant 8 bits of each integer is used to store the data. This means each integer represents only 1 byte of data.

Example 1:

Input: data = [197,130,1]
Output: true
Explanation: data represents the octet sequence: 11000101 10000010 00000001.
It is a valid utf-8 encoding for a 2-bytes character followed by a 1-byte character.

Example 2:

Input: data = [235,140,4]
Output: false
Explanation: data represented the octet sequence: 11101011 10001100 00000100.
The first 3 bits are all one's and the 4th bit is 0 means it is a 3-bytes character.
The next byte is a continuation byte which starts with 10 and that's correct.
But the second continuation byte does not start with 10, so it is invalid.

Constraints:

1 <= data.length <= 2 * 104
0 <= data[i] <= 255

---

394. Decode String

Given an encoded string, return its decoded string.
The encoding rule is: k[encoded_string], where the encoded_string inside the square brackets is being repeated exact ly k times. Note that k is guaranteed to be a positive integer.
You may assume that the input string is always valid; No extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repe at numbers, k. For example, there won't be input like 3a or 2[4].

Example 1:
Input: s = "3[a]2[bc]"
Output: "aaabcbc"
Example 2:
Input: s = "3[a2[c]]"
Output: "accaccacc"
Example 3:
Input: s = "2[abc]3[cd]ef"
Output: "abcabccdcdcdef"
Example 4:
Input: s = "abc3[cd]xyz"
Output: "abccdcdcdxyz"

Constraints:

1 <= s.length <= 30
s consists of lowercase English letters, digits, and square brackets '[]'.

s is guaranteed to be a valid input.
All the integers in s are in the range [1, 300].

---

395. Longest Substring with At Least K Repeating Characters

Given a string s and an integer k, return the length of the longest substring of s such that the frequency of each chara cter in this substring is greater than or equal to k.

Example 1:

Input: s = "aaabb", k = 3
Output: 3
Explanation: The longest substring is "aaa", as 'a' is repeated 3 times.

Example 2:

Input: s = "ababbc", k = 2
Output: 5
Explanation: The longest substring is "ababb", as 'a' is repeated 2 times and 'b' is repeated 3 times.

Constraints:

1 <= s.length <= 104
s consists of only lowercase English letters.
1 <= k <= 105

---

396. Rotate Function

You are given an integer array nums of length n.
Assume arrk to be an array obtained by rotating nums by k positions clock-wise. We define the rotation function F o n nums as follow:

F(k) = 0 * arrk[0] + 1 * arrk[1] + ... + (n - 1) * arrk[n - 1].

Return the maximum value of F(0), F(1), ..., F(n-1).
The test cases are generated so that the answer fits in a 32-bit integer.

Example 1:

Input: nums = [4,3,2,6]
Output: 26
Explanation:
F(0) = (0 * 4) + (1 * 3) + (2 * 2) + (3 * 6) = 0 + 3 + 4 + 18 = 25
F(1) = (0 * 6) + (1 * 4) + (2 * 3) + (3 * 2) = 0 + 4 + 6 + 6 = 16
F(2) = (0 * 2) + (1 * 6) + (2 * 4) + (3 * 3) = 0 + 6 + 8 + 9 = 23
F(3) = (0 * 3) + (1 * 2) + (2 * 6) + (3 * 4) = 0 + 2 + 12 + 12 = 26
So the maximum value of F(0), F(1), F(2), F(3) is F(3) = 26.

Example 2:

Input: nums = [100]
Output: 0

Constraints:

n == nums.length
1 <= n <= 105
-100 <= nums[i] <= 100

---

397. Integer Replacement

Given a positive integer n, you can apply one of the following operations:

If n is even, replace n with n / 2.
If n is odd, replace n with either n + 1 or n - 1.

Return the minimum number of operations needed for n to become 1.

Example 1:

Input: n = 8
Output: 3
Explanation: 8 -> 4 -> 2 -> 1

Example 2:

Input: n = 7
Output: 4
Explanation: 7 -> 8 -> 4 -> 2 -> 1
or 7 -> 6 -> 3 -> 2 -> 1

Example 3:

Input: n = 4

Output: 2

Constraints:

1 <= n <= 231 - 1

---

398. Random Pick Index

Given an integer array nums with possible duplicates, randomly output the index of a given target number. You can assume that the given target number must exist in the array.
Implement the Solution class:

Solution(int[] nums) Initializes the object with the array nums.
int pick(int target) Picks a random index i from nums where nums[i] == target. If there are multiple valid i's, then eac h index should have an equal probability of returning.

Example 1:

Input
["Solution", "pick", "pick", "pick"]
[[[1, 2, 3, 3, 3]], [3], [1], [3]]
Output
[null, 4, 0, 2]

Explanation
Solution solution = new Solution([1, 2, 3, 3, 3]);
solution.pick(3); // It should return either index 2, 3, or 4 randomly. Each index should have equal probability of retu rning.
solution.pick(1); // It should return 0. Since in the array only nums[0] is equal to 1.
solution.pick(3); // It should return either index 2, 3, or 4 randomly. Each index should have equal probability of retu rning.

Constraints:

1 <= nums.length <= 2 * 104
-231 <= nums[i] <= 231 - 1
target is an integer from nums.
At most 104 calls will be made to pick.

---

399. Evaluate Division

You are given an array of variable pairs equations and an array of real numbers values, where equations[i] = [Ai, Bi] and values[i] represent the equation Ai / Bi = values[i]. Each Ai or Bi is a string that represents a single variable.
You are also given some queries, where queries[j] = [Cj, Dj] represents the jth query where you must find the answer for Cj / Dj = ?.
Return the answers to all queries. If a single answer cannot be determined, return -1.0.
Note: The input is always valid. You may assume that evaluating the queries will not result in division by zero and that there is no contradiction.

Example 1:

Input: equations = [["a","b"],["b","c"]], values = [2.0,3.0], queries = [["a","c"],["b","a"],["a","e"],["a","a"],["x","x"]]
Output: [6.00000,0.50000,-1.00000,1.00000,-1.00000]
Explanation:
Given: a / b = 2.0, b / c = 3.0
queries are: a / c = ?, b / a = ?, a / e = ?, a / a = ?, x / x = ?
return: [6.0, 0.5, -1.0, 1.0, -1.0]

Example 2:

Input: equations = [["a","b"],["b","c"],["bc","cd"]], values = [1.5,2.5,5.0], queries = [["a","c"],["c","b"],["bc","cd"],["cd","bc"]]
Output: [3.75000,0.40000,5.00000,0.20000]

Example 3:

Input: equations = [["a","b"]], values = [0.5], queries = [["a","b"],["b","a"],["a","c"],["x","y"]]
Output: [0.50000,2.00000,-1.00000,-1.00000]

Constraints:

1 <= equations.length <= 20
equations[i].length == 2
1 <= Ai.length, Bi.length <= 5
values.length == equations.length
0.0 < values[i] <= 20.0
1 <= queries.length <= 20
queries[i].length == 2
1 <= Cj.length, Dj.length <= 5
Ai, Bi, Cj, Dj consist of lower case English letters and digits.

---

400. Nth Digit

Given an integer n, return the nth digit of the infinite integer sequence [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ...].

Example 1:

Input: n = 3
Output: 3

Example 2:

Input: n = 11
Output: 0
Explanation: The 11th digit of the sequence 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, ... is a 0, which is part of the number 10.

Constraints:

1 <= n <= 231 - 1

---

401. Binary Watch

A binary watch has 4 LEDs on the top which represent the hours (0-11), and the 6 LEDs on the bottom represent the minutes (0-59). Each LED represents a zero or one, with the least significant bit on the right.

For example, the below binary watch reads "4:51".

Given an integer turnedOn which represents the number of LEDs that are currently on, return all possible times the watch could represent. You may return the answer in any order.
The hour must not contain a leading zero.

For example, "01:00" is not valid. It should be "1:00".

The minute must be consist of two digits and may contain a leading zero.

For example, "10:2" is not valid. It should be "10:02".

Example 1:
Input: turnedOn = 1
Output: ["0:01","0:02","0:04","0:08","0:16","0:32","1:00","2:00","4:00","8:00"]
Example 2:
Input: turnedOn = 9
Output: []

Constraints:

0 <= turnedOn <= 10

---

402. Remove K Digits

Given string num representing a non-negative integer num, and an integer k, return the smallest possible integer after removing k digits from num.

Example 1:

Input: num = "1432219", k = 3
Output: "1219"
Explanation: Remove the three digits 4, 3, and 2 to form the new number 1219 which is the smallest.

Example 2:

Input: num = "10200", k = 1
Output: "200"
Explanation: Remove the leading 1 and the number is 200. Note that the output must not contain leading zeroes.

Example 3:

Input: num = "10", k = 2
Output: "0"
Explanation: Remove all the digits from the number and it is left with nothing which is 0.

Constraints:

1 <= k <= num.length <= 105
num consists of only digits.
num does not have any leading zeros except for the zero itself.

---

403. Frog Jump

A frog is crossing a river. The river is divided into some number of units, and at each unit, there may or may not exis t a stone. The frog can jump on a stone, but it must not jump into the water.
Given a list of stones' positions (in units) in sorted ascending order, determine if the frog can cross the river by landi ng on the last stone. Initially, the frog is on the first stone and assumes the first jump must be 1 unit.

If the frog's last jump was k units, its next jump must be either k - 1, k, or k + 1 units. The frog can only jump in the forward direction.

Example 1:

Input: stones = [0,1,3,5,6,8,12,17]
Output: true
Explanation: The frog can jump to the last stone by jumping 1 unit to the 2nd stone, then 2 units to the 3rd stone, the n 2 units to the 4th stone, then 3 units to the 6th stone, 4 units to the 7th stone, and 5 units to the 8th stone.

Example 2:

Input: stones = [0,1,2,3,4,8,9,11]
Output: false
Explanation: There is no way to jump to the last stone as the gap between the 5th and 6th stone is too large.

Constraints:

2 <= stones.length <= 2000
0 <= stones[i] <= 231 - 1
stones[0] == 0
stones is sorted in a strictly increasing order.

---

404. Sum of Left Leaves

Given the root of a binary tree, return the sum of all left leaves.

Example 1:

Input: root = [3,9,20,null,null,15,7]
Output: 24
Explanation: There are two left leaves in the binary tree, with values 9 and 15 respectively.

Example 2:

Input: root = [1]
Output: 0

Constraints:

The number of nodes in the tree is in the range [1, 1000].
-1000 <= Node.val <= 1000

---

405. Convert a Number to Hexadecimal

Given an integer num, return a string representing its hexadecimal representation. For negative integers, two’s compl ement method is used.
All the letters in the answer string should be lowercase characters, and there should not be any leading zeros in the a nswer except for the zero itself.
Note: You are not allowed to use any built-in library method to directly solve this problem.

Example 1:
Input: num = 26
Output: "1a"
Example 2:
Input: num = -1
Output: "ffffffff"

Constraints:

-231 <= num <= 231 - 1

---

406. Queue Reconstruction by Height

You are given an array of people, people, which are the attributes of some people in a queue (not necessarily in order). Each people[i] = [hi, ki] represents the ith person of height hi with exactly ki other people in front who have a heig ht greater than or equal to hi.
Reconstruct and return the queue that is represented by the input array people. The returned queue should be formatt ed as an array queue, where queue[j] = [hj, kj] is the attributes of the jth person in the queue (queue[0] is the person at the front of the queue).

Example 1:

Input: people = [[7,0],[4,4],[7,1],[5,0],[6,1],[5,2]]
Output: [[5,0],[7,0],[5,2],[6,1],[4,4],[7,1]]
Explanation:
Person 0 has height 5 with no other people taller or the same height in front.
Person 1 has height 7 with no other people taller or the same height in front.
Person 2 has height 5 with two persons taller or the same height in front, which is person 0 and 1.
Person 3 has height 6 with one person taller or the same height in front, which is person 1.
Person 4 has height 4 with four people taller or the same height in front, which are people 0, 1, 2, and 3.
Person 5 has height 7 with one person taller or the same height in front, which is person 1.

Hence [[5,0],[7,0],[5,2],[6,1],[4,4],[7,1]] is the reconstructed queue.

Example 2:

Input: people = [[6,0],[5,0],[4,0],[3,2],[2,2],[1,4]]
Output: [[4,0],[5,0],[2,2],[3,2],[1,4],[6,0]]

Constraints:

1 <= people.length <= 2000
0 <= hi <= 106
0 <= ki < people.length
It is guaranteed that the queue can be reconstructed.

---

407. Trapping Rain Water II

Given an m x n integer matrix heightMap representing the height of each unit cell in a 2D elevation map, return the volume of water it can trap after raining.

Example 1:

Input: heightMap = [[1,4,3,1,3,2],[3,2,1,3,2,4],[2,3,3,2,3,1]]
Output: 4
Explanation: After the rain, water is trapped between the blocks.
We have two small ponds 1 and 3 units trapped.
The total volume of water trapped is 4.

Example 2:

Input: heightMap = [[3,3,3,3,3],[3,2,2,2,3],[3,2,1,2,3],[3,2,2,2,3],[3,3,3,3,3]]
Output: 10

Constraints:

m == heightMap.length
n == heightMap[i].length
1 <= m, n <= 200
0 <= heightMap[i][j] <= 2 * 104

---

409. Longest Palindrome

Given a string s which consists of lowercase or uppercase letters, return the length of the longest palindrome that can be built with those letters.
Letters are case sensitive, for example, "Aa" is not considered a palindrome here.

Example 1:

Input: s = "abccccdd"
Output: 7
Explanation:
One longest palindrome that can be built is "dccaccd", whose length is 7.

Example 2:

Input: s = "a"
Output: 1

Example 3:

Input: s = "bb"
Output: 2

Constraints:

1 <= s.length <= 2000
s consists of lowercase and/or uppercase English letters only.

---

410. Split Array Largest Sum

Given an array nums which consists of non-negative integers and an integer m, you can split the array into m non-empty continuous subarrays.
Write an algorithm to minimize the largest sum among these m subarrays.

Example 1:

Input: nums = [7,2,5,10,8], m = 2
Output: 18
Explanation:
There are four ways to split nums into two subarrays.
The best way is to split it into [7,2,5] and [10,8],
where the largest sum among the two subarrays is only 18.

Example 2:

Input: nums = [1,2,3,4,5], m = 2
Output: 9

Example 3:

Input: nums = [1,4,4], m = 3
Output: 4

Constraints:

1 <= nums.length <= 1000
0 <= nums[i] <= 106
1 <= m <= min(50, nums.length)

---

412. Fizz Buzz

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 if non 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

---

413. Arithmetic Slices

An integer array is called arithmetic if it consists of at least three elements and if the difference between any two con secutive elements is the same.

For example, [1,3,5,7,9], [7,7,7,7], and [3,-1,-5,-9] are arithmetic sequences.

Given an integer array nums, return the number of arithmetic subarrays of nums.
A subarray is a contiguous subsequence of the array.

Example 1:

Input: nums = [1,2,3,4]
Output: 3
Explanation: We have 3 arithmetic slices in nums: [1, 2, 3], [2, 3, 4] and [1,2,3,4] itself.

Example 2:

Input: nums = [1]
Output: 0

Constraints:

1 <= nums.length <= 5000
-1000 <= nums[i] <= 1000

---

414. Third Maximum Number

Given an integer array nums, return the third distinct maximum number in this array. If the third maximum does not exist, return the maximum number.

Example 1:

Input: nums = [3,2,1]
Output: 1
Explanation:
The first distinct maximum is 3.
The second distinct maximum is 2.
The third distinct maximum is 1.

Example 2:

Input: nums = [1,2]
Output: 2
Explanation:
The first distinct maximum is 2.
The second distinct maximum is 1.
The third distinct maximum does not exist, so the maximum (2) is returned instead.

Example 3:

Input: nums = [2,2,3,1]
Output: 1
Explanation:
The first distinct maximum is 3.
The second distinct maximum is 2 (both 2's are counted together since they have the same value).
The third distinct maximum is 1.

Constraints:

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

Follow up: Can you find an O(n) solution?

---

415. Add Strings

Given two non-negative integers, num1 and num2 represented as string, return the sum of num1 and num2 as a strin
g.
You must solve the problem without using any built-in library for handling large integers (such as BigInteger). You must also not convert the inputs to integers directly.

Example 1:

Input: num1 = "11", num2 = "123"
Output: "134"

Example 2:

Input: num1 = "456", num2 = "77"
Output: "533"

Example 3:

Input: num1 = "0", num2 = "0"
Output: "0"

Constraints:

1 <= num1.length, num2.length <= 104
num1 and num2 consist of only digits.
num1 and num2 don't have any leading zeros except for the zero itself.

---

416. Partition Equal Subset Sum

Given a non-empty array nums containing only positive integers, find if the array can be partitioned into two subsets such that the sum of elements in both subsets is equal.

Example 1:

Input: nums = [1,5,11,5]
Output: true
Explanation: The array can be partitioned as [1, 5, 5] and [11].

Example 2:

Input: nums = [1,2,3,5]
Output: false
Explanation: The array cannot be partitioned into equal sum subsets.

Constraints:

1 <= nums.length <= 200
1 <= nums[i] <= 100

---

417. Pacific Atlantic Water Flow

There is an m x n rectangular island that borders both the Pacific Ocean and Atlantic Ocean. The Pacific Ocean touc hes the island's left and top edges, and the Atlantic Ocean touches the island's right and bottom edges.
The island is partitioned into a grid of square cells. You are given an m x n integer matrix heights where heights[r][c] represents the height above sea level of the cell at coordinate (r, c).
The island receives a lot of rain, and the rain water can flow to neighboring cells directly north, south, east, and west if the neighboring cell's height is less than or equal to the current cell's height. Water can flow from any cell adjacent to an ocean into the ocean.
Return a 2D list of grid coordinates result where result[i] = [ri, ci] denotes that rain water can flow from cell (ri, ci) t o both the Pacific and Atlantic oceans.

Example 1:

Input: heights = [[1,2,2,3,5],[3,2,3,4,4],[2,4,5,3,1],[6,7,1,4,5],[5,1,1,2,4]]
Output: [[0,4],[1,3],[1,4],[2,2],[3,0],[3,1],[4,0]]

Example 2:

Input: heights = [[2,1],[1,2]]
Output: [[0,0],[0,1],[1,0],[1,1]]

Constraints:

m == heights.length
n == heights[r].length
1 <= m, n <= 200
0 <= heights[r][c] <= 105

---

419. Battleships in a Board

Given an m x n matrix board where each cell is a battleship 'X' or empty '.', return the number of the battleships on b oard.
Battleships can only be placed horizontally or vertically on board. In other words, they can only be made of the shap e 1 x k (1 row, k columns) or k x 1 (k rows, 1 column), where k can be of any size. At least one horizontal or vertical cell separates between two battleships (i.e., there are no adjacent battleships).

Example 1:

Input: board = [["X",".",".","X"],[".",".",".","X"],[".",".",".","X"]]
Output: 2

Example 2:

Input: board = [["."]]
Output: 0

Constraints:

m == board.length
n == board[i].length
1 <= m, n <= 200
board[i][j] is either '.' or 'X'.

Follow up: Could you do it in one-pass, using only O(1) extra memory and without modifying the values board?

---

420. Strong Password Checker

A password is considered strong if the below conditions are all met:

It has at least 6 characters and at most 20 characters.
It contains at least one lowercase letter, at least one uppercase letter, and at least one digit.
It does not contain three repeating characters in a row (i.e., "...aaa..." is weak, but "...aa...a..." is strong, assuming oth er conditions are met).

Given a string password, return the minimum number of steps required to make password strong. if password is alre ady strong, return 0.
In one step, you can:

Insert one character to password,
Delete one character from password, or
Replace one character of password with another character.

Example 1:
Input: password = "a"
Output: 5
Example 2:
Input: password = "aA1"
Output: 3
Example 3:
Input: password = "1337C0d3"
Output: 0

Constraints:

1 <= password.length <= 50
password consists of letters, digits, dot '.' or exclamation mark '!'.

---

421. Maximum XOR of Two Numbers in an Array

Given an integer array nums, return the maximum result of nums[i] XOR nums[j], where 0 <= i <= j < n.

Example 1:

Input: nums = [3,10,5,25,2,8]
Output: 28
Explanation: The maximum result is 5 XOR 25 = 28.
Example 2:

Input: nums = [0]
Output: 0

Example 3:

Input: nums = [2,4]
Output: 6

Example 4:

Input: nums = [8,10,2]
Output: 10

Example 5:

Input: nums = [14,70,53,83,49,91,36,80,92,51,66,70]
Output: 127

Constraints:

1 <= nums.length <= 2 * 105
0 <= nums[i] <= 231 - 1

---

423. Reconstruct Original Digits from English

Given a string s containing an out-of-order English representation of digits 0-9, return the digits in ascending order.

Example 1:
Input: s = "owoztneoer"
Output: "012"
Example 2:
Input: s = "fviefuro"
Output: "45"

Constraints:

1 <= s.length <= 105

s[i] is one of the characters ["e","g","f","i","h","o","n","s","r","u","t","w","v","x","z"].
s is guaranteed to be valid.

---

424. Longest Repeating Character Replacement

You are given a string s and an integer k. You can choose any character of the string and change it to any other uppe rcase English character. You can perform this operation at most k times.
Return the length of the longest substring containing the same letter you can get after performing the above operatio ns.

Example 1:

Input: s = "ABAB", k = 2
Output: 4
Explanation: Replace the two 'A's with two 'B's or vice versa.

Example 2:

Input: s = "AABABBA", k = 1
Output: 4
Explanation: Replace the one 'A' in the middle with 'B' and form "AABBBBA".
The substring "BBBB" has the longest repeating letters, which is 4.

Constraints:

1 <= s.length <= 105
s consists of only uppercase English letters.
0 <= k <= s.length

---

427. Construct Quad Tree

Given a n * n matrix grid of 0's and 1's only. We want to represent the grid with a Quad-Tree.
Return the root of the Quad-Tree representing the grid.
Notice that you can assign the value of a node to True or False when isLeaf is False, and both are accepted in the ans wer.
A Quad-Tree is a tree data structure in which each internal node has exactly four children. Besides, each node has tw o attributes:

val: True if the node represents a grid of 1's or False if the node represents a grid of 0's.
isLeaf: True if the node is leaf node on the tree or False if the node has the four children.

```
class Node {
    public boolean val;
    public boolean isLeaf;
    public Node topLeft;
    public Node topRight;
    public Node bottomLeft;
    public Node bottomRight;
}
```

We can construct a Quad-Tree from a two-dimensional area using the following steps:

If the current grid has the same value (i.e all 1's or all 0's) set isLeaf True and set val to the value of the grid and set t he four children to Null and stop.
If the current grid has different values, set isLeaf to False and set val to any value and divide the current grid into fou r sub-grids as shown in the photo.
Recurse for each of the children with the proper sub-grid.

If you want to know more about the Quad-Tree, you can refer to the wiki.
Quad-Tree format:
The output represents the serialized format of a Quad-Tree using level order traversal, where null signifies a path ter minator where no node exists below.
It is very similar to the serialization of the binary tree. The only difference is that the node is represented as a list [is Leaf, val].
If the value of isLeaf or val is True we represent it as 1 in the list [isLeaf, val] and if the value of isLeaf or val is Fals e we represent it as 0.

Example 1:

Input: grid = [[0,1],[1,0]]
Output: [[0,1],[1,0],[1,1],[1,1],[1,0]]
Explanation: The explanation of this example is shown below:
Notice that 0 represnts False and 1 represents True in the photo representing the Quad-Tree.

Example 2:

Input: grid = [[1,1,1,1,0,0,0,0],[1,1,1,1,0,0,0,0],[1,1,1,1,1,1,1,1],[1,1,1,1,1,1,1,1],[1,1,1,1,0,0,0,0],[1,1,1,1,0,0,0,0],[1,1,1,1,0,0,0,0],[1,1,1,1,0,0,0,0]]
Output: [[0,1],[1,1],[0,1],[1,1],[1,0],null,null,null,null,[1,0],[1,0],[1,1],[1,1]]
Explanation: All values in the grid are not the same. We divide the grid into four sub-grids.
The topLeft, bottomLeft and bottomRight each has the same value.
The topRight have different values so we divide it into 4 sub-grids where each has the same value.
Explanation is shown in the photo below:

Example 3:

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

Example 4:

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

Example 5:

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

Constraints:

n == grid.length == grid[i].length
n == 2^x where 0 <= x <= 6

---

429. N-ary Tree Level Order Traversal

Given an n-ary tree, return the level order traversal of its nodes' values.
Nary-Tree input serialization is represented in their level order traversal, each group of children is separated by the n ull value (See examples).

Example 1:

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

Example 2:

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

Constraints:

The height of the n-ary tree is less than or equal to 1000
The total number of nodes is between [0, 104]

---

430. Flatten a Multilevel Doubly Linked List

You are given a doubly linked list, which contains nodes that have a next pointer, a previous pointer, and an addition al child pointer. This child pointer may or may not point to a separate doubly linked list, also containing these specia l nodes. These child lists may have one or more children of their own, and so on, to produce a multilevel data structu re as shown in the example below.
Given the head of the first level of the list, flatten the list so that all the nodes appear in a single-level, doubly linked list. Let curr be a node with a child list. The nodes in the child list should appear after curr and before curr.next in th e flattened list.
Return the head of the flattened list. The nodes in the list must have all of their child pointers set to null.

Example 1:

Input: head = [1,2,3,4,5,6,null,null,null,7,8,9,10,null,null,11,12]
Output: [1,2,3,7,8,11,12,9,10,4,5,6]
Explanation: The multilevel linked list in the input is shown.
After flattening the multilevel linked list it becomes:

Example 2:

Input: head = [1,2,null,3]
Output: [1,3,2]
Explanation: The multilevel linked list in the input is shown.
After flattening the multilevel linked list it becomes:

Example 3:

Input: head = []
Output: []
Explanation: There could be empty list in the input.

Constraints:

The number of Nodes will not exceed 1000.
1 <= Node.val <= 105

How the multilevel linked list is represented in test cases:
We use the multilevel linked list from Example 1 above:

```
 1---2---3---4---5---6--NULL
        |
        7---8---9---10--NULL
            |
            11--12--NULL
```

The serialization of each level is as follows:

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

[7,8,9,10,null]
[11,12,null]

To serialize all levels together, we will add nulls in each level to signify no node connects to the upper node of the pr
evious level. The serialization becomes:

```
[1,    2,    3, 4, 5, 6, null]
          |
[null, null, 7,    8, 9, 10, null]
               |
[           null, 11, 12, null]
```

Merging the serialization of each level and removing trailing nulls we obtain:

[1,2,3,4,5,6,null,null,null,7,8,9,10,null,null,11,12]

---

432. All O`one Data Structure

Design a data structure to store the strings' count with the ability to return the strings with minimum and maximum c
ounts.
Implement the AllOne class:

AllOne() Initializes the object of the data structure.
inc(String key) Increments the count of the string key by 1. If key does not exist in the data structure, insert it with c
ount 1.
dec(String key) Decrements the count of the string key by 1. If the count of key is 0 after the decrement, remove it fr
om the data structure. It is guaranteed that key exists in the data structure before the decrement.
getMaxKey() Returns one of the keys with the maximal count. If no element exists, return an empty string "".
getMinKey() Returns one of the keys with the minimum count. If no element exists, return an empty string "".

Example 1:

Input
["AllOne", "inc", "inc", "getMaxKey", "getMinKey", "inc", "getMaxKey", "getMinKey"]
[[], ["hello"], ["hello"], [], [], ["leet"], [], []]
Output
[null, null, null, "hello", "hello", null, "hello", "leet"]

Explanation
AllOne allOne = new AllOne();
allOne.inc("hello");
allOne.inc("hello");
allOne.getMaxKey(); // return "hello"
allOne.getMinKey(); // return "hello"
allOne.inc("leet");
allOne.getMaxKey(); // return "hello"

allOne.getMinKey(); // return "leet"

Constraints:

1 <= key.length <= 10
key consists of lowercase English letters.
It is guaranteed that for each call to dec, key is existing in the data structure.
At most 5 * 104 calls will be made to inc, dec, getMaxKey, and getMinKey.

---

433. Minimum Genetic Mutation

A gene string can be represented by an 8-character long string, with choices from 'A', 'C', 'G', and 'T'.
Suppose we need to investigate a mutation from a gene string start to a gene string end where one mutation is define d as one single character changed in the gene string.

For example, "AACCGGTT" --> "AACCGGTA" is one mutation.

There is also a gene bank bank that records all the valid gene mutations. A gene must be in bank to make it a valid g ene string.
Given the two gene strings start and end and the gene bank bank, return the minimum number of mutations needed t o mutate from start to end. If there is no such a mutation, return -1.
Note that the starting point is assumed to be valid, so it might not be included in the bank.

Example 1:

Input: start = "AACCGGTT", end = "AACCGGTA", bank = ["AACCGGTA"]
Output: 1

Example 2:

Input: start = "AACCGGTT", end = "AAACGGTA", bank = ["AACCGGTA","AACCGCTA","AAACGGTA"]
Output: 2

Example 3:

Input: start = "AAAAACCC", end = "AACCCCCC", bank = ["AAAACCCC","AAACCCCC","AACCCCCC"]
Output: 3

Constraints:

start.length == 8
end.length == 8
0 <= bank.length <= 10
bank[i].length == 8
start, end, and bank[i] consist of only the characters ['A', 'C', 'G', 'T'].

---

434. Number of Segments in a String

You are given a string s, return the number of segments in the string.
A segment is defined to be a contiguous sequence of non-space characters.

Example 1:

Input: s = "Hello, my name is John"
Output: 5
Explanation: The five segments are ["Hello,", "my", "name", "is", "John"]

Example 2:

Input: s = "Hello"
Output: 1

Example 3:

Input: s = "love live! mu'sic forever"
Output: 4

Example 4:

Input: s = ""
Output: 0

Constraints:

0 <= s.length <= 300
s consists of lower-case and upper-case English letters, digits or one of the following characters "!@#$%^&*()_+-=',
.:".
The only space character in s is ' '.

---

435. Non-overlapping Intervals

Given an array of intervals intervals where intervals[i] = [starti, endi], return the minimum number of intervals you n
eed to remove to make the rest of the intervals non-overlapping.

Example 1:

Input: intervals = [[1,2],[2,3],[3,4],[1,3]]
Output: 1
Explanation: [1,3] can be removed and the rest of the intervals are non-overlapping.

Example 2:

Input: intervals = [[1,2],[1,2],[1,2]]
Output: 2
Explanation: You need to remove two [1,2] to make the rest of the intervals non-overlapping.

Example 3:

Input: intervals = [[1,2],[2,3]]
Output: 0
Explanation: You don't need to remove any of the intervals since they're already non-overlapping.

Constraints:

1 <= intervals.length <= 105
intervals[i].length == 2
-5 * 104 <= starti < endi <= 5 * 104

---

436. Find Right Interval

You are given an array of intervals, where intervals[i] = [starti, endi] and each starti is unique.
The right interval for an interval i is an interval j such that startj >= endi and startj is minimized.
Return an array of right interval indices for each interval i. If no right interval exists for interval i, then put -1 at inde
x i.

Example 1:

Input: intervals = [[1,2]]
Output: [-1]
Explanation: There is only one interval in the collection, so it outputs -1.

Example 2:

Input: intervals = [[3,4],[2,3],[1,2]]
Output: [-1,0,1]
Explanation: There is no right interval for [3,4].
The right interval for [2,3] is [3,4] since start0 = 3 is the smallest start that is >= end1 = 3.
The right interval for [1,2] is [2,3] since start1 = 2 is the smallest start that is >= end2 = 2.

Example 3:

Input: intervals = [[1,4],[2,3],[3,4]]
Output: [-1,2,-1]
Explanation: There is no right interval for [1,4] and [3,4].
The right interval for [2,3] is [3,4] since start2 = 3 is the smallest start that is >= end1 = 3.

Constraints:

1 <= intervals.length <= 2 * 104
intervals[i].length == 2
-106 <= starti <= endi <= 106
The start point of each interval is unique.

---

437. Path Sum III

Given the root of a binary tree and an integer targetSum, return the number of paths where the sum of the values alo ng the path equals targetSum.
The path does not need to start or end at the root or a leaf, but it must go downwards (i.e., traveling only from parent nodes to child nodes).

Example 1:

Input: root = [10,5,-3,3,2,null,11,3,-2,null,1], targetSum = 8
Output: 3
Explanation: The paths that sum to 8 are shown.

Example 2:

Input: root = [5,4,8,11,null,13,4,7,2,null,null,5,1], targetSum = 22
Output: 3

Constraints:

The number of nodes in the tree is in the range [0, 1000].
-109 <= Node.val <= 109
-1000 <= targetSum <= 1000

---

438. Find All Anagrams in a String

Given two strings s and p, return an array of all the start indices of p's anagrams in s. You may return the answer in a ny order.
An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all t he original letters exactly once.

Example 1:

Input: s = "cbaebabacd", p = "abc"
Output: [0,6]
Explanation:
The substring with start index = 0 is "cba", which is an anagram of "abc".
The substring with start index = 6 is "bac", which is an anagram of "abc".

Example 2:

Input: s = "abab", p = "ab"
Output: [0,1,2]
Explanation:
The substring with start index = 0 is "ab", which is an anagram of "ab".
The substring with start index = 1 is "ba", which is an anagram of "ab".
The substring with start index = 2 is "ab", which is an anagram of "ab".

Constraints:

1 <= s.length, p.length <= 3 * 104
s and p consist of lowercase English letters.

---

440. K-th Smallest in Lexicographical Order

Given two integers n and k, return the kth lexicographically smallest integer in the range [1, n].

Example 1:

Input: n = 13, k = 2
Output: 10
Explanation: The lexicographical order is [1, 10, 11, 12, 13, 2, 3, 4, 5, 6, 7, 8, 9], so the second smallest number is 1 0.

Example 2:

Input: n = 1, k = 1
Output: 1

Constraints:

1 <= k <= n <= 109

---

441. Arranging Coins

You have n coins and you want to build a staircase with these coins. The staircase consists of k rows where the ith ro w has exactly i coins. The last row of the staircase may be incomplete.
Given the integer n, return the number of complete rows of the staircase you will build.

Example 1:

Input: n = 5
Output: 2
Explanation: Because the 3rd row is incomplete, we return 2.

Example 2:

Input: n = 8
Output: 3
Explanation: Because the 4th row is incomplete, we return 3.

Constraints:

1 <= n <= 231 - 1

---

442. Find All Duplicates in an Array

Given an integer array nums of length n where all the integers of nums are in the range [1, n] and each integer appear s once or twice, return an array of all the integers that appears twice.
You must write an algorithm that runs in O(n) time and uses only constant extra space.

Example 1:
Input: nums = [4,3,2,7,8,2,3,1]

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

Constraints:

n == nums.length
1 <= n <= 105
1 <= nums[i] <= n
Each element in nums appears once or twice.

---

443. String Compression

Given an array of characters chars, compress it using the following algorithm:
Begin with an empty string s. For each group of consecutive repeating characters in chars:

If the group's length is 1, append the character to s.
Otherwise, append the character followed by the group's length.

The compressed string s should not be returned separately, but instead, be stored in the input character array chars. N ote that group lengths that are 10 or longer will be split into multiple characters in chars.
After you are done modifying the input array, return the new length of the array.
You must write an algorithm that uses only constant extra space.

Example 1:

Input: chars = ["a","a","b","b","c","c","c"]
Output: Return 6, and the first 6 characters of the input array should be: ["a","2","b","2","c","3"]
Explanation: The groups are "aa", "bb", and "ccc". This compresses to "a2b2c3".

Example 2:

Input: chars = ["a"]
Output: Return 1, and the first character of the input array should be: ["a"]
Explanation: The only group is "a", which remains uncompressed since it's a single character.

Example 3:

Input: chars = ["a","b","b","b","b","b","b","b","b","b","b","b","b"]
Output: Return 4, and the first 4 characters of the input array should be: ["a","b","1","2"].
Explanation: The groups are "a" and "bbbbbbbbbbbb". This compresses to "ab12".
Example 4:

Input: chars = ["a","a","a","b","b","a","a"]
Output: Return 6, and the first 6 characters of the input array should be: ["a","3","b","2","a","2"].
Explanation: The groups are "aaa", "bb", and "aa". This compresses to "a3b2a2". Note that each group is independen t even if two groups have the same character.

Constraints:

1 <= chars.length <= 2000
chars[i] is a lowercase English letter, uppercase English letter, digit, or symbol.

---

445. Add Two Numbers II

You are given two non-empty linked lists representing two non-negative integers. The most significant digit comes f irst 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 = [7,2,4,3], l2 = [5,6,4]
Output: [7,8,0,7]

Example 2:

Input: l1 = [2,4,3], l2 = [5,6,4]
Output: [8,0,7]

Example 3:

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

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.

Follow up: Could you solve it without reversing the input lists?

---

446. Arithmetic Slices II - Subsequence

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 t
wo 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

---

447. Number of Boomerangs

You are given n points in the plane that are all distinct, where points[i] = [xi, yi]. A boomerang is a tuple of points (i, j, k) such that the distance between i and j equals the distance between i and k (the order of the tuple matters).
Return the number of boomerangs.

Example 1:

Input: points = [[0,0],[1,0],[2,0]]
Output: 2
Explanation: The two boomerangs are [[1,0],[0,0],[2,0]] and [[1,0],[2,0],[0,0]].

Example 2:

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

Example 3:

Input: points = [[1,1]]
Output: 0

Constraints:

n == points.length
1 <= n <= 500
points[i].length == 2
-104 <= xi, yi <= 104
All the points are unique.

---

448. Find All Numbers Disappeared in an Array

Given an array nums of n integers where nums[i] is in the range [1, n], return an array of all the integers in the range [1, n] that do not appear in nums.

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

Constraints:

n == nums.length
1 <= n <= 105
1 <= nums[i] <= n

Follow up: Could you do it without extra space and in O(n) runtime? You may assume the returned list does not cou nt as extra space.

---

449. Serialize and Deserialize BST

Serialization is converting a data structure or object into a sequence of bits so that it can be stored in a file or memor y buffer, or transmitted across a network connection link to be reconstructed later in the same or another computer e nvironment.
Design an algorithm to serialize and deserialize a binary search tree. There is no restriction on how your serialization /deserialization algorithm should work. You need to ensure that a binary search tree can be serialized to a string, and this string can be deserialized to the original tree structure.
The encoded string should be as compact as possible.

Example 1:
Input: root = [2,1,3]
Output: [2,1,3]
Example 2:
Input: root = []
Output: []

Constraints:

The number of nodes in the tree is in the range [0, 104].
0 <= Node.val <= 104
The input tree is guaranteed to be a binary search tree.

---

450. Delete Node in a BST

Given a root node reference of a BST and a key, delete the node with the given key in the BST. Return the root node reference (possibly updated) of the BST.
Basically, the deletion can be divided into two stages:

Search for a node to remove.
If the node is found, delete the node.

Example 1:

Input: root = [5,3,6,2,4,null,7], key = 3
Output: [5,4,6,2,null,null,7]
Explanation: Given key to delete is 3. So we find the node with value 3 and delete it.
One valid answer is [5,4,6,2,null,null,7], shown in the above BST.
Please notice that another valid answer is [5,2,6,null,4,null,7] and it's also accepted.

Example 2:

Input: root = [5,3,6,2,4,null,7], key = 0
Output: [5,3,6,2,4,null,7]
Explanation: The tree does not contain a node with value = 0.

Example 3:

Input: root = [], key = 0
Output: []

Constraints:

The number of nodes in the tree is in the range [0, 104].
-105 <= Node.val <= 105
Each node has a unique value.
root is a valid binary search tree.
-105 <= key <= 105

Follow up: Could you solve it with time complexity O(height of tree)?

---

451. Sort Characters By Frequency

Given a string s, sort it in decreasing order based on the frequency of the characters. The frequency of a character is t he number of times it appears in the string.
Return the sorted string. If there are multiple answers, return any of them.

Example 1:

Input: s = "tree"
Output: "eert"
Explanation: 'e' appears twice while 'r' and 't' both appear once.
So 'e' must appear before both 'r' and 't'. Therefore "eetr" is also a valid answer.

Example 2:

Input: s = "cccaaa"

Output: "aaaccc"
Explanation: Both 'c' and 'a' appear three times, so both "cccaaa" and "aaaccc" are valid answers.
Note that "cacaca" is incorrect, as the same characters must be together.

Example 3:

Input: s = "Aabb"
Output: "bbAa"
Explanation: "bbaA" is also a valid answer, but "Aabb" is incorrect.
Note that 'A' and 'a' are treated as two different characters.

Constraints:

1 <= s.length <= 5 * 105
s consists of uppercase and lowercase English letters and digits.

---

452. Minimum Number of Arrows to Burst Balloons

There are some spherical balloons taped onto a flat wall that represents the XY-plane. The balloons are represented a s a 2D integer array points where points[i] = [xstart, xend] denotes a balloon whose horizontal diameter stretches bet ween xstart and xend. You do not know the exact y-coordinates of the balloons.
Arrows can be shot up directly vertically (in the positive y-direction) from different points along the x-axis. A balloo n with xstart and xend is burst by an arrow shot at x if xstart <= x <= xend. There is no limit to the number of arrows that can be shot. A shot arrow keeps traveling up infinitely, bursting any balloons in its path.
Given the array points, return the minimum number of arrows that must be shot to burst all balloons.

Example 1:

Input: points = [[10,16],[2,8],[1,6],[7,12]]
Output: 2
Explanation: The balloons can be burst by 2 arrows:
- Shoot an arrow at x = 6, bursting the balloons [2,8] and [1,6].
- Shoot an arrow at x = 11, bursting the balloons [10,16] and [7,12].

Example 2:

Input: points = [[1,2],[3,4],[5,6],[7,8]]
Output: 4
Explanation: One arrow needs to be shot for each balloon for a total of 4 arrows.

Example 3:

Input: points = [[1,2],[2,3],[3,4],[4,5]]
Output: 2
Explanation: The balloons can be burst by 2 arrows:
- Shoot an arrow at x = 2, bursting the balloons [1,2] and [2,3].

- Shoot an arrow at x = 4, bursting the balloons [3,4] and [4,5].

Constraints:

1 <= points.length <= 105
points[i].length == 2
-231 <= xstart < xend <= 231 - 1

---

453. Minimum Moves to Equal Array Elements

Given an integer array nums of size n, return the minimum number of moves required to make all array elements equ al.
In one move, you can increment n - 1 elements of the array by 1.

Example 1:

Input: nums = [1,2,3]
Output: 3
Explanation: Only three moves are needed (remember each move increments two elements):
[1,2,3] => [2,3,3] => [3,4,3] => [4,4,4]

Example 2:

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

Constraints:

n == nums.length
1 <= nums.length <= 105
-109 <= nums[i] <= 109
The answer is guaranteed to fit in a 32-bit integer.

---

454. 4Sum II

Given four integer arrays nums1, nums2, nums3, and nums4 all of length n, return the number of tuples (i, j, k, l) suc h that:

0 <= i, j, k, l < n
nums1[i] + nums2[j] + nums3[k] + nums4[l] == 0

Example 1:

Input: nums1 = [1,2], nums2 = [-2,-1], nums3 = [-1,2], nums4 = [0,2]
Output: 2
Explanation:
The two tuples are:
1. (0, 0, 0, 1) -> nums1[0] + nums2[0] + nums3[0] + nums4[1] = 1 + (-2) + (-1) + 2 = 0
2. (1, 1, 0, 0) -> nums1[1] + nums2[1] + nums3[0] + nums4[0] = 2 + (-1) + (-1) + 0 = 0

Example 2:

Input: nums1 = [0], nums2 = [0], nums3 = [0], nums4 = [0]
Output: 1

Constraints:

n == nums1.length
n == nums2.length
n == nums3.length
n == nums4.length
1 <= n <= 200
-228 <= nums1[i], nums2[i], nums3[i], nums4[i] <= 228

---

455. Assign Cookies

Assume you are an awesome parent and want to give your children some cookies. But, you should give each child at most one cookie.
Each child i has a greed factor g[i], which is the minimum size of a cookie that the child will be content with; and ea ch cookie j has a size s[j]. If s[j] >= g[i], we can assign the cookie j to the child i, and the child i will be content. You r goal is to maximize the number of your content children and output the maximum number.

Example 1:

Input: g = [1,2,3], s = [1,1]
Output: 1
Explanation: You have 3 children and 2 cookies. The greed factors of 3 children are 1, 2, 3.
And even though you have 2 cookies, since their size is both 1, you could only make the child whose greed factor is 1 content.
You need to output 1.

Example 2:

Input: g = [1,2], s = [1,2,3]
Output: 2
Explanation: You have 2 children and 3 cookies. The greed factors of 2 children are 1, 2.
You have 3 cookies and their sizes are big enough to gratify all of the children,
You need to output 2.

Constraints:

1 <= g.length <= 3 * 104
0 <= s.length <= 3 * 104
1 <= g[i], s[j] <= 231 - 1

---

456. 132 Pattern

Given an array of n integers nums, a 132 pattern is a subsequence of three integers nums[i], nums[j] and nums[k] suc
h that i < j < k and nums[i] < nums[k] < nums[j].
Return true if there is a 132 pattern in nums, otherwise, return false.

Example 1:

Input: nums = [1,2,3,4]
Output: false
Explanation: There is no 132 pattern in the sequence.

Example 2:

Input: nums = [3,1,4,2]
Output: true
Explanation: There is a 132 pattern in the sequence: [1, 4, 2].

Example 3:

Input: nums = [-1,3,2,0]
Output: true
Explanation: There are three 132 patterns in the sequence: [-1, 3, 2], [-1, 3, 0] and [-1, 2, 0].

Constraints:

n == nums.length
1 <= n <= 2 * 105
-109 <= nums[i] <= 109

---

457. Circular Array Loop

You are playing a game involving a circular array of non-zero integers nums. Each nums[i] denotes the number of in dices forward/backward you must move if you are located at index i:

If nums[i] is positive, move nums[i] steps forward, and
If nums[i] is negative, move nums[i] steps backward.

Since the array is circular, you may assume that moving forward from the last element puts you on the first element, and moving backwards from the first element puts you on the last element.
A cycle in the array consists of a sequence of indices seq of length k where:

Following the movement rules above results in the repeating index sequence seq[0] -> seq[1] -> ... -> seq[k - 1] -> se q[0] -> ...
Every nums[seq[j]] is either all positive or all negative.
k > 1

Return true if there is a cycle in nums, or false otherwise.

Example 1:

Input: nums = [2,-1,1,2,2]
Output: true
Explanation:
There is a cycle from index 0 -> 2 -> 3 -> 0 -> ...
The cycle's length is 3.

Example 2:

Input: nums = [-1,2]
Output: false
Explanation:
The sequence from index 1 -> 1 -> 1 -> ... is not a cycle because the sequence's length is 1.
By definition the sequence's length must be strictly greater than 1 to be a cycle.

Example 3:

Input: nums = [-2,1,-1,-2,-2]
Output: false
Explanation:
The sequence from index 1 -> 2 -> 1 -> ... is not a cycle because nums[1] is positive, but nums[2] is negative.
Every nums[seq[j]] must be either all positive or all negative.

Constraints:

1 <= nums.length <= 5000
-1000 <= nums[i] <= 1000
nums[i] != 0

Follow up: Could you solve it in O(n) time complexity and O(1) extra space complexity?

---

458. Poor Pigs

There are buckets buckets of liquid, where exactly one of the buckets is poisonous. To figure out which one is poisonous, you feed some number of (poor) pigs the liquid to see whether they will die or not. Unfortunately, you only have minutesToTest minutes to determine which bucket is poisonous.
You can feed the pigs according to these steps:

Choose some live pigs to feed.
For each pig, choose which buckets to feed it. The pig will consume all the chosen buckets simultaneously and will take no time.
Wait for minutesToDie minutes. You may not feed any other pigs during this time.
After minutesToDie minutes have passed, any pigs that have been fed the poisonous bucket will die, and all others will survive.
Repeat this process until you run out of time.

Given buckets, minutesToDie, and minutesToTest, return the minimum number of pigs needed to figure out which bucket is poisonous within the allotted time.

Example 1:
Input: buckets = 1000, minutesToDie = 15, minutesToTest = 60
Output: 5
Example 2:
Input: buckets = 4, minutesToDie = 15, minutesToTest = 15
Output: 2
Example 3:
Input: buckets = 4, minutesToDie = 15, minutesToTest = 30
Output: 2

Constraints:

1 <= buckets <= 1000
1 <= minutesToDie <= minutesToTest <= 100

---

459. Repeated Substring Pattern

Given a string s, check if it can be constructed by taking a substring of it and appending multiple copies of the substr ing together.

Example 1:

Input: s = "abab"
Output: true
Explanation: It is the substring "ab" twice.

Example 2:

Input: s = "aba"
Output: false

Example 3:

Input: s = "abcabcabcabc"
Output: true
Explanation: It is the substring "abc" four times or the substring "abcabc" twice.

Constraints:

1 <= s.length <= 104
s consists of lowercase English letters.

---

460. LFU Cache

Design and implement a data structure for a Least Frequently Used (LFU) cache.
Implement the LFUCache class:

LFUCache(int capacity) Initializes the object with the capacity of the data structure.
int get(int key) Gets the value of the key if the key exists in the cache. Otherwise, returns -1.
void put(int key, int value) Update the value of the key if present, or inserts the key if not already present. When the cache reaches its capacity, it should invalidate and remove the least frequently used key before inserting a new item. For this problem, when there is a tie (i.e., two or more keys with the same frequency), the least recently used key wo uld be invalidated.

To determine the least frequently used key, a use counter is maintained for each key in the cache. The key with the s mallest use counter is the least frequently used key.
When a key is first inserted into the cache, its use counter is set to 1 (due to the put operation). The use counter for a key in the cache is incremented either a get or put operation is called on it.
The functions get and put must each run in O(1) average time complexity.

Example 1:

Input

["LFUCache", "put", "put", "get", "put", "get", "get", "put", "get", "get", "get"]
[[2], [1, 1], [2, 2], [1], [3, 3], [2], [3], [4, 4], [1], [3], [4]]
Output
[null, null, null, 1, null, -1, 3, null, -1, 3, 4]

Explanation
// cnt(x) = the use counter for key x
// cache=[] will show the last used order for tiebreakers (leftmost element is most recent)
LFUCache lfu = new LFUCache(2);
lfu.put(1, 1);   // cache=[1,_], cnt(1)=1
lfu.put(2, 2);   // cache=[2,1], cnt(2)=1, cnt(1)=1
lfu.get(1);   // return 1
// cache=[1,2], cnt(2)=1, cnt(1)=2
lfu.put(3, 3);   // 2 is the LFU key because cnt(2)=1 is the smallest, invalidate 2.
// cache=[3,1], cnt(3)=1, cnt(1)=2
lfu.get(2);   // return -1 (not found)
lfu.get(3);   // return 3
// cache=[3,1], cnt(3)=2, cnt(1)=2
lfu.put(4, 4);   // Both 1 and 3 have the same cnt, but 1 is LRU, invalidate 1.
// cache=[4,3], cnt(4)=1, cnt(3)=2
lfu.get(1);   // return -1 (not found)
lfu.get(3);   // return 3
// cache=[3,4], cnt(4)=1, cnt(3)=3
lfu.get(4);   // return 4
// cache=[3,4], cnt(4)=2, cnt(3)=3

Constraints:

0 <= capacity <= 104
0 <= key <= 105
0 <= value <= 109
At most 2 * 105 calls will be made to get and put.

---

461. Hamming Distance

The Hamming distance between two integers is the number of positions at which the corresponding bits are different
.
Given two integers x and y, return the Hamming distance between them.

Example 1:

Input: x = 1, y = 4
Output: 2
Explanation:
1   (0 0 0 1)

4   (0 1 0 0)
      ↑   ↑
The above arrows point to positions where the corresponding bits are different.

Example 2:

Input: x = 3, y = 1
Output: 1

Constraints:

0 <= x, y <= 231 - 1

---

462. Minimum Moves to Equal Array Elements II

Given an integer array nums of size n, return the minimum number of moves required to make all array elements equ
al.
In one move, you can increment or decrement an element of the array by 1.
Test cases are designed so that the answer will fit in a 32-bit integer.

Example 1:

Input: nums = [1,2,3]
Output: 2
Explanation:
Only two moves are needed (remember each move increments or decrements one element):
[1,2,3] => [2,2,3] => [2,2,2]

Example 2:

Input: nums = [1,10,2,9]
Output: 16

Constraints:

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

---

463. Island Perimeter

You are given row x col grid representing a map where grid[i][j] = 1 represents land and grid[i][j] = 0 represents wat
er.
Grid cells are connected horizontally/vertically (not diagonally). The grid is completely surrounded by water, and th
ere is exactly one island (i.e., one or more connected land cells).
The island doesn't have "lakes", meaning the water inside isn't connected to the water around the island. One cell is a
square with side length 1. The grid is rectangular, width and height don't exceed 100. Determine the perimeter of the
island.

Example 1:

Input: grid = [[0,1,0,0],[1,1,1,0],[0,1,0,0],[1,1,0,0]]
Output: 16
Explanation: The perimeter is the 16 yellow stripes in the image above.

Example 2:

Input: grid = [[1]]
Output: 4

Example 3:

Input: grid = [[1,0]]
Output: 4

Constraints:

row == grid.length
col == grid[i].length
1 <= row, col <= 100
grid[i][j] is 0 or 1.
There is exactly one island in grid.

---

464. Can I Win

In the "100 game" two players take turns adding, to a running total, any integer from 1 to 10. The player who first ca
uses the running total to reach or exceed 100 wins.
What if we change the game so that players cannot re-use integers?
For example, two players might take turns drawing from a common pool of numbers from 1 to 15 without replaceme
nt until they reach a total >= 100.
Given two integers maxChoosableInteger and desiredTotal, return true if the first player to move can force a win, oth
erwise, return false. Assume both players play optimally.

Example 1:

Input: maxChoosableInteger = 10, desiredTotal = 11
Output: false
Explanation:
No matter which integer the first player choose, the first player will lose.
The first player can choose an integer from 1 up to 10.
If the first player choose 1, the second player can only choose integers from 2 up to 10.
The second player will win by choosing 10 and get a total = 11, which is >= desiredTotal.
Same with other integers chosen by the first player, the second player will always win.

Example 2:

Input: maxChoosableInteger = 10, desiredTotal = 0
Output: true

Example 3:

Input: maxChoosableInteger = 10, desiredTotal = 1
Output: true

Constraints:

1 <= maxChoosableInteger <= 20
0 <= desiredTotal <= 300

---

466. Count The Repetitions

We define str = [s, n] as the string str which consists of the string s concatenated n times.

For example, str == ["abc", 3] =="abcabcabc".

We define that string s1 can be obtained from string s2 if we can remove some characters from s2 such that it becom es s1.

For example, s1 = "abc" can be obtained from s2 = "abdbec" based on our definition by removing the bolded underli ned characters.

You are given two strings s1 and s2 and two integers n1 and n2. You have the two strings str1 = [s1, n1] and str2 = [s2, n2].
Return the maximum integer m such that str = [str2, m] can be obtained from str1.

Example 1:
Input: s1 = "acb", n1 = 4, s2 = "ab", n2 = 2
Output: 2

Example 2:
Input: s1 = "acb", n1 = 1, s2 = "acb", n2 = 1
Output: 1

Constraints:

1 <= s1.length, s2.length <= 100
s1 and s2 consist of lowercase English letters.
1 <= n1, n2 <= 106

---

467. Unique Substrings in Wraparound String

We define the string s to be the infinite wraparound string of "abcdefghijklmnopqrstuvwxyz", so s will look like this:

"...zabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcd....".

Given a string p, return the number of unique non-empty substrings of p are present in s.

Example 1:

Input: p = "a"
Output: 1
Explanation: Only the substring "a" of p is in s.

Example 2:

Input: p = "cac"
Output: 2
Explanation: There are two substrings ("a", "c") of p in s.

Example 3:

Input: p = "zab"
Output: 6
Explanation: There are six substrings ("z", "a", "b", "za", "ab", and "zab") of p in s.

Constraints:

1 <= p.length <= 105
p consists of lowercase English letters.

---

468. Validate IP Address

Given a string queryIP, return "IPv4" if IP is a valid IPv4 address, "IPv6" if IP is a valid IPv6 address or "Neither" if IP is not a correct IP of any type.
A valid IPv4 address is an IP in the form "x1.x2.x3.x4" where 0 <= xi <= 255 and xi cannot contain leading zeros. For example, "192.168.1.1" and "192.168.1.0" are valid IPv4 addresses but "192.168.01.1", while "192.168.1.00" and "192.168@1.1" are invalid IPv4 addresses.
A valid IPv6 address is an IP in the form "x1:x2:x3:x4:x5:x6:x7:x8" where:

1 <= xi.length <= 4
xi is a hexadecimal string which may contain digits, lower-case English letter ('a' to 'f') and upper-case English letters ('A' to 'F').
Leading zeros are allowed in xi.

For example, "2001:0db8:85a3:0000:0000:8a2e:0370:7334" and "2001:db8:85a3:0:0:8A2E:0370:7334" are valid IPv6 addresses, while "2001:0db8:85a3::8A2E:037j:7334" and "02001:0db8:85a3:0000:0000:8a2e:0370:7334" are invalid IPv6 addresses.

Example 1:

Input: queryIP = "172.16.254.1"
Output: "IPv4"
Explanation: This is a valid IPv4 address, return "IPv4".

Example 2:

Input: queryIP = "2001:0db8:85a3:0:0:8A2E:0370:7334"
Output: "IPv6"
Explanation: This is a valid IPv6 address, return "IPv6".

Example 3:

Input: queryIP = "256.256.256.256"
Output: "Neither"
Explanation: This is neither a IPv4 address nor a IPv6 address.

Example 4:

Input: queryIP = "2001:0db8:85a3:0:0:8A2E:0370:7334:"
Output: "Neither"

Example 5:

Input: queryIP = "1e1.4.5.6"
Output: "Neither"

Constraints:

queryIP consists only of English letters, digits and the characters '.' and ':'.

---

470. Implement Rand10() Using Rand7()

Given the API rand7() that generates a uniform random integer in the range [1, 7], write a function rand10() that gen
erates a uniform random integer in the range [1, 10]. You can only call the API rand7(), and you shouldn't call any ot
her API. Please do not use a language's built-in random API.
Each test case will have one internal argument n, the number of times that your implemented function rand10() will
be called while testing. Note that this is not an argument passed to rand10().

Example 1:
Input: n = 1
Output: [2]
Example 2:
Input: n = 2
Output: [2,8]
Example 3:
Input: n = 3
Output: [3,8,10]

Constraints:

1 <= n <= 105

Follow up:

What is the expected value for the number of calls to rand7() function?
Could you minimize the number of calls to rand7()?

---

472. Concatenated Words

Given an array of strings words (without duplicates), return all the concatenated words in the given list of words.
A concatenated word is defined as a string that is comprised entirely of at least two shorter words in the given array.

Example 1:

Input: words = ["cat","cats","catsdogcats","dog","dogcatsdog","hippopotamuses","rat","ratcatdogcat"]
Output: ["catsdogcats","dogcatsdog","ratcatdogcat"]
Explanation: "catsdogcats" can be concatenated by "cats", "dog" and "cats";

"dogcatsdog" can be concatenated by "dog", "cats" and "dog";
"ratcatdogcat" can be concatenated by "rat", "cat", "dog" and "cat".
Example 2:

Input: words = ["cat","dog","catdog"]
Output: ["catdog"]

Constraints:

1 <= words.length <= 104
0 <= words[i].length <= 1000
words[i] consists of only lowercase English letters.
0 <= sum(words[i].length) <= 105

---

473. Matchsticks to Square

You are given an integer array matchsticks where matchsticks[i] is the length of the ith matchstick. You want to use all the matchsticks to make one square. You should not break any stick, but you can link them up, and each matchsti ck must be used exactly one time.
Return true if you can make this square and false otherwise.

Example 1:

Input: matchsticks = [1,1,2,2,2]
Output: true
Explanation: You can form a square with length 2, one side of the square came two sticks with length 1.

Example 2:

Input: matchsticks = [3,3,3,3,4]
Output: false
Explanation: You cannot find a way to form a square with all the matchsticks.

Constraints:

1 <= matchsticks.length <= 15
1 <= matchsticks[i] <= 108

---

474. Ones and Zeroes

You are given an array of binary strings strs and two integers m and n.
Return the size of the largest subset of strs such that there are at most m 0's and n 1's in the subset.
A set x is a subset of a set y if all elements of x are also elements of y.

Example 1:

Input: strs = ["10","0001","111001","1","0"], m = 5, n = 3
Output: 4
Explanation: The largest subset with at most 5 0's and 3 1's is {"10", "0001", "1", "0"}, so the answer is 4.
Other valid but smaller subsets include {"0001", "1"} and {"10", "1", "0"}.
{"111001"} is an invalid subset because it contains 4 1's, greater than the maximum of 3.

Example 2:

Input: strs = ["10","0","1"], m = 1, n = 1
Output: 2
Explanation: The largest subset is {"0", "1"}, so the answer is 2.

Constraints:

1 <= strs.length <= 600
1 <= strs[i].length <= 100
strs[i] consists only of digits '0' and '1'.
1 <= m, n <= 100

---

475. Heaters

Winter is coming! During the contest, your first job is to design a standard heater with a fixed warm radius to warm all the houses.
Every house can be warmed, as long as the house is within the heater's warm radius range.
Given the positions of houses and heaters on a horizontal line, return the minimum radius standard of heaters so that those heaters could cover all houses.
Notice that all the heaters follow your radius standard, and the warm radius will the same.

Example 1:

Input: houses = [1,2,3], heaters = [2]
Output: 1
Explanation: The only heater was placed in the position 2, and if we use the radius 1 standard, then all the houses ca n be warmed.

Example 2:

Input: houses = [1,2,3,4], heaters = [1,4]
Output: 1
Explanation: The two heater was placed in the position 1 and 4. We need to use radius 1 standard, then all the houses can be warmed.

Example 3:

Input: houses = [1,5], heaters = [2]
Output: 3

Constraints:

1 <= houses.length, heaters.length <= 3 * 104
1 <= houses[i], heaters[i] <= 109

---

476. Number Complement

The complement of an integer is the integer you get when you flip all the 0's to 1's and all the 1's to 0's in its binary r epresentation.

For example, The integer 5 is "101" in binary and its complement is "010" which is the integer 2.

Given an integer num, return its complement.

Example 1:

Input: num = 5
Output: 2
Explanation: The binary representation of 5 is 101 (no leading zero bits), and its complement is 010. So you need to output 2.

Example 2:

Input: num = 1
Output: 0
Explanation: The binary representation of 1 is 1 (no leading zero bits), and its complement is 0. So you need to outp ut 0.

Constraints:

1 <= num < 231

Note: This question is the same as 1009: https://leetcode.com/problems/complement-of-base-10-integer/

---

477. Total Hamming Distance

The Hamming distance between two integers is the number of positions at which the corresponding bits are different
.
Given an integer array nums, return the sum of Hamming distances between all the pairs of the integers in nums.

Example 1:

Input: nums = [4,14,2]
Output: 6
Explanation: In binary representation, the 4 is 0100, 14 is 1110, and 2 is 0010 (just
showing the four bits relevant in this case).
The answer will be:
HammingDistance(4, 14) + HammingDistance(4, 2) + HammingDistance(14, 2) = 2 + 2 + 2 = 6.

Example 2:

Input: nums = [4,14,4]
Output: 4

Constraints:

1 <= nums.length <= 104
0 <= nums[i] <= 109
The answer for the given input will fit in a 32-bit integer.

---

478. Generate Random Point in a Circle

Given the radius and the position of the center of a circle, implement the function randPoint which generates a unifor
m random point inside the circle.
Implement the Solution class:

Solution(double radius, double x_center, double y_center) initializes the object with the radius of the circle radius an
d the position of the center (x_center, y_center).
randPoint() returns a random point inside the circle. A point on the circumference of the circle is considered to be in
the circle. The answer is returned as an array [x, y].

Example 1:

Input
["Solution", "randPoint", "randPoint", "randPoint"]
[[1.0, 0.0, 0.0], [], [], []]
Output
[null, [-0.02493, -0.38077], [0.82314, 0.38945], [0.36572, 0.17248]]

Explanation
Solution solution = new Solution(1.0, 0.0, 0.0);
solution.randPoint(); // return [-0.02493, -0.38077]
solution.randPoint(); // return [0.82314, 0.38945]
solution.randPoint(); // return [0.36572, 0.17248]

Constraints:

0 < radius <= 108
-107 <= x_center, y_center <= 107
At most 3 * 104 calls will be made to randPoint.

---

479. Largest Palindrome Product

Given an integer n, return the largest palindromic integer that can be represented as the product of two n-digits integ ers. Since the answer can be very large, return it modulo 1337.

Example 1:

Input: n = 2
Output: 987
Explanation: 99 x 91 = 9009, 9009 % 1337 = 987

Example 2:

Input: n = 1
Output: 9

Constraints:

1 <= n <= 8

---

480. Sliding Window Median

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

For examples, if arr = [2,3,4], the median is 3.
For examples, if arr = [1,2,3,4], the median is (2 + 3) / 2 = 2.5.

You are given an integer array nums and an integer k. There is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.
Return the median array for each window in the original array. Answers within 10-5 of the actual value will be accep ted.

Example 1:

Input: nums = [1,3,-1,-3,5,3,6,7], k = 3
Output: [1.00000,-1.00000,-1.00000,3.00000,5.00000,6.00000]
Explanation:

Window position	Median
[1 3 -1] -3 5 3 6 7	1
1 [3 -1 -3] 5 3 6 7	-1
1 3 [-1 -3 5] 3 6 7	-1
1 3 -1 [-3 5 3] 6 7	3
1 3 -1 -3 [5 3 6] 7	5
1 3 -1 -3 5 [3 6 7]	6

Example 2:

Input: nums = [1,2,3,4,2,3,1,4,2], k = 3
Output: [2.00000,3.00000,3.00000,3.00000,2.00000,3.00000,2.00000]

Constraints:

1 <= k <= nums.length <= 105
-231 <= nums[i] <= 231 - 1

---

481. Magical String

A magical string s consists of only '1' and '2' and obeys the following rules:

The string s is magical because concatenating the number of contiguous occurrences of characters '1' and '2' generate

s the string s itself.

The first few elements of s is s = "1221121221221121122……". If we group the consecutive 1's and 2's in s, it will b e "1 22 11 2 1 22 1 22 11 2 11 22 ....." and the occurrences of 1's or 2's in each group are "1 2 2 1 1 2 1 2 2 1 2 2 ..... ". You can see that the occurrence sequence is s itself.
Given an integer n, return the number of 1's in the first n number in the magical string s.

Example 1:

Input: n = 6
Output: 3
Explanation: The first 6 elements of magical string s is "122112" and it contains three 1's, so return 3.

Example 2:

Input: n = 1
Output: 1

Constraints:

1 <= n <= 105

---

482. License Key Formatting

You are given a license key represented as a string s that consists of only alphanumeric characters and dashes. The st ring is separated into n + 1 groups by n dashes. You are also given an integer k.
We want to reformat the string s such that each group contains exactly k characters, except for the first group, which could be shorter than k but still must contain at least one character. Furthermore, there must be a dash inserted betwe en two groups, and you should convert all lowercase letters to uppercase.
Return the reformatted license key.

Example 1:

Input: s = "5F3Z-2e-9-w", k = 4
Output: "5F3Z-2E9W"
Explanation: The string s has been split into two parts, each part has 4 characters.
Note that the two extra dashes are not needed and can be removed.

Example 2:

Input: s = "2-5g-3-J", k = 2
Output: "2-5G-3J"
Explanation: The string s has been split into three parts, each part has 2 characters except the first part as it could be shorter as mentioned above.

Constraints:

1 <= s.length <= 105
s consists of English letters, digits, and dashes '-'.
1 <= k <= 104

---

483. Smallest Good Base

Given an integer n represented as a string, return the smallest good base of n.
We call k >= 2 a good base of n, if all digits of n base k are 1's.

Example 1:

Input: n = "13"
Output: "3"
Explanation: 13 base 3 is 111.

Example 2:

Input: n = "4681"
Output: "8"
Explanation: 4681 base 8 is 11111.

Example 3:

Input: n = "1000000000000000000"
Output: "999999999999999999"
Explanation: 1000000000000000000 base 999999999999999999 is 11.

Constraints:

n is an integer in the range [3, 1018].
n does not contain any leading zeros.

---

485. Max Consecutive Ones

Given a binary array nums, return the maximum number of consecutive 1's in the array.

Example 1:

Input: nums = [1,1,0,1,1,1]
Output: 3
Explanation: The first two digits or the last three digits are consecutive 1s. The maximum number of consecutive 1s i s 3.

Example 2:

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

Constraints:

1 <= nums.length <= 105
nums[i] is either 0 or 1.

---

486. Predict the Winner

You are given an integer array nums. Two players are playing a game with this array: player 1 and player 2.
Player 1 and player 2 take turns, with player 1 starting first. Both players start the game with a score of 0. At each tur n, the player takes one of the numbers from either end of the array (i.e., nums[0] or nums[nums.length - 1]) which re duces the size of the array by 1. The player adds the chosen number to their score. The game ends when there are no more elements in the array.
Return true if Player 1 can win the game. If the scores of both players are equal, then player 1 is still the winner, and you should also return true. You may assume that both players are playing optimally.

Example 1:

Input: nums = [1,5,2]
Output: false
Explanation: Initially, player 1 can choose between 1 and 2.
If he chooses 2 (or 1), then player 2 can choose from 1 (or 2) and 5. If player 2 chooses 5, then player 1 will be left with 1 (or 2).
So, final score of player 1 is 1 + 2 = 3, and player 2 is 5.
Hence, player 1 will never be the winner and you need to return false.

Example 2:

Input: nums = [1,5,233,7]
Output: true
Explanation: Player 1 first chooses 1. Then player 2 has to choose between 5 and 7. No matter which number player 2 choose, player 1 can choose 233.
Finally, player 1 has more score (234) than player 2 (12), so you need to return True representing player1 can win.

Constraints:

1 <= nums.length <= 20
0 <= nums[i] <= 107

---

488. Zuma Game

You are playing a variation of the game Zuma.
In this variation of Zuma, there is a single row of colored balls on a board, where each ball can be colored red 'R', yel low 'Y', blue 'B', green 'G', or white 'W'. You also have several colored balls in your hand.
Your goal is to clear all of the balls from the board. On each turn:

Pick any ball from your hand and insert it in between two balls in the row or on either end of the row.
If there is a group of three or more consecutive balls of the same color, remove the group of balls from the board.

If this removal causes more groups of three or more of the same color to form, then continue removing each group u ntil there are none left.

If there are no more balls on the board, then you win the game.
Repeat this process until you either win or do not have any more balls in your hand.

Given a string board, representing the row of balls on the board, and a string hand, representing the balls in your han d, return the minimum number of balls you have to insert to clear all the balls from the board. If you cannot clear all the balls from the board using the balls in your hand, return -1.

Example 1:

Input: board = "WRRBBW", hand = "RB"
Output: -1
Explanation: It is impossible to clear all the balls. The best you can do is:
- Insert 'R' so the board becomes WRRRBBW. WRRRBBW -> WBBW.
- Insert 'B' so the board becomes WBBBW. WBBBW -> WW.
There are still balls remaining on the board, and you are out of balls to insert.
Example 2:

Input: board = "WWRRBBWW", hand = "WRBRW"
Output: 2
Explanation: To make the board empty:
- Insert 'R' so the board becomes WWRRRBBWW. WWRRRBBWW -> WWBBWW.
- Insert 'B' so the board becomes WWBBBWW. WWBBBWW -> WWWW -> empty.
2 balls from your hand were needed to clear the board.

Example 3:

Input: board = "G", hand = "GGGGG"
Output: 2

Explanation: To make the board empty:
- Insert 'G' so the board becomes GG.
- Insert 'G' so the board becomes GGG. GGG -> empty.
2 balls from your hand were needed to clear the board.

Example 4:

Input: board = "RBYYBBRRB", hand = "YRBGB"
Output: 3
Explanation: To make the board empty:
- Insert 'Y' so the board becomes RBYYYBBRRB. RBYYYBBRRB -> RBBBRRB -> RRRB -> B.
- Insert 'B' so the board becomes BB.
- Insert 'B' so the board becomes BBB. BBB -> empty.
3 balls from your hand were needed to clear the board.

Constraints:

1 <= board.length <= 16
1 <= hand.length <= 5
board and hand consist of the characters 'R', 'Y', 'B', 'G', and 'W'.
The initial row of balls on the board will not have any groups of three or more consecutive balls of the same color.

---

491. Increasing Subsequences

Given an integer array nums, return all the different possible increasing subsequences of the given array with at least two elements. You may return the answer in any order.
The given array may contain duplicates, and two equal integers should also be considered a special case of increasing sequence.

Example 1:

Input: nums = [4,6,7,7]
Output: [[4,6],[4,6,7],[4,6,7,7],[4,7],[4,7,7],[6,7],[6,7,7],[7,7]]

Example 2:

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

Constraints:

1 <= nums.length <= 15
-100 <= nums[i] <= 100

---

492. Construct the Rectangle

A web developer needs to know how to design a web page's size. So, given a specific rectangular web page’s area, y our job by now is to design a rectangular web page, whose length L and width W satisfy the following requirements:

The area of the rectangular web page you designed must equal to the given target area.
The width W should not be larger than the length L, which means L >= W.
The difference between length L and width W should be as small as possible.

Return an array [L, W] where L and W are the length and width of the web page you designed in sequence.

Example 1:

Input: area = 4
Output: [2,2]
Explanation: The target area is 4, and all the possible ways to construct it are [1,4], [2,2], [4,1].
But according to requirement 2, [1,4] is illegal; according to requirement 3, [4,1] is not optimal compared to [2,2]. S o the length L is 2, and the width W is 2.

Example 2:

Input: area = 37
Output: [37,1]

Example 3:

Input: area = 122122
Output: [427,286]

Constraints:

1 <= area <= 107

---

493. Reverse Pairs

Given an integer array nums, return the number of reverse pairs in the array.
A reverse pair is a pair (i, j) where 0 <= i < j < nums.length and nums[i] > 2 * nums[j].

Example 1:

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

Constraints:

1 <= nums.length <= 5 * 104
-231 <= nums[i] <= 231 - 1

---

494. Target Sum

You are given an integer array nums and an integer target.
You want to build an expression out of nums by adding one of the symbols '+' and '-' before each integer in nums an d then concatenate all the integers.

For example, if nums = [2, 1], you can add a '+' before 2 and a '-' before 1 and concatenate them to build the expressi on "+2-1".

Return the number of different expressions that you can build, which evaluates to target.

Example 1:

Input: nums = [1,1,1,1,1], target = 3
Output: 5
Explanation: There are 5 ways to assign symbols to make the sum of nums be target 3.
-1 + 1 + 1 + 1 + 1 = 3
+1 - 1 + 1 + 1 + 1 = 3
+1 + 1 - 1 + 1 + 1 = 3
+1 + 1 + 1 - 1 + 1 = 3
+1 + 1 + 1 + 1 - 1 = 3

Example 2:

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

Constraints:

1 <= nums.length <= 20
0 <= nums[i] <= 1000
0 <= sum(nums[i]) <= 1000
-1000 <= target <= 1000

---

495. Teemo Attacking

Our hero Teemo is attacking an enemy Ashe with poison attacks! When Teemo attacks Ashe, Ashe gets poisoned for a exactly duration seconds. More formally, an attack at second t will mean Ashe is poisoned during the inclusive time interval [t, t + duration - 1]. If Teemo attacks again before the poison effect ends, the timer for it is reset, and the poison effect will end duration seconds after the new attack.
You are given a non-decreasing integer array timeSeries, where timeSeries[i] denotes that Teemo attacks Ashe at second timeSeries[i], and an integer duration.
Return the total number of seconds that Ashe is poisoned.

Example 1:

Input: timeSeries = [1,4], duration = 2
Output: 4
Explanation: Teemo's attacks on Ashe go as follows:
- At second 1, Teemo attacks, and Ashe is poisoned for seconds 1 and 2.
- At second 4, Teemo attacks, and Ashe is poisoned for seconds 4 and 5.
Ashe is poisoned for seconds 1, 2, 4, and 5, which is 4 seconds in total.

Example 2:

Input: timeSeries = [1,2], duration = 2
Output: 3
Explanation: Teemo's attacks on Ashe go as follows:
- At second 1, Teemo attacks, and Ashe is poisoned for seconds 1 and 2.
- At second 2 however, Teemo attacks again and resets the poison timer. Ashe is poisoned for seconds 2 and 3.
Ashe is poisoned for seconds 1, 2, and 3, which is 3 seconds in total.

Constraints:

1 <= timeSeries.length <= 104
0 <= timeSeries[i], duration <= 107
timeSeries is sorted in non-decreasing order.

---

496. Next Greater Element I

The next greater element of some element x in an array is the first greater element that is to the right of x in the same array.
You are given two distinct 0-indexed integer arrays nums1 and nums2, where nums1 is a subset of nums2.

For each 0 <= i < nums1.length, find the index j such that nums1[i] == nums2[j] and determine the next greater elem ent of nums2[j] in nums2. If there is no next greater element, then the answer for this query is -1.
Return an array ans of length nums1.length such that ans[i] is the next greater element as described above.

Example 1:

Input: nums1 = [4,1,2], nums2 = [1,3,4,2]
Output: [-1,3,-1]
Explanation: The next greater element for each value of nums1 is as follows:
- 4 is underlined in nums2 = [1,3,4,2]. There is no next greater element, so the answer is -1.
- 1 is underlined in nums2 = [1,3,4,2]. The next greater element is 3.
- 2 is underlined in nums2 = [1,3,4,2]. There is no next greater element, so the answer is -1.

Example 2:

Input: nums1 = [2,4], nums2 = [1,2,3,4]
Output: [3,-1]
Explanation: The next greater element for each value of nums1 is as follows:
- 2 is underlined in nums2 = [1,2,3,4]. The next greater element is 3.
- 4 is underlined in nums2 = [1,2,3,4]. There is no next greater element, so the answer is -1.

Constraints:

1 <= nums1.length <= nums2.length <= 1000
0 <= nums1[i], nums2[i] <= 104
All integers in nums1 and nums2 are unique.
All the integers of nums1 also appear in nums2.

Follow up: Could you find an O(nums1.length + nums2.length) solution?

---

497. Random Point in Non-overlapping Rectangles

You are given an array of non-overlapping axis-aligned rectangles rects where rects[i] = [ai, bi, xi, yi] indicates that (ai, bi) is the bottom-left corner point of the ith rectangle and (xi, yi) is the top-right corner point of the ith rectangle.
Design an algorithm to pick a random integer point inside the space covered by one of the given rectangles. A point on the perimeter of a rectangle is included in the space covered by the rectangle.
Any integer point inside the space covered by one of the given rectangles should be equally likely to be returned.
Note that an integer point is a point that has integer coordinates.
Implement the Solution class:

Solution(int[][] rects) Initializes the object with the given rectangles rects.
int[] pick() Returns a random integer point [u, v] inside the space covered by one of the given rectangles.

Example 1:

Input
["Solution", "pick", "pick", "pick", "pick", "pick"]
[[[[-2, -2, 1, 1], [2, 2, 4, 6]]], [], [], [], [], []]
Output
[null, [1, -2], [1, -1], [-1, -2], [-2, -2], [0, 0]]

Explanation
Solution solution = new Solution([[-2, -2, 1, 1], [2, 2, 4, 6]]);
solution.pick(); // return [1, -2]
solution.pick(); // return [1, -1]
solution.pick(); // return [-1, -2]
solution.pick(); // return [-2, -2]
solution.pick(); // return [0, 0]

Constraints:

1 <= rects.length <= 100
rects[i].length == 4
-109 <= ai < xi <= 109
-109 <= bi < yi <= 109
xi - ai <= 2000
yi - bi <= 2000
All the rectangles do not overlap.
At most 104 calls will be made to pick.

---

498. Diagonal Traverse

Given an m x n matrix mat, return an array of all the elements of the array in a diagonal order.

Example 1:

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

Example 2:

Input: mat = [[1,2],[3,4]]
Output: [1,2,3,4]

Constraints:

m == mat.length
n == mat[i].length
1 <= m, n <= 104

1 <= m * n <= 104
-105 <= mat[i][j] <= 105

---

500. Keyboard Row

Given an array of strings words, return the words that can be typed using letters of the alphabet on only one row of American keyboard like the image below.
In the American keyboard:

the first row consists of the characters "qwertyuiop",
the second row consists of the characters "asdfghjkl", and
the third row consists of the characters "zxcvbnm".

Example 1:

Input: words = ["Hello","Alaska","Dad","Peace"]
Output: ["Alaska","Dad"]

Example 2:

Input: words = ["omk"]
Output: []

Example 3:

Input: words = ["adsdf","sfd"]
Output: ["adsdf","sfd"]

Constraints:

1 <= words.length <= 20
1 <= words[i].length <= 100
words[i] consists of English letters (both lowercase and uppercase).

---

501. Find Mode in Binary Search Tree

Given the root of a binary search tree (BST) with duplicates, return all the mode(s) (i.e., the most frequently occurre d element) in it.
If the tree has more than one mode, return them in any order.
Assume a BST is defined as follows:

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

Example 1:

Input: root = [1,null,2,2]
Output: [2]

Example 2:

Input: root = [0]
Output: [0]

Constraints:

The number of nodes in the tree is in the range [1, 104].
-105 <= Node.val <= 105

Follow up: Could you do that without using any extra space? (Assume that the implicit stack space incurred due to r ecursion does not count).

---

502. IPO

Suppose LeetCode will start its IPO soon. In order to sell a good price of its shares to Venture Capital, LeetCode wo uld like to work on some projects to increase its capital before the IPO. Since it has limited resources, it can only fini sh at most k distinct projects before the IPO. Help LeetCode design the best way to maximize its total capital after fi nishing at most k distinct projects.
You are given n projects where the ith project has a pure profit profits[i] and a minimum capital of capital[i] is neede d to start it.
Initially, you have w capital. When you finish a project, you will obtain its pure profit and the profit will be added to your total capital.
Pick a list of at most k distinct projects from given projects to maximize your final capital, and return the final maxi mized capital.
The answer is guaranteed to fit in a 32-bit signed integer.

Example 1:

Input: k = 2, w = 0, profits = [1,2,3], capital = [0,1,1]
Output: 4

Explanation: Since your initial capital is 0, you can only start the project indexed 0.
After finishing it you will obtain profit 1 and your capital becomes 1.
With capital 1, you can either start the project indexed 1 or the project indexed 2.
Since you can choose at most 2 projects, you need to finish the project indexed 2 to get the maximum capital.
Therefore, output the final maximized capital, which is 0 + 1 + 3 = 4.

Example 2:

Input: k = 3, w = 0, profits = [1,2,3], capital = [0,1,2]
Output: 6

Constraints:

1 <= k <= 105
0 <= w <= 109
n == profits.length
n == capital.length
1 <= n <= 105
0 <= profits[i] <= 104
0 <= capital[i] <= 109

---

503. Next Greater Element II

Given a circular integer array nums (i.e., the next element of nums[nums.length - 1] is nums[0]), return the next grea ter number for every element in nums.
The next greater number of a number x is the first greater number to its traversing-order next in the array, which mea ns you could search circularly to find its next greater number. If it doesn't exist, return -1 for this number.

Example 1:

Input: nums = [1,2,1]
Output: [2,-1,2]
Explanation: The first 1's next greater number is 2;
The number 2 can't find next greater number.
The second 1's next greater number needs to search circularly, which is also 2.

Example 2:

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

Constraints:

1 <= nums.length <= 104
-109 <= nums[i] <= 109

---

504. Base 7

Given an integer num, return a string of its base 7 representation.

Example 1:
Input: num = 100
Output: "202"
Example 2:
Input: num = -7
Output: "-10"

Constraints:

-107 <= num <= 107

---

506. Relative Ranks

You are given an integer array score of size n, where score[i] is the score of the ith athlete in a competition. All the s cores are guaranteed to be unique.
The athletes are placed based on their scores, where the 1st place athlete has the highest score, the 2nd place athlete has the 2nd highest score, and so on. The placement of each athlete determines their rank:

The 1st place athlete's rank is "Gold Medal".
The 2nd place athlete's rank is "Silver Medal".
The 3rd place athlete's rank is "Bronze Medal".
For the 4th place to the nth place athlete, their rank is their placement number (i.e., the xth place athlete's rank is "x") .

Return an array answer of size n where answer[i] is the rank of the ith athlete.

Example 1:

Input: score = [5,4,3,2,1]
Output: ["Gold Medal","Silver Medal","Bronze Medal","4","5"]
Explanation: The placements are [1st, 2nd, 3rd, 4th, 5th].
Example 2:

Input: score = [10,3,8,9,4]
Output: ["Gold Medal","5","Bronze Medal","Silver Medal","4"]
Explanation: The placements are [1st, 5th, 3rd, 2nd, 4th].

Constraints:

n == score.length
1 <= n <= 104
0 <= score[i] <= 106
All the values in score are unique.

---

507. Perfect Number

A perfect number is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. A divisor of an integer x is an integer that can divide x evenly.
Given an integer n, return true if n is a perfect number, otherwise return false.

Example 1:

Input: num = 28
Output: true
Explanation: 28 = 1 + 2 + 4 + 7 + 14
1, 2, 4, 7, and 14 are all divisors of 28.

Example 2:

Input: num = 6
Output: true

Example 3:

Input: num = 496
Output: true

Example 4:

Input: num = 8128
Output: true

Example 5:

Input: num = 2
Output: false

Constraints:

1 <= num <= 108

---

508. Most Frequent Subtree Sum

Given the root of a binary tree, return the most frequent subtree sum. If there is a tie, return all the values with the hi ghest frequency in any order.
The subtree sum of a node is defined as the sum of all the node values formed by the subtree rooted at that node (incl uding the node itself).

Example 1:

Input: root = [5,2,-3]
Output: [2,-3,4]

Example 2:

Input: root = [5,2,-5]
Output: [2]

Constraints:

The number of nodes in the tree is in the range [1, 104].
-105 <= Node.val <= 105

---

509. Fibonacci Number

The Fibonacci numbers, commonly denoted F(n) form a sequence, called the Fibonacci sequence, such that each nu mber is the sum of the two preceding ones, starting from 0 and 1. That is,

F(0) = 0, F(1) = 1
F(n) = F(n - 1) + F(n - 2), for n > 1.

Given n, calculate F(n).

Example 1:

Input: n = 2
Output: 1
Explanation: F(2) = F(1) + F(0) = 1 + 0 = 1.

Example 2:

Input: n = 3
Output: 2
Explanation: F(3) = F(2) + F(1) = 1 + 1 = 2.

Example 3:

Input: n = 4
Output: 3
Explanation: F(4) = F(3) + F(2) = 2 + 1 = 3.

Constraints:

0 <= n <= 30

---

513. Find Bottom Left Tree Value

Given the root of a binary tree, return the leftmost value in the last row of the tree.

Example 1:

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

Example 2:

Input: root = [1,2,3,4,null,5,6,null,null,7]
Output: 7

Constraints:

The number of nodes in the tree is in the range [1, 104].
-231 <= Node.val <= 231 - 1

---

514. Freedom Trail

In the video game Fallout 4, the quest "Road to Freedom" requires players to reach a metal dial called the "Freedom Trail Ring" and use the dial to spell a specific keyword to open the door.
Given a string ring that represents the code engraved on the outer ring and another string key that represents the key word that needs to be spelled, return the minimum number of steps to spell all the characters in the keyword.
Initially, the first character of the ring is aligned at the "12:00" direction. You should spell all the characters in key o ne by one by rotating ring clockwise or anticlockwise to make each character of the string key aligned at the "12:00" direction and then by pressing the center button.
At the stage of rotating the ring to spell the key character key[i]:

You can rotate the ring clockwise or anticlockwise by one place, which counts as one step. The final purpose of the r otation is to align one of ring's characters at the "12:00" direction, where this character must equal key[i].
If the character key[i] has been aligned at the "12:00" direction, press the center button to spell, which also counts as one step. After the pressing, you could begin to spell the next character in the key (next stage). Otherwise, you have finished all the spelling.

Example 1:

Input: ring = "godding", key = "gd"
Output: 4
Explanation:
For the first key character 'g', since it is already in place, we just need 1 step to spell this character.
For the second key character 'd', we need to rotate the ring "godding" anticlockwise by two steps to make it become "ddinggo".
Also, we need 1 more step for spelling.
So the final output is 4.

Example 2:

Input: ring = "godding", key = "godding"
Output: 13

Constraints:

1 <= ring.length, key.length <= 100
ring and key consist of only lower case English letters.
It is guaranteed that key could always be spelled by rotating ring.

---

515. Find Largest Value in Each Tree Row

Given the root of a binary tree, return an array of the largest value in each row of the tree (0-indexed).

Example 1:

Input: root = [1,3,2,5,3,null,9]
Output: [1,3,9]

Example 2:

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

Example 3:

Input: root = [1]
Output: [1]

Example 4:

Input: root = [1,null,2]
Output: [1,2]

Example 5:

Input: root = []
Output: []

Constraints:

The number of nodes in the tree will be in the range [0, 104].
-231 <= Node.val <= 231 - 1

---

516. Longest Palindromic Subsequence

Given a string s, find the longest palindromic subsequence's length in s.
A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without ch anging the order of the remaining elements.

Example 1:

Input: s = "bbbab"

Output: 4
Explanation: One possible longest palindromic subsequence is "bbbb".

Example 2:

Input: s = "cbbd"
Output: 2
Explanation: One possible longest palindromic subsequence is "bb".

Constraints:

1 <= s.length <= 1000
s consists only of lowercase English letters.

---

517. Super Washing Machines

You have n super washing machines on a line. Initially, each washing machine has some dresses or is empty.
For each move, you could choose any m (1 <= m <= n) washing machines, and pass one dress of each washing mach ine to one of its adjacent washing machines at the same time.
Given an integer array machines representing the number of dresses in each washing machine from left to right on th e line, return the minimum number of moves to make all the washing machines have the same number of dresses. If i t is not possible to do it, return -1.

Example 1:

Input: machines = [1,0,5]
Output: 3
Explanation:
1st move: 1 0 <-- 5 => 1 1 4
2nd move: 1 <-- 1 <-- 4 => 2 1 3
3rd move: 2 1 <-- 3 => 2 2 2

Example 2:

Input: machines = [0,3,0]
Output: 2
Explanation:
1st move: 0 <-- 3 0 => 1 2 0
2nd move: 1 2 --> 0 => 1 1 1

Example 3:

Input: machines = [0,2,0]
Output: -1
Explanation:
It's impossible to make all three washing machines have the same number of dresses.

Constraints:

n == machines.length
1 <= n <= 104
0 <= machines[i] <= 105

---

518. Coin Change 2

You are given an integer array coins representing coins of different denominations and an integer amount representin g a total amount of money.
Return the number of combinations that make up that amount. If that amount of money cannot be made up by any co mbination of the coins, return 0.
You may assume that you have an infinite number of each kind of coin.
The answer is guaranteed to fit into a signed 32-bit integer.

Example 1:

Input: amount = 5, coins = [1,2,5]
Output: 4
Explanation: there are four ways to make up the amount:
5=5
5=2+2+1
5=2+1+1+1
5=1+1+1+1+1

Example 2:

Input: amount = 3, coins = [2]
Output: 0
Explanation: the amount of 3 cannot be made up just with coins of 2.

Example 3:

Input: amount = 10, coins = [10]
Output: 1

Constraints:

1 <= coins.length <= 300
1 <= coins[i] <= 5000
All the values of coins are unique.
0 <= amount <= 5000

---

519. Random Flip Matrix

There is an m x n binary grid matrix with all the values set 0 initially. Design an algorithm to randomly pick an inde
x (i, j) where matrix[i][j] == 0 and flips it to 1. All the indices (i, j) where matrix[i][j] == 0 should be equally likely t
o be returned.
Optimize your algorithm to minimize the number of calls made to the built-in random function of your language and
optimize the time and space complexity.
Implement the Solution class:

Solution(int m, int n) Initializes the object with the size of the binary matrix m and n.
int[] flip() Returns a random index [i, j] of the matrix where matrix[i][j] == 0 and flips it to 1.
void reset() Resets all the values of the matrix to be 0.

Example 1:

Input
["Solution", "flip", "flip", "flip", "reset", "flip"]
[[3, 1], [], [], [], [], []]
Output
[null, [1, 0], [2, 0], [0, 0], null, [2, 0]]

Explanation
Solution solution = new Solution(3, 1);
solution.flip(); // return [1, 0], [0,0], [1,0], and [2,0] should be equally likely to be returned.
solution.flip(); // return [2, 0], Since [1,0] was returned, [2,0] and [0,0]
solution.flip(); // return [0, 0], Based on the previously returned indices, only [0,0] can be returned.
solution.reset(); // All the values are reset to 0 and can be returned.
solution.flip(); // return [2, 0], [0,0], [1,0], and [2,0] should be equally likely to be returned.

Constraints:

1 <= m, n <= 104
There will be at least one free cell for each call to flip.
At most 1000 calls will be made to flip and reset.

---

520. Detect Capital

We define the usage of capitals in a word to be right when one of the following cases holds:

All letters in this word are capitals, like "USA".
All letters in this word are not capitals, like "leetcode".
Only the first letter in this word is capital, like "Google".

Given a string word, return true if the usage of capitals in it is right.

Example 1:
Input: word = "USA"
Output: true
Example 2:
Input: word = "FlaG"
Output: false

Constraints:

1 <= word.length <= 100
word consists of lowercase and uppercase English letters.

---

521. Longest Uncommon Subsequence I

Given two strings a and b, return the length of the longest uncommon subsequence between a and b. If the longest un common subsequence does not exist, return -1.
An uncommon subsequence between two strings is a string that is a subsequence of one but not the other.
A subsequence of a string s is a string that can be obtained after deleting any number of characters from s.

For example, "abc" is a subsequence of "aebdc" because you can delete the underlined characters in "aebdc" to get " abc". Other subsequences of "aebdc" include "aebdc", "aeb", and "" (empty string).

Example 1:

Input: a = "aba", b = "cdc"
Output: 3
Explanation: One longest uncommon subsequence is "aba" because "aba" is a subsequence of "aba" but not "cdc".
Note that "cdc" is also a longest uncommon subsequence.

Example 2:

Input: a = "aaa", b = "bbb"
Output: 3
Explanation: The longest uncommon subsequences are "aaa" and "bbb".

Example 3:

Input: a = "aaa", b = "aaa"

Output: -1
Explanation: Every subsequence of string a is also a subsequence of string b. Similarly, every subsequence of string b is also a subsequence of string a.

Constraints:

1 <= a.length, b.length <= 100
a and b consist of lower-case English letters.

---

522. Longest Uncommon Subsequence II

Given an array of strings strs, return the length of the longest uncommon subsequence between them. If the longest u ncommon subsequence does not exist, return -1.
An uncommon subsequence between an array of strings is a string that is a subsequence of one string but not the oth ers.
A subsequence of a string s is a string that can be obtained after deleting any number of characters from s.

For example, "abc" is a subsequence of "aebdc" because you can delete the underlined characters in "aebdc" to get " abc". Other subsequences of "aebdc" include "aebdc", "aeb", and "" (empty string).

Example 1:
Input: strs = ["aba","cdc","eae"]
Output: 3
Example 2:
Input: strs = ["aaa","aaa","aa"]
Output: -1

Constraints:

2 <= strs.length <= 50
1 <= strs[i].length <= 10
strs[i] consists of lowercase English letters.

---

523. Continuous Subarray Sum

Given an integer array nums and an integer k, return true if nums has a continuous subarray of size at least two whos

e elements sum up to a multiple of k, or false otherwise.
An integer x is a multiple of k if there exists an integer n such that x = n * k. 0 is always a multiple of k.

Example 1:

Input: nums = [23,2,4,6,7], k = 6
Output: true
Explanation: [2, 4] is a continuous subarray of size 2 whose elements sum up to 6.

Example 2:

Input: nums = [23,2,6,4,7], k = 6
Output: true
Explanation: [23, 2, 6, 4, 7] is an continuous subarray of size 5 whose elements sum up to 42.
42 is a multiple of 6 because 42 = 7 * 6 and 7 is an integer.

Example 3:

Input: nums = [23,2,6,4,7], k = 13
Output: false

Constraints:

1 <= nums.length <= 105
0 <= nums[i] <= 109
0 <= sum(nums[i]) <= 231 - 1
1 <= k <= 231 - 1

---

524. Longest Word in Dictionary through Deleting

Given a string s and a string array dictionary, return the longest string in the dictionary that can be formed by deletin g some of the given string characters. If there is more than one possible result, return the longest word with the small est lexicographical order. If there is no possible result, return the empty string.

Example 1:

Input: s = "abpcplea", dictionary = ["ale","apple","monkey","plea"]
Output: "apple"

Example 2:

Input: s = "abpcplea", dictionary = ["a","b","c"]
Output: "a"

Constraints:

1 <= s.length <= 1000
1 <= dictionary.length <= 1000
1 <= dictionary[i].length <= 1000
s and dictionary[i] consist of lowercase English letters.

---

525. Contiguous Array

Given a binary array nums, return the maximum length of a contiguous subarray with an equal number of 0 and 1.

Example 1:

Input: nums = [0,1]
Output: 2
Explanation: [0, 1] is the longest contiguous subarray with an equal number of 0 and 1.

Example 2:

Input: nums = [0,1,0]
Output: 2
Explanation: [0, 1] (or [1, 0]) is a longest contiguous subarray with equal number of 0 and 1.

Constraints:

1 <= nums.length <= 105
nums[i] is either 0 or 1.

---

526. Beautiful Arrangement

Suppose you have n integers labeled 1 through n. A permutation of those n integers perm (1-indexed) is considered a beautiful arrangement if for every i (1 <= i <= n), either of the following is true:

perm[i] is divisible by i.
i is divisible by perm[i].

Given an integer n, return the number of the beautiful arrangements that you can construct.

Example 1:

Input: n = 2
Output: 2
Explanation:
The first beautiful arrangement is [1,2]:
    - perm[1] = 1 is divisible by i = 1
    - perm[2] = 2 is divisible by i = 2
The second beautiful arrangement is [2,1]:
    - perm[1] = 2 is divisible by i = 1
    - i = 2 is divisible by perm[2] = 1

Example 2:

Input: n = 1
Output: 1

Constraints:

1 <= n <= 15

---

528. Random Pick with Weight

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] (inclu sive) 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 p icking 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.

---

529. Minesweeper

Let's play the minesweeper game (Wikipedia, online game)!
You are given an m x n char matrix board representing the game board where:

'M' represents an unrevealed mine,
'E' represents an unrevealed empty square,
'B' represents a revealed blank square that has no adjacent mines (i.e., above, below, left, right, and all 4 diagonals),
digit ('1' to '8') represents how many mines are adjacent to this revealed square, and
'X' represents a revealed mine.

You are also given an integer array click where click = [clickr, clickc] represents the next click position among all th e unrevealed squares ('M' or 'E').
Return the board after revealing this position according to the following rules:

If a mine 'M' is revealed, then the game is over. You should change it to 'X'.
If an empty square 'E' with no adjacent mines is revealed, then change it to a revealed blank 'B' and all of its adjacent unrevealed squares should be revealed recursively.

If an empty square 'E' with at least one adjacent mine is revealed, then change it to a digit ('1' to '8') representing the number of adjacent mines.
Return the board when no more squares will be revealed.

Example 1:

Input: board = [["E","E","E","E","E"],["E","E","M","E","E"],["E","E","E","E","E"],["E","E","E","E","E"]], click = [3,0]
Output: [["B","1","E","1","B"],["B","1","M","1","B"],["B","1","1","1","B"],["B","B","B","B","B"]]

Example 2:

Input: board = [["B","1","E","1","B"],["B","1","M","1","B"],["B","1","1","1","B"],["B","B","B","B","B"]], click = [1,2]
Output: [["B","1","E","1","B"],["B","1","X","1","B"],["B","1","1","1","B"],["B","B","B","B","B"]]

Constraints:

m == board.length
n == board[i].length
1 <= m, n <= 50
board[i][j] is either 'M', 'E', 'B', or a digit from '1' to '8'.
click.length == 2
0 <= clickr < m
0 <= clickc < n
board[clickr][clickc] is either 'M' or 'E'.

---

530. Minimum Absolute Difference in BST

Given the root of a Binary Search Tree (BST), return the minimum absolute difference between the values of any two different nodes in the tree.

Example 1:

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

Example 2:

Input: root = [1,0,48,null,null,12,49]
Output: 1

Constraints:

The number of nodes in the tree is in the range [2, 104].
0 <= Node.val <= 105

Note: This question is the same as 783: https://leetcode.com/problems/minimum-distance-between-bst-nodes/

---

532. K-diff Pairs in an Array

Given an array of integers nums and an integer k, return the number of unique k-diff pairs in the array.
A k-diff pair is an integer pair (nums[i], nums[j]), where the following are true:

0 <= i < j < nums.length
|nums[i] - nums[j]| == k

Notice that |val| denotes the absolute value of val.

Example 1:

Input: nums = [3,1,4,1,5], k = 2
Output: 2
Explanation: There are two 2-diff pairs in the array, (1, 3) and (3, 5).
Although we have two 1s in the input, we should only return the number of unique pairs.

Example 2:

Input: nums = [1,2,3,4,5], k = 1
Output: 4
Explanation: There are four 1-diff pairs in the array, (1, 2), (2, 3), (3, 4) and (4, 5).

Example 3:

Input: nums = [1,3,1,5,4], k = 0
Output: 1
Explanation: There is one 0-diff pair in the array, (1, 1).

Example 4:

Input: nums = [1,2,4,4,3,3,0,9,2,3], k = 3
Output: 2

Example 5:

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

Constraints:

1 <= nums.length <= 104
-107 <= nums[i] <= 107
0 <= k <= 107

---

535. Encode and Decode TinyURL

Note: This is a companion problem to the System Design problem: Design TinyURL.
TinyURL is a URL shortening service where you enter a URL such as https://leetcode.com/problems/design-tinyurl and it returns a short URL such as http://tinyurl.com/4e9iAk. Design a class to encode a URL and decode a tiny URL.
There is no restriction on how your encode/decode algorithm should work. You just need to ensure that a URL can be encoded to a tiny URL and the tiny URL can be decoded to the original URL.
Implement the Solution class:

Solution() Initializes the object of the system.
String encode(String longUrl) Returns a tiny URL for the given longUrl.
String decode(String shortUrl) Returns the original long URL for the given shortUrl. It is guaranteed that the given shortUrl was encoded by the same object.

Example 1:

Input: url = "https://leetcode.com/problems/design-tinyurl"
Output: "https://leetcode.com/problems/design-tinyurl"

Explanation:
Solution obj = new Solution();
string tiny = obj.encode(url); // returns the encoded tiny url.
string ans = obj.decode(tiny); // returns the original url after deconding it.

Constraints:

1 <= url.length <= 104
url is guranteed to be a valid URL.

---

537. Complex Number Multiplication

A complex number can be represented as a string on the form "real+imaginaryi" where:

real is the real part and is an integer in the range [-100, 100].
imaginary is the imaginary part and is an integer in the range [-100, 100].
i2 == -1.

Given two complex numbers num1 and num2 as strings, return a string of the complex number that represents their multiplications.

Example 1:

Input: num1 = "1+1i", num2 = "1+1i"
Output: "0+2i"
Explanation: (1 + i) * (1 + i) = 1 + i2 + 2 * i = 2i, and you need convert it to the form of 0+2i.

Example 2:

Input: num1 = "1+-1i", num2 = "1+-1i"
Output: "0+-2i"
Explanation: (1 - i) * (1 - i) = 1 + i2 - 2 * i = -2i, and you need convert it to the form of 0+-2i.

Constraints:

num1 and num2 are valid complex numbers.

---

538. Convert BST to Greater Tree

Given the root of a Binary Search Tree (BST), convert it to a Greater Tree such that every key of the original BST is changed to the original key plus the sum of all keys greater than the original key in BST.
As a reminder, a binary search tree is a tree that satisfies these constraints:

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

Example 1:

Input: root = [4,1,6,0,2,5,7,null,null,null,3,null,null,null,8]
Output: [30,36,21,36,35,26,15,null,null,null,33,null,null,null,8]

Example 2:

Input: root = [0,null,1]
Output: [1,null,1]

Example 3:

Input: root = [1,0,2]
Output: [3,3,2]

Example 4:

Input: root = [3,2,4,1]
Output: [7,9,4,10]

Constraints:

The number of nodes in the tree is in the range [0, 104].
-104 <= Node.val <= 104
All the values in the tree are unique.
root is guaranteed to be a valid binary search tree.

Note: This question is the same as 1038: https://leetcode.com/problems/binary-search-tree-to-greater-sum-tree/

---

539. Minimum Time Difference

Given a list of 24-hour clock time points in "HH:MM" format, return the minimum minutes difference between any t
wo time-points in the list.

Example 1:
Input: timePoints = ["23:59","00:00"]
Output: 1
Example 2:
Input: timePoints = ["00:00","23:59","00:00"]
Output: 0

Constraints:

2 <= timePoints <= 2 * 104
timePoints[i] is in the format "HH:MM".

---

540. Single Element in a Sorted Array

You are given a sorted array consisting of only integers where every element appears exactly twice, except for one el ement which appears exactly once.
Return the single element that appears only once.
Your solution must run in O(log n) time and O(1) space.

Example 1:
Input: nums = [1,1,2,3,3,4,4,8,8]
Output: 2
Example 2:
Input: nums = [3,3,7,7,10,11,11]
Output: 10

Constraints:

1 <= nums.length <= 105
0 <= nums[i] <= 105

---

541. Reverse String II

Given a string s and an integer k, reverse the first k characters for every 2k characters counting from the start of the s tring.
If there are fewer than k characters left, reverse all of them. If there are less than 2k but greater than or equal to k cha racters, then reverse the first k characters and left the other as original.

Example 1:
Input: s = "abcdefg", k = 2
Output: "bacdfeg"
Example 2:
Input: s = "abcd", k = 2
Output: "bacd"

Constraints:

1 <= s.length <= 104
s consists of only lowercase English letters.
1 <= k <= 104

---

542. 01 Matrix

Given an m x n binary matrix mat, return the distance of the nearest 0 for each cell.
The distance between two adjacent cells is 1.

Example 1:

Input: mat = [[0,0,0],[0,1,0],[0,0,0]]
Output: [[0,0,0],[0,1,0],[0,0,0]]

Example 2:

Input: mat = [[0,0,0],[0,1,0],[1,1,1]]
Output: [[0,0,0],[0,1,0],[1,2,1]]

Constraints:

m == mat.length
n == mat[i].length
1 <= m, n <= 104
1 <= m * n <= 104
mat[i][j] is either 0 or 1.
There is at least one 0 in mat.

---

543. Diameter of Binary Tree

Given the root of a binary tree, return the length of the diameter of the tree.
The diameter of a binary tree is the length of the longest path between any two nodes in a tree. This path may or may not pass through the root.
The length of a path between two nodes is represented by the number of edges between them.

Example 1:

Input: root = [1,2,3,4,5]
Output: 3
Explanation: 3 is the length of the path [4,2,1,3] or [5,2,1,3].

Example 2:

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

Constraints:

The number of nodes in the tree is in the range [1, 104].
-100 <= Node.val <= 100

---

546. Remove Boxes

You are given several boxes with different colors represented by different positive numbers.
You may experience several rounds to remove boxes until there is no box left. Each time you can choose some conti nuous boxes with the same color (i.e., composed of k boxes, k >= 1), remove them and get k * k points.
Return the maximum points you can get.

Example 1:

Input: boxes = [1,3,2,2,2,3,4,3,1]
Output: 23
Explanation:
[1, 3, 2, 2, 2, 3, 4, 3, 1]
----> [1, 3, 3, 4, 3, 1] (3*3=9 points)
----> [1, 3, 3, 3, 1] (1*1=1 points)
----> [1, 1] (3*3=9 points)
----> [] (2*2=4 points)

Example 2:

Input: boxes = [1,1,1]
Output: 9

Example 3:

Input: boxes = [1]
Output: 1

Constraints:

1 <= boxes.length <= 100
1 <= boxes[i] <= 100

---

547. Number of Provinces

There are n cities. Some of them are connected, while some are not. If city a is connected directly with city b, and cit
y b is connected directly with city c, then city a is connected indirectly with city c.
A province is a group of directly or indirectly connected cities and no other cities outside of the group.
You are given an n x n matrix isConnected where isConnected[i][j] = 1 if the ith city and the jth city are directly con
nected, and isConnected[i][j] = 0 otherwise.
Return the total number of provinces.

Example 1:

Input: isConnected = [[1,1,0],[1,1,0],[0,0,1]]
Output: 2

Example 2:

Input: isConnected = [[1,0,0],[0,1,0],[0,0,1]]
Output: 3

Constraints:

1 <= n <= 200
n == isConnected.length
n == isConnected[i].length
isConnected[i][j] is 1 or 0.
isConnected[i][i] == 1
isConnected[i][j] == isConnected[j][i]

---

551. Student Attendance Record I

You are given a string s representing an attendance record for a student where each character signifies whether the st
udent was absent, late, or present on that day. The record only contains the following three characters:

'A': Absent.
'L': Late.
'P': Present.

The student is eligible for an attendance award if they meet both of the following criteria:

The student was absent ('A') for strictly fewer than 2 days total.
The student was never late ('L') for 3 or more consecutive days.

Return true if the student is eligible for an attendance award, or false otherwise.

Example 1:

Input: s = "PPALLP"
Output: true
Explanation: The student has fewer than 2 absences and was never late 3 or more consecutive days.

Example 2:

Input: s = "PPALLL"
Output: false
Explanation: The student was late 3 consecutive days in the last 3 days, so is not eligible for the award.

Constraints:

1 <= s.length <= 1000
s[i] is either 'A', 'L', or 'P'.

---

552. Student Attendance Record II

An attendance record for a student can be represented as a string where each character signifies whether the student was absent, late, or present on that day. The record only contains the following three characters:

'A': Absent.
'L': Late.
'P': Present.

Any student is eligible for an attendance award if they meet both of the following criteria:

The student was absent ('A') for strictly fewer than 2 days total.
The student was never late ('L') for 3 or more consecutive days.

Given an integer n, return the number of possible attendance records of length n that make a student eligible for an at tendance award. The answer may be very large, so return it modulo 109 + 7.

Example 1:

Input: n = 2
Output: 8
Explanation: There are 8 records with length 2 that are eligible for an award:
"PP", "AP", "PA", "LP", "PL", "AL", "LA", "LL"
Only "AA" is not eligible because there are 2 absences (there need to be fewer than 2).

Example 2:

Input: n = 1
Output: 3

Example 3:

Input: n = 10101
Output: 183236316

Constraints:

1 <= n <= 105

---

553. Optimal Division

You are given an integer array nums. The adjacent integers in nums will perform the float division.

For example, for nums = [2,3,4], we will evaluate the expression "2/3/4".

However, you can add any number of parenthesis at any position to change the priority of operations. You want to a dd these parentheses such the value of the expression after the evaluation is maximum.
Return the corresponding expression that has the maximum value in string format.
Note: your expression should not contain redundant parenthesis.

Example 1:

Input: nums = [1000,100,10,2]
Output: "1000/(100/10/2)"
Explanation:
1000/(100/10/2) = 1000/((100/10)/2) = 200
However, the bold parenthesis in "1000/((100/10)/2)" are redundant, since they don't influence the operation priority.
So you should return "1000/(100/10/2)".
Other cases:
1000/(100/10)/2 = 50
1000/(100/(10/2)) = 50
1000/100/10/2 = 0.5
1000/100/(10/2) = 2

Example 2:

Input: nums = [2,3,4]
Output: "2/(3/4)"

Example 3:

Input: nums = [2]
Output: "2"

Constraints:

1 <= nums.length <= 10
2 <= nums[i] <= 1000
There is only one optimal division for the given iput.

---

554. Brick Wall

There is a rectangular brick wall in front of you with n rows of bricks. The ith row has some number of bricks each o f the same height (i.e., one unit) but they can be of different widths. The total width of each row is the same.
Draw a vertical line from the top to the bottom and cross the least bricks. If your line goes through the edge of a bric k, then the brick is not considered as crossed. You cannot draw a line just along one of the two vertical edges of the wall, in which case the line will obviously cross no bricks.
Given the 2D array wall that contains the information about the wall, return the minimum number of crossed bricks a fter drawing such a vertical line.

Example 1:

Input: wall = [[1,2,2,1],[3,1,2],[1,3,2],[2,4],[3,1,2],[1,3,1,1]]
Output: 2

Example 2:

Input: wall = [[1],[1],[1]]
Output: 3

Constraints:

n == wall.length
1 <= n <= 104
1 <= wall[i].length <= 104
1 <= sum(wall[i].length) <= 2 * 104
sum(wall[i]) is the same for each row i.
1 <= wall[i][j] <= 231 - 1

---

556. Next Greater Element III

Given a positive integer n, find the smallest integer which has exactly the same digits existing in the integer n and is greater in value than n. If no such positive integer exists, return -1.
Note that the returned integer should fit in 32-bit integer, if there is a valid answer but it does not fit in 32-bit integer, return -1.

Example 1:
Input: n = 12
Output: 21
Example 2:
Input: n = 21
Output: -1

Constraints:

1 <= n <= 231 - 1

---

557. Reverse Words in a String III

Given a string s, reverse the order of characters in each word within a sentence while still preserving whitespace and initial word order.

Example 1:
Input: s = "Let's take LeetCode contest"
Output: "s'teL ekat edoCteeL tsetnoc"
Example 2:
Input: s = "God Ding"
Output: "doG gniD"

Constraints:

1 <= s.length <= 5 * 104
s contains printable ASCII characters.
s does not contain any leading or trailing spaces.
There is at least one word in s.
All the words in s are separated by a single space.

---

558. Logical OR of Two Binary Grids Represented as Quad-Trees

A Binary Matrix is a matrix in which all the elements are either 0 or 1.
Given quadTree1 and quadTree2. quadTree1 represents a n * n binary matrix and quadTree2 represents another n * n binary matrix.
Return a Quad-Tree representing the n * n binary matrix which is the result of logical bitwise OR of the two binary matrixes represented by quadTree1 and quadTree2.
Notice that you can assign the value of a node to True or False when isLeaf is False, and both are accepted in the ans wer.
A Quad-Tree is a tree data structure in which each internal node has exactly four children. Besides, each node has tw o attributes:

val: True if the node represents a grid of 1's or False if the node represents a grid of 0's.
isLeaf: True if the node is leaf node on the tree or False if the node has the four children.

```
class Node {
    public boolean val;
    public boolean isLeaf;
    public Node topLeft;
    public Node topRight;
    public Node bottomLeft;
    public Node bottomRight;
}
```

We can construct a Quad-Tree from a two-dimensional area using the following steps:

If the current grid has the same value (i.e all 1's or all 0's) set isLeaf True and set val to the value of the grid and set t he four children to Null and stop.
If the current grid has different values, set isLeaf to False and set val to any value and divide the current grid into fou r sub-grids as shown in the photo.
Recurse for each of the children with the proper sub-grid.

If you want to know more about the Quad-Tree, you can refer to the wiki.
Quad-Tree format:
The input/output represents the serialized format of a Quad-Tree using level order traversal, where null signifies a pa th terminator where no node exists below.
It is very similar to the serialization of the binary tree. The only difference is that the node is represented as a list [is Leaf, val].
If the value of isLeaf or val is True we represent it as 1 in the list [isLeaf, val] and if the value of isLeaf or val is Fals e we represent it as 0.

Example 1:

Input: quadTree1 = [[0,1],[1,1],[1,1],[1,0],[1,0]]
, quadTree2 = [[0,1],[1,1],[0,1],[1,1],[1,0],null,null,null,null,[1,0],[1,0],[1,1],[1,1]]
Output: [[0,0],[1,1],[1,1],[1,1],[1,0]]
Explanation: quadTree1 and quadTree2 are shown above. You can see the binary matrix which is represented by eac h Quad-Tree.
If we apply logical bitwise OR on the two binary matrices we get the binary matrix below which is represented by th

e result Quad-Tree.
Notice that the binary matrices shown are only for illustration, you don't have to construct the binary matrix to get th e result tree.

Example 2:

Input: quadTree1 = [[1,0]]
, quadTree2 = [[1,0]]
Output: [[1,0]]
Explanation: Each tree represents a binary matrix of size 1*1. Each matrix contains only zero.
The resulting matrix is of size 1*1 with also zero.

Example 3:

Input: quadTree1 = [[0,0],[1,0],[1,0],[1,1],[1,1]]
, quadTree2 = [[0,0],[1,1],[1,1],[1,0],[1,1]]
Output: [[1,1]]

Example 4:

Input: quadTree1 = [[0,0],[1,1],[1,0],[1,1],[1,1]]
, quadTree2 = [[0,0],[1,1],[0,1],[1,1],[1,1],null,null,null,null,[1,1],[1,0],[1,0],[1,1]]
Output: [[0,0],[1,1],[0,1],[1,1],[1,1],null,null,null,null,[1,1],[1,0],[1,0],[1,1]]

Example 5:

Input: quadTree1 = [[0,1],[1,0],[0,1],[1,1],[1,0],null,null,null,null,[1,0],[1,0],[1,1],[1,1]]
, quadTree2 = [[0,1],[0,1],[1,0],[1,1],[1,0],[1,0],[1,0],[1,1],[1,1]]
Output: [[0,0],[0,1],[0,1],[1,1],[1,0],[1,0],[1,0],[1,1],[1,1],[1,0],[1,0],[1,1],[1,1]]

Constraints:

quadTree1 and quadTree2 are both valid Quad-Trees each representing a n * n grid.
n == 2^x where 0 <= x <= 9.

---

559. Maximum Depth of N-ary Tree

Given a n-ary tree, find its maximum depth.
The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node .
Nary-Tree input serialization is represented in their level order traversal, each group of children is separated by the n ull value (See examples).

Example 1:

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

Example 2:

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

Constraints:

The total number of nodes is in the range [0, 104].
The depth of the n-ary tree is less than or equal to 1000.

---

560. Subarray Sum Equals K

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

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

---

561. Array Partition I

Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2), ..., (an, bn) such that t he sum of min(ai, bi) for all i is maximized. Return the maximized sum.

Example 1:

Input: nums = [1,4,3,2]
Output: 4
Explanation: All possible pairings (ignoring the ordering of elements) are:
1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3
2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3
3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4
So the maximum possible sum is 4.
Example 2:

Input: nums = [6,2,6,5,1,2]
Output: 9
Explanation: The optimal pairing is (2, 1), (2, 5), (6, 6). min(2, 1) + min(2, 5) + min(6, 6) = 1 + 2 + 6 = 9.

Constraints:

1 <= n <= 104
nums.length == 2 * n
-104 <= nums[i] <= 104

---

563. Binary Tree Tilt

Given the root of a binary tree, return the sum of every tree node's tilt.
The tilt of a tree node is the absolute difference between the sum of all left subtree node values and all right subtree node values. If a node does not have a left child, then the sum of the left subtree node values is treated as 0. The rule is similar if there the node does not have a right child.

Example 1:

Input: root = [1,2,3]
Output: 1
Explanation:
Tilt of node 2 : |0-0| = 0 (no children)
Tilt of node 3 : |0-0| = 0 (no children)
Tilt of node 1 : |2-3| = 1 (left subtree is just left child, so sum is 2; right subtree is just right child, so sum is 3)
Sum of every tilt : 0 + 0 + 1 = 1

Example 2:

Input: root = [4,2,9,3,5,null,7]
Output: 15
Explanation:

Tilt of node 3 : |0-0| = 0 (no children)
Tilt of node 5 : |0-0| = 0 (no children)
Tilt of node 7 : |0-0| = 0 (no children)
Tilt of node 2 : |3-5| = 2 (left subtree is just left child, so sum is 3; right subtree is just right child, so sum is 5)
Tilt of node 9 : |0-7| = 7 (no left child, so sum is 0; right subtree is just right child, so sum is 7)
Tilt of node 4 : |(3+5+2)-(9+7)| = |10-16| = 6 (left subtree values are 3, 5, and 2, which sums to 10; right subtree values are 9 and 7, which sums to 16)
Sum of every tilt : 0 + 0 + 0 + 2 + 7 + 6 = 15

Example 3:

Input: root = [21,7,14,1,1,2,2,3,3]
Output: 9

Constraints:

The number of nodes in the tree is in the range [0, 104].
-1000 <= Node.val <= 1000

---

564. Find the Closest Palindrome

Given a string n representing an integer, return the closest integer (not including itself), which is a palindrome. If there is a tie, return the smaller one.
The closest is defined as the absolute difference minimized between two integers.

Example 1:

Input: n = "123"
Output: "121"

Example 2:

Input: n = "1"
Output: "0"
Explanation: 0 and 2 are the closest palindromes but we return the smallest which is 0.

Constraints:

1 <= n.length <= 18
n consists of only digits.
n does not have leading zeros.
n is representing an integer in the range [1, 1018 - 1].

---

565. Array Nesting

You are given an integer array nums of length n where nums is a permutation of the numbers in the range [0, n - 1].
You should build a set s[k] = {nums[k], nums[nums[k]], nums[nums[nums[k]]], ... } subjected to the following rule:

The first element in s[k] starts with the selection of the element nums[k] of index = k.
The next element in s[k] should be nums[nums[k]], and then nums[nums[nums[k]]], and so on.
We stop adding right before a duplicate element occurs in s[k].

Return the longest length of a set s[k].

Example 1:

Input: nums = [5,4,0,3,1,6,2]
Output: 4
Explanation:
nums[0] = 5, nums[1] = 4, nums[2] = 0, nums[3] = 3, nums[4] = 1, nums[5] = 6, nums[6] = 2.
One of the longest sets s[k]:
s[0] = {nums[0], nums[5], nums[6], nums[2]} = {5, 6, 2, 0}

Example 2:

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

Constraints:

1 <= nums.length <= 105
0 <= nums[i] < nums.length
All the values of nums are unique.

---

566. Reshape the Matrix

In MATLAB, there is a handy function called reshape which can reshape an m x n matrix into a new one with a diffe rent size r x c keeping its original data.
You are given an m x n matrix mat and two integers r and c representing the number of rows and the number of colu mns of the wanted reshaped matrix.
The reshaped matrix should be filled with all the elements of the original matrix in the same row-traversing order as they were.

If the reshape operation with given parameters is possible and legal, output the new reshaped matrix; Otherwise, out put the original matrix.

Example 1:

Input: mat = [[1,2],[3,4]], r = 1, c = 4
Output: [[1,2,3,4]]

Example 2:

Input: mat = [[1,2],[3,4]], r = 2, c = 4
Output: [[1,2],[3,4]]

Constraints:

m == mat.length
n == mat[i].length
1 <= m, n <= 100
-1000 <= mat[i][j] <= 1000
1 <= r, c <= 300

---

567. Permutation in String

Given two strings s1 and s2, return true if s2 contains a permutation of s1, or false otherwise.
In other words, return true if one of s1's permutations is the substring of s2.

Example 1:

Input: s1 = "ab", s2 = "eidbaooo"
Output: true
Explanation: s2 contains one permutation of s1 ("ba").

Example 2:

Input: s1 = "ab", s2 = "eidboaoo"
Output: false

Constraints:

1 <= s1.length, s2.length <= 104
s1 and s2 consist of lowercase English letters.

---

572. Subtree of Another Tree

Given the roots of two binary trees root and subRoot, return true if there is a subtree of root with the same structure a nd node values of subRoot and false otherwise.
A subtree of a binary tree tree is a tree that consists of a node in tree and all of this node's descendants. The tree tree could also be considered as a subtree of itself.

Example 1:

Input: root = [3,4,5,1,2], subRoot = [4,1,2]
Output: true

Example 2:

Input: root = [3,4,5,1,2,null,null,null,null,0], subRoot = [4,1,2]
Output: false

Constraints:

The number of nodes in the root tree is in the range [1, 2000].
The number of nodes in the subRoot tree is in the range [1, 1000].
-104 <= root.val <= 104
-104 <= subRoot.val <= 104

---

575. Distribute Candies

Alice has n candies, where the ith candy is of type candyType[i]. Alice noticed that she started to gain weight, so she visited a doctor.
The doctor advised Alice to only eat n / 2 of the candies she has (n is always even). Alice likes her candies very muc h, and she wants to eat the maximum number of different types of candies while still following the doctor's advice.
Given the integer array candyType of length n, return the maximum number of different types of candies she can eat if she only eats n / 2 of them.

Example 1:

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

Explanation: Alice can only eat 6 / 2 = 3 candies. Since there are only 3 types, she can eat one of each type.

Example 2:

Input: candyType = [1,1,2,3]
Output: 2
Explanation: Alice can only eat 4 / 2 = 2 candies. Whether she eats types [1,2], [1,3], or [2,3], she still can only eat 2 different types.

Example 3:

Input: candyType = [6,6,6,6]
Output: 1
Explanation: Alice can only eat 4 / 2 = 2 candies. Even though she can eat 2 candies, she only has 1 type.

Constraints:

n == candyType.length
2 <= n <= 104
n is even.
-105 <= candyType[i] <= 105

---

576. Out of Boundary Paths

There is an m x n grid with a ball. The ball is initially at the position [startRow, startColumn]. You are allowed to m ove the ball to one of the four adjacent cells in the grid (possibly out of the grid crossing the grid boundary). You ca n apply at most maxMove moves to the ball.
Given the five integers m, n, maxMove, startRow, startColumn, return the number of paths to move the ball out of th e grid boundary. Since the answer can be very large, return it modulo 109 + 7.

Example 1:

Input: m = 2, n = 2, maxMove = 2, startRow = 0, startColumn = 0
Output: 6

Example 2:

Input: m = 1, n = 3, maxMove = 3, startRow = 0, startColumn = 1
Output: 12

Constraints:

1 <= m, n <= 50

0 <= maxMove <= 50
0 <= startRow < m
0 <= startColumn < n

---

581. Shortest Unsorted Continuous Subarray

Given an integer array nums, you need to find one continuous subarray that if you only sort this subarray in ascendin g order, then the whole array will be sorted in ascending order.
Return the shortest such subarray and output its length.

Example 1:

Input: nums = [2,6,4,8,10,9,15]
Output: 5
Explanation: You need to sort [6, 4, 8, 10, 9] in ascending order to make the whole array sorted in ascending order.

Example 2:

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

Example 3:

Input: nums = [1]
Output: 0

Constraints:

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

Follow up: Can you solve it in O(n) time complexity?

---

583. Delete Operation for Two Strings

Given two strings word1 and word2, return the minimum number of steps required to make word1 and word2 the sa me.
In one step, you can delete exactly one character in either string.

Example 1:

Input: word1 = "sea", word2 = "eat"
Output: 2
Explanation: You need one step to make "sea" to "ea" and another step to make "eat" to "ea".

Example 2:

Input: word1 = "leetcode", word2 = "etco"
Output: 4

Constraints:

1 <= word1.length, word2.length <= 500
word1 and word2 consist of only lowercase English letters.

---

587. Erect the Fence

You are given an array trees where trees[i] = [xi, yi] represents the location of a tree in the garden.
You are asked to fence the entire garden using the minimum length of rope as it is expensive. The garden is well fen ced only if all the trees are enclosed.
Return the coordinates of trees that are exactly located on the fence perimeter.

Example 1:

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

Example 2:

Input: points = [[1,2],[2,2],[4,2]]
Output: [[4,2],[2,2],[1,2]]

Constraints:

1 <= points.length <= 3000
points[i].length == 2
0 <= xi, yi <= 100
All the given points are unique.

---

589. N-ary Tree Preorder Traversal

Given the root of an n-ary tree, return the preorder traversal of its nodes' values.
Nary-Tree input serialization is represented in their level order traversal. Each group of children is separated by the n ull value (See examples)

Example 1:

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

Example 2:

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

Constraints:

The number of nodes in the tree is in the range [0, 104].
0 <= Node.val <= 104
The height of the n-ary tree is less than or equal to 1000.

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

---

590. N-ary Tree Postorder Traversal

Given the root of an n-ary tree, return the postorder traversal of its nodes' values.
Nary-Tree input serialization is represented in their level order traversal. Each group of children is separated by the n ull value (See examples)

Example 1:

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

Example 2:

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

Constraints:

The number of nodes in the tree is in the range [0, 104].
0 <= Node.val <= 104
The height of the n-ary tree is less than or equal to 1000.

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

---

591. Tag Validator

Given a string representing a code snippet, implement a tag validator to parse the code and return whether it is valid. A code snippet is valid if all the following rules hold:

The code must be wrapped in a valid closed tag. Otherwise, the code is invalid.
A closed tag (not necessarily valid) has exactly the following format : <TAG_NAME>TAG_CONTENT</TAG_NAME>. Among them, <TAG_NAME> is the start tag, and </TAG_NAME> is the end tag. The TAG_NAME in start and end tags should be the same. A closed tag is valid if and only if the TAG_NAME and TAG_CONTENT are valid.
A valid TAG_NAME only contain upper-case letters, and has length in range [1,9]. Otherwise, the TAG_NAME is invalid.
A valid TAG_CONTENT may contain other valid closed tags, cdata and any characters (see note1) EXCEPT unmatched <, unmatched start and end tag, and unmatched or closed tags with invalid TAG_NAME. Otherwise, the TAG_CONTENT is invalid.
A start tag is unmatched if no end tag exists with the same TAG_NAME, and vice versa. However, you also need to consider the issue of unbalanced when tags are nested.
A < is unmatched if you cannot find a subsequent >. And when you find a < or </, all the subsequent characters until the next > should be parsed as TAG_NAME (not necessarily valid).
The cdata has the following format : <![CDATA[CDATA_CONTENT]]>. The range of CDATA_CONTENT is defined as the characters between <![CDATA[and the first subsequent]]>.
CDATA_CONTENT may contain any characters. The function of cdata is to forbid the validator to parse CDATA_CONTENT, so even it has some characters that can be parsed as tag (no matter valid or invalid), you should treat it as regular characters.

Example 1:

Input: code = "<DIV>This is the first line <![CDATA[<div>]]></DIV>"
Output: true
Explanation:

The code is wrapped in a closed tag : <DIV> and </DIV>.
The TAG_NAME is valid, the TAG_CONTENT consists of some characters and cdata.
Although CDATA_CONTENT has an unmatched start tag with invalid TAG_NAME, it should be considered as pla in text, not parsed as a tag.
So TAG_CONTENT is valid, and then the code is valid. Thus return true.

Example 2:

Input: code = "<DIV>>>> ![cdata[]] <![CDATA[<div>]>]]>]]>>]</DIV>"
Output: true
Explanation:
We first separate the code into : start_tag|tag_content|end_tag.
start_tag -> "<DIV>"
end_tag -> "</DIV>"
tag_content could also be separated into : text1|cdata|text2.
text1 -> ">> ![cdata[]] "
cdata -> "<![CDATA[<div>]>]]>", where the CDATA_CONTENT is "<div>]>"
text2 -> "]]>>]"
The reason why start_tag is NOT "<DIV>>>>" is because of the rule 6.
The reason why cdata is NOT "<![CDATA[<div>]>]]>]]>" is because of the rule 7.

Example 3:

Input: code = "<A> <B> </A>   </B>"
Output: false
Explanation: Unbalanced. If "<A>" is closed, then "<B>" must be unmatched, and vice versa.

Example 4:

Input: code = "<DIV> div tag is not closed <DIV>"
Output: false

Constraints:

1 <= code.length <= 500
code consists of English letters, digits, '<', '>', '/', '!', '[', ']', '.', and ' '.

---

592. Fraction Addition and Subtraction

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

Example 1:

Input: expression = "-1/2+1/2"
Output: "0/1"

Example 2:

Input: expression = "-1/2+1/2+1/3"
Output: "1/3"

Example 3:

Input: expression = "1/3-1/2"
Output: "-1/6"

Example 4:

Input: expression = "5/3+1/3"
Output: "2/1"

Constraints:

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

---

593. Valid Square

Given the coordinates of four points in 2D space p1, p2, p3 and p4, return true if the four points construct a square.
The coordinate of a point pi is represented as [xi, yi]. The input is not given in any order.
A valid square has four equal sides with positive length and four equal angles (90-degree angles).

Example 1:

Input: p1 = [0,0], p2 = [1,1], p3 = [1,0], p4 = [0,1]
Output: true

Example 2:

Input: p1 = [0,0], p2 = [1,1], p3 = [1,0], p4 = [0,12]
Output: false

Example 3:

Input: p1 = [1,0], p2 = [-1,0], p3 = [0,1], p4 = [0,-1]
Output: true

Constraints:

p1.length == p2.length == p3.length == p4.length == 2
-104 <= xi, yi <= 104

---

594. Longest Harmonious Subsequence

We define a harmonious array as an array where the difference between its maximum value and its minimum value i s exactly 1.
Given an integer array nums, return the length of its longest harmonious subsequence among all its possible subsequ ences.
A subsequence of array is a sequence that can be derived from the array by deleting some or no elements without ch anging the order of the remaining elements.

Example 1:

Input: nums = [1,3,2,2,5,2,3,7]
Output: 5
Explanation: The longest harmonious subsequence is [3,2,2,2,3].

Example 2:

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

Example 3:

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

Constraints:

1 <= nums.length <= 2 * 104
-109 <= nums[i] <= 109

---

595. Big Countries

Table: World

+-------------+---------+
| Column Name | Type |
+-------------+---------+
name	varchar
continent	varchar
area	int
population	int
gdp	int
+-------------+---------+
name is the primary key column for this table.
Each row of this table gives information about the name of a country, the continent to which it belongs, its area, the population, and its GDP value.

A country is big if:

it has an area of at least three million (i.e., 3000000 km2), or
it has a population of at least twenty-five million (i.e., 25000000).

Write an SQL query to report the name, population, and area of the big countries.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
World table:
+-------------+-----------+---------+------------+--------------+
| name | continent | area | population | gdp |
+-------------+-----------+---------+------------+--------------+
Afghanistan	Asia	652230	25500100	20343000000
Albania	Europe	28748	2831741	12960000000
Algeria	Africa	2381741	37100000	188681000000
Andorra	Europe	468	78115	3712000000
Angola	Africa	1246700	20609294	100990000000
+-------------+-----------+---------+------------+--------------+
Output:
+-------------+------------+---------+
| name | population | area |
+-------------+------------+---------+
| Afghanistan | 25500100 | 652230 |
| Algeria | 37100000 | 2381741 |
+-------------+------------+---------+

---

596. Classes More Than 5 Students

Table: Courses

+-------------+---------+
| Column Name | Type |
+-------------+---------+
| student | varchar |
| class | varchar |
+-------------+---------+
(student, class) is the primary key column for this table.
Each row of this table indicates the name of a student and the class in which they are enrolled.

Write an SQL query to report all the classes that have at least five students.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Courses table:
+---------+----------+
| student | class |
+---------+----------+
A	Math
B	English
C	Math
D	Biology
E	Math
F	Computer
G	Math
H	Math
I	Math
+---------+----------+
Output:
+---------+
| class |
+---------+
| Math |
+---------+
Explanation:
- Math has 6 students, so we include it.
- English has 1 student, so we do not include it.
- Biology has 1 student, so we do not include it.
- Computer has 1 student, so we do not include it.

---

598. Range Addition II

You are given an m x n matrix M initialized with all 0's and an array of operations ops, where ops[i] = [ai, bi] means M[x][y] should be incremented by one for all 0 <= x < ai and 0 <= y < bi.
Count and return the number of maximum integers in the matrix after performing all the operations.

Example 1:

Input: m = 3, n = 3, ops = [[2,2],[3,3]]
Output: 4
Explanation: The maximum integer in M is 2, and there are four of it in M. So return 4.

Example 2:

Input: m = 3, n = 3, ops = [[2,2],[3,3],[3,3],[3,3],[2,2],[3,3],[3,3],[3,3],[2,2],[3,3],[3,3],[3,3]]
Output: 4

Example 3:

Input: m = 3, n = 3, ops = []
Output: 9

Constraints:

1 <= m, n <= 4 * 104
0 <= ops.length <= 104
ops[i].length == 2
1 <= ai <= m
1 <= bi <= n

---

599. Minimum Index Sum of Two Lists

Suppose Andy and Doris want to choose a restaurant for dinner, and they both have a list of favorite restaurants repr esented by strings.
You need to help them find out their common interest with the least list index sum. If there is a choice tie between an swers, output all of them with no order requirement. You could assume there always exists an answer.

Example 1:

Input: list1 = ["Shogun","Tapioca Express","Burger King","KFC"], list2 = ["Piatti","The Grill at Torrey Pines","Hungry Hunter Steakhouse","Shogun"]
Output: ["Shogun"]
Explanation: The only restaurant they both like is "Shogun".

Example 2:

Input: list1 = ["Shogun","Tapioca Express","Burger King","KFC"], list2 = ["KFC","Shogun","Burger King"]
Output: ["Shogun"]
Explanation: The restaurant they both like and have the least index sum is "Shogun" with index sum 1 (0+1).

Example 3:

Input: list1 = ["Shogun","Tapioca Express","Burger King","KFC"], list2 = ["KFC","Burger King","Tapioca Express","Shogun"]
Output: ["KFC","Burger King","Tapioca Express","Shogun"]

Example 4:

Input: list1 = ["Shogun","Tapioca Express","Burger King","KFC"], list2 = ["KNN","KFC","Burger King","Tapioca Express","Shogun"]
Output: ["KFC","Burger King","Tapioca Express","Shogun"]

Example 5:

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

Constraints:

1 <= list1.length, list2.length <= 1000
1 <= list1[i].length, list2[i].length <= 30
list1[i] and list2[i] consist of spaces ' ' and English letters.
All the stings of list1 are unique.
All the stings of list2 are unique.

---

600. Non-negative Integers without Consecutive Ones

Given a positive integer n, return the number of the integers in the range [0, n] whose binary representations do not contain consecutive ones.

Example 1:

Input: n = 5
Output: 5
Explanation:
Here are the non-negative integers <= 5 with their corresponding binary representations:
0 : 0
1 : 1
2 : 10
3 : 11
4 : 100
5 : 101
Among them, only integer 3 disobeys the rule (two consecutive ones) and the other 5 satisfy the rule.

Example 2:

Input: n = 1
Output: 2

Example 3:

Input: n = 2
Output: 3

Constraints:

1 <= n <= 109

---

601. Human Traffic of Stadium

Table: Stadium

+---------------+---------+
| Column Name | Type |
+---------------+---------+
id	int
visit_date	date
people	int
+---------------+---------+
visit_date is the primary key for this table.
Each row of this table contains the visit date and visit id to the stadium with the number of people during the visit.
No two rows will have the same visit_date, and as the id increases, the dates increase as well.

Write an SQL query to display the records with three or more rows with consecutive id's, and the number of people i s greater than or equal to 100 for each.
Return the result table ordered by visit_date in ascending order.

The query result format is in the following example.

Example 1:

Input:
Stadium table:

id	visit_date	people
1	2017-01-01	10
2	2017-01-02	109
3	2017-01-03	150
4	2017-01-04	99
5	2017-01-05	145
6	2017-01-06	1455
7	2017-01-07	199
8	2017-01-09	188

Output:

id	visit_date	people
5	2017-01-05	145
6	2017-01-06	1455
7	2017-01-07	199
8	2017-01-09	188

Explanation:
The four rows with ids 5, 6, 7, and 8 have consecutive ids and each of them has >= 100 people attended. Note that ro w 8 was included even though the visit_date was not the next day after row 7.
The rows with ids 2 and 3 are not included because we need at least three consecutive ids.

---

605. Can Place Flowers

You have a long flowerbed in which some of the plots are planted, and some are not. However, flowers cannot be pl anted in adjacent plots.
Given an integer array flowerbed containing 0's and 1's, where 0 means empty and 1 means not empty, and an intege r n, return if n new flowers can be planted in the flowerbed without violating the no-adjacent-flowers rule.

Example 1:
Input: flowerbed = [1,0,0,0,1], n = 1
Output: true
Example 2:
Input: flowerbed = [1,0,0,0,1], n = 2
Output: false

Constraints:

1 <= flowerbed.length <= 2 * 104
flowerbed[i] is 0 or 1.
There are no two adjacent flowers in flowerbed.
0 <= n <= flowerbed.length

---

606. Construct String from Binary Tree

Given the root of a binary tree, construct a string consisting of parenthesis and integers from a binary tree with the pr eorder traversal way, and return it.
Omit all the empty parenthesis pairs that do not affect the one-to-one mapping relationship between the string and th e original binary tree.

Example 1:

Input: root = [1,2,3,4]
Output: "1(2(4))(3)"
Explanation: Originally, it needs to be "1(2(4())(3()())", but you need to omit all the unnecessary empty parenthesis pairs. And it will be "1(2(4))(3)"

Example 2:

Input: root = [1,2,3,null,4]
Output: "1(2()(4))(3)"
Explanation: Almost the same as the first example, except we cannot omit the first parenthesis pair to break the one-t o-one mapping relationship between the input and the output.

Constraints:

The number of nodes in the tree is in the range [1, 104].
-1000 <= Node.val <= 1000

---

609. Find Duplicate File in System

Given a list paths of directory info, including the directory path, and all the files with contents in this directory, retur n all the duplicate files in the file system in terms of their paths. You may return the answer in any order.
A group of duplicate files consists of at least two files that have the same content.
A single directory info string in the input list has the following format:

"root/d1/d2/.../dm f1.txt(f1_content) f2.txt(f2_content) ... fn.txt(fn_content)"

It means there are n files (f1.txt, f2.txt ... fn.txt) with content (f1_content, f2_content ... fn_content) respectively in th e directory "root/d1/d2/.../dm". Note that n >= 1 and m >= 0. If m = 0, it means the directory is just the root directory .
The output is a list of groups of duplicate file paths. For each group, it contains all the file paths of the files that have the same content. A file path is a string that has the following format:

"directory_path/file_name.txt"

Example 1:
Input: paths = ["root/a 1.txt(abcd) 2.txt(efgh)","root/c 3.txt(abcd)","root/c/d 4.txt(efgh)","root 4.txt(efgh)"]
Output: [["root/a/2.txt","root/c/d/4.txt","root/4.txt"],["root/a/1.txt","root/c/3.txt"]]
Example 2:
Input: paths = ["root/a 1.txt(abcd) 2.txt(efgh)","root/c 3.txt(abcd)","root/c/d 4.txt(efgh)"]
Output: [["root/a/2.txt","root/c/d/4.txt"],["root/a/1.txt","root/c/3.txt"]]

Constraints:

1 <= paths.length <= 2 * 104
1 <= paths[i].length <= 3000
1 <= sum(paths[i].length) <= 5 * 105
paths[i] consist of English letters, digits, '/', '.', '(', ')', and ' '.
You may assume no files or directories share the same name in the same directory.
You may assume each given directory info represents a unique directory. A single blank space separates the director y path and file info.

Follow up:

Imagine you are given a real file system, how will you search files? DFS or BFS?
If the file content is very large (GB level), how will you modify your solution?
If you can only read the file by 1kb each time, how will you modify your solution?
What is the time complexity of your modified solution? What is the most time-consuming part and memory-consumi ng part of it? How to optimize?
How to make sure the duplicated files you find are not false positive?

---

611. Valid Triangle Number

Given an integer array nums, return the number of triplets chosen from the array that can make triangles if we take th em as side lengths of a triangle.

Example 1:

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

Example 2:

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

Constraints:

1 <= nums.length <= 1000
0 <= nums[i] <= 1000

---

617. Merge Two Binary Trees

You are given two binary trees root1 and root2.
Imagine that when you put one of them to cover the other, some nodes of the two trees are overlapped while the othe rs are not. You need to merge the two trees into a new binary tree. The merge rule is that if two nodes overlap, then s um node values up as the new value of the merged node. Otherwise, the NOT null node will be used as the node of t he new tree.
Return the merged tree.
Note: The merging process must start from the root nodes of both trees.

Example 1:

Input: root1 = [1,3,2,5], root2 = [2,1,3,null,4,null,7]
Output: [3,4,5,5,4,null,7]

Example 2:

Input: root1 = [1], root2 = [1,2]
Output: [2,2]

Constraints:

The number of nodes in both trees is in the range [0, 2000].
-104 <= Node.val <= 104

---

620. Not Boring Movies

Table: Cinema

+----------------+----------+
| Column Name | Type |
+----------------+----------+
id	int
movie	varchar
description	varchar
rating	float
+----------------+----------+
id is the primary key for this table.
Each row contains information about the name of a movie, its genre, and its rating.
rating is a 2 decimal places float in the range [0, 10]

Write an SQL query to report the movies with an odd-numbered ID and a description that is not "boring".
Return the result table ordered by rating in descending order.
The query result format is in the following example.

Example 1:

Input:
Cinema table:
+----+------------+-------------+--------+
| id | movie | description | rating |
+----+------------+-------------+--------+
1	War	great 3D	8.9
2	Science	fiction	8.5
3	irish	boring	6.2
4	Ice song	Fantacy	8.6
5	House card	Interesting	9.1
+----+------------+-------------+--------+
Output:
+----+------------+-------------+--------+
| id | movie | description | rating |
+----+------------+-------------+--------+
| 5 | House card | Interesting | 9.1 |
| 1 | War | great 3D | 8.9 |
+----+------------+-------------+--------+

Explanation:
We have three movies with odd-numbered IDs: 1, 3, and 5. The movie with ID = 3 is boring so we do not include it i
n the answer.

---

621. Task Scheduler

Given a characters array tasks, representing the tasks a CPU needs to do, where each letter represents a different task
. Tasks could be done in any order. Each task is done in one unit of time. For each unit of time, the CPU could comp
lete either one task or just be idle.
However, there is a non-negative integer n that represents the cooldown period between two same tasks (the same let
ter in the array), that is that there must be at least n units of time between any two same tasks.
Return the least number of units of times that the CPU will take to finish all the given tasks.

Example 1:

Input: tasks = ["A","A","A","B","B","B"], n = 2
Output: 8
Explanation:
A -> B -> idle -> A -> B -> idle -> A -> B
There is at least 2 units of time between any two same tasks.

Example 2:

Input: tasks = ["A","A","A","B","B","B"], n = 0
Output: 6
Explanation: On this case any permutation of size 6 would work since n = 0.
["A","A","A","B","B","B"]
["A","B","A","B","A","B"]
["B","B","B","A","A","A"]
...
And so on.

Example 3:

Input: tasks = ["A","A","A","A","A","A","B","C","D","E","F","G"], n = 2
Output: 16
Explanation:
One possible solution is
A -> B -> C -> A -> D -> E -> A -> F -> G -> A -> idle -> idle -> A -> idle -> idle -> A

Constraints:

1 <= task.length <= 104
tasks[i] is upper-case English letter.
The integer n is in the range [0, 100].

---

622. Design Circular Queue

Design your implementation of the circular queue. The circular queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called "Ring Buffer".
One of the benefits of the circular queue is that we can make use of the spaces in front of the queue. In a normal queue, once the queue becomes full, we cannot insert the next element even if there is a space in front of the queue. But using the circular queue, we can use the space to store new values.
Implementation the MyCircularQueue class:

MyCircularQueue(k) Initializes the object with the size of the queue to be k.
int Front() Gets the front item from the queue. If the queue is empty, return -1.
int Rear() Gets the last item from the queue. If the queue is empty, return -1.
boolean enQueue(int value) Inserts an element into the circular queue. Return true if the operation is successful.
boolean deQueue() Deletes an element from the circular queue. Return true if the operation is successful.
boolean isEmpty() Checks whether the circular queue is empty or not.
boolean isFull() Checks whether the circular queue is full or not.

You must solve the problem without using the built-in queue data structure in your programming language.

Example 1:

Input
["MyCircularQueue", "enQueue", "enQueue", "enQueue", "enQueue", "Rear", "isFull", "deQueue", "enQueue", "Rear"]
[[3], [1], [2], [3], [4], [], [], [], [4], []]
Output
[null, true, true, true, false, 3, true, true, true, 4]

Explanation
MyCircularQueue myCircularQueue = new MyCircularQueue(3);
myCircularQueue.enQueue(1); // return True
myCircularQueue.enQueue(2); // return True
myCircularQueue.enQueue(3); // return True
myCircularQueue.enQueue(4); // return False
myCircularQueue.Rear();   // return 3
myCircularQueue.isFull();   // return True
myCircularQueue.deQueue(); // return True
myCircularQueue.enQueue(4); // return True
myCircularQueue.Rear();   // return 4

Constraints:

1 <= k <= 1000

0 <= value <= 1000
At most 3000 calls will be made to enQueue, deQueue, Front, Rear, isEmpty, and isFull.

---

623. Add One Row to Tree

Given the root of a binary tree and two integers val and depth, add a row of nodes with value val at the given depth d epth.
Note that the root node is at depth 1.
The adding rule is:

Given the integer depth, for each not null tree node cur at the depth depth - 1, create two tree nodes with value val as cur's left subtree root and right subtree root.
cur's original left subtree should be the left subtree of the new left subtree root.
cur's original right subtree should be the right subtree of the new right subtree root.
If depth == 1 that means there is no depth depth - 1 at all, then create a tree node with value val as the new root of th e whole original tree, and the original tree is the new root's left subtree.

Example 1:

Input: root = [4,2,6,3,1,5], val = 1, depth = 2
Output: [4,1,1,2,null,null,6,3,1,5]

Example 2:

Input: root = [4,2,null,3,1], val = 1, depth = 3
Output: [4,2,null,1,1,3,null,null,1]

Constraints:

The number of nodes in the tree is in the range [1, 104].
The depth of the tree is in the range [1, 104].
-100 <= Node.val <= 100
-105 <= val <= 105
1 <= depth <= the depth of tree + 1

---

626. Exchange Seats

Table: Seat

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| name        | varchar |
+-------------+---------+
```

id is the primary key column for this table.
Each row of this table indicates the name and the ID of a student.
id is a continuous increment.

Write an SQL query to swap the seat id of every two consecutive students. If the number of students is odd, the id of the last student is not swapped.
Return the result table ordered by id in ascending order.
The query result format is in the following example.

Example 1:

Input:
Seat table:

```
+----+---------+
| id | student |
+----+---------+
| 1  | Abbot   |
| 2  | Doris   |
| 3  | Emerson |
| 4  | Green   |
| 5  | Jeames  |
+----+---------+
```

Output:

```
+----+---------+
| id | student |
+----+---------+
| 1  | Doris   |
| 2  | Abbot   |
| 3  | Green   |
| 4  | Emerson |
| 5  | Jeames  |
+----+---------+
```

Explanation:
Note that if the number of students is odd, there is no need to change the last one's seat.

---

627. Swap Salary

Table: Salary

```
+-------------+----------+
| Column Name | Type     |
+-------------+----------+
| id          | int      |
| name        | varchar  |
| sex         | ENUM     |
| salary      | int      |
+-------------+----------+
```

id is the primary key for this table.
The sex column is ENUM value of type ('m', 'f').
The table contains information about an employee.

Write an SQL query to swap all 'f' and 'm' values (i.e., change all 'f' values to 'm' and vice versa) with a single update statement and no intermediate temporary tables.
Note that you must write a single update statement, do not write any select statement for this problem.
The query result format is in the following example.

Example 1:

Input:
Salary table:

```
+----+------+-----+--------+
| id | name | sex | salary |
+----+------+-----+--------+
| 1  | A    | m   | 2500   |
| 2  | B    | f   | 1500   |
| 3  | C    | m   | 5500   |
| 4  | D    | f   | 500    |
+----+------+-----+--------+
```

Output:

```
+----+------+-----+--------+
| id | name | sex | salary |
+----+------+-----+--------+
| 1  | A    | f   | 2500   |
| 2  | B    | m   | 1500   |
| 3  | C    | f   | 5500   |
| 4  | D    | m   | 500    |
+----+------+-----+--------+
```

Explanation:
(1, A) and (3, C) were changed from 'm' to 'f'.
(2, B) and (4, D) were changed from 'f' to 'm'.

---

628. Maximum Product of Three Numbers

Given an integer array nums, find three numbers whose product is maximum and return the maximum product.

Example 1:
Input: nums = [1,2,3]
Output: 6
Example 2:
Input: nums = [1,2,3,4]
Output: 24
Example 3:
Input: nums = [-1,-2,-3]
Output: -6

Constraints:

3 <= nums.length <= 104
-1000 <= nums[i] <= 1000

---

629. K Inverse Pairs Array

For an integer array nums, an inverse pair is a pair of integers [i, j] where 0 <= i < j < nums.length and nums[i] > nu ms[j].
Given two integers n and k, return the number of different arrays consist of numbers from 1 to n such that there are e xactly 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

---

630. Course Schedule III

There are n different online courses numbered from 1 to n. You are given an array courses where courses[i] = [durati oni, lastDayi] indicate that the ith course should be taken continuously for durationi days and must be finished befor e or on lastDayi.
You will start on the 1st day and you cannot take two or more courses simultaneously.
Return the maximum number of courses that you can take.

Example 1:

Input: courses = [[100,200],[200,1300],[1000,1250],[2000,3200]]
Output: 3
Explanation:
There are totally 4 courses, but you can take 3 courses at most:
First, take the 1st course, it costs 100 days so you will finish it on the 100th day, and ready to take the next course on the 101st day.
Second, take the 3rd course, it costs 1000 days so you will finish it on the 1100th day, and ready to take the next cou rse on the 1101st day.
Third, take the 2nd course, it costs 200 days so you will finish it on the 1300th day.
The 4th course cannot be taken now, since you will finish it on the 3300th day, which exceeds the closed date.

Example 2:

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

Example 3:

Input: courses = [[3,2],[4,3]]
Output: 0

Constraints:

1 <= courses.length <= 104
1 <= durationi, lastDayi <= 104

---

632. Smallest Range Covering Elements from K Lists

You have k lists of sorted integers in non-decreasing order. Find the smallest range that includes at least one number from each of the k lists.
We define the range [a, b] is smaller than range [c, d] if b - a < d - c or a < c if b - a == d - c.

Example 1:

Input: nums = [[4,10,15,24,26],[0,9,12,20],[5,18,22,30]]
Output: [20,24]
Explanation:
List 1: [4, 10, 15, 24,26], 24 is in range [20,24].
List 2: [0, 9, 12, 20], 20 is in range [20,24].
List 3: [5, 18, 22, 30], 22 is in range [20,24].

Example 2:

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

Example 3:

Input: nums = [[10,10],[11,11]]
Output: [10,11]

Example 4:

Input: nums = [[10],[11]]
Output: [10,11]

Example 5:

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

Constraints:

nums.length == k
1 <= k <= 3500
1 <= nums[i].length <= 50
-105 <= nums[i][j] <= 105
nums[i] is sorted in non-decreasing order.

---

633. Sum of Square Numbers

Given a non-negative integer c, decide whether there're two integers a and b such that a2 + b2 = c.

Example 1:

Input: c = 5
Output: true
Explanation: 1 * 1 + 2 * 2 = 5

Example 2:

Input: c = 3
Output: false

Example 3:

Input: c = 4
Output: true

Example 4:

Input: c = 2
Output: true

Example 5:

Input: c = 1
Output: true

Constraints:

0 <= c <= 231 - 1

---

636. Exclusive Time of Functions

On a single-threaded CPU, we execute a program containing n functions. Each function has a unique ID between 0 a nd n-1.
Function calls are stored in a call stack: when a function call starts, its ID is pushed onto the stack, and when a functi on call ends, its ID is popped off the stack. The function whose ID is at the top of the stack is the current function bei ng executed. Each time a function starts or ends, we write a log with the ID, whether it started or ended, and the time stamp.
You are given a list logs, where logs[i] represents the ith log message formatted as a string "{function_id}:{"start" | " end"}:{timestamp}". For example, "0:start:3" means a function call with function ID 0 started at the beginning of ti mestamp 3, and "1:end:2" means a function call with function ID 1 ended at the end of timestamp 2. Note that a func tion can be called multiple times, possibly recursively.
A function's exclusive time is the sum of execution times for all function calls in the program. For example, if a func

tion is called twice, one call executing for 2 time units and another call executing for 1 time unit, the exclusive time i s 2 + 1 = 3.
Return the exclusive time of each function in an array, where the value at the ith index represents the exclusive time for the function with ID i.

Example 1:

Input: n = 2, logs = ["0:start:0","1:start:2","1:end:5","0:end:6"]
Output: [3,4]
Explanation:
Function 0 starts at the beginning of time 0, then it executes 2 for units of time and reaches the end of time 1.
Function 1 starts at the beginning of time 2, executes for 4 units of time, and ends at the end of time 5.
Function 0 resumes execution at the beginning of time 6 and executes for 1 unit of time.
So function 0 spends 2 + 1 = 3 units of total time executing, and function 1 spends 4 units of total time executing.

Example 2:

Input: n = 1, logs = ["0:start:0","0:start:2","0:end:5","0:start:6","0:end:6","0:end:7"]
Output: [8]
Explanation:
Function 0 starts at the beginning of time 0, executes for 2 units of time, and recursively calls itself.
Function 0 (recursive call) starts at the beginning of time 2 and executes for 4 units of time.
Function 0 (initial call) resumes execution then immediately calls itself again.
Function 0 (2nd recursive call) starts at the beginning of time 6 and executes for 1 unit of time.
Function 0 (initial call) resumes execution at the beginning of time 7 and executes for 1 unit of time.
So function 0 spends 2 + 4 + 1 + 1 = 8 units of total time executing.

Example 3:

Input: n = 2, logs = ["0:start:0","0:start:2","0:end:5","1:start:6","1:end:6","0:end:7"]
Output: [7,1]
Explanation:
Function 0 starts at the beginning of time 0, executes for 2 units of time, and recursively calls itself.
Function 0 (recursive call) starts at the beginning of time 2 and executes for 4 units of time.
Function 0 (initial call) resumes execution then immediately calls function 1.
Function 1 starts at the beginning of time 6, executes 1 units of time, and ends at the end of time 6.
Function 0 resumes execution at the beginning of time 6 and executes for 2 units of time.
So function 0 spends 2 + 4 + 1 = 7 units of total time executing, and function 1 spends 1 unit of total time executing.

Example 4:

Input: n = 2, logs = ["0:start:0","0:start:2","0:end:5","1:start:7","1:end:7","0:end:8"]
Output: [8,1]

Example 5:

Input: n = 1, logs = ["0:start:0","0:end:0"]
Output: [1]

Constraints:

1 <= n <= 100

1 <= logs.length <= 500
0 <= function_id < n
0 <= timestamp <= 109
No two start events will happen at the same timestamp.
No two end events will happen at the same timestamp.
Each function has an "end" log for each "start" log.

---

637. Average of Levels in Binary Tree

Given the root of a binary tree, return the average value of the nodes on each level in the form of an array. Answers within 10-5 of the actual answer will be accepted.

Example 1:

Input: root = [3,9,20,null,null,15,7]
Output: [3.00000,14.50000,11.00000]
Explanation: The average value of nodes on level 0 is 3, on level 1 is 14.5, and on level 2 is 11.
Hence return [3, 14.5, 11].

Example 2:

Input: root = [3,9,20,15,7]
Output: [3.00000,14.50000,11.00000]

Constraints:

The number of nodes in the tree is in the range [1, 104].
-231 <= Node.val <= 231 - 1

---

638. Shopping Offers

In LeetCode Store, there are n items to sell. Each item has a price. However, there are some special offers, and a spe cial offer consists of one or more different kinds of items with a sale price.
You are given an integer array price where price[i] is the price of the ith item, and an integer array needs where need s[i] is the number of pieces of the ith item you want to buy.

You are also given an array special where special[i] is of size n + 1 where special[i][j] is the number of pieces of the jth item in the ith offer and special[i][n] (i.e., the last integer in the array) is the price of the ith offer.
Return the lowest price you have to pay for exactly certain items as given, where you could make optimal use of the special offers. You are not allowed to buy more items than you want, even if that would lower the overall price. You could use any of the special offers as many times as you want.

Example 1:

Input: price = [2,5], special = [[3,0,5],[1,2,10]], needs = [3,2]
Output: 14
Explanation: There are two kinds of items, A and B. Their prices are $2 and $5 respectively.
In special offer 1, you can pay $5 for 3A and 0B
In special offer 2, you can pay $10 for 1A and 2B.
You need to buy 3A and 2B, so you may pay $10 for 1A and 2B (special offer #2), and $4 for 2A.

Example 2:

Input: price = [2,3,4], special = [[1,1,0,4],[2,2,1,9]], needs = [1,2,1]
Output: 11
Explanation: The price of A is $2, and $3 for B, $4 for C.
You may pay $4 for 1A and 1B, and $9 for 2A ,2B and 1C.
You need to buy 1A ,2B and 1C, so you may pay $4 for 1A and 1B (special offer #1), and $3 for 1B, $4 for 1C.
You cannot add more items, though only $9 for 2A ,2B and 1C.

Constraints:

n == price.length
n == needs.length
1 <= n <= 6
0 <= price[i] <= 10
0 <= needs[i] <= 10
1 <= special.length <= 100
special[i].length == n + 1
0 <= special[i][j] <= 50

---

639. Decode Ways II

A message containing letters from A-Z can be encoded into numbers using the following mapping:

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

To decode an encoded message, all the digits must be grouped then mapped back into letters using the reverse of the

mapping above (there may be multiple ways). For example, "11106" can be mapped into:

"AAJF" with the grouping (1 1 10 6)
"KJF" with the grouping (11 10 6)

Note that the grouping (1 11 06) is invalid because "06" cannot be mapped into 'F' since "6" is different from "06".
In addition to the mapping above, an encoded message may contain the '*' character, which can represent any digit fr om '1' to '9' ('0' is excluded). For example, the encoded message "1*" may represent any of the encoded messages "1 1", "12", "13", "14", "15", "16", "17", "18", or "19". Decoding "1*" is equivalent to decoding any of the encoded me ssages it can represent.
Given a string s consisting of digits and '*' characters, return the number of ways to decode it.
Since the answer may be very large, return it modulo 109 + 7.

Example 1:

Input: s = "*"
Output: 9
Explanation: The encoded message can represent any of the encoded messages "1", "2", "3", "4", "5", "6", "7", "8", o r "9".
Each of these can be decoded to the strings "A", "B", "C", "D", "E", "F", "G", "H", and "I" respectively.
Hence, there are a total of 9 ways to decode "*".

Example 2:

Input: s = "1*"
Output: 18
Explanation: The encoded message can represent any of the encoded messages "11", "12", "13", "14", "15", "16", "1 7", "18", or "19".
Each of these encoded messages have 2 ways to be decoded (e.g. "11" can be decoded to "AA" or "K").
Hence, there are a total of 9 * 2 = 18 ways to decode "1*".

Example 3:

Input: s = "2*"
Output: 15
Explanation: The encoded message can represent any of the encoded messages "21", "22", "23", "24", "25", "26", "2 7", "28", or "29".
"21", "22", "23", "24", "25", and "26" have 2 ways of being decoded, but "27", "28", and "29" only have 1 way.
Hence, there are a total of (6 * 2) + (3 * 1) = 12 + 3 = 15 ways to decode "2*".

Constraints:

1 <= s.length <= 105
s[i] is a digit or '*'.

---

640. Solve the Equation

Solve a given equation and return the value of 'x' in the form of a string "x=#value". The equation contains only '+', '-' operation, the variable 'x' and its coefficient. You should return "No solution" if there is no solution for the equation, or "Infinite solutions" if there are infinite solutions for the equation.
If there is exactly one solution for the equation, we ensure that the value of 'x' is an integer.

Example 1:
Input: equation = "x+5-3+x=6+x-2"
Output: "x=2"
Example 2:
Input: equation = "x=x"
Output: "Infinite solutions"
Example 3:
Input: equation = "2x=x"
Output: "x=0"
Example 4:
Input: equation = "2x+3x-6x=x+2"
Output: "x=-1"
Example 5:
Input: equation = "x=x+2"
Output: "No solution"

Constraints:

3 <= equation.length <= 1000
equation has exactly one '='.
equation consists of integers with an absolute value in the range [0, 100] without any leading zeros, and the variable 'x'.

---

641. Design Circular Deque

Design your implementation of the circular double-ended queue (deque).
Implement the MyCircularDeque class:

MyCircularDeque(int k) Initializes the deque with a maximum size of k.
boolean insertFront() Adds an item at the front of Deque. Returns true if the operation is successful, or false otherwise.
boolean insertLast() Adds an item at the rear of Deque. Returns true if the operation is successful, or false otherwise.
boolean deleteFront() Deletes an item from the front of Deque. Returns true if the operation is successful, or false otherwise.
boolean deleteLast() Deletes an item from the rear of Deque. Returns true if the operation is successful, or false otherwise.
int getFront() Returns the front item from the Deque. Returns -1 if the deque is empty.
int getRear() Returns the last item from Deque. Returns -1 if the deque is empty.

boolean isEmpty() Returns true if the deque is empty, or false otherwise.
boolean isFull() Returns true if the deque is full, or false otherwise.

Example 1:

Input
["MyCircularDeque", "insertLast", "insertLast", "insertFront", "insertFront", "getRear", "isFull", "deleteLast", "inser
tFront", "getFront"]
[[3], [1], [2], [3], [4], [], [], [], [4], []]
Output
[null, true, true, true, false, 2, true, true, true, 4]

Explanation
MyCircularDeque myCircularDeque = new MyCircularDeque(3);
myCircularDeque.insertLast(1); // return True
myCircularDeque.insertLast(2); // return True
myCircularDeque.insertFront(3); // return True
myCircularDeque.insertFront(4); // return False, the queue is full.
myCircularDeque.getRear();   // return 2
myCircularDeque.isFull();   // return True
myCircularDeque.deleteLast();   // return True
myCircularDeque.insertFront(4); // return True
myCircularDeque.getFront();   // return 4

Constraints:

1 <= k <= 1000
0 <= value <= 1000
At most 2000 calls will be made to insertFront, insertLast, deleteFront, deleteLast, getFront, getRear, isEmpty, isFull
.

---

643. Maximum Average Subarray I

You are given an integer array nums consisting of n elements, and an integer k.
Find a contiguous subarray whose length is equal to k that has the maximum average value and return this value. An
y answer with a calculation error less than 10-5 will be accepted.

Example 1:

Input: nums = [1,12,-5,-6,50,3], k = 4
Output: 12.75000
Explanation: Maximum average is (12 - 5 - 6 + 50) / 4 = 51 / 4 = 12.75

Example 2:

Input: nums = [5], k = 1
Output: 5.00000

Constraints:

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

---

645. Set Mismatch

You have a set of integers s, which originally contains all the numbers from 1 to n. Unfortunately, due to some error, one of the numbers in s got duplicated to another number in the set, which results in repetition of one number and loss of another number.
You are given an integer array nums representing the data status of this set after the error.
Find the number that occurs twice and the number that is missing and return them in the form of an array.

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

Constraints:

2 <= nums.length <= 104
1 <= nums[i] <= 104

---

646. Maximum Length of Pair Chain

You are given an array of n pairs pairs where pairs[i] = [lefti, righti] and lefti < righti.
A pair p2 = [c, d] follows a pair p1 = [a, b] if b < c. A chain of pairs can be formed in this fashion.
Return the length longest chain which can be formed.
You do not need to use up all the given intervals. You can select pairs in any order.

Example 1:

Input: pairs = [[1,2],[2,3],[3,4]]
Output: 2
Explanation: The longest chain is [1,2] -> [3,4].

Example 2:

Input: pairs = [[1,2],[7,8],[4,5]]
Output: 3
Explanation: The longest chain is [1,2] -> [4,5] -> [7,8].

Constraints:

n == pairs.length
1 <= n <= 1000
-1000 <= lefti < righti <= 1000

---

647. Palindromic Substrings

Given a string s, return the number of palindromic substrings in it.
A string is a palindrome when it reads the same backward as forward.
A substring is a contiguous sequence of characters within the string.

Example 1:

Input: s = "abc"
Output: 3
Explanation: Three palindromic strings: "a", "b", "c".

Example 2:

Input: s = "aaa"
Output: 6
Explanation: Six palindromic strings: "a", "a", "a", "aa", "aa", "aaa".

Constraints:

1 <= s.length <= 1000
s consists of lowercase English letters.

---

648. Replace Words

In English, we have a concept called root, which can be followed by some other word to form another longer word - let's call this word successor. For example, when the root "an" is followed by the successor word "other", we can for m a new word "another".
Given a dictionary consisting of many roots and a sentence consisting of words separated by spaces, replace all the s uccessors in the sentence with the root forming it. If a successor can be replaced by more than one root, replace it wit h the root that has the shortest length.
Return the sentence after the replacement.

Example 1:
Input: dictionary = ["cat","bat","rat"], sentence = "the cattle was rattled by the battery"
Output: "the cat was rat by the bat"
Example 2:
Input: dictionary = ["a","b","c"], sentence = "aadsfasf absbs bbab cadsfafs"
Output: "a a b c"
Example 3:
Input: dictionary = ["a", "aa", "aaa", "aaaa"], sentence = "a aa a aaaa aaa aaa aaa aaaaaa bbb baba ababa"
Output: "a a a a a a a a bbb baba a"
Example 4:
Input: dictionary = ["catt","cat","bat","rat"], sentence = "the cattle was rattled by the battery"
Output: "the cat was rat by the bat"
Example 5:
Input: dictionary = ["ac","ab"], sentence = "it is abnormal that this solution is accepted"
Output: "it is ab that this solution is ac"

Constraints:

1 <= dictionary.length <= 1000
1 <= dictionary[i].length <= 100
dictionary[i] consists of only lower-case letters.
1 <= sentence.length <= 10^6
sentence consists of only lower-case letters and spaces.
The number of words in sentence is in the range [1, 1000]
The length of each word in sentence is in the range [1, 1000]
Each two consecutive words in sentence will be separated by exactly one space.
sentence does not have leading or trailing spaces.

---

649. Dota2 Senate

In the world of Dota2, there are two parties: the Radiant and the Dire.
The Dota2 senate consists of senators coming from two parties. Now the Senate wants to decide on a change in the Dota2 game. The voting for this change is a round-based procedure. In each round, each senator can exercise one of the two rights:

Ban one senator's right: A senator can make another senator lose all his rights in this and all the following rounds.
Announce the victory: If this senator found the senators who still have rights to vote are all from the same party, he c an announce the victory and decide on the change in the game.

Given a string senate representing each senator's party belonging. The character 'R' and 'D' represent the Radiant part y and the Dire party. Then if there are n senators, the size of the given string will be n.
The round-based procedure starts from the first senator to the last senator in the given order. This procedure will last until the end of voting. All the senators who have lost their rights will be skipped during the procedure.
Suppose every senator is smart enough and will play the best strategy for his own party. Predict which party will fina lly announce the victory and change the Dota2 game. The output should be "Radiant" or "Dire".

Example 1:

Input: senate = "RD"
Output: "Radiant"
Explanation:
The first senator comes from Radiant and he can just ban the next senator's right in round 1.
And the second senator can't exercise any rights anymore since his right has been banned.
And in round 2, the first senator can just announce the victory since he is the only guy in the senate who can vote.

Example 2:

Input: senate = "RDD"
Output: "Dire"
Explanation:
The first senator comes from Radiant and he can just ban the next senator's right in round 1.
And the second senator can't exercise any rights anymore since his right has been banned.
And the third senator comes from Dire and he can ban the first senator's right in round 1.
And in round 2, the third senator can just announce the victory since he is the only guy in the senate who can vote.

Constraints:

n == senate.length
1 <= n <= 104
senate[i] is either 'R' or 'D'.

---

650. 2 Keys Keyboard

There is only one character 'A' on the screen of a notepad. You can perform two operations on this notepad for each

step:

Copy All: You can copy all the characters present on the screen (a partial copy is not allowed).
Paste: You can paste the characters which are copied last time.

Given an integer n, return the minimum number of operations to get the character 'A' exactly n times on the screen.

Example 1:

Input: n = 3
Output: 3
Explanation: Intitally, we have one character 'A'.
In step 1, we use Copy All operation.
In step 2, we use Paste operation to get 'AA'.
In step 3, we use Paste operation to get 'AAA'.

Example 2:

Input: n = 1
Output: 0

Constraints:

1 <= n <= 1000

---

652. Find Duplicate Subtrees

Given the root of a binary tree, return all duplicate subtrees.
For each kind of duplicate subtrees, you only need to return the root node of any one of them.
Two trees are duplicate if they have the same structure with the same node values.

Example 1:

Input: root = [1,2,3,4,null,2,4,null,null,4]
Output: [[2,4],[4]]

Example 2:

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

Example 3:

Input: root = [2,2,2,3,null,3,null]
Output: [[2,3],[3]]

Constraints:

The number of the nodes in the tree will be in the range [1, 10^4]
-200 <= Node.val <= 200

---

653. Two Sum IV - Input is a BST

Given the root of a Binary Search Tree and a target number k, return true if there exist two elements in the BST such that their sum is equal to the given target.

Example 1:

Input: root = [5,3,6,2,4,null,7], k = 9
Output: true

Example 2:

Input: root = [5,3,6,2,4,null,7], k = 28
Output: false

Example 3:

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

Example 4:

Input: root = [2,1,3], k = 1
Output: false

Example 5:

Input: root = [2,1,3], k = 3
Output: true

Constraints:

The number of nodes in the tree is in the range [1, 104].

-104 <= Node.val <= 104
root is guaranteed to be a valid binary search tree.
-105 <= k <= 105

---

654. Maximum Binary Tree

You are given an integer array nums with no duplicates. A maximum binary tree can be built recursively from nums using the following algorithm:

Create a root node whose value is the maximum value in nums.
Recursively build the left subtree on the subarray prefix to the left of the maximum value.
Recursively build the right subtree on the subarray suffix to the right of the maximum value.

Return the maximum binary tree built from nums.

Example 1:

Input: nums = [3,2,1,6,0,5]
Output: [6,3,5,null,2,0,null,null,1]
Explanation: The recursive calls are as follow:
- The largest value in [3,2,1,6,0,5] is 6. Left prefix is [3,2,1] and right suffix is [0,5].
  - The largest value in [3,2,1] is 3. Left prefix is [] and right suffix is [2,1].
    - Empty array, so no child.
    - The largest value in [2,1] is 2. Left prefix is [] and right suffix is [1].
      - Empty array, so no child.
      - Only one element, so child is a node with value 1.
  - The largest value in [0,5] is 5. Left prefix is [0] and right suffix is [].
    - Only one element, so child is a node with value 0.
    - Empty array, so no child.

Example 2:

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

Constraints:

1 <= nums.length <= 1000
0 <= nums[i] <= 1000
All integers in nums are unique.

---

655. Print Binary Tree

Given the root of a binary tree, construct a 0-indexed m x n string matrix res that represents a formatted layout of the tree. The formatted layout matrix should be constructed using the following rules:

The height of the tree is height and the number of rows m should be equal to height + 1.
The number of columns n should be equal to 2height+1 - 1.
Place the root node in the middle of the top row (more formally, at location res[0][(n-1)/2]).
For each node that has been placed in the matrix at position res[r][c], place its left child at res[r+1][c-2height-r-1] and its right child at res[r+1][c+2height-r-1].
Continue this process until all the nodes in the tree have been placed.
Any empty cells should contain the empty string "".

Return the constructed matrix res.

Example 1:

Input: root = [1,2]
Output:
[["","1",""],
 ["2","",""]]

Example 2:

Input: root = [1,2,3,null,4]
Output:
[["","","","1","","","",""],
 ["","2","","","","","3",""],
 ["","","4","","","","",""]]

Constraints:

The number of nodes in the tree is in the range [1, 210].
-99 <= Node.val <= 99
The depth of the tree will be in the range [1, 10].

---

657. Robot Return to Origin

There is a robot starting at the position (0, 0), the origin, on a 2D plane. Given a sequence of its moves, judge if this robot ends up at (0, 0) after it completes its moves.
You are given a string moves that represents the move sequence of the robot where moves[i] represents its ith move. Valid moves are 'R' (right), 'L' (left), 'U' (up), and 'D' (down).
Return true if the robot returns to the origin after it finishes all of its moves, or false otherwise.
Note: The way that the robot is "facing" is irrelevant. 'R' will always make the robot move to the right once, 'L' will a lways make it move left, etc. Also, assume that the magnitude of the robot's movement is the same for each move.

Example 1:

Input: moves = "UD"
Output: true
Explanation: The robot moves up once, and then down once. All moves have the same magnitude, so it ended up at t he origin where it started. Therefore, we return true.

Example 2:

Input: moves = "LL"
Output: false
Explanation: The robot moves left twice. It ends up two "moves" to the left of the origin. We return false because it i s not at the origin at the end of its moves.

Example 3:

Input: moves = "RRDD"
Output: false

Example 4:

Input: moves = "LDRRLRUULR"
Output: false

Constraints:

1 <= moves.length <= 2 * 104
moves only contains the characters 'U', 'D', 'L' and 'R'.

---

658. Find K Closest Elements

Given a sorted integer array arr, two integers k and x, return the k closest integers to x in the array. The result should also be sorted in ascending order.
An integer a is closer to x than an integer b if:

|a - x| < |b - x|, or

|a - x| == |b - x| and a < b

Example 1:
Input: arr = [1,2,3,4,5], k = 4, x = 3
Output: [1,2,3,4]
Example 2:
Input: arr = [1,2,3,4,5], k = 4, x = -1
Output: [1,2,3,4]

Constraints:

1 <= k <= arr.length
1 <= arr.length <= 104
arr is sorted in ascending order.
-104 <= arr[i], x <= 104

---

659. Split Array into Consecutive Subsequences

You are given an integer array nums that is sorted in non-decreasing order.
Determine if it is possible to split nums into one or more subsequences such that both of the following conditions are true:

Each subsequence is a consecutive increasing sequence (i.e. each integer is exactly one more than the previous integ er).
All subsequences have a length of 3 or more.

Return true if you can split nums according to the above conditions, or false otherwise.
A subsequence of an array is a new array that is formed from the original array by deleting some (can be none) of th e elements without disturbing the relative positions of the remaining elements. (i.e., [1,3,5] is a subsequence of [1,2, 3,4,5] while [1,3,2] is not).

Example 1:

Input: nums = [1,2,3,3,4,5]
Output: true
Explanation: nums can be split into the following subsequences:
[1,2,3,3,4,5] --> 1, 2, 3
[1,2,3,3,4,5] --> 3, 4, 5

Example 2:

Input: nums = [1,2,3,3,4,4,5,5]
Output: true
Explanation: nums can be split into the following subsequences:

[1,2,3,3,4,4,5,5] --> 1, 2, 3, 4, 5
[1,2,3,3,4,4,5,5] --> 3, 4, 5

Example 3:

Input: nums = [1,2,3,4,4,5]
Output: false
Explanation: It is impossible to split nums into consecutive increasing subsequences of length 3 or more.

Constraints:

1 <= nums.length <= 104
-1000 <= nums[i] <= 1000
nums is sorted in non-decreasing order.

---

661. Image Smoother

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 aver age 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 cell s in the red smoother).

Given an m x n integer matrix img representing the grayscale of an image, return the image after applying the smoot her 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

---

662. Maximum Width of Binary Tree

Given the root of a binary tree, return the maximum width of the given tree.
The maximum width of a tree is the maximum width among all levels.
The width of one level is defined as the length between the end-nodes (the leftmost and rightmost non-null nodes), w
here the null nodes between the end-nodes are also counted into the length calculation.
It is guaranteed that the answer will in the range of 32-bit signed integer.

Example 1:

Input: root = [1,3,2,5,3,null,9]
Output: 4
Explanation: The maximum width existing in the third level with the length 4 (5,3,null,9).

Example 2:

Input: root = [1,3,null,5,3]
Output: 2
Explanation: The maximum width existing in the third level with the length 2 (5,3).

Example 3:

Input: root = [1,3,2,5]
Output: 2
Explanation: The maximum width existing in the second level with the length 2 (3,2).

Example 4:

Input: root = [1,3,2,5,null,null,9,6,null,null,7]
Output: 8
Explanation: The maximum width existing in the fourth level with the length 8 (6,null,null,null,null,null,null,7).

Constraints:

The number of nodes in the tree is in the range [1, 3000].
-100 <= Node.val <= 100

---

664. Strange Printer

There is a strange printer with the following two special properties:

The printer can only print a sequence of the same character each time.
At each turn, the printer can print new characters starting from and ending at any place and will cover the original ex isting characters.

Given a string s, return the minimum number of turns the printer needed to print it.

Example 1:

Input: s = "aaabbb"
Output: 2
Explanation: Print "aaa" first and then print "bbb".

Example 2:

Input: s = "aba"
Output: 2
Explanation: Print "aaa" first and then print "b" from the second place of the string, which will cover the existing cha racter 'a'.

Constraints:

1 <= s.length <= 100
s consists of lowercase English letters.

---

665. Non-decreasing Array

Given an array nums with n integers, your task is to check if it could become non-decreasing by modifying at most o ne element.
We define an array is non-decreasing if nums[i] <= nums[i + 1] holds for every i (0-based) such that (0 <= i <= n - 2

).

Example 1:

Input: nums = [4,2,3]
Output: true
Explanation: You could modify the first 4 to 1 to get a non-decreasing array.

Example 2:

Input: nums = [4,2,1]
Output: false
Explanation: You can't get a non-decreasing array by modify at most one element.

Constraints:

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

---

667. Beautiful Arrangement II

Given two integers n and k, construct a list answer that contains n different positive integers ranging from 1 to n and obeys the following requirement:

Suppose this list is answer = [a1, a2, a3, ... , an], then the list [|a1 - a2|, |a2 - a3|, |a3 - a4|, ... , |an-1 - an|] has exactly k distinct integers.

Return the list answer. If there multiple valid answers, return any of them.

Example 1:

Input: n = 3, k = 1
Output: [1,2,3]
Explanation: The [1,2,3] has three different positive integers ranging from 1 to 3, and the [1,1] has exactly 1 distinct integer: 1

Example 2:

Input: n = 3, k = 2
Output: [1,3,2]
Explanation: The [1,3,2] has three different positive integers ranging from 1 to 3, and the [2,1] has exactly 2 distinct integers: 1 and 2.

Constraints:

1 <= k < n <= 104

---

668. Kth Smallest Number in Multiplication Table

Nearly everyone has used the Multiplication Table. The multiplication table of size m x n is an integer matrix mat w here mat[i][j] == i * j (1-indexed).
Given three integers m, n, and k, return the kth smallest element in the m x n multiplication table.

Example 1:

Input: m = 3, n = 3, k = 5
Output: 3
Explanation: The 5th smallest number is 3.

Example 2:

Input: m = 2, n = 3, k = 6
Output: 6
Explanation: The 6th smallest number is 6.

Constraints:

1 <= m, n <= 3 * 104
1 <= k <= m * n

---

669. Trim a Binary Search Tree

Given the root of a binary search tree and the lowest and highest boundaries as low and high, trim the tree so that all its elements lies in [low, high]. Trimming the tree should not change the relative structure of the elements that will re main in the tree (i.e., any node's descendant should remain a descendant). It can be proven that there is a unique ans wer.
Return the root of the trimmed binary search tree. Note that the root may change depending on the given bounds.

Example 1:

Input: root = [1,0,2], low = 1, high = 2
Output: [1,null,2]

Example 2:

Input: root = [3,0,4,null,2,null,null,1], low = 1, high = 3
Output: [3,2,null,1]

Example 3:

Input: root = [1], low = 1, high = 2
Output: [1]

Example 4:

Input: root = [1,null,2], low = 1, high = 3
Output: [1,null,2]

Example 5:

Input: root = [1,null,2], low = 2, high = 4
Output: [2]

Constraints:

The number of nodes in the tree in the range [1, 104].
0 <= Node.val <= 104
The value of each node in the tree is unique.
root is guaranteed to be a valid binary search tree.
0 <= low <= high <= 104

---

670. Maximum Swap

You are given an integer num. You can swap two digits at most once to get the maximum valued number.
Return the maximum valued number you can get.

Example 1:

Input: num = 2736
Output: 7236
Explanation: Swap the number 2 and the number 7.

Example 2:

Input: num = 9973
Output: 9973
Explanation: No swap.

Constraints:

0 <= num <= 108

---

671. Second Minimum Node In a Binary Tree

Given a non-empty special binary tree consisting of nodes with the non-negative value, where each node in this tree has exactly two or zero sub-node. If the node has two sub-nodes, then this node's value is the smaller value among it s two sub-nodes. More formally, the property root.val = min(root.left.val, root.right.val) always holds.
Given such a binary tree, you need to output the second minimum value in the set made of all the nodes' value in the whole tree.
If no such second minimum value exists, output -1 instead.

Example 1:

Input: root = [2,2,5,null,null,5,7]
Output: 5
Explanation: The smallest value is 2, the second smallest value is 5.

Example 2:

Input: root = [2,2,2]
Output: -1
Explanation: The smallest value is 2, but there isn't any second smallest value.

Constraints:

The number of nodes in the tree is in the range [1, 25].
1 <= Node.val <= 231 - 1
root.val == min(root.left.val, root.right.val) for each internal node of the tree.

---

672. Bulb Switcher II

There is a room with n bulbs labeled from 1 to n that all are turned on initially, and four buttons on the wall. Each of the four buttons has a different functionality where:

Button 1: Flips the status of all the bulbs.
Button 2: Flips the status of all the bulbs with even labels (i.e., 2, 4, ...).
Button 3: Flips the status of all the bulbs with odd labels (i.e., 1, 3, ...).
Button 4: Flips the status of all the bulbs with a label j = 3k + 1 where k = 0, 1, 2, ... (i.e., 1, 4, 7, 10, ...).

You must make exactly presses button presses in total. For each press, you may pick any of the four buttons to press. Given the two integers n and presses, return the number of different possible statuses after performing all presses but ton presses.

Example 1:

Input: n = 1, presses = 1
Output: 2
Explanation: Status can be:
- [off] by pressing button 1
- [on] by pressing button 2

Example 2:

Input: n = 2, presses = 1
Output: 3
Explanation: Status can be:
- [off, off] by pressing button 1
- [on, off] by pressing button 2
- [off, on] by pressing button 3

Example 3:

Input: n = 3, presses = 1
Output: 4
Explanation: Status can be:
- [off, off, off] by pressing button 1
- [off, on, off] by pressing button 2
- [on, off, on] by pressing button 3
- [off, on, on] by pressing button 4

Example 4:

Input: n = 1, presses = 0
Output: 1
Explanation: Status can only be [on] since you cannot press any of the buttons.

Example 5:

Input: n = 1, presses = 2
Output: 2
Explanation: Status can be:
- [off] by pressing button 1 then button 1 again
- [on] by pressing button 1 then button 2

Constraints:

1 <= n <= 1000
0 <= presses <= 1000

---

673. Number of Longest Increasing Subsequence

Given an integer array nums, return the number of longest increasing subsequences.
Notice that the sequence has to be strictly increasing.

Example 1:

Input: nums = [1,3,5,4,7]
Output: 2
Explanation: The two longest increasing subsequences are [1, 3, 4, 7] and [1, 3, 5, 7].

Example 2:

Input: nums = [2,2,2,2,2]
Output: 5
Explanation: The length of longest continuous increasing subsequence is 1, and there are 5 subsequences' length is 1, so output 5.

Constraints:

1 <= nums.length <= 2000
-106 <= nums[i] <= 106

---

674. Longest Continuous Increasing Subsequence

Given an unsorted array of integers nums, return the length of the longest continuous increasing subsequence (i.e. su barray). The subsequence must be strictly increasing.
A continuous increasing subsequence is defined by two indices l and r (l < r) such that it is [nums[l], nums[l + 1], ..., nums[r - 1], nums[r]] and for each l <= i < r, nums[i] < nums[i + 1].

Example 1:

Input: nums = [1,3,5,4,7]
Output: 3
Explanation: The longest continuous increasing subsequence is [1,3,5] with length 3.
Even though [1,3,5,7] is an increasing subsequence, it is not continuous as elements 5 and 7 are separated by elemen t
4.

Example 2:

Input: nums = [2,2,2,2,2]
Output: 1
Explanation: The longest continuous increasing subsequence is [2] with length 1. Note that it must be strictly increasing.

Constraints:

1 <= nums.length <= 104
-109 <= nums[i] <= 109

---

675. Cut Off Trees for Golf Event

You are asked to cut off all the trees in a forest for a golf event. The forest is represented as an m x n matrix. In this matrix:

0 means the cell cannot be walked through.
1 represents an empty cell that can be walked through.
A number greater than 1 represents a tree in a cell that can be walked through, and this number is the tree's height.

In one step, you can walk in any of the four directions: north, east, south, and west. If you are standing in a cell with a tree, you can choose whether to cut it off.
You must cut off the trees in order from shortest to tallest. When you cut off a tree, the value at its cell becomes 1 (a n empty cell).
Starting from the point (0, 0), return the minimum steps you need to walk to cut off all the trees. If you cannot cut of f all the trees, return -1.
You are guaranteed that no two trees have the same height, and there is at least one tree needs to be cut off.

Example 1:

Input: forest = [[1,2,3],[0,0,4],[7,6,5]]
Output: 6
Explanation: Following the path above allows you to cut off the trees from shortest to tallest in 6 steps.

Example 2:

Input: forest = [[1,2,3],[0,0,0],[7,6,5]]
Output: -1
Explanation: The trees in the bottom row cannot be accessed as the middle row is blocked.

Example 3:

Input: forest = [[2,3,4],[0,0,5],[8,7,6]]
Output: 6
Explanation: You can follow the same path as Example 1 to cut off all the trees.
Note that you can cut off the first tree at (0, 0) before making any steps.

Constraints:

m == forest.length
n == forest[i].length
1 <= m, n <= 50
0 <= forest[i][j] <= 109

---

676. Implement Magic Dictionary

Design a data structure that is initialized with a list of different words. Provided a string, you should determine if yo u can change exactly one character in this string to match any word in the data structure.
Implement the MagicDictionary class:

MagicDictionary() Initializes the object.
void buildDict(String[] dictionary) Sets the data structure with an array of distinct strings dictionary.
bool search(String searchWord) Returns true if you can change exactly one character in searchWord to match any str ing in the data structure, otherwise returns false.

Example 1:

Input
["MagicDictionary", "buildDict", "search", "search", "search", "search"]
[[], [["hello", "leetcode"]], ["hello"], ["hhllo"], ["hell"], ["leetcoded"]]
Output

[null, null, false, true, false, false]

Explanation
MagicDictionary magicDictionary = new MagicDictionary();
magicDictionary.buildDict(["hello", "leetcode"]);
magicDictionary.search("hello"); // return False
magicDictionary.search("hhllo"); // We can change the second 'h' to 'e' to match "hello" so we return True
magicDictionary.search("hell"); // return False
magicDictionary.search("leetcoded"); // return False

Constraints:

1 <= dictionary.length <= 100
1 <= dictionary[i].length <= 100
dictionary[i] consists of only lower-case English letters.
All the strings in dictionary are distinct.
1 <= searchWord.length <= 100
searchWord consists of only lower-case English letters.
buildDict will be called only once before search.
At most 100 calls will be made to search.

---

677. Map Sum Pairs

Design a map that allows you to do the following:

Maps a string key to a given value.
Returns the sum of the values that have a key with a prefix equal to a given string.

Implement the MapSum class:

MapSum() Initializes the MapSum object.
void insert(String key, int val) Inserts the key-val pair into the map. If the key already existed, the original key-value pair will be overridden to the new one.
int sum(string prefix) Returns the sum of all the pairs' value whose key starts with the prefix.

Example 1:

Input
["MapSum", "insert", "sum", "insert", "sum"]
[[], ["apple", 3], ["ap"], ["app", 2], ["ap"]]
Output
[null, null, 3, null, 5]

Explanation

```
MapSum mapSum = new MapSum();
mapSum.insert("apple", 3);
mapSum.sum("ap");           // return 3 (apple = 3)
mapSum.insert("app", 2);
mapSum.sum("ap");           // return 5 (apple + app = 3 + 2 = 5)
```

Constraints:

1 <= key.length, prefix.length <= 50
key and prefix consist of only lowercase English letters.
1 <= val <= 1000
At most 50 calls will be made to insert and sum.

---

678. Valid Parenthesis String

Given a string s containing only three types of characters: '(', ')' and '*', return true if s is valid.
The following rules define a valid string:

Any left parenthesis '(' must have a corresponding right parenthesis ')'.
Any right parenthesis ')' must have a corresponding left parenthesis '('.
Left parenthesis '(' must go before the corresponding right parenthesis ')'.
'*' could be treated as a single right parenthesis ')' or a single left parenthesis '(' or an empty string "".

Example 1:
Input: s = "()"
Output: true
Example 2:
Input: s = "(*)"
Output: true
Example 3:
Input: s = "(*))"
Output: true

Constraints:

1 <= s.length <= 100
s[i] is '(', ')' or '*'.

---

679. 24 Game

You are given an integer array cards of length 4. You have four cards, each containing a number in the range [1, 9]. You should arrange the numbers on these cards in a mathematical expression using the operators ['+', '-', '*', '/'] and t he parentheses '(' and ')' to get the value 24.
You are restricted with the following rules:

The division operator '/' represents real division, not integer division.

For example, 4 / (1 - 2 / 3) = 4 / (1 / 3) = 12.

Every operation done is between two numbers. In particular, we cannot use '-' as a unary operator.

For example, if cards = [1, 1, 1, 1], the expression "-1 - 1 - 1 - 1" is not allowed.

You cannot concatenate numbers together

For example, if cards = [1, 2, 1, 2], the expression "12 + 12" is not valid.

Return true if you can get such expression that evaluates to 24, and false otherwise.

Example 1:

Input: cards = [4,1,8,7]
Output: true
Explanation: (8-4) * (7-1) = 24

Example 2:

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

Constraints:

cards.length == 4
1 <= cards[i] <= 9

---

680. Valid Palindrome II

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.

---

682. Baseball Game

You are keeping score for a baseball game with strange rules. The game consists of several rounds, where the scores of past rounds may affect future rounds' scores.
At the beginning of the game, you start with an empty record. You are given a list of strings ops, where ops[i] is the i th operation you must apply to the record and is one of the following:

An integer x - Record a new score of x.
"+" - Record a new score that is the sum of the previous two scores. It is guaranteed there will always be two previou s scores.
"D" - Record a new score that is double the previous score. It is guaranteed there will always be a previous score.
"C" - Invalidate the previous score, removing it from the record. It is guaranteed there will always be a previous scor e.

Return the sum of all the scores on the record.

Example 1:

Input: ops = ["5","2","C","D","+"]
Output: 30

Explanation:
"5" - Add 5 to the record, record is now [5].
"2" - Add 2 to the record, record is now [5, 2].
"C" - Invalidate and remove the previous score, record is now [5].
"D" - Add 2 * 5 = 10 to the record, record is now [5, 10].
"+" - Add 5 + 10 = 15 to the record, record is now [5, 10, 15].
The total sum is 5 + 10 + 15 = 30.

Example 2:

Input: ops = ["5","-2","4","C","D","9","+","+"]
Output: 27
Explanation:
"5" - Add 5 to the record, record is now [5].
"-2" - Add -2 to the record, record is now [5, -2].
"4" - Add 4 to the record, record is now [5, -2, 4].
"C" - Invalidate and remove the previous score, record is now [5, -2].
"D" - Add 2 * -2 = -4 to the record, record is now [5, -2, -4].
"9" - Add 9 to the record, record is now [5, -2, -4, 9].
"+" - Add -4 + 9 = 5 to the record, record is now [5, -2, -4, 9, 5].
"+" - Add 9 + 5 = 14 to the record, record is now [5, -2, -4, 9, 5, 14].
The total sum is 5 + -2 + -4 + 9 + 5 + 14 = 27.

Example 3:

Input: ops = ["1"]
Output: 1

Constraints:

1 <= ops.length <= 1000
ops[i] is "C", "D", "+", or a string representing an integer in the range [-3 * 104, 3 * 104].
For operation "+", there will always be at least two previous scores on the record.
For operations "C" and "D", there will always be at least one previous score on the record.

---

684. Redundant Connection

In this problem, a tree is an undirected graph that is connected and has no cycles.
You are given a graph that started as a tree with n nodes labeled from 1 to n, with one additional edge added. The ad ded edge has two different vertices chosen from 1 to n, and was not an edge that already existed. The graph is repres ented as an array edges of length n where edges[i] = [ai, bi] indicates that there is an edge between nodes ai and bi in the graph.
Return an edge that can be removed so that the resulting graph is a tree of n nodes. If there are multiple answers, retu rn the answer that occurs last in the input.

Example 1:

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

Example 2:

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

Constraints:

n == edges.length
3 <= n <= 1000
edges[i].length == 2
1 <= ai < bi <= edges.length
ai != bi
There are no repeated edges.
The given graph is connected.

---

685. Redundant Connection II

In this problem, a rooted tree is a directed graph such that, there is exactly one node (the root) for which all other no des are descendants of this node, plus every node has exactly one parent, except for the root node which has no pare nts.
The given input is a directed graph that started as a rooted tree with n nodes (with distinct values from 1 to n), with o ne additional directed edge added. The added edge has two different vertices chosen from 1 to n, and was not an edg e that already existed.
The resulting graph is given as a 2D-array of edges. Each element of edges is a pair [ui, vi] that represents a directed edge connecting nodes ui and vi, where ui is a parent of child vi.
Return an edge that can be removed so that the resulting graph is a rooted tree of n nodes. If there are multiple answe rs, return the answer that occurs last in the given 2D-array.

Example 1:

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

Example 2:

Input: edges = [[1,2],[2,3],[3,4],[4,1],[1,5]]

Output: [4,1]

Constraints:

n == edges.length
3 <= n <= 1000
edges[i].length == 2
1 <= ui, vi <= n
ui != vi

---

686. Repeated String Match

Given two strings a and b, return the minimum number of times you should repeat string a so that string b is a substri ng of it. If it is impossible for b to be a substring of a after repeating it, return -1.
Notice: string "abc" repeated 0 times is "", repeated 1 time is "abc" and repeated 2 times is "abcabc".

Example 1:

Input: a = "abcd", b = "cdabcdab"
Output: 3
Explanation: We return 3 because by repeating a three times "abcdabcdabcd", b is a substring of it.

Example 2:

Input: a = "a", b = "aa"
Output: 2

Example 3:

Input: a = "a", b = "a"
Output: 1

Example 4:

Input: a = "abc", b = "wxyz"
Output: -1

Constraints:

1 <= a.length <= 104
1 <= b.length <= 104
a and b consist of lower-case English letters.

---

687. Longest Univalue Path

Given the root of a binary tree, return the length of the longest path, where each node in the path has the same value. This path may or may not pass through the root.
The length of the path between two nodes is represented by the number of edges between them.

Example 1:

Input: root = [5,4,5,1,1,5]
Output: 2

Example 2:

Input: root = [1,4,5,4,4,5]
Output: 2

Constraints:

The number of nodes in the tree is in the range [0, 104].
-1000 <= Node.val <= 1000
The depth of the tree will not exceed 1000.

---

688. Knight Probability in Chessboard

On an n x n chessboard, a knight starts at the cell (row, column) and attempts to make exactly k moves. The rows an d columns are 0-indexed, so the top-left cell is (0, 0), and the bottom-right cell is (n - 1, n - 1).
A chess knight has eight possible moves it can make, as illustrated below. Each move is two cells in a cardinal direct ion, then one cell in an orthogonal direction.

Each time the knight is to move, it chooses one of eight possible moves uniformly at random (even if the piece woul d go off the chessboard) and moves there.
The knight continues moving until it has made exactly k moves or has moved off the chessboard.
Return the probability that the knight remains on the board after it has stopped moving.

Example 1:

Input: n = 3, k = 2, row = 0, column = 0
Output: 0.06250
Explanation: There are two moves (to (1,2), (2,1)) that will keep the knight on the board.
From each of those positions, there are also two moves that will keep the knight on the board.
The total probability the knight stays on the board is 0.0625.

Example 2:

Input: n = 1, k = 0, row = 0, column = 0
Output: 1.00000

Constraints:

1 <= n <= 25
0 <= k <= 100
0 <= row, column <= n

---

689. Maximum Sum of 3 Non-Overlapping Subarrays

Given an integer array nums and an integer k, find three non-overlapping subarrays of length k with maximum sum and return them.
Return the result as a list of indices representing the starting position of each interval (0-indexed). If there are multipl e answers, return the lexicographically smallest one.

Example 1:

Input: nums = [1,2,1,2,6,7,5,1], k = 2
Output: [0,3,5]
Explanation: Subarrays [1, 2], [2, 6], [7, 5] correspond to the starting indices [0, 3, 5].
We could have also taken [2, 1], but an answer of [1, 3, 5] would be lexicographically larger.

Example 2:

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

Constraints:

1 <= nums.length <= 2 * 104
1 <= nums[i] < 216
1 <= k <= floor(nums.length / 3)

---

690. Employee Importance

You have a data structure of employee information, including the employee's unique ID, importance value, and direc t subordinates' IDs.
You are given an array of employees employees where:

employees[i].id is the ID of the ith employee.
employees[i].importance is the importance value of the ith employee.
employees[i].subordinates is a list of the IDs of the direct subordinates of the ith employee.

Given an integer id that represents an employee's ID, return the total importance value of this employee and all their direct and indirect subordinates.

Example 1:

Input: employees = [[1,5,[2,3]],[2,3,[]],[3,3,[]]], id = 1
Output: 11
Explanation: Employee 1 has an importance value of 5 and has two direct subordinates: employee 2 and employee 3.
They both have an importance value of 3.
Thus, the total importance value of employee 1 is 5 + 3 + 3 = 11.

Example 2:

Input: employees = [[1,2,[5]],[5,-3,[]]], id = 5
Output: -3
Explanation: Employee 5 has an importance value of -3 and has no direct subordinates.
Thus, the total importance value of employee 5 is -3.

Constraints:

1 <= employees.length <= 2000
1 <= employees[i].id <= 2000
All employees[i].id are unique.
-100 <= employees[i].importance <= 100
One employee has at most one direct leader and may have several subordinates.
The IDs in employees[i].subordinates are valid IDs.

---

691. Stickers to Spell Word

We are given n different types of stickers. Each sticker has a lowercase English word on it.
You would like to spell out the given string target by cutting individual letters from your collection of stickers and re arranging them. You can use each sticker more than once if you want, and you have infinite quantities of each sticke r.
Return the minimum number of stickers that you need to spell out target. If the task is impossible, return -1.
Note: In all test cases, all words were chosen randomly from the 1000 most common US English words, and target was chosen as a concatenation of two random words.

Example 1:

Input: stickers = ["with","example","science"], target = "thehat"
Output: 3
Explanation:
We can use 2 "with" stickers, and 1 "example" sticker.
After cutting and rearrange the letters of those stickers, we can form the target "thehat".
Also, this is the minimum number of stickers necessary to form the target string.

Example 2:

Input: stickers = ["notice","possible"], target = "basicbasic"
Output: -1
Explanation:
We cannot form the target "basicbasic" from cutting letters from the given stickers.

Constraints:

n == stickers.length
1 <= n <= 50
1 <= stickers[i].length <= 10
1 <= target <= 15
stickers[i] and target consist of lowercase English letters.

---

692. Top K Frequent Words

Given an array of strings words and an integer k, return the k most frequent strings.
Return the answer sorted by the frequency from highest to lowest. Sort the words with the same frequency by their le xicographical order.

Example 1:

Input: words = ["i","love","leetcode","i","love","coding"], k = 2
Output: ["i","love"]
Explanation: "i" and "love" are the two most frequent words.

Note that "i" comes before "love" due to a lower alphabetical order.

Example 2:

Input: words = ["the","day","is","sunny","the","the","the","sunny","is","is"], k = 4
Output: ["the","is","sunny","day"]
Explanation: "the", "is", "sunny" and "day" are the four most frequent words, with the number of occurrence being 4, 3, 2 and 1 respectively.

Constraints:

1 <= words.length <= 500
1 <= words[i] <= 10
words[i] consists of lowercase English letters.
k is in the range [1, The number of unique words[i]]

Follow-up: Could you solve it in O(n log(k)) time and O(n) extra space?

---

693. Binary Number with Alternating Bits

Given a positive integer, check whether it has alternating bits: namely, if two adjacent bits will always have different values.

Example 1:

Input: n = 5
Output: true
Explanation: The binary representation of 5 is: 101

Example 2:

Input: n = 7
Output: false
Explanation: The binary representation of 7 is: 111.
Example 3:

Input: n = 11
Output: false
Explanation: The binary representation of 11 is: 1011.
Example 4:

Input: n = 10
Output: true
Explanation: The binary representation of 10 is: 1010.
Example 5:

Input: n = 3
Output: false

Constraints:

1 <= n <= 231 - 1

---

695. Max Area of Island

You are given an m x n binary matrix grid. An island is a group of 1's (representing land) connected 4-directionally (horizontal or vertical.) You may assume all four edges of the grid are surrounded by water.
The area of an island is the number of cells with a value 1 in the island.
Return the maximum area of an island in grid. If there is no island, return 0.

Example 1:

Input: grid = [[0,0,1,0,0,0,0,1,0,0,0,0,0],[0,0,0,0,0,0,0,1,1,1,0,0,0],[0,1,1,0,1,0,0,0,0,0,0,0,0],[0,1,0,0,1,1,0,0,1,0,1,0,0],[0,1,0,0,1,1,0,0,1,1,1,0,0],[0,0,0,0,0,0,0,0,0,0,1,0,0],[0,0,0,0,0,0,0,1,1,1,0,0,0],[0,0,0,0,0,0,0,1,1,0,0,0,0]]
Output: 6
Explanation: The answer is not 11, because the island must be connected 4-directionally.

Example 2:

Input: grid = [[0,0,0,0,0,0,0,0]]
Output: 0

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 50
grid[i][j] is either 0 or 1.

---

696. Count Binary Substrings

Give a binary string s, return the number of non-empty substrings that have the same number of 0's and 1's, and all th e 0's and all the 1's in these substrings are grouped consecutively.
Substrings that occur multiple times are counted the number of times they occur.

Example 1:

Input: s = "00110011"
Output: 6
Explanation: There are 6 substrings that have equal number of consecutive 1's and 0's: "0011", "01", "1100", "10", " 0011", and "01".
Notice that some of these substrings repeat and are counted the number of times they occur.
Also, "00110011" is not a valid substring because all the 0's (and 1's) are not grouped together.

Example 2:

Input: s = "10101"
Output: 4
Explanation: There are 4 substrings: "10", "01", "10", "01" that have equal number of consecutive 1's and 0's.

Constraints:

1 <= s.length <= 105
s[i] is either '0' or '1'.

---

697. Degree of an Array

Given a non-empty array of non-negative integers nums, the degree of this array is defined as the maximum frequen cy of any one of its elements.
Your task is to find the smallest possible length of a (contiguous) subarray of nums, that has the same degree as num s.

Example 1:

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

Example 2:

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

Output: 6
Explanation:
The degree is 3 because the element 2 is repeated 3 times.
So [2,2,3,1,4,2] is the shortest subarray, therefore returning 6.

Constraints:

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

---

698. Partition to K Equal Sum Subsets

Given an integer array nums and an integer k, return true if it is possible to divide this array into k non-empty subset s whose sums are all equal.

Example 1:

Input: nums = [4,3,2,3,5,2,1], k = 4
Output: true
Explanation: It's possible to divide it into 4 subsets (5), (1, 4), (2,3), (2,3) with equal sums.

Example 2:

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

Constraints:

1 <= k <= nums.length <= 16
1 <= nums[i] <= 104
The frequency of each element is in the range [1, 4].

---

699. Falling Squares

There are several squares being dropped onto the X-axis of a 2D plane.
You are given a 2D integer array positions where positions[i] = [lefti, sideLengthi] represents the ith square with a si

de length of sideLengthi that is dropped with its left edge aligned with X-coordinate lefti.
Each square is dropped one at a time from a height above any landed squares. It then falls downward (negative Y dir ection) until it either lands on the top side of another square or on the X-axis. A square brushing the left/right side of another square does not count as landing on it. Once it lands, it freezes in place and cannot be moved.
After each square is dropped, you must record the height of the current tallest stack of squares.
Return an integer array ans where ans[i] represents the height described above after dropping the ith square.

Example 1:

Input: positions = [[1,2],[2,3],[6,1]]
Output: [2,5,5]
Explanation:
After the first drop, the tallest stack is square 1 with a height of 2.
After the second drop, the tallest stack is squares 1 and 2 with a height of 5.
After the third drop, the tallest stack is still squares 1 and 2 with a height of 5.
Thus, we return an answer of [2, 5, 5].

Example 2:

Input: positions = [[100,100],[200,100]]
Output: [100,100]
Explanation:
After the first drop, the tallest stack is square 1 with a height of 100.
After the second drop, the tallest stack is either square 1 or square 2, both with heights of 100.
Thus, we return an answer of [100, 100].
Note that square 2 only brushes the right side of square 1, which does not count as landing on it.

Constraints:

1 <= positions.length <= 1000
1 <= lefti <= 108
1 <= sideLengthi <= 106

---

700. Search in a Binary Search Tree

You are given the root of a binary search tree (BST) and an integer val.
Find the node in the BST that the node's value equals val and return the subtree rooted with that node. If such a node does not exist, return null.

Example 1:

Input: root = [4,2,7,1,3], val = 2
Output: [2,1,3]

Example 2:

Input: root = [4,2,7,1,3], val = 5
Output: []

Constraints:

The number of nodes in the tree is in the range [1, 5000].
1 <= Node.val <= 107
root is a binary search tree.
1 <= val <= 107

---

701. Insert into a Binary Search Tree

You are given the root node of a binary search tree (BST) and a value to insert into the tree. Return the root node of t he BST after the insertion. It is guaranteed that the new value does not exist in the original BST.
Notice that there may exist multiple valid ways for the insertion, as long as the tree remains a BST after insertion. Y ou can return any of them.

Example 1:

Input: root = [4,2,7,1,3], val = 5
Output: [4,2,7,1,3,5]
Explanation: Another accepted tree is:

Example 2:

Input: root = [40,20,60,10,30,50,70], val = 25
Output: [40,20,60,10,30,50,70,null,null,25]

Example 3:

Input: root = [4,2,7,1,3,null,null,null,null,null,null], val = 5
Output: [4,2,7,1,3,5]

Constraints:

The number of nodes in the tree will be in the range [0, 104].
-108 <= Node.val <= 108
All the values Node.val are unique.

-108 <= val <= 108
It's guaranteed that val does not exist in the original BST.

---

703. Kth Largest Element in a Stream

Design a class to find the kth largest element in a stream. Note that it is the kth largest element in the sorted order, no t the kth distinct element.
Implement KthLargest class:

KthLargest(int k, int[] nums) Initializes the object with the integer k and the stream of integers nums.
int add(int val) Appends the integer val to the stream and returns the element representing the kth largest element in t he stream.

Example 1:

Input
["KthLargest", "add", "add", "add", "add", "add"]
[[3, [4, 5, 8, 2]], [3], [5], [10], [9], [4]]
Output
[null, 4, 5, 5, 8, 8]

Explanation
KthLargest kthLargest = new KthLargest(3, [4, 5, 8, 2]);
kthLargest.add(3);   // return 4
kthLargest.add(5);   // return 5
kthLargest.add(10); // return 5
kthLargest.add(9);   // return 8
kthLargest.add(4);   // return 8

Constraints:

1 <= k <= 104
0 <= nums.length <= 104
-104 <= nums[i] <= 104
-104 <= val <= 104
At most 104 calls will be made to add.
It is guaranteed that there will be at least k elements in the array when you search for the kth element.

---

704. Binary Search

Given an array of integers nums which is sorted in ascending order, and an integer target, write a function to search t
arget in nums. If target exists, then return its index. Otherwise, return -1.
You must write an algorithm with O(log n) runtime complexity.

Example 1:

Input: nums = [-1,0,3,5,9,12], target = 9
Output: 4
Explanation: 9 exists in nums and its index is 4

Example 2:

Input: nums = [-1,0,3,5,9,12], target = 2
Output: -1
Explanation: 2 does not exist in nums so return -1

Constraints:

1 <= nums.length <= 104
-104 < nums[i], target < 104
All the integers in nums are unique.
nums is sorted in ascending order.

---

705. Design HashSet

Design a HashSet without using any built-in hash table libraries.
Implement MyHashSet class:

void add(key) Inserts the value key into the HashSet.
bool contains(key) Returns whether the value key exists in the HashSet or not.
void remove(key) Removes the value key in the HashSet. If key does not exist in the HashSet, do nothing.

Example 1:

Input
["MyHashSet", "add", "add", "contains", "contains", "add", "contains", "remove", "contains"]
[[], [1], [2], [1], [3], [2], [2], [2], [2]]
Output
[null, null, null, true, false, null, true, null, false]

Explanation
MyHashSet myHashSet = new MyHashSet();
myHashSet.add(1);   // set = [1]
myHashSet.add(2);   // set = [1, 2]
myHashSet.contains(1); // return True
myHashSet.contains(3); // return False, (not found)
myHashSet.add(2);   // set = [1, 2]
myHashSet.contains(2); // return True
myHashSet.remove(2);   // set = [1]
myHashSet.contains(2); // return False, (already removed)

Constraints:

0 <= key <= 106
At most 104 calls will be made to add, remove, and contains.

---

706. Design HashMap

Design a HashMap without using any built-in hash table libraries.
Implement the MyHashMap class:

MyHashMap() initializes the object with an empty map.
void put(int key, int value) inserts a (key, value) pair into the HashMap. If the key already exists in the map, update t he corresponding value.
int get(int key) returns the value to which the specified key is mapped, or -1 if this map contains no mapping for the key.
void remove(key) removes the key and its corresponding value if the map contains the mapping for the key.

Example 1:

Input
["MyHashMap", "put", "put", "get", "get", "put", "get", "remove", "get"]
[[], [1, 1], [2, 2], [1], [3], [2, 1], [2], [2], [2]]
Output
[null, null, null, 1, -1, null, 1, null, -1]

Explanation
MyHashMap myHashMap = new MyHashMap();
myHashMap.put(1, 1); // The map is now [[1,1]]
myHashMap.put(2, 2); // The map is now [[1,1], [2,2]]
myHashMap.get(1);   // return 1, The map is now [[1,1], [2,2]]
myHashMap.get(3);   // return -1 (i.e., not found), The map is now [[1,1], [2,2]]
myHashMap.put(2, 1); // The map is now [[1,1], [2,1]] (i.e., update the existing value)
myHashMap.get(2);   // return 1, The map is now [[1,1], [2,1]]
myHashMap.remove(2); // remove the mapping for 2, The map is now [[1,1]]

myHashMap.get(2);   // return -1 (i.e., not found), The map is now [[1,1]]

Constraints:

0 <= key, value <= 106
At most 104 calls will be made to put, get, and remove.

---

707. Design Linked List

Design your implementation of the linked list. You can choose to use a singly or doubly linked list.
A node in a singly linked list should have two attributes: val and next. val is the value of the current node, and next i s a pointer/reference to the next node.
If you want to use the doubly linked list, you will need one more attribute prev to indicate the previous node in the li nked list. Assume all nodes in the linked list are 0-indexed.
Implement the MyLinkedList class:

MyLinkedList() Initializes the MyLinkedList object.
int get(int index) Get the value of the indexth node in the linked list. If the index is invalid, return -1.
void addAtHead(int val) Add a node of value val before the first element of the linked list. After the insertion, the ne w node will be the first node of the linked list.
void addAtTail(int val) Append a node of value val as the last element of the linked list.
void addAtIndex(int index, int val) Add a node of value val before the indexth node in the linked list. If index equals the length of the linked list, the node will be appended to the end of the linked list. If index is greater than the length , the node will not be inserted.
void deleteAtIndex(int index) Delete the indexth node in the linked list, if the index is valid.

Example 1:

Input
["MyLinkedList", "addAtHead", "addAtTail", "addAtIndex", "get", "deleteAtIndex", "get"]
[[], [1], [3], [1, 2], [1], [1], [1]]
Output
[null, null, null, null, 2, null, 3]

Explanation
MyLinkedList myLinkedList = new MyLinkedList();
myLinkedList.addAtHead(1);
myLinkedList.addAtTail(3);
myLinkedList.addAtIndex(1, 2);   // linked list becomes 1->2->3
myLinkedList.get(1);   // return 2
myLinkedList.deleteAtIndex(1);   // now the linked list is 1->3
myLinkedList.get(1);   // return 3

Constraints:

0 <= index, val <= 1000
Please do not use the built-in LinkedList library.
At most 2000 calls will be made to get, addAtHead, addAtTail, addAtIndex and deleteAtIndex.

---

709. To Lower Case

Given a string s, return the string after replacing every uppercase letter with the same lowercase letter.

Example 1:

Input: s = "Hello"
Output: "hello"

Example 2:

Input: s = "here"
Output: "here"

Example 3:

Input: s = "LOVELY"
Output: "lovely"

Constraints:

1 <= s.length <= 100
s consists of printable ASCII characters.

---

710. Random Pick with Blacklist

You are given an integer n and an array of unique integers blacklist. Design an algorithm to pick a random integer in the range [0, n - 1] that is not in blacklist. Any integer that is in the mentioned range and not in blacklist should be equally likely to be returned.
Optimize your algorithm such that it minimizes the number of calls to the built-in random function of your language.
Implement the Solution class:

Solution(int n, int[] blacklist) Initializes the object with the integer n and the blacklisted integers blacklist.
int pick() Returns a random integer in the range [0, n - 1] and not in blacklist.

Example 1:

Input
["Solution", "pick", "pick", "pick", "pick", "pick", "pick", "pick"]
[[7, [2, 3, 5]], [], [], [], [], [], [], []]
Output
[null, 0, 4, 1, 6, 1, 0, 4]

Explanation
Solution solution = new Solution(7, [2, 3, 5]);
solution.pick(); // return 0, any integer from [0,1,4,6] should be ok. Note that for every call of pick,
// 0, 1, 4, and 6 must be equally likely to be returned (i.e., with probability 1/4).
solution.pick(); // return 4
solution.pick(); // return 1
solution.pick(); // return 6
solution.pick(); // return 1
solution.pick(); // return 0
solution.pick(); // return 4

Constraints:

1 <= n <= 109
0 <= blacklist.length <- min(105, n - 1)
0 <= blacklist[i] < n
All the values of blacklist are unique.
At most 2 * 104 calls will be made to pick.

---

712. Minimum ASCII Delete Sum for Two Strings

Given two strings s1 and s2, return the lowest ASCII sum of deleted characters to make two strings equal.

Example 1:

Input: s1 = "sea", s2 = "eat"
Output: 231
Explanation: Deleting "s" from "sea" adds the ASCII value of "s" (115) to the sum.
Deleting "t" from "eat" adds 116 to the sum.
At the end, both strings are equal, and 115 + 116 = 231 is the minimum sum possible to achieve this.

Example 2:

Input: s1 = "delete", s2 = "leet"
Output: 403
Explanation: Deleting "dee" from "delete" to turn the string into "let",
adds 100[d] + 101[e] + 101[e] to the sum.
Deleting "e" from "leet" adds 101[e] to the sum.
At the end, both strings are equal to "let", and the answer is 100+101+101+101 = 403.
If instead we turned both strings into "lee" or "eet", we would get answers of 433 or 417, which are higher.

Constraints:

1 <= s1.length, s2.length <= 1000
s1 and s2 consist of lowercase English letters.

---

713. Subarray Product Less Than K

Given an array of integers nums and an integer k, return the number of contiguous subarrays where the product of all the elements in the subarray is strictly less than k.

Example 1:

Input: nums = [10,5,2,6], k = 100
Output: 8
Explanation: The 8 subarrays that have product less than 100 are:
[10], [5], [2], [6], [10, 5], [5, 2], [2, 6], [5, 2, 6]
Note that [10, 5, 2] is not included as the product of 100 is not strictly less than k.

Example 2:

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

Constraints:

1 <= nums.length <= 3 * 104
1 <= nums[i] <= 1000
0 <= k <= 106

---

714. Best Time to Buy and Sell Stock with Transaction Fee

You are given an array prices where prices[i] is the price of a given stock on the ith day, and an integer fee representi ng a transaction fee.
Find the maximum profit you can achieve. You may complete as many transactions as you like, but you need to pay the transaction fee for each transaction.
Note: You may not engage in multiple transactions simultaneously (i.e., you must sell the stock before you buy agai n).

Example 1:

Input: prices = [1,3,2,8,4,9], fee = 2
Output: 8
Explanation: The maximum profit can be achieved by:
- Buying at prices[0] = 1
- Selling at prices[3] = 8
- Buying at prices[4] = 4
- Selling at prices[5] = 9
The total profit is ((8 - 1) - 2) + ((9 - 4) - 2) = 8.

Example 2:

Input: prices = [1,3,7,5,10,3], fee = 3
Output: 6

Constraints:

1 <= prices.length <= 5 * 104
1 <= prices[i] < 5 * 104
0 <= fee < 5 * 104

---

715. Range Module

A Range Module is a module that tracks ranges of numbers. Design a data structure to track the ranges represented a s half-open intervals and query about them.
A half-open interval [left, right) denotes all the real numbers x where left <= x < right.
Implement the RangeModule class:

RangeModule() Initializes the object of the data structure.
void addRange(int left, int right) Adds the half-open interval [left, right), tracking every real number in that interval. Adding an interval that partially overlaps with currently tracked numbers should add any numbers in the interval [lef t, right) that are not already tracked.
boolean queryRange(int left, int right) Returns true if every real number in the interval [left, right) is currently being

tracked, and false otherwise.
void removeRange(int left, int right) Stops tracking every real number currently being tracked in the half-open interv al [left, right).

Example 1:

Input
["RangeModule", "addRange", "removeRange", "queryRange", "queryRange", "queryRange"]
[[], [10, 20], [14, 16], [10, 14], [13, 15], [16, 17]]
Output
[null, null, null, true, false, true]

Explanation
RangeModule rangeModule = new RangeModule();
rangeModule.addRange(10, 20);
rangeModule.removeRange(14, 16);
rangeModule.queryRange(10, 14); // return True,(Every number in [10, 14) is being tracked)
rangeModule.queryRange(13, 15); // return False,(Numbers like 14, 14.03, 14.17 in [13, 15) are not being tracked)
rangeModule.queryRange(16, 17); // return True, (The number 16 in [16, 17) is still being tracked, despite the remov e operation)

Constraints:

1 <= left < right <= 109
At most 104 calls will be made to addRange, queryRange, and removeRange.

---

717. 1-bit and 2-bit Characters

We have two special characters:

The first character can be represented by one bit 0.
The second character can be represented by two bits (10 or 11).

Given a binary array bits that ends with 0, return true if the last character must be a one-bit character.

Example 1:

Input: bits = [1,0,0]
Output: true
Explanation: The only way to decode it is two-bit character and one-bit character.
So the last character is one-bit character.

Example 2:

Input: bits = [1,1,1,0]
Output: false
Explanation: The only way to decode it is two-bit character and two-bit character.
So the last character is not one-bit character.

Constraints:

1 <= bits.length <= 1000
bits[i] is either 0 or 1.

---

718. Maximum Length of Repeated Subarray

Given two integer arrays nums1 and nums2, return the maximum length of a subarray that appears in both arrays.

Example 1:

Input: nums1 = [1,2,3,2,1], nums2 = [3,2,1,4,7]
Output: 3
Explanation: The repeated subarray with maximum length is [3,2,1].

Example 2:

Input: nums1 = [0,0,0,0,0], nums2 = [0,0,0,0,0]
Output: 5

Constraints:

1 <= nums1.length, nums2.length <= 1000
0 <= nums1[i], nums2[i] <= 100

---

719. Find K-th Smallest Pair Distance

The distance of a pair of integers a and b is defined as the absolute difference between a and b.
Given an integer array nums and an integer k, return the kth smallest distance among all the pairs nums[i] and nums[j] where 0 <= i < j < nums.length.

Example 1:

Input: nums = [1,3,1], k = 1
Output: 0
Explanation: Here are all the pairs:
(1,3) -> 2
(1,1) -> 0
(3,1) -> 2
Then the 1st smallest distance pair is (1,1), and its distance is 0.

Example 2:

Input: nums = [1,1,1], k = 2
Output: 0

Example 3:

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

Constraints:

n == nums.length
2 <= n <= 104
0 <= nums[i] <= 106
1 <= k <= n * (n - 1) / 2

---

720. Longest Word in Dictionary

Given an array of strings words representing an English Dictionary, return the longest word in words that can be buil t one character at a time by other words in words.
If there is more than one possible answer, return the longest word with the smallest lexicographical order. If there is no answer, return the empty string.

Example 1:

Input: words = ["w","wo","wor","worl","world"]
Output: "world"
Explanation: The word "world" can be built one character at a time by "w", "wo", "wor", and "worl".

Example 2:

Input: words = ["a","banana","app","appl","ap","apply","apple"]
Output: "apple"
Explanation: Both "apply" and "apple" can be built from other words in the dictionary. However, "apple" is lexicogr

aphically smaller than "apply".

Constraints:

1 <= words.length <= 1000
1 <= words[i].length <= 30
words[i] consists of lowercase English letters.

---

721. Accounts Merge

Given a list of accounts where each element accounts[i] is a list of strings, where the first element accounts[i][0] is a name, and the rest of the elements are emails representing emails of the account.
Now, we would like to merge these accounts. Two accounts definitely belong to the same person if there is some co mmon email to both accounts. Note that even if two accounts have the same name, they may belong to different peo ple as people could have the same name. A person can have any number of accounts initially, but all of their account s definitely have the same name.
After merging the accounts, return the accounts in the following format: the first element of each account is the nam e, and the rest of the elements are emails in sorted order. The accounts themselves can be returned in any order.

Example 1:

Input: accounts = [["John","johnsmith@mail.com","john_newyork@mail.com"],["John","johnsmith@mail.com","jo hn00@mail.com"],["Mary","mary@mail.com"],["John","johnnybravo@mail.com"]]
Output: [["John","john00@mail.com","john_newyork@mail.com","johnsmith@mail.com"],["Mary","mary@mail.c om"],["John","johnnybravo@mail.com"]]
Explanation:
The first and second John's are the same person as they have the common email "johnsmith@mail.com".
The third John and Mary are different people as none of their email addresses are used by other accounts.
We could return these lists in any order, for example the answer [['Mary', 'mary@mail.com'], ['John', 'johnnybravo@ mail.com'],
['John', 'john00@mail.com', 'john_newyork@mail.com', 'johnsmith@mail.com']] would still be accepted.

Example 2:

Input: accounts = [["Gabe","Gabe0@m.co","Gabe3@m.co","Gabe1@m.co"],["Kevin","Kevin3@m.co","Kevin5@ m.co","Kevin0@m.co"],["Ethan","Ethan5@m.co","Ethan4@m.co","Ethan0@m.co"],["Hanzo","Hanzo3@m.co","H anzo1@m.co","Hanzo0@m.co"],["Fern","Fern5@m.co","Fern1@m.co","Fern0@m.co"]]
Output: [["Ethan","Ethan0@m.co","Ethan4@m.co","Ethan5@m.co"],["Gabe","Gabe0@m.co","Gabe1@m.co","Gab e3@m.co"],["Hanzo","Hanzo0@m.co","Hanzo1@m.co","Hanzo3@m.co"],["Kevin","Kevin0@m.co","Kevin3@m.c o","Kevin5@m.co"],["Fern","Fern0@m.co","Fern1@m.co","Fern5@m.co"]]

Constraints:

1 <= accounts.length <= 1000

2 <= accounts[i].length <= 10
1 <= accounts[i][j] <= 30
accounts[i][0] consists of English letters.
accounts[i][j] (for j > 0) is a valid email.

---

722. Remove Comments

Given a C++ program, remove comments from it. The program source is an array of strings source where source[i] i s the ith line of the source code. This represents the result of splitting the original source code string by the newline c haracter '\n'.
In C++, there are two types of comments, line comments, and block comments.

The string "//" denotes a line comment, which represents that it and the rest of the characters to the right of it in the s ame line should be ignored.
The string "/*" denotes a block comment, which represents that all characters until the next (non-overlapping) occurr ence of "*/" should be ignored. (Here, occurrences happen in reading order: line by line from left to right.) To be cle ar, the string "/*/" does not yet end the block comment, as the ending would be overlapping the beginning.

The first effective comment takes precedence over others.

For example, if the string "//" occurs in a block comment, it is ignored.
Similarly, if the string "/*" occurs in a line or block comment, it is also ignored.

If a certain line of code is empty after removing comments, you must not output that line: each string in the answer li st will be non-empty.
There will be no control characters, single quote, or double quote characters.

For example, source = "string s = "/* Not a comment. */";" will not be a test case.

Also, nothing else such as defines or macros will interfere with the comments.
It is guaranteed that every open block comment will eventually be closed, so "/*" outside of a line or block comment always starts a new comment.
Finally, implicit newline characters can be deleted by block comments. Please see the examples below for details.
After removing the comments from the source code, return the source code in the same format.

Example 1:

Input: source = ["/*Test program */", "int main()", "{ ", " // variable declaration ", "int a, b, c;", "/* This is a test", " multiline ", "   comment for ", "   testing */", "a = b + c;", "}"]
Output: ["int main()","{ "," ","int a, b, c;","a = b + c;","}"]
Explanation: The line by line code is visualized as below:
/*Test program */
int main()
{
  // variable declaration
int a, b, c;

```
/* This is a test
   multiline
   comment for
   testing */
a = b + c;
}
```

The string /* denotes a block comment, including line 1 and lines 6-9. The string // denotes line 4 as comments.
The line by line output code is visualized as below:

```
int main()
{

int a, b, c;
a = b + c;
}
```

Example 2:

Input: source = ["a/*comment", "line", "more_comment*/b"]
Output: ["ab"]
Explanation: The original source string is "a/*comment\nline\nmore_comment*/b", where we have bolded the newli ne characters. After deletion, the implicit newline characters are deleted, leaving the string "ab", which when delimit ed by newline characters becomes ["ab"].

Constraints:

1 <= source.length <= 100
0 <= source[i].length <= 80
source[i] consists of printable ASCII characters.
Every open block comment is eventually closed.
There are no single-quote or double-quote in the input.

---

724. Find Pivot Index

Given an array of integers nums, calculate the pivot index of this array.
The pivot index is the index where the sum of all the numbers strictly to the left of the index is equal to the sum of al l the numbers strictly to the index's right.
If the index is on the left edge of the array, then the left sum is 0 because there are no elements to the left. This also a pplies to the right edge of the array.
Return the leftmost pivot index. If no such index exists, return -1.

Example 1:

Input: nums = [1,7,3,6,5,6]
Output: 3
Explanation:

The pivot index is 3.
Left sum = nums[0] + nums[1] + nums[2] = 1 + 7 + 3 = 11
Right sum = nums[4] + nums[5] = 5 + 6 = 11

Example 2:

Input: nums = [1,2,3]
Output: -1
Explanation:
There is no index that satisfies the conditions in the problem statement.
Example 3:

Input: nums = [2,1,-1]
Output: 0
Explanation:
The pivot index is 0.
Left sum = 0 (no elements to the left of index 0)
Right sum = nums[1] + nums[2] = 1 + -1 = 0

Constraints:

1 <= nums.length <= 104
-1000 <= nums[i] <= 1000

Note: This question is the same as 1991: https://leetcode.com/problems/find-the-middle-index-in-array/

---

725. Split Linked List in Parts

Given the head of a singly linked list and an integer k, split the linked list into k consecutive linked list parts.
The length of each part should be as equal as possible: no two parts should have a size differing by more than one. T his may lead to some parts being null.
The parts should be in the order of occurrence in the input list, and parts occurring earlier should always have a size greater than or equal to parts occurring later.
Return an array of the k parts.

Example 1:

Input: head = [1,2,3], k = 5
Output: [[1],[2],[3],[],[]]
Explanation:
The first element output[0] has output[0].val = 1, output[0].next = null.
The last element output[4] is null, but its string representation as a ListNode is [].

Example 2:

Input: head = [1,2,3,4,5,6,7,8,9,10], k = 3
Output: [[1,2,3,4],[5,6,7],[8,9,10]]
Explanation:
The input has been split into consecutive parts with size difference at most 1, and earlier parts are a larger size than t he later parts.

Constraints:

The number of nodes in the list is in the range [0, 1000].
0 <= Node.val <= 1000
1 <= k <= 50

---

726. Number of Atoms

Given a string formula representing a chemical formula, return the count of each atom.
The atomic element always starts with an uppercase character, then zero or more lowercase letters, representing the name.
One or more digits representing that element's count may follow if the count is greater than 1. If the count is 1, no di gits will follow.

For example, "H2O" and "H2O2" are possible, but "H1O2" is impossible.

Two formulas are concatenated together to produce another formula.

For example, "H2O2He3Mg4" is also a formula.

A formula placed in parentheses, and a count (optionally added) is also a formula.

For example, "(H2O2)" and "(H2O2)3" are formulas.

Return the count of all elements as a string in the following form: the first name (in sorted order), followed by its co unt (if that count is more than 1), followed by the second name (in sorted order), followed by its count (if that count i s more than 1), and so on.

Example 1:

Input: formula = "H2O"
Output: "H2O"
Explanation: The count of elements are {'H': 2, 'O': 1}.

Example 2:

Input: formula = "Mg(OH)2"

Output: "H2MgO2"
Explanation: The count of elements are {'H': 2, 'Mg': 1, 'O': 2}.

Example 3:

Input: formula = "K4(ON(SO3)2)2"
Output: "K4N2O14S4"
Explanation: The count of elements are {'K': 4, 'N': 2, 'O': 14, 'S': 4}.

Example 4:

Input: formula = "Be32"
Output: "Be32"

Constraints:

1 <= formula.length <= 1000
formula consists of English letters, digits, '(', and ')'.
formula is always valid.
All the values in the output will fit in a 32-bit integer.

---

728. Self Dividing Numbers

A self-dividing number is a number that is divisible by every digit it contains.

For example, 128 is a self-dividing number because 128 % 1 == 0, 128 % 2 == 0, and 128 % 8 == 0.

A self-dividing number is not allowed to contain the digit zero.
Given two integers left and right, return a list of all the self-dividing numbers in the range [left, right].

Example 1:
Input: left = 1, right = 22
Output: [1,2,3,4,5,6,7,8,9,11,12,15,22]
Example 2:
Input: left = 47, right = 85
Output: [48,55,66,77]

Constraints:

1 <= left <= right <= 104

---

729. My Calendar I

You are implementing a program to use as your calendar. We can add a new event if adding the event will not cause a double booking.
A double booking happens when two events have some non-empty intersection (i.e., some moment is common to bo th events.).
The event can be represented as a pair of integers start and end that represents a booking on the half-open interval [st art, end), the range of real numbers x such that start <= x < end.
Implement the MyCalendar class:

MyCalendar() Initializes the calendar object.
boolean book(int start, int end) Returns true if the event can be added to the calendar successfully without causing a double booking. Otherwise, return false and do not add the event to the calendar.

Example 1:

Input
["MyCalendar", "book", "book", "book"]
[[], [10, 20], [15, 25], [20, 30]]
Output
[null, true, false, true]

Explanation
MyCalendar myCalendar = new MyCalendar();
myCalendar.book(10, 20); // return True
myCalendar.book(15, 25); // return False, It can not be booked because time 15 is already booked by another event.
myCalendar.book(20, 30); // return True, The event can be booked, as the first event takes every time less than 20, b ut not including 20.

Constraints:

0 <= start < end <= 109
At most 1000 calls will be made to book.

---

730. Count Different Palindromic Subsequences

Given a string s, return the number of different non-empty palindromic subsequences in s. Since the answer may be very large, return it modulo 109 + 7.
A subsequence of a string is obtained by deleting zero or more characters from the string.
A sequence is palindromic if it is equal to the sequence reversed.

Two sequences a1, a2, ... and b1, b2, ... are different if there is some i for which ai != bi.

Example 1:

Input: s = "bccb"
Output: 6
Explanation: The 6 different non-empty palindromic subsequences are 'b', 'c', 'bb', 'cc', 'bcb', 'bccb'.
Note that 'bcb' is counted only once, even though it occurs twice.

Example 2:

Input: s = "abcdabcdabcdabcdabcdabcdabcdabcddcbadcbadcbadcbadcbadcbadcbadcba"
Output: 104860361
Explanation: There are 3104860382 different non-empty palindromic subsequences, which is 104860361 modulo 10 9 + 7.

Constraints:

1 <= s.length <= 1000
s[i] is either 'a', 'b', 'c', or 'd'.

---

731. My Calendar II

You are implementing a program to use as your calendar. We can add a new event if adding the event will not cause a triple booking.
A triple booking happens when three events have some non-empty intersection (i.e., some moment is common to all the three events.).
The event can be represented as a pair of integers start and end that represents a booking on the half-open interval [st art, end), the range of real numbers x such that start <= x < end.
Implement the MyCalendarTwo class:

MyCalendarTwo() Initializes the calendar object.
boolean book(int start, int end) Returns true if the event can be added to the calendar successfully without causing a t riple booking. Otherwise, return false and do not add the event to the calendar.

Example 1:

Input
["MyCalendarTwo", "book", "book", "book", "book", "book", "book"]
[[], [10, 20], [50, 60], [10, 40], [5, 15], [5, 10], [25, 55]]
Output
[null, true, true, true, false, true, true]

Explanation

MyCalendarTwo myCalendarTwo = new MyCalendarTwo();
myCalendarTwo.book(10, 20); // return True, The event can be booked.
myCalendarTwo.book(50, 60); // return True, The event can be booked.
myCalendarTwo.book(10, 40); // return True, The event can be double booked.
myCalendarTwo.book(5, 15); // return False, The event ca not be booked, because it would result in a triple booking
.
myCalendarTwo.book(5, 10); // return True, The event can be booked, as it does not use time 10 which is already do
uble booked.
myCalendarTwo.book(25, 55); // return True, The event can be booked, as the time in [25, 40) will be double booke
d with the third event, the time [40, 50) will be single booked, and the time [50, 55) will be double booked with the s
econd event.

Constraints:

0 <= start < end <= 109
At most 1000 calls will be made to book.

---

732. My Calendar III

A k-booking happens when k events have some non-empty intersection (i.e., there is some time that is common to al
l k events.)
You are given some events [start, end), after each given event, return an integer k representing the maximum k-book
ing between all the previous events.
Implement the MyCalendarThree class:

MyCalendarThree() Initializes the object.
int book(int start, int end) Returns an integer k representing the largest integer such that there exists a k-booking in t
he calendar.

Example 1:

Input
["MyCalendarThree", "book", "book", "book", "book", "book", "book"]
[[], [10, 20], [50, 60], [10, 40], [5, 15], [5, 10], [25, 55]]
Output
[null, 1, 1, 2, 3, 3, 3]

Explanation
MyCalendarThree myCalendarThree = new MyCalendarThree();
myCalendarThree.book(10, 20); // return 1, The first event can be booked and is disjoint, so the maximum k-booking
 is a 1-booking.
myCalendarThree.book(50, 60); // return 1, The second event can be booked and is disjoint, so the maximum k-book
ing is a 1-booking.
myCalendarThree.book(10, 40); // return 2, The third event [10, 40) intersects the first event, and the maximum k-bo

oking is a 2-booking.
myCalendarThree.book(5, 15); // return 3, The remaining events cause the maximum K-booking to be only a 3-book ing.
myCalendarThree.book(5, 10); // return 3
myCalendarThree.book(25, 55); // return 3

Constraints:

0 <= start < end <= 109
At most 400 calls will be made to book.

---

733. Flood Fill

An image is represented by an m x n integer grid image where image[i][j] represents the pixel value of the image.
You are also given three integers sr, sc, and newColor. You should perform a flood fill on the image starting from th e pixel image[sr][sc].
To perform a flood fill, consider the starting pixel, plus any pixels connected 4-directionally to the starting pixel of t he same color as the starting pixel, plus any pixels connected 4-directionally to those pixels (also with the same color), and so on. Replace the color of all of the aforementioned pixels with newColor.
Return the modified image after performing the flood fill.

Example 1:

Input: image = [[1,1,1],[1,1,0],[1,0,1]], sr = 1, sc = 1, newColor = 2
Output: [[2,2,2],[2,2,0],[2,0,1]]
Explanation: From the center of the image with position (sr, sc) = (1, 1) (i.e., the red pixel), all pixels connected by a path of the same color as the starting pixel (i.e., the blue pixels) are colored with the new color.
Note the bottom corner is not colored 2, because it is not 4-directionally connected to the starting pixel.

Example 2:

Input: image = [[0,0,0],[0,0,0]], sr = 0, sc = 0, newColor = 2
Output: [[2,2,2],[2,2,2]]

Constraints:

m == image.length
n == image[i].length
1 <= m, n <= 50
0 <= image[i][j], newColor < 216
0 <= sr < m
0 <= sc < n

---

735. Asteroid Collision

We are given an array asteroids of integers representing asteroids in a row.
For each asteroid, the absolute value represents its size, and the sign represents its direction (positive meaning right, negative meaning left). Each asteroid moves at the same speed.
Find out the state of the asteroids after all collisions. If two asteroids meet, the smaller one will explode. If both are t he same size, both will explode. Two asteroids moving in the same direction will never meet.

Example 1:

Input: asteroids = [5,10,-5]
Output: [5,10]
Explanation: The 10 and -5 collide resulting in 10. The 5 and 10 never collide.

Example 2:

Input: asteroids = [8,-8]
Output: []
Explanation: The 8 and -8 collide exploding each other.

Example 3:

Input: asteroids = [10,2,-5]
Output: [10]
Explanation: The 2 and -5 collide resulting in -5. The 10 and -5 collide resulting in 10.

Example 4:

Input: asteroids = [-2,-1,1,2]
Output: [-2,-1,1,2]
Explanation: The -2 and -1 are moving left, while the 1 and 2 are moving right. Asteroids moving the same direction never meet, so no asteroids will meet each other.

Constraints:

2 <= asteroids.length <= 104
-1000 <= asteroids[i] <= 1000
asteroids[i] != 0

---

736. Parse Lisp Expression

You are given a string expression representing a Lisp-like expression to return the integer value of.
The syntax for these expressions is given as follows.

An expression is either an integer, let expression, add expression, mult expression, or an assigned variable. Expressi ons always evaluate to a single integer.
(An integer could be positive or negative.)
A let expression takes the form "(let v1 e1 v2 e2 ... vn en expr)", where let is always the string "let", then there are o ne or more pairs of alternating variables and expressions, meaning that the first variable v1 is assigned the value of t he expression e1, the second variable v2 is assigned the value of the expression e2, and so on sequentially; and then t he value of this let expression is the value of the expression expr.
An add expression takes the form "(add e1 e2)" where add is always the string "add", there are always two expressio ns e1, e2 and the result is the addition of the evaluation of e1 and the evaluation of e2.
A mult expression takes the form "(mult e1 e2)" where mult is always the string "mult", there are always two express ions e1, e2 and the result is the multiplication of the evaluation of e1 and the evaluation of e2.
For this question, we will use a smaller subset of variable names. A variable starts with a lowercase letter, then zero or more lowercase letters or digits. Additionally, for your convenience, the names "add", "let", and "mult" are protec ted and will never be used as variable names.
Finally, there is the concept of scope. When an expression of a variable name is evaluated, within the context of that evaluation, the innermost scope (in terms of parentheses) is checked first for the value of that variable, and then oute r scopes are checked sequentially. It is guaranteed that every expression is legal. Please see the examples for more de tails on the scope.

Example 1:

Input: expression = "(let x 2 (mult x (let x 3 y 4 (add x y))))"
Output: 14
Explanation: In the expression (add x y), when checking for the value of the variable x,
we check from the innermost scope to the outermost in the context of the variable we are trying to evaluate.
Since x = 3 is found first, the value of x is 3.

Example 2:

Input: expression = "(let x 3 x 2 x)"
Output: 2
Explanation: Assignment in let statements is processed sequentially.

Example 3:

Input: expression = "(let x 1 y 2 x (add x y) (add x y))"
Output: 5
Explanation: The first (add x y) evaluates as 3, and is assigned to x.
The second (add x y) evaluates as 3+2 = 5.

Example 4:

Input: expression = "(let x 2 (add (let x 3 (let x 4 x)) x))"
Output: 6
Explanation: Even though (let x 4 x) has a deeper scope, it is outside the context
of the final x in the add-expression. That final x will equal 2.

Example 5:

Input: expression = "(let a1 3 b2 (add a1 1) b2)"
Output: 4
Explanation: Variable names can contain digits after the first character.

Constraints:

1 <= expression.length <= 2000
There are no leading or trailing spaces in exprssion.
All tokens are separated by a single space in expressoin.
The answer and all intermediate calculations of that answer are guaranteed to fit in a 32-bit integer.
The expression is guaranteed to be legal and evaluate to an integer.

---

738. Monotone Increasing Digits

An integer has monotone increasing digits if and only if each pair of adjacent digits x and y satisfy x <= y.
Given an integer n, return the largest number that is less than or equal to n with monotone increasing digits.

Example 1:

Input: n = 10
Output: 9

Example 2:

Input: n = 1234
Output: 1234

Example 3:

Input: n = 332
Output: 299

Constraints:

0 <= n <= 109

---

739. Daily Temperatures

Given an array of integers temperatures represents the daily temperatures, return an array answer such that answer[i] is the number of days you have to wait after the ith day to get a warmer temperature. If there is no future day for whi ch this is possible, keep answer[i] == 0 instead.

Example 1:
Input: temperatures = [73,74,75,71,69,72,76,73]
Output: [1,1,4,2,1,1,0,0]
Example 2:
Input: temperatures = [30,40,50,60]
Output: [1,1,1,0]
Example 3:
Input: temperatures = [30,60,90]
Output: [1,1,0]

Constraints:

1 <= temperatures.length <= 105
30 <= temperatures[i] <= 100

---

740. Delete and Earn

You are given an integer array nums. You want to maximize the number of points you get by performing the followi ng operation any number of times:

Pick any nums[i] and delete it to earn nums[i] points. Afterwards, you must delete every element equal to nums[i] - 1 and every element equal to nums[i] + 1.

Return the maximum number of points you can earn by applying the above operation some number of times.

Example 1:

Input: nums = [3,4,2]
Output: 6
Explanation: You can perform the following operations:
- Delete 4 to earn 4 points. Consequently, 3 is also deleted. nums = [2].
- Delete 2 to earn 2 points. nums = [].
You earn a total of 6 points.

Example 2:

Input: nums = [2,2,3,3,3,4]
Output: 9
Explanation: You can perform the following operations:
- Delete a 3 to earn 3 points. All 2's and 4's are also deleted. nums = [3,3].
- Delete a 3 again to earn 3 points. nums = [3].
- Delete a 3 once more to earn 3 points. nums = [].
You earn a total of 9 points.

Constraints:

1 <= nums.length <= 2 * 104
1 <= nums[i] <= 104

---

741. Cherry Pickup

You are given an n x n grid representing a field of cherries, each cell is one of three possible integers.

0 means the cell is empty, so you can pass through,
1 means the cell contains a cherry that you can pick up and pass through, or
-1 means the cell contains a thorn that blocks your way.

Return the maximum number of cherries you can collect by following the rules below:

Starting at the position (0, 0) and reaching (n - 1, n - 1) by moving right or down through valid path cells (cells with value 0 or 1).
After reaching (n - 1, n - 1), returning to (0, 0) by moving left or up through valid path cells.
When passing through a path cell containing a cherry, you pick it up, and the cell becomes an empty cell 0.
If there is no valid path between (0, 0) and (n - 1, n - 1), then no cherries can be collected.

Example 1:

Input: grid = [[0,1,-1],[1,0,-1],[1,1,1]]
Output: 5
Explanation: The player started at (0, 0) and went down, down, right right to reach (2, 2).
4 cherries were picked up during this single trip, and the matrix becomes [[0,1,-1],[0,0,-1],[0,0,0]].
Then, the player went left, up, up, left to return home, picking up one more cherry.
The total number of cherries picked up is 5, and this is the maximum possible.

Example 2:

Input: grid = [[1,1,-1],[1,-1,1],[-1,1,1]]
Output: 0

Constraints:

n == grid.length
n == grid[i].length
1 <= n <= 50
grid[i][j] is -1, 0, or 1.
grid[0][0] != -1
grid[n - 1][n - 1] != -1

---

743. Network Delay Time

You are given a network of n nodes, labeled from 1 to n. You are also given times, a list of travel times as directed e dges times[i] = (ui, vi, wi), where ui is the source node, vi is the target node, and wi is the time it takes for a signal to travel from source to target.
We will send a signal from a given node k. Return the time it takes for all the n nodes to receive the signal. If it is im possible for all the n nodes to receive the signal, return -1.

Example 1:

Input: times = [[2,1,1],[2,3,1],[3,4,1]], n = 4, k = 2
Output: 2

Example 2:

Input: times = [[1,2,1]], n = 2, k = 1
Output: 1

Example 3:

Input: times = [[1,2,1]], n = 2, k = 2
Output: -1

Constraints:

1 <= k <= n <= 100
1 <= times.length <= 6000
times[i].length == 3
1 <= ui, vi <= n
ui != vi
0 <= wi <= 100
All the pairs (ui, vi) are unique. (i.e., no multiple edges.)

---

744. Find Smallest Letter Greater Than Target

Given a characters array letters that is sorted in non-decreasing order and a character target, return the smallest chara
cter in the array that is larger than target.
Note that the letters wrap around.

For example, if target == 'z' and letters == ['a', 'b'], the answer is 'a'.

Example 1:

Input: letters = ["c","f","j"], target = "a"
Output: "c"

Example 2:

Input: letters = ["c","f","j"], target = "c"
Output: "f"

Example 3:

Input: letters = ["c","f","j"], target = "d"
Output: "f"

Example 4:

Input: letters = ["c","f","j"], target = "g"
Output: "j"

Example 5:

Input: letters = ["c","f","j"], target = "j"
Output: "c"

Constraints:

2 <= letters.length <= 104
letters[i] is a lowercase English letter.
letters is sorted in non-decreasing order.
letters contains at least two different characters.
target is a lowercase English letter.

---

745. Prefix and Suffix Search

Design a special dictionary with some words that searchs the words in it by a prefix and a suffix.
Implement the WordFilter class:

WordFilter(string[] words) Initializes the object with the words in the dictionary.
f(string prefix, string suffix) Returns the index of the word in the dictionary, which has the prefix prefix and the suffi
x suffix. If there is more than one valid index, return the largest of them. If there is no such word in the dictionary, re
turn -1.

Example 1:

Input
["WordFilter", "f"]
[[["apple"]], ["a", "e"]]
Output
[null, 0]

Explanation
WordFilter wordFilter = new WordFilter(["apple"]);
wordFilter.f("a", "e"); // return 0, because the word at index 0 has prefix = "a" and suffix = 'e".

Constraints:

1 <= words.length <= 15000
1 <= words[i].length <= 10
1 <= prefix.length, suffix.length <= 10
words[i], prefix and suffix consist of lower-case English letters only.
At most 15000 calls will be made to the function f.

---

746. Min Cost Climbing Stairs

You are given an integer array cost where cost[i] is the cost of ith step on a staircase. Once you pay the cost, you can
either climb one or two steps.
You can either start from the step with index 0, or the step with index 1.
Return the minimum cost to reach the top of the floor.

Example 1:

Input: cost = [10,15,20]
Output: 15

Explanation: You will start at index 1.
- Pay 15 and climb two steps to reach the top.
The total cost is 15.

Example 2:

Input: cost = [1,100,1,1,1,100,1,1,100,1]
Output: 6
Explanation: You will start at index 0.
- Pay 1 and climb two steps to reach index 2.
- Pay 1 and climb two steps to reach index 4.
- Pay 1 and climb two steps to reach index 6.
- Pay 1 and climb one step to reach index 7.
- Pay 1 and climb two steps to reach index 9.
- Pay 1 and climb one step to reach the top.
The total cost is 6.

Constraints:

2 <= cost.length <= 1000
0 <= cost[i] <= 999

---

747. Largest Number At Least Twice of Others

You are given an integer array nums where the largest integer is unique.
Determine whether the largest element in the array is at least twice as much as every other number in the array. If it i s, return the index of the largest element, or return -1 otherwise.

Example 1:

Input: nums = [3,6,1,0]
Output: 1
Explanation: 6 is the largest integer.
For every other number in the array x, 6 is at least twice as big as x.
The index of value 6 is 1, so we return 1.

Example 2:

Input: nums = [1,2,3,4]
Output: -1
Explanation: 4 is less than twice the value of 3, so we return -1.
Example 3:

Input: nums = [1]
Output: 0

Explanation: 1 is trivially at least twice the value as any other number because there are no other numbers.

Constraints:

1 <= nums.length <= 50
0 <= nums[i] <= 100
The largest element in nums is unique.

---

748. Shortest Completing Word

Given a string licensePlate and an array of strings words, find the shortest completing word in words.
A completing word is a word that contains all the letters in licensePlate. Ignore numbers and spaces in licensePlate, a nd treat letters as case insensitive. If a letter appears more than once in licensePlate, then it must appear in the word t he same number of times or more.
For example, if licensePlate = "aBc 12c", then it contains letters 'a', 'b' (ignoring case), and 'c' twice. Possible comple ting words are "abccdef", "caaacab", and "cbca".
Return the shortest completing word in words. It is guaranteed an answer exists. If there are multiple shortest comple ting words, return the first one that occurs in words.

Example 1:

Input: licensePlate = "1s3 PSt", words = ["step","steps","stripe","stepple"]
Output: "steps"
Explanation: licensePlate contains letters 's', 'p', 's' (ignoring case), and 't'.
"step" contains 't' and 'p', but only contains 1 's'.
"steps" contains 't', 'p', and both 's' characters.
"stripe" is missing an 's'.
"stepple" is missing an 's'.
Since "steps" is the only word containing all the letters, that is the answer.

Example 2:

Input: licensePlate = "1s3 456", words = ["looks","pest","stew","show"]
Output: "pest"
Explanation: licensePlate only contains the letter 's'. All the words contain 's', but among these "pest", "stew", and "s how" are shortest. The answer is "pest" because it is the word that appears earliest of the 3.

Example 3:

Input: licensePlate = "Ah71752", words = ["suggest","letter","of","husband","easy","education","drug","prevent","w riter","old"]
Output: "husband"

Example 4:

Input: licensePlate = "OgEu755", words = ["enough","these","play","wide","wonder","box","arrive","money","tax","thus"]
Output: "enough"

Example 5:

Input: licensePlate = "iMSlpe4", words = ["claim","consumer","student","camera","public","never","wonder","simple","thought","use"]
Output: "simple"

Constraints:

1 <= licensePlate.length <= 7
licensePlate contains digits, letters (uppercase or lowercase), or space ' '.
1 <= words.length <= 1000
1 <= words[i].length <= 15
words[i] consists of lower case English letters.

---

749. Contain Virus

A virus is spreading rapidly, and your task is to quarantine the infected area by installing walls.
The world is modeled as an m x n binary grid isInfected, where isInfected[i][j] == 0 represents uninfected cells, and isInfected[i][j] == 1 represents cells contaminated with the virus. A wall (and only one wall) can be installed between any two 4-directionally adjacent cells, on the shared boundary.
Every night, the virus spreads to all neighboring cells in all four directions unless blocked by a wall. Resources are limited. Each day, you can install walls around only one region (i.e., the affected area (continuous block of infected cells) that threatens the most uninfected cells the following night). There will never be a tie.
Return the number of walls used to quarantine all the infected regions. If the world will become fully infected, return the number of walls used.

Example 1:

Input: isInfected = [[0,1,0,0,0,0,0,1],[0,1,0,0,0,0,0,1],[0,0,0,0,0,0,0,1],[0,0,0,0,0,0,0,0]]
Output: 10
Explanation: There are 2 contaminated regions.
On the first day, add 5 walls to quarantine the viral region on the left. The board after the virus spreads is:

On the second day, add 5 walls to quarantine the viral region on the right. The virus is fully contained.

Example 2:

Input: isInfected = [[1,1,1],[1,0,1],[1,1,1]]

Output: 4
Explanation: Even though there is only one cell saved, there are 4 walls built.
Notice that walls are only built on the shared boundary of two different cells.

Example 3:

Input: isInfected = [[1,1,1,0,0,0,0,0,0],[1,0,1,0,1,1,1,1,1],[1,1,1,0,0,0,0,0,0]]
Output: 13
Explanation: The region on the left only builds two new walls.

Constraints:

m == isInfected.length
n == isInfected[i].length
1 <= m, n <= 50
isInfected[i][j] is either 0 or 1.
There is always a contiguous viral region throughout the described process that will infect strictly more uncontamina ted squares in the next round.

---

752. Open the Lock

You have a lock in front of you with 4 circular wheels. Each wheel has 10 slots: '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'. T he wheels can rotate freely and wrap around: for example we can turn '9' to be '0', or '0' to be '9'. Each move consists of turning one wheel one slot.
The lock initially starts at '0000', a string representing the state of the 4 wheels.
You are given a list of deadends dead ends, meaning if the lock displays any of these codes, the wheels of the lock w ill stop turning and you will be unable to open it.
Given a target representing the value of the wheels that will unlock the lock, return the minimum total number of tur ns required to open the lock, or -1 if it is impossible.

Example 1:

Input: deadends = ["0201","0101","0102","1212","2002"], target = "0202"
Output: 6
Explanation:
A sequence of valid moves would be "0000" -> "1000" -> "1100" -> "1200" -> "1201" -> "1202" -> "0202".
Note that a sequence like "0000" -> "0001" -> "0002" -> "0102" -> "0202" would be invalid,
because the wheels of the lock become stuck after the display becomes the dead end "0102".

Example 2:

Input: deadends = ["8888"], target = "0009"
Output: 1
Explanation:
We can turn the last wheel in reverse to move from "0000" -> "0009".

Example 3:

Input: deadends = ["8887","8889","8878","8898","8788","8988","7888","9888"], target = "8888"
Output: -1
Explanation:
We can't reach the target without getting stuck.

Example 4:

Input: deadends = ["0000"], target = "8888"
Output: -1

Constraints:

1 <= deadends.length <= 500
deadends[i].length == 4
target.length == 4
target will not be in the list deadends.
target and deadends[i] consist of digits only.

---

753. Cracking the Safe

There is a safe protected by a password. The password is a sequence of n digits where each digit can be in the range [0, k - 1].
The safe has a peculiar way of checking the password. When you enter in a sequence, it checks the most recent n dig its that were entered each time you type a digit.

For example, the correct password is "345" and you enter in "012345":

After typing 0, the most recent 3 digits is "0", which is incorrect.
After typing 1, the most recent 3 digits is "01", which is incorrect.
After typing 2, the most recent 3 digits is "012", which is incorrect.
After typing 3, the most recent 3 digits is "123", which is incorrect.
After typing 4, the most recent 3 digits is "234", which is incorrect.
After typing 5, the most recent 3 digits is "345", which is correct and the safe unlocks.

Return any string of minimum length that will unlock the safe at some point of entering it.

Example 1:

Input: n = 1, k = 2

Output: "10"
Explanation: The password is a single digit, so enter each digit. "01" would also unlock the safe.

Example 2:

Input: n = 2, k = 2
Output: "01100"
Explanation: For each possible password:
- "00" is typed in starting from the 4th digit.
- "01" is typed in starting from the 1st digit.
- "10" is typed in starting from the 3rd digit.
- "11" is typed in starting from the 2nd digit.
Thus "01100" will unlock the safe. "01100", "10011", and "11001" would also unlock the safe.

Constraints:

1 <= n <= 4
1 <= k <= 10
1 <= kn <= 4096

---

754. Reach a Number

You are standing at position 0 on an infinite number line. There is a destination at position target.
You can make some number of moves numMoves so that:

On each move, you can either go left or right.
During the ith move (starting from i == 1 to i == numMoves), you take i steps in the chosen direction.

Given the integer target, return the minimum number of moves required (i.e., the minimum numMoves) to reach the destination.

Example 1:

Input: target = 2
Output: 3
Explanation:
On the 1st move, we step from 0 to 1 (1 step).
On the 2nd move, we step from 1 to -1 (2 steps).
On the 3rd move, we step from -1 to 2 (3 steps).

Example 2:

Input: target = 3
Output: 2
Explanation:

On the 1st move, we step from 0 to 1 (1 step).
On the 2nd move, we step from 1 to 3 (2 steps).

Constraints:

-109 <= target <= 109
target != 0

---

756. Pyramid Transition Matrix

You are stacking blocks to form a pyramid. Each block has a color, which is represented by a single letter. Each row of blocks contains one less block than the row beneath it and is centered on top.
To make the pyramid aesthetically pleasing, there are only specific triangular patterns that are allowed. A triangular pattern consists of a single block stacked on top of two blocks. The patterns are given as a list of three-letter strings a llowed, where the first two characters of a pattern represent the left and right bottom blocks respectively, and the thir d character is the top block.

For example, "ABC" represents a triangular pattern with a 'C' block stacked on top of an 'A' (left) and 'B' (right) bloc k. Note that this is different from "BAC" where 'B' is on the left bottom and 'A' is on the right bottom.

You start with a bottom row of blocks bottom, given as a single string, that you must use as the base of the pyramid. Given bottom and allowed, return true if you can build the pyramid all the way to the top such that every triangular p attern in the pyramid is in allowed, or false otherwise.

Example 1:

Input: bottom = "BCD", allowed = ["BCC","CDE","CEA","FFF"]
Output: true
Explanation: The allowed triangular patterns are shown on the right.
Starting from the bottom (level 3), we can build "CE" on level 2 and then build "E" on level 1.
There are three triangular patterns in the pyramid, which are "BCC", "CDE", and "CEA". All are allowed.

Example 2:

Input: bottom = "AAAA", allowed = ["AAB","AAC","BCD","BBE","DEF"]
Output: false
Explanation: The allowed triangular patterns are shown on the right.
Starting from the bottom (level 4), there are multiple ways to build level 3, but trying all the possibilites, you will get always stuck before building level 1.

Constraints:

2 <= bottom.length <= 6
0 <= allowed.length <= 216
allowed[i].length == 3
The letters in all input strings are from the set {'A', 'B', 'C', 'D', 'E', 'F'}.
All the values of allowed are unique.

---

757. Set Intersection Size At Least Two

An integer interval [a, b] (for integers a < b) is a set of all consecutive integers from a to b, including a and b.
Find the minimum size of a set S such that for every integer interval A in intervals, the intersection of S with A has a size of at least two.

Example 1:

Input: intervals = [[1,3],[1,4],[2,5],[3,5]]
Output: 3
Explanation: Consider the set S = {2, 3, 4}. For each interval, there are at least 2 elements from S in the interval.
Also, there isn't a smaller size set that fulfills the above condition.
Thus, we output the size of this set, which is 3.

Example 2:

Input: intervals = [[1,2],[2,3],[2,4],[4,5]]
Output: 5
Explanation: An example of a minimum sized set is {1, 2, 3, 4, 5}.

Constraints:

1 <= intervals.length <= 3000
intervals[i].length == 2
0 <= ai < bi <= 108

---

761. Special Binary String

Special binary strings are binary strings with the following two properties:

The number of 0's is equal to the number of 1's.

Every prefix of the binary string has at least as many 1's as 0's.

You are given a special binary string s.
A move consists of choosing two consecutive, non-empty, special substrings of s, and swapping them. Two strings a re consecutive if the last character of the first string is exactly one index before the first character of the second strin g.
Return the lexicographically largest resulting string possible after applying the mentioned operations on the string.

Example 1:

Input: s = "11011000"
Output: "11100100"
Explanation: The strings "10" [occuring at s[1]] and "1100" [at s[3]] are swapped.
This is the lexicographically largest string possible after some number of swaps.

Example 2:

Input: s = "10"
Output: "10"

Constraints:

1 <= s.length <= 50
s[i] is either '0' or '1'.
s is a special binary string.

---

762. Prime Number of Set Bits in Binary Representation

Given two integers left and right, return the count of numbers in the inclusive range [left, right] having a prime num ber of set bits in their binary representation.
Recall that the number of set bits an integer has is the number of 1's present when written in binary.

For example, 21 written in binary is 10101, which has 3 set bits.

Example 1:

Input: left = 6, right = 10
Output: 4
Explanation:
6 -> 110 (2 set bits, 2 is prime)
7 -> 111 (3 set bits, 3 is prime)
8 -> 1000 (1 set bit, 1 is not prime)
9 -> 1001 (2 set bits, 2 is prime)
10 -> 1010 (2 set bits, 2 is prime)

4 numbers have a prime number of set bits.

Example 2:

Input: left = 10, right = 15
Output: 5
Explanation:
10 -> 1010 (2 set bits, 2 is prime)
11 -> 1011 (3 set bits, 3 is prime)
12 -> 1100 (2 set bits, 2 is prime)
13 -> 1101 (3 set bits, 3 is prime)
14 -> 1110 (3 set bits, 3 is prime)
15 -> 1111 (4 set bits, 4 is not prime)
5 numbers have a prime number of set bits.

Constraints:

1 <= left <= right <= 106
0 <= right - left <= 104

---

763. Partition Labels

You are given a string s. We want to partition the string into as many parts as possible so that each letter appears in a t most one part.
Return a list of integers representing the size of these parts.

Example 1:

Input: s = "ababcbacadefegdehijhklij"
Output: [9,7,8]
Explanation:
The partition is "ababcbaca", "defegde", "hijhklij".
This is a partition so that each letter appears in at most one part.
A partition like "ababcbacadefegde", "hijhklij" is incorrect, because it splits s into less parts.

Example 2:

Input: s = "eccbbbbdec"
Output: [10]

Constraints:

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

---

764. Largest Plus Sign

You are given an integer n. You have an n x n binary grid grid with all values initially 1's except for some indices gi ven in the array mines. The ith element of the array mines is defined as mines[i] = [xi, yi] where grid[xi][yi] == 0.
Return the order of the largest axis-aligned plus sign of 1's contained in grid. If there is none, return 0.
An axis-aligned plus sign of 1's of order k has some center grid[r][c] == 1 along with four arms of length k - 1 going up, down, left, and right, and made of 1's. Note that there could be 0's or 1's beyond the arms of the plus sign, only th e relevant area of the plus sign is checked for 1's.

Example 1:

Input: n = 5, mines = [[4,2]]
Output: 2
Explanation: In the above grid, the largest plus sign can only be of order 2. One of them is shown.

Example 2:

Input: n = 1, mines = [[0,0]]
Output: 0
Explanation: There is no plus sign, so return 0.

Constraints:

1 <= n <= 500
1 <= mines.length <= 5000
0 <= xi, yi < n
All the pairs (xi, yi) are unique.

---

765. Couples Holding Hands

There are n couples sitting in 2n seats arranged in a row and want to hold hands.
The people and seats are represented by an integer array row where row[i] is the ID of the person sitting in the ith se at. The couples are numbered in order, the first couple being (0, 1), the second couple being (2, 3), and so on with th e last couple being (2n - 2, 2n - 1).

Return the minimum number of swaps so that every couple is sitting side by side. A swap consists of choosing any t
wo people, then they stand up and switch seats.

Example 1:

Input: row = [0,2,1,3]
Output: 1
Explanation: We only need to swap the second (row[1]) and third (row[2]) person.

Example 2:

Input: row = [3,2,0,1]
Output: 0
Explanation: All couples are already seated side by side.

Constraints:

2n == row.length
2 <= n <= 30
n is even.
0 <= row[i] < 2n
All the elements of row are unique.

---

766. Toeplitz Matrix

Given an m x n matrix, return true if the matrix is Toeplitz. Otherwise, return false.
A matrix is Toeplitz if every diagonal from top-left to bottom-right has the same elements.

Example 1:

Input: matrix = [[1,2,3,4],[5,1,2,3],[9,5,1,2]]
Output: true
Explanation:
In the above grid, the diagonals are:
"[9]", "[5, 5]", "[1, 1, 1]", "[2, 2, 2]", "[3, 3]", "[4]".
In each diagonal all elements are the same, so the answer is True.

Example 2:

Input: matrix = [[1,2],[2,2]]
Output: false
Explanation:
The diagonal "[1, 2]" has different elements.

Constraints:

m == matrix.length
n == matrix[i].length
1 <= m, n <= 20
0 <= matrix[i][j] <= 99

Follow up:

What if the matrix is stored on disk, and the memory is limited such that you can only load at most one row of the m atrix into the memory at once?
What if the matrix is so large that you can only load up a partial row into the memory at once?

---

767. Reorganize String

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.

---

768. Max Chunks To Make Sorted II

You are given an integer array arr.
We split arr into some number of chunks (i.e., partitions), and individually sort each chunk. After concatenating the

m, the result should equal the sorted array.
Return the largest number of chunks we can make to sort the array.

Example 1:

Input: arr = [5,4,3,2,1]
Output: 1
Explanation:
Splitting into two or more chunks will not return the required result.
For example, splitting into [5, 4], [3, 2, 1] will result in [4, 5, 1, 2, 3], which isn't sorted.

Example 2:

Input: arr = [2,1,3,4,4]
Output: 4
Explanation:
We can split into two chunks, such as [2, 1], [3, 4, 4].
However, splitting into [2, 1], [3], [4], [4] is the highest number of chunks possible.

Constraints:

1 <= arr.length <= 2000
0 <= arr[i] <= 108

---

769. Max Chunks To Make Sorted

You are given an integer array arr of length n that represents a permutation of the integers in the range [0, n - 1].
We split arr into some number of chunks (i.e., partitions), and individually sort each chunk. After concatenating the m, the result should equal the sorted array.
Return the largest number of chunks we can make to sort the array.

Example 1:

Input: arr = [4,3,2,1,0]
Output: 1
Explanation:
Splitting into two or more chunks will not return the required result.
For example, splitting into [4, 3], [2, 1, 0] will result in [3, 4, 0, 1, 2], which isn't sorted.

Example 2:

Input: arr = [1,0,2,3,4]
Output: 4
Explanation:
We can split into two chunks, such as [1, 0], [2, 3, 4].

However, splitting into [1, 0], [2], [3], [4] is the highest number of chunks possible.

Constraints:

n == arr.length
1 <= n <= 10
0 <= arr[i] < n
All the elements of arr are unique.

---

770. Basic Calculator IV

Given an expression such as expression = "e + 8 - a + 5" and an evaluation map such as {"e": 1} (given in terms of e valvars = ["e"] and evalints = [1]), return a list of tokens representing the simplified expression, such as ["-1*a","14"]

An expression alternates chunks and symbols, with a space separating each chunk and symbol.
A chunk is either an expression in parentheses, a variable, or a non-negative integer.
A variable is a string of lowercase letters (not including digits.) Note that variables can be multiple letters, and note t hat variables never have a leading coefficient or unary operator like "2x" or "-x".

Expressions are evaluated in the usual order: brackets first, then multiplication, then addition and subtraction.

For example, expression = "1 + 2 * 3" has an answer of ["7"].

The format of the output is as follows:

For each term of free variables with a non-zero coefficient, we write the free variables within a term in sorted order l exicographically.

For example, we would never write a term like "b*a*c", only "a*b*c".

Terms have degrees equal to the number of free variables being multiplied, counting multiplicity. We write the large st degree terms of our answer first, breaking ties by lexicographic order ignoring the leading coefficient of the term.

For example, "a*a*b*c" has degree 4.

The leading coefficient of the term is placed directly to the left with an asterisk separating it from the variables (if th ey exist.) A leading coefficient of 1 is still printed.
An example of a well-formatted answer is ["-2*a*a*a", "3*a*a*b", "3*b*b", "4*a", "5*c", "-6"].
Terms (including constant terms) with coefficient 0 are not included.

For example, an expression of "0" has an output of [].

Example 1:

Input: expression = "e + 8 - a + 5", evalvars = ["e"], evalints = [1]
Output: ["-1*a","14"]

Example 2:

Input: expression = "e - 8 + temperature - pressure", evalvars = ["e", "temperature"], evalints = [1, 12]
Output: ["-1*pressure","5"]

Example 3:

Input: expression = "(e + 8) * (e - 8)", evalvars = [], evalints = []
Output: ["1*e*e","-64"]

Example 4:

Input: expression = "a * b * c + b * a * c * 4", evalvars = [], evalints = []
Output: ["5*a*b*c"]

Example 5:

Input: expression = "((a - b) * (b - c) + (c - a)) * ((a - b) + (b - c) * (c - a))", evalvars = [], evalints = []
Output: ["-1*a*a*b*b","2*a*a*b*c","-1*a*a*c*c","1*a*b*b*b","-1*a*b*b*c","-1*a*b*c*c","1*a*c*c*c","-1*b*b*b*c","2*b*b*c*c","-1*b*c*c*c","2*a*a*b","-2*a*a*c","-2*a*b*b","2*a*c*c","1*b*b*b","-1*b*b*c","1*b*c*c","-1*c*c*c","-1*a*a","1*a*b","1*a*c","-1*b*c"]

Constraints:

1 <= expression.length <= 250
expression consists of lowercase English letters, digits, '+', '-', '*', '(', ')', ' '.
expression does not contain any leading or trailing spaces.
All the tokens in expression are separated by a single space.
0 <= evalvars.length <= 100
1 <= evalvars[i].length <= 20
evalvars[i] consists of lowercase English letters.
evalints.length == evalvars.length
-100 <= evalints[i] <= 100

---

771. Jewels and Stones

You're given strings jewels representing the types of stones that are jewels, and stones representing the stones you ha

ve. Each character in stones is a type of stone you have. You want to know how many of the stones you have are als o jewels.
Letters are case sensitive, so "a" is considered a different type of stone from "A".

Example 1:
Input: jewels = "aA", stones = "aAAbbbb"
Output: 3
Example 2:
Input: jewels = "z", stones = "ZZ"
Output: 0

Constraints:

1 <= jewels.length, stones.length <= 50
jewels and stones consist of only English letters.
All the characters of jewels are unique.

---

773. Sliding Puzzle

On an 2 x 3 board, there are five tiles labeled from 1 to 5, and an empty square represented by 0. A move consists of choosing 0 and a 4-directionally adjacent number and swapping it.
The state of the board is solved if and only if the board is [[1,2,3],[4,5,0]].
Given the puzzle board board, return the least number of moves required so that the state of the board is solved. If it is impossible for the state of the board to be solved, return -1.

Example 1:

Input: board = [[1,2,3],[4,0,5]]
Output: 1
Explanation: Swap the 0 and the 5 in one move.

Example 2:

Input: board = [[1,2,3],[5,4,0]]
Output: -1
Explanation: No number of moves will make the board solved.

Example 3:

Input: board = [[4,1,2],[5,0,3]]
Output: 5
Explanation: 5 is the smallest number of moves that solves the board.

An example path:
After move 0: [[4,1,2],[5,0,3]]
After move 1: [[4,1,2],[0,5,3]]
After move 2: [[0,1,2],[4,5,3]]
After move 3: [[1,0,2],[4,5,3]]
After move 4: [[1,2,0],[4,5,3]]
After move 5: [[1,2,3],[4,5,0]]

Example 4:

Input: board = [[3,2,4],[1,5,0]]
Output: 14

Constraints:

board.length == 2
board[i].length == 3
0 <= board[i][j] <= 5
Each value board[i][j] is unique.

---

775. Global and Local Inversions

You are given an integer array nums of length n which represents a permutation of all the integers in the range [0, n - 1].
The number of global inversions is the number of the different pairs (i, j) where:

0 <= i < j < n
nums[i] > nums[j]

The number of local inversions is the number of indices i where:

0 <= i < n - 1
nums[i] > nums[i + 1]

Return true if the number of global inversions is equal to the number of local inversions.

Example 1:

Input: nums = [1,0,2]
Output: true
Explanation: There is 1 global inversion and 1 local inversion.

Example 2:

Input: nums = [1,2,0]
Output: false
Explanation: There are 2 global inversions and 1 local inversion.

Constraints:

n == nums.length
1 <= n <= 105
0 <= nums[i] < n
All the integers of nums are unique.
nums is a permutation of all the numbers in the range [0, n - 1].

---

777. Swap Adjacent in LR String

In a string composed of 'L', 'R', and 'X' characters, like "RXXLRXRXL", a move consists of either replacing one occ urrence of "XL" with "LX", or replacing one occurrence of "RX" with "XR". Given the starting string start and the e nding string end, return True if and only if there exists a sequence of moves to transform one string to the other.

Example 1:

Input: start = "RXXLRXRXL", end = "XRLXXRRLX"
Output: true
Explanation: We can transform start to end following these steps:
RXXLRXRXL ->
XRXLRXRXL ->
XRLXRXRXL ->
XRLXXRRXL ->
XRLXXRRLX

Example 2:

Input: start = "X", end = "L"
Output: false

Example 3:

Input: start = "LLR", end = "RRL"
Output: false

Example 4:

Input: start = "XL", end = "LX"
Output: true

Example 5:

Input: start = "XLLR", end = "LXLX"
Output: false

Constraints:

1 <= start.length <= 104
start.length == end.length
Both start and end will only consist of characters in 'L', 'R', and 'X'.

---

778. Swim in Rising Water

You are given an n x n integer matrix grid where each value grid[i][j] represents the elevation at that point (i, j).
The rain starts to fall. At time t, the depth of the water everywhere is t. You can swim from a square to another 4-dir ectionally adjacent square if and only if the elevation of both squares individually are at most t. You can swim infinit e distances in zero time. Of course, you must stay within the boundaries of the grid during your swim.
Return the least time until you can reach the bottom right square (n - 1, n - 1) if you start at the top left square (0, 0).

Example 1:

Input: grid = [[0,2],[1,3]]
Output: 3
Explanation:
At time 0, you are in grid location (0, 0).
You cannot go anywhere else because 4-directionally adjacent neighbors have a higher elevation than t = 0.
You cannot reach point (1, 1) until time 3.
When the depth of water is 3, we can swim anywhere inside the grid.

Example 2:

Input: grid = [[0,1,2,3,4],[24,23,22,21,5],[12,13,14,15,16],[11,17,18,19,20],[10,9,8,7,6]]
Output: 16
Explanation: The final route is shown.
We need to wait until time 16 so that (0, 0) and (4, 4) are connected.

Constraints:

n == grid.length
n == grid[i].length
1 <= n <= 50
0 <= grid[i][j] < n2
Each value grid[i][j] is unique.

---

779. K-th Symbol in Grammar

We build a table of n rows (1-indexed). We start by writing 0 in the 1st row. Now in every subsequent row, we look at the previous row and replace each occurrence of 0 with 01, and each occurrence of 1 with 10.

For example, for n = 3, the 1st row is 0, the 2nd row is 01, and the 3rd row is 0110.

Given two integer n and k, return the kth (1-indexed) symbol in the nth row of a table of n rows.

Example 1:

Input: n = 1, k = 1
Output: 0
Explanation: row 1: 0

Example 2:

Input: n = 2, k = 1
Output: 0
Explanation:
row 1: 0
row 2: 01

Example 3:

Input: n = 2, k = 2
Output: 1
Explanation:
row 1: 0
row 2: 01

Example 4:

Input: n = 3, k = 1
Output: 0
Explanation:
row 1: 0
row 2: 01
row 3: 0110

Constraints:

1 <= n <= 30
1 <= k <= 2n - 1

---

780. Reaching Points

Given four integers sx, sy, tx, and ty, return true if it is possible to convert the point (sx, sy) to the point (tx, ty) throu gh some operations, or false otherwise.
The allowed operation on some point (x, y) is to convert it to either (x, x + y) or (x + y, y).

Example 1:

Input: sx = 1, sy = 1, tx = 3, ty = 5
Output: true
Explanation:
One series of moves that transforms the starting point to the target is:
(1, 1) -> (1, 2)
(1, 2) -> (3, 2)
(3, 2) -> (3, 5)

Example 2:

Input: sx = 1, sy = 1, tx = 2, ty = 2
Output: false

Example 3:

Input: sx = 1, sy = 1, tx = 1, ty = 1
Output: true

Constraints:

1 <= sx, sy, tx, ty <= 109

---

781. Rabbits in Forest

There is a forest with an unknown number of rabbits. We asked n rabbits "How many rabbits have the same color as you?" and collected the answers in an integer array answers where answers[i] is the answer of the ith rabbit.
Given the array answers, return the minimum number of rabbits that could be in the forest.

Example 1:

Input: answers = [1,1,2]
Output: 5
Explanation:
The two rabbits that answered "1" could both be the same color, say red.
The rabbit that answered "2" can't be red or the answers would be inconsistent.
Say the rabbit that answered "2" was blue.
Then there should be 2 other blue rabbits in the forest that didn't answer into the array.
The smallest possible number of rabbits in the forest is therefore 5: 3 that answered plus 2 that didn't.

Example 2:

Input: answers = [10,10,10]
Output: 11

Constraints:

1 <= answers.length <= 1000
0 <= answers[i] < 1000

---

782. Transform to Chessboard

You are given an n x n binary grid board. In each move, you can swap any two rows with each other, or any two col
umns with each other.
Return the minimum number of moves to transform the board into a chessboard board. If the task is impossible, retur
n -1.
A chessboard board is a board where no 0's and no 1's are 4-directionally adjacent.

Example 1:

Input: board = [[0,1,1,0],[0,1,1,0],[1,0,0,1],[1,0,0,1]]
Output: 2
Explanation: One potential sequence of moves is shown.
The first move swaps the first and second column.
The second move swaps the second and third row.

Example 2:

Input: board = [[0,1],[1,0]]
Output: 0
Explanation: Also note that the board with 0 in the top left corner, is also a valid chessboard.

Example 3:

Input: board = [[1,0],[1,0]]
Output: -1
Explanation: No matter what sequence of moves you make, you cannot end with a valid chessboard.

Constraints:

n == board.length
n == board[i].length
2 <= n <= 30
board[i][j] is either 0 or 1.

---

783. Minimum Distance Between BST Nodes

Given the root of a Binary Search Tree (BST), return the minimum difference between the values of any two differe nt nodes in the tree.

Example 1:

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

Example 2:

Input: root = [1,0,48,null,null,12,49]
Output: 1

Constraints:

The number of nodes in the tree is in the range [2, 100].
0 <= Node.val <= 105

Note: This question is the same as 530: https://leetcode.com/problems/minimum-absolute-difference-in-bst/

---

784. Letter Case Permutation

Given a string s, we can transform every letter individually to be lowercase or uppercase to create another string. Return a list of all possible strings we could create. You can return the output in any order.

Example 1:

Input: s = "a1b2"
Output: ["a1b2","a1B2","A1b2","A1B2"]

Example 2:

Input: s = "3z4"
Output: ["3z4","3Z4"]

Example 3:

Input: s = "12345"
Output: ["12345"]

Example 4:

Input: s = "0"
Output: ["0"]

Constraints:

s will be a string with length between 1 and 12.
s will consist only of letters or digits.

---

785. Is Graph Bipartite?

There is an undirected graph with n nodes, where each node is numbered between 0 and n - 1. You are given a 2D ar ray graph, where graph[u] is an array of nodes that node u is adjacent to. More formally, for each v in graph[u], there is an undirected edge between node u and node v. The graph has the following properties:

There are no self-edges (graph[u] does not contain u).
There are no parallel edges (graph[u] does not contain duplicate values).
If v is in graph[u], then u is in graph[v] (the graph is undirected).
The graph may not be connected, meaning there may be two nodes u and v such that there is no path between them.

A graph is bipartite if the nodes can be partitioned into two independent sets A and B such that every edge in the gra ph connects a node in set A and a node in set B.

Return true if and only if it is bipartite.

Example 1:

Input: graph = [[1,2,3],[0,2],[0,1,3],[0,2]]
Output: false
Explanation: There is no way to partition the nodes into two independent sets such that every edge connects a node i n one and a node in the other.
Example 2:

Input: graph = [[1,3],[0,2],[1,3],[0,2]]
Output: true
Explanation: We can partition the nodes into two sets: {0, 2} and {1, 3}.

Constraints:

graph.length == n
1 <= n <= 100
0 <= graph[u].length < n
0 <= graph[u][i] <= n - 1
graph[u] does not contain u.
All the values of graph[u] are unique.
If graph[u] contains v, then graph[v] contains u.

---

786. K-th Smallest Prime Fraction

You are given a sorted integer array arr containing 1 and prime numbers, where all the integers of arr are unique. Yo u are also given an integer k.
For every i and j where 0 <= i < j < arr.length, we consider the fraction arr[i] / arr[j].
Return the kth smallest fraction considered. Return your answer as an array of integers of size 2, where answer[0] == arr[i] and answer[1] == arr[j].

Example 1:

Input: arr = [1,2,3,5], k = 3
Output: [2,5]
Explanation: The fractions to be considered in sorted order are:
1/5, 1/3, 2/5, 1/2, 3/5, and 2/3.
The third fraction is 2/5.

Example 2:

Input: arr = [1,7], k = 1
Output: [1,7]

Constraints:

2 <= arr.length <= 1000
1 <= arr[i] <= 3 * 104
arr[0] == 1
arr[i] is a prime number for i > 0.
All the numbers of arr are unique and sorted in strictly increasing order.
1 <= k <= arr.length * (arr.length - 1) / 2

---

787. Cheapest Flights Within K Stops

There are n cities connected by some number of flights. You are given an array flights where flights[i] = [fromi, toi, pricei] indicates that there is a flight from city fromi to city toi with cost pricei.
You are also given three integers src, dst, and k, return the cheapest price from src to dst with at most k stops. If ther e is no such route, return -1.

Example 1:

Input: n = 3, flights = [[0,1,100],[1,2,100],[0,2,500]], src = 0, dst = 2, k = 1
Output: 200
Explanation: The graph is shown.
The cheapest price from city 0 to city 2 with at most 1 stop costs 200, as marked red in the picture.

Example 2:

Input: n = 3, flights = [[0,1,100],[1,2,100],[0,2,500]], src = 0, dst = 2, k = 0
Output: 500
Explanation: The graph is shown.
The cheapest price from city 0 to city 2 with at most 0 stop costs 500, as marked blue in the picture.

Constraints:

1 <= n <= 100
0 <= flights.length <= (n * (n - 1) / 2)
flights[i].length == 3
0 <= fromi, toi < n
fromi != toi
1 <= pricei <= 104
There will not be any multiple flights between two cities.
0 <= src, dst, k < n
src != dst

---

788. Rotated Digits

An integer x is a good if after rotating each digit individually by 180 degrees, we get a valid number that is different from x. Each digit must be rotated - we cannot choose to leave it alone.
A number is valid if each digit remains a digit after rotation. For example:

0, 1, and 8 rotate to themselves,
2 and 5 rotate to each other (in this case they are rotated in a different direction, in other words, 2 or 5 gets mirrored),
6 and 9 rotate to each other, and
the rest of the numbers do not rotate to any other number and become invalid.

Given an integer n, return the number of good integers in the range [1, n].

Example 1:

Input: n = 10
Output: 4
Explanation: There are four good numbers in the range [1, 10] : 2, 5, 6, 9.
Note that 1 and 10 are not good numbers, since they remain unchanged after rotating.

Example 2:

Input: n = 1
Output: 0

Example 3:

Input: n = 2
Output: 1

Constraints:

1 <= n <= 104

---

789. Escape The Ghosts

You are playing a simplified PAC-MAN game on an infinite 2-D grid. You start at the point [0, 0], and you are give n a destination point target = [xtarget, ytarget] that you are trying to get to. There are several ghosts on the map with their starting positions given as a 2D array ghosts, where ghosts[i] = [xi, yi] represents the starting position of the ith ghost. All inputs are integral coordinates.
Each turn, you and all the ghosts may independently choose to either move 1 unit in any of the four cardinal directio ns: north, east, south, or west, or stay still. All actions happen simultaneously.
You escape if and only if you can reach the target before any ghost reaches you. If you reach any square (including t he target) at the same time as a ghost, it does not count as an escape.
Return true if it is possible to escape regardless of how the ghosts move, otherwise return false.

Example 1:

Input: ghosts = [[1,0],[0,3]], target = [0,1]
Output: true
Explanation: You can reach the destination (0, 1) after 1 turn, while the ghosts located at (1, 0) and (0, 3) cannot catc h up with you.

Example 2:

Input: ghosts = [[1,0]], target = [2,0]
Output: false
Explanation: You need to reach the destination (2, 0), but the ghost at (1, 0) lies between you and the destination.

Example 3:

Input: ghosts = [[2,0]], target = [1,0]
Output: false
Explanation: The ghost can reach the target at the same time as you.

Example 4:

Input: ghosts = [[5,0],[-10,-2],[0,-5],[-2,-2],[-7,1]], target = [7,7]
Output: false

Example 5:

Input: ghosts = [[-1,0],[0,1],[-1,0],[0,1],[-1,0]], target = [0,0]
Output: true

Constraints:

1 <= ghosts.length <= 100
ghosts[i].length == 2
-104 <= xi, yi <= 104
There can be multiple ghosts in the same location.
target.length == 2
-104 <= xtarget, ytarget <= 104

---

790. Domino and Tromino Tiling

You have two types of tiles: a 2 x 1 domino shape and a tromino shape. You may rotate these shapes.

Given an integer n, return the number of ways to tile an 2 x n board. Since the answer may be very large, return it m odulo 109 + 7.
In a tiling, every square must be covered by a tile. Two tilings are different if and only if there are two 4-directionall y adjacent cells on the board such that exactly one of the tilings has both squares occupied by a tile.

Example 1:

Input: n = 3
Output: 5
Explanation: The five different ways are show above.

Example 2:

Input: n = 1
Output: 1

Constraints:

1 <= n <= 1000

---

791. Custom Sort String

You are given two strings order and s. All the words of order are unique and were sorted in some custom order previ ously.
Permute the characters of s so that they match the order that order was sorted. More specifically, if a character x occ urs before a character y in order, then x should occur before y in the permuted string.
Return any permutation of s that satisfies this property.

Example 1:

Input: order = "cba", s = "abcd"
Output: "cbad"
Explanation:
"a", "b", "c" appear in order, so the order of "a", "b", "c" should be "c", "b", and "a".
Since "d" does not appear in order, it can be at any position in the returned string. "dcba", "cdba", "cbda" are also val id outputs.

Example 2:

Input: order = "cbafg", s = "abcd"
Output: "cbad"

Constraints:

1 <= order.length <= 26
1 <= s.length <= 200
order and s consist of lowercase English letters.
All the characters of order are unique.

---

792. Number of Matching Subsequences

Given a string s and an array of strings words, return the number of words[i] that is a subsequence of s.
A subsequence of a string is a new string generated from the original string with some characters (can be none) delet ed without changing the relative order of the remaining characters.

For example, "ace" is a subsequence of "abcde".

Example 1:

Input: s = "abcde", words = ["a","bb","acd","ace"]
Output: 3
Explanation: There are three strings in words that are a subsequence of s: "a", "acd", "ace".

Example 2:

Input: s = "dsahjpjauf", words = ["ahjpjau","ja","ahbwzgqnuk","tnmlanowax"]
Output: 2

Constraints:

1 <= s.length <= 5 * 104
1 <= words.length <= 5000
1 <= words[i].length <= 50
s and words[i] consist of only lowercase English letters.

---

793. Preimage Size of Factorial Zeroes Function

Let f(x) be the number of zeroes at the end of x!. Recall that x! = 1 * 2 * 3 * ... * x and by convention, 0! = 1.

For example, f(3) = 0 because 3! = 6 has no zeroes at the end, while f(11) = 2 because 11! = 39916800 has two zeroe s at the end.

Given an integer k, return the number of non-negative integers x have the property that f(x) = k.

Example 1:

Input: k = 0
Output: 5
Explanation: 0!, 1!, 2!, 3!, and 4! end with k = 0 zeroes.

Example 2:

Input: k = 5
Output: 0
Explanation: There is no x such that x! ends in k = 5 zeroes.

Example 3:

Input: k = 3
Output: 5

Constraints:

0 <= k <= 109

---

794. Valid Tic-Tac-Toe State

Given a Tic-Tac-Toe board as a string array board, return true if and only if it is possible to reach this board position during the course of a valid tic-tac-toe game.
The board is a 3 x 3 array that consists of characters ' ', 'X', and 'O'. The ' ' character represents an empty square.
Here are the rules of Tic-Tac-Toe:

Players take turns placing characters into empty squares ' '.
The first player always places 'X' characters, while the second player always places 'O' characters.
'X' and 'O' characters are always placed into empty squares, never filled ones.
The game ends when there are three of the same (non-empty) character filling any row, column, or diagonal.

The game also ends if all squares are non-empty.
No more moves can be played if the game is over.

Example 1:

Input: board = ["O "," "," "]
Output: false
Explanation: The first player always plays "X".

Example 2:

Input: board = ["XOX"," X "," "]
Output: false
Explanation: Players take turns making moves.

Example 3:

Input: board = ["XXX"," ","OOO"]
Output: false

Example 4:

Input: board = ["XOX","O O","XOX"]
Output: true

Constraints:

board.length == 3
board[i].length == 3
board[i][j] is either 'X', 'O', or ' '.

---

795. Number of Subarrays with Bounded Maximum

Given an integer array nums and two integers left and right, return the number of contiguous non-empty subarrays su ch that the value of the maximum array element in that subarray is in the range [left, right].
The test cases are generated so that the answer will fit in a 32-bit integer.

Example 1:

Input: nums = [2,1,4,3], left = 2, right = 3

Output: 3
Explanation: There are three subarrays that meet the requirements: [2], [2, 1], [3].

Example 2:

Input: nums = [2,9,2,5,6], left = 2, right = 8
Output: 7

Constraints:

1 <= nums.length <= 105
0 <= nums[i] <= 109
0 <= left <= right <= 109

---

796. Rotate String

Given two strings s and goal, return true if and only if s can become goal after some number of shifts on s.
A shift on s consists of moving the leftmost character of s to the rightmost position.

For example, if s = "abcde", then it will be "bcdea" after one shift.

Example 1:
Input: s = "abcde", goal = "cdeab"
Output: true
Example 2:
Input: s = "abcde", goal = "abced"
Output: false

Constraints:

1 <= s.length, goal.length <= 100
s and goal consist of lowercase English letters.

---

797. All Paths From Source to Target

Given a directed acyclic graph (DAG) of n nodes labeled from 0 to n - 1, find all possible paths from node 0 to node n - 1 and return them in any order.
The graph is given as follows: graph[i] is a list of all nodes you can visit from node i (i.e., there is a directed edge fro m node i to node graph[i][j]).

Example 1:

Input: graph = [[1,2],[3],[3],[]]
Output: [[0,1,3],[0,2,3]]
Explanation: There are two paths: 0 -> 1 -> 3 and 0 -> 2 -> 3.

Example 2:

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

Example 3:

Input: graph = [[1],[]]
Output: [[0,1]]

Example 4:

Input: graph = [[1,2,3],[2],[3],[]]
Output: [[0,1,2,3],[0,2,3],[0,3]]

Example 5:

Input: graph = [[1,3],[2],[3],[]]
Output: [[0,1,2,3],[0,3]]

Constraints:

n == graph.length
2 <= n <= 15
0 <= graph[i][j] < n
graph[i][j] != i (i.e., there will be no self-loops).
All the elements of graph[i] are unique.
The input graph is guaranteed to be a DAG.

---

798. Smallest Rotation with Highest Score

You are given an array nums. You can rotate it by a non-negative integer k so that the array becomes [nums[k], num

s[k + 1], ... nums[nums.length - 1], nums[0], nums[1], ..., nums[k-1]]. Afterward, any entries that are less than or equ al to their index are worth one point.

For example, if we have nums = [2,4,1,3,0], and we rotate by k = 2, it becomes [1,3,0,2,4]. This is worth 3 points be cause 1 > 0 [no points], 3 > 1 [no points], 0 <= 2 [one point], 2 <= 3 [one point], 4 <= 4 [one point].

Return the rotation index k that corresponds to the highest score we can achieve if we rotated nums by it. If there are multiple answers, return the smallest such index k.

Example 1:

Input: nums = [2,3,1,4,0]
Output: 3
Explanation: Scores for each k are listed below:
k = 0, nums = [2,3,1,4,0],   score 2
k = 1, nums = [3,1,4,0,2],   score 3
k = 2, nums = [1,4,0,2,3],   score 3
k = 3, nums = [4,0,2,3,1],   score 4
k = 4, nums = [0,2,3,1,4],   score 3
So we should choose k = 3, which has the highest score.

Example 2:

Input: nums = [1,3,0,2,4]
Output: 0
Explanation: nums will always have 3 points no matter how it shifts.
So we will choose the smallest k, which is 0.

Constraints:

1 <= nums.length <= 105
0 <= nums[i] < nums.length

---

799. Champagne Tower

We stack glasses in a pyramid, where the first row has 1 glass, the second row has 2 glasses, and so on until the 100t h row. Each glass holds one cup of champagne.
Then, some champagne is poured into the first glass at the top. When the topmost glass is full, any excess liquid pou red will fall equally to the glass immediately to the left and right of it. When those glasses become full, any excess c hampagne will fall equally to the left and right of those glasses, and so on. (A glass at the bottom row has its excess champagne fall on the floor.)
For example, after one cup of champagne is poured, the top most glass is full. After two cups of champagne are pou red, the two glasses on the second row are half full. After three cups of champagne are poured, those two cups beco me full - there are 3 full glasses total now. After four cups of champagne are poured, the third row has the middle gl ass half full, and the two outside glasses are a quarter full, as pictured below.

Now after pouring some non-negative integer cups of champagne, return how full the jth glass in the ith row is (both i and j are 0-indexed.)

Example 1:

Input: poured = 1, query_row = 1, query_glass = 1
Output: 0.00000
Explanation: We poured 1 cup of champange to the top glass of the tower (which is indexed as (0, 0)). There will be no excess liquid so all the glasses under the top glass will remain empty.

Example 2:

Input: poured = 2, query_row = 1, query_glass = 1
Output: 0.50000
Explanation: We poured 2 cups of champange to the top glass of the tower (which is indexed as (0, 0)). There is one cup of excess liquid. The glass indexed as (1, 0) and the glass indexed as (1, 1) will share the excess liquid equally, a nd each will get half cup of champange.

Example 3:

Input: poured = 100000009, query_row = 33, query_glass = 17
Output: 1.00000

Constraints:

0 <= poured <= 109
0 <= query_glass <= query_row < 100

---

801. Minimum Swaps To Make Sequences Increasing

You are given two integer arrays of the same length nums1 and nums2. In one operation, you are allowed to swap nu ms1[i] with nums2[i].

For example, if nums1 = [1,2,3,8], and nums2 = [5,6,7,4], you can swap the element at i = 3 to obtain nums1 = [1,2,3 ,4] and nums2 = [5,6,7,8].

Return the minimum number of needed operations to make nums1 and nums2 strictly increasing. The test cases are g enerated so that the given input always makes it possible.
An array arr is strictly increasing if and only if arr[0] < arr[1] < arr[2] < ... < arr[arr.length - 1].

Example 1:

Input: nums1 = [1,3,5,4], nums2 = [1,2,3,7]
Output: 1
Explanation:

Swap nums1[3] and nums2[3]. Then the sequences are:
nums1 = [1, 3, 5, 7] and nums2 = [1, 2, 3, 4]
which are both strictly increasing.

Example 2:

Input: nums1 = [0,3,5,8,9], nums2 = [2,1,4,6,9]
Output: 1

Constraints:

2 <= nums1.length <= 105
nums2.length == nums1.length
0 <= nums1[i], nums2[i] <= 2 * 105

---

802. Find Eventual Safe States

There is a directed graph of n nodes with each node labeled from 0 to n - 1. The graph is represented by a 0-indexed 2D integer array graph where graph[i] is an integer array of nodes adjacent to node i, meaning there is an edge from node i to each node in graph[i].
A node is a terminal node if there are no outgoing edges. A node is a safe node if every possible path starting from th at node leads to a terminal node.
Return an array containing all the safe nodes of the graph. The answer should be sorted in ascending order.

Example 1:

Input: graph = [[1,2],[2,3],[5],[0],[5],[],[]]
Output: [2,4,5,6]
Explanation: The given graph is shown above.
Nodes 5 and 6 are terminal nodes as there are no outgoing edges from either of them.
Every path starting at nodes 2, 4, 5, and 6 all lead to either node 5 or 6.
Example 2:

Input: graph = [[1,2,3,4],[1,2],[3,4],[0,4],[]]
Output: [4]
Explanation:
Only node 4 is a terminal node, and every path starting at node 4 leads to node 4.

Constraints:

n == graph.length
1 <= n <= 104
0 <= graph[i].length <= n

0 <= graph[i][j] <= n - 1
graph[i] is sorted in a strictly increasing order.
The graph may contain self-loops.
The number of edges in the graph will be in the range [1, 4 * 104].

---

803. Bricks Falling When Hit

You are given an m x n binary grid, where each 1 represents a brick and 0 represents an empty space. A brick is stab le if:

It is directly connected to the top of the grid, or
At least one other brick in its four adjacent cells is stable.

You are also given an array hits, which is a sequence of erasures we want to apply. Each time we want to erase the b rick at the location hits[i] = (rowi, coli). The brick on that location (if it exists) will disappear. Some other bricks ma y no longer be stable because of that erasure and will fall. Once a brick falls, it is immediately erased from the grid (i .e., it does not land on other stable bricks).
Return an array result, where each result[i] is the number of bricks that will fall after the ith erasure is applied.
Note that an erasure may refer to a location with no brick, and if it does, no bricks drop.

Example 1:

Input: grid = [[1,0,0,0],[1,1,1,0]], hits = [[1,0]]
Output: [2]
Explanation: Starting with the grid:
[[1,0,0,0],
 [1,1,1,0]]
We erase the underlined brick at (1,0), resulting in the grid:
[[1,0,0,0],
 [0,1,1,0]]
The two underlined bricks are no longer stable as they are no longer connected to the top nor adjacent to another stab le brick, so they will fall. The resulting grid is:
[[1,0,0,0],
 [0,0,0,0]]
Hence the result is [2].

Example 2:

Input: grid = [[1,0,0,0],[1,1,0,0]], hits = [[1,1],[1,0]]
Output: [0,0]
Explanation: Starting with the grid:
[[1,0,0,0],
 [1,1,0,0]]
We erase the underlined brick at (1,1), resulting in the grid:
[[1,0,0,0],
 [1,0,0,0]]

All remaining bricks are still stable, so no bricks fall. The grid remains the same:
[[1,0,0,0],
 [1,0,0,0]]
Next, we erase the underlined brick at (1,0), resulting in the grid:
[[1,0,0,0],
 [0,0,0,0]]
Once again, all remaining bricks are still stable, so no bricks fall.
Hence the result is [0,0].

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 200
grid[i][j] is 0 or 1.
1 <= hits.length <= 4 * 104
hits[i].length == 2
0 <= xi <= m - 1
0 <= yi <= n - 1
All (xi, yi) are unique.

---

804. Unique Morse Code Words

International Morse Code defines a standard encoding where each letter is mapped to a series of dots and dashes, as f ollows:

'a' maps to ".-",
'b' maps to "-...",
'c' maps to "-.-.", and so on.

For convenience, the full table for the 26 letters of the English alphabet is given below:

[".-","-...","-.-.","-..",".","..-.","--.","....","..",".---","-.-",".-..","--","-.","---",".--.","--.-",".-.","...","-","..-","...-",".--","-..-","-.--","--.."]
Given an array of strings words where each word can be written as a concatenation of the Morse code of each letter.

For example, "cab" can be written as "-.-..--...", which is the concatenation of "-.-.", ".-", and "-...". We will call such a concatenation the transformation of a word.

Return the number of different transformations among all words we have.

Example 1:

Input: words = ["gin","zen","gig","msg"]
Output: 2

Explanation: The transformation of each word is:
"gin" -> "--...-."
"zen" -> "--...-."
"gig" -> "--...--."
"msg" -> "--...--."
There are 2 different transformations: "--...-." and "--...--.".

Example 2:

Input: words = ["a"]
Output: 1

Constraints:

1 <= words.length <= 100
1 <= words[i].length <= 12
words[i] consists of lowercase English letters.

---

805. Split Array With Same Average

You are given an integer array nums.
You should move each element of nums into one of the two arrays A and B such that A and B are non-empty, and av erage(A) == average(B).
Return true if it is possible to achieve that and false otherwise.
Note that for an array arr, average(arr) is the sum of all the elements of arr over the length of arr.

Example 1:

Input: nums = [1,2,3,4,5,6,7,8]
Output: true
Explanation: We can split the array into [1,4,5,8] and [2,3,6,7], and both of them have an average of 4.5.

Example 2:

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

Constraints:

1 <= nums.length <= 30
0 <= nums[i] <= 104

---

806. Number of Lines To Write String

You are given a string s of lowercase English letters and an array widths denoting how many pixels wide each lower case English letter is. Specifically, widths[0] is the width of 'a', widths[1] is the width of 'b', and so on.
You are trying to write s across several lines, where each line is no longer than 100 pixels. Starting at the beginning of s, write as many letters on the first line such that the total width does not exceed 100 pixels. Then, from where yo u stopped in s, continue writing as many letters as you can on the second line. Continue this process until you have written all of s.
Return an array result of length 2 where:

result[0] is the total number of lines.
result[1] is the width of the last line in pixels.

Example 1:

Input: widths = [10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10], s = "abcdefghijkl mnopqrstuvwxyz"
Output: [3,60]
Explanation: You can write s as follows:
abcdefghij // 100 pixels wide
klmnopqrst // 100 pixels wide
uvwxyz   // 60 pixels wide
There are a total of 3 lines, and the last line is 60 pixels wide.
Example 2:

Input: widths = [4,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10], s = "bbbcccdddaaa "
Output: [2,4]
Explanation: You can write s as follows:
bbbcccdddaa // 98 pixels wide
a   // 4 pixels wide
There are a total of 2 lines, and the last line is 4 pixels wide.

Constraints:

widths.length == 26
2 <= widths[i] <= 10
1 <= s.length <= 1000
s contains only lowercase English letters.

---

807. Max Increase to Keep City Skyline

There is a city composed of n x n blocks, where each block contains a single building shaped like a vertical square prism. You are given a 0-indexed n x n integer matrix grid where grid[r][c] represents the height of the building located in the block at row r and column c.
A city's skyline is the the outer contour formed by all the building when viewing the side of the city from a distance. The skyline from each cardinal direction north, east, south, and west may be different.
We are allowed to increase the height of any number of buildings by any amount (the amount can be different per building). The height of a 0-height building can also be increased. However, increasing the height of a building should not affect the city's skyline from any cardinal direction.
Return the maximum total sum that the height of the buildings can be increased by without changing the city's skyline from any cardinal direction.

Example 1:

Input: grid = [[3,0,8,4],[2,4,5,7],[9,2,6,3],[0,3,1,0]]
Output: 35
Explanation: The building heights are shown in the center of the above image.
The skylines when viewed from each cardinal direction are drawn in red.
The grid after increasing the height of buildings without affecting skylines is:
gridNew = [[8, 4, 8, 7],
[7, 4, 7, 7],
[9, 4, 8, 7],
[3, 3, 3, 3]]

Example 2:

Input: grid = [[0,0,0],[0,0,0],[0,0,0]]
Output: 0
Explanation: Increasing the height of any building will result in the skyline changing.

Constraints:

n == grid.length
n == grid[r].length
2 <= n <= 50
0 <= grid[r][c] <= 100

---

808. Soup Servings

There are two types of soup: type A and type B. Initially, we have n ml of each type of soup. There are four kinds of operations:

Serve 100 ml of soup A and 0 ml of soup B,
Serve 75 ml of soup A and 25 ml of soup B,
Serve 50 ml of soup A and 50 ml of soup B, and
Serve 25 ml of soup A and 75 ml of soup B.

When we serve some soup, we give it to someone, and we no longer have it. Each turn, we will choose from the four operations with an equal probability 0.25. If the remaining volume of soup is not enough to complete the operation, we will serve as much as possible. We stop once we no longer have some quantity of both types of soup.
Note that we do not have an operation where all 100 ml's of soup B are used first.
Return the probability that soup A will be empty first, plus half the probability that A and B become empty at the sa me time. Answers within 10-5 of the actual answer will be accepted.

Example 1:

Input: n = 50
Output: 0.62500
Explanation: If we choose the first two operations, A will become empty first.
For the third operation, A and B will become empty at the same time.
For the fourth operation, B will become empty first.
So the total probability of A becoming empty first plus half the probability that A and B become empty at the same ti me, is 0.25 * (1 + 1 + 0.5 + 0) = 0.625.

Example 2:

Input: n = 100
Output: 0.71875

Constraints:

0 <= n <= 109

---

809. Expressive Words

Sometimes people repeat letters to represent extra feeling. For example:

"hello" -> "heeellooo"
"hi" -> "hiiii"

In these strings like "heeellooo", we have groups of adjacent letters that are all the same: "h", "eee", "ll", "ooo".
You are given a string s and an array of query strings words. A query word is stretchy if it can be made to be equal t o s by any number of applications of the following extension operation: choose a group consisting of characters c, an d add some number of characters c to the group so that the size of the group is three or more.

For example, starting with "hello", we could do an extension on the group "o" to get "hellooo", but we cannot get "h elloo" since the group "oo" has a size less than three. Also, we could do another extension like "ll" -> "lllll" to get "h

ellllooo". If s = "hellllooo", then the query word "hello" would be stretchy because of these two extension operation s: query = "hello" -> "hellooo" -> "hellllooo" = s.

Return the number of query strings that are stretchy.

Example 1:

Input: s = "heeellooo", words = ["hello", "hi", "helo"]
Output: 1
Explanation:
We can extend "e" and "o" in the word "hello" to get "heeellooo".
We can't extend "helo" to get "heeellooo" because the group "ll" is not size 3 or more.

Example 2:

Input: s = "zzzzzyyyyy", words = ["zzyy","zy","zyy"]
Output: 3

Constraints:

1 <= s.length, words.length <= 100
1 <= words[i].length <= 100
s and words[i] consist of lowercase letters.

---

810. Chalkboard XOR Game

You are given an array of integers nums represents the numbers written on a chalkboard.
Alice and Bob take turns erasing exactly one number from the chalkboard, with Alice starting first. If erasing a num ber causes the bitwise XOR of all the elements of the chalkboard to become 0, then that player loses. The bitwise X OR of one element is that element itself, and the bitwise XOR of no elements is 0.
Also, if any player starts their turn with the bitwise XOR of all the elements of the chalkboard equal to 0, then that pl ayer wins.
Return true if and only if Alice wins the game, assuming both players play optimally.

Example 1:

Input: nums = [1,1,2]
Output: false
Explanation:
Alice has two choices: erase 1 or erase 2.
If she erases 1, the nums array becomes [1, 2]. The bitwise XOR of all the elements of the chalkboard is 1 XOR 2 = 3. Now Bob can remove any element he wants, because Alice will be the one to erase the last element and she will lo se.
If Alice erases 2 first, now nums become [1, 1]. The bitwise XOR of all the elements of the chalkboard is 1 XOR 1 = 0. Alice will lose.

Example 2:

Input: nums = [0,1]
Output: true

Example 3:

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

Constraints:

1 <= nums.length <= 1000
0 <= nums[i] < 216

---

811. Subdomain Visit Count

A website domain "discuss.leetcode.com" consists of various subdomains. At the top level, we have "com", at the ne xt level, we have "leetcode.com" and at the lowest level, "discuss.leetcode.com". When we visit a domain like "discu ss.leetcode.com", we will also visit the parent domains "leetcode.com" and "com" implicitly.
A count-paired domain is a domain that has one of the two formats "rep d1.d2.d3" or "rep d1.d2" where rep is the nu mber of visits to the domain and d1.d2.d3 is the domain itself.

For example, "9001 discuss.leetcode.com" is a count-paired domain that indicates that discuss.leetcode.com was visi ted 9001 times.

Given an array of count-paired domains cpdomains, return an array of the count-paired domains of each subdomain i n the input. You may return the answer in any order.

Example 1:

Input: cpdomains = ["9001 discuss.leetcode.com"]
Output: ["9001 leetcode.com","9001 discuss.leetcode.com","9001 com"]
Explanation: We only have one website domain: "discuss.leetcode.com".
As discussed above, the subdomain "leetcode.com" and "com" will also be visited. So they will all be visited 9001 ti mes.

Example 2:

Input: cpdomains = ["900 google.mail.com", "50 yahoo.com", "1 intel.mail.com", "5 wiki.org"]
Output: ["901 mail.com","50 yahoo.com","900 google.mail.com","5 wiki.org","5 org","1 intel.mail.com","951 com"]
Explanation: We will visit "google.mail.com" 900 times, "yahoo.com" 50 times, "intel.mail.com" once and "wiki.org " 5 times.

For the subdomains, we will visit "mail.com" 900 + 1 = 901 times, "com" 900 + 50 + 1 = 951 times, and "org" 5 tim
es.

Constraints:

1 <= cpdomain.length <= 100
1 <= cpdomain[i].length <= 100
cpdomain[i] follows either the "repi d1i.d2i.d3i" format or the "repi d1i.d2i" format.
repi is an integer in the range [1, 104].
d1i, d2i, and d3i consist of lowercase English letters.

---

812. Largest Triangle Area

Given an array of points on the X-Y plane points where points[i] = [xi, yi], return the area of the largest triangle that can be formed by any three different points. Answers within 10-5 of the actual answer will be accepted.

Example 1:

Input: points = [[0,0],[0,1],[1,0],[0,2],[2,0]]
Output: 2.00000
Explanation: The five points are shown in the above figure. The red triangle is the largest.

Example 2:

Input: points = [[1,0],[0,0],[0,1]]
Output: 0.50000

Constraints:

3 <= points.length <= 50
-50 <= xi, yi <= 50
All the given points are unique.

---

813. Largest Sum of Averages

You are given an integer array nums and an integer k. You can partition the array into at most k non-empty adjacent subarrays. The score of a partition is the sum of the averages of each subarray.
Note that the partition must use every integer in nums, and that the score is not necessarily an integer.
Return the maximum score you can achieve of all the possible partitions. Answers within 10-6 of the actual answer will be accepted.

Example 1:

Input: nums = [9,1,2,3,9], k = 3
Output: 20.00000
Explanation:
The best choice is to partition nums into [9], [1, 2, 3], [9]. The answer is 9 + (1 + 2 + 3) / 3 + 9 = 20.
We could have also partitioned nums into [9, 1], [2], [3, 9], for example.
That partition would lead to a score of 5 + 2 + 6 = 13, which is worse.

Example 2:

Input: nums = [1,2,3,4,5,6,7], k = 4
Output: 20.50000

Constraints:

1 <= nums.length <= 100
1 <= nums[i] <= 104
1 <= k <= nums.length

---

814. Binary Tree Pruning

Given the root of a binary tree, return the same tree where every subtree (of the given tree) not containing a 1 has be en removed.
A subtree of a node node is node plus every node that is a descendant of node.

Example 1:

Input: root = [1,null,0,0,1]
Output: [1,null,0,null,1]
Explanation:
Only the red nodes satisfy the property "every subtree not containing a 1".
The diagram on the right represents the answer.

Example 2:

Input: root = [1,0,1,0,0,0,1]

Output: [1,null,1,null,1]

Example 3:

Input: root = [1,1,0,1,1,0,1,0]
Output: [1,1,0,1,1,null,1]

Constraints:

The number of nodes in the tree is in the range [1, 200].
Node.val is either 0 or 1.

---

815. Bus Routes

You are given an array routes representing bus routes where routes[i] is a bus route that the ith bus repeats forever.

For example, if routes[0] = [1, 5, 7], this means that the 0th bus travels in the sequence 1 -> 5 -> 7 -> 1 -> 5 -> 7 -> 1 -> ... forever.

You will start at the bus stop source (You are not on any bus initially), and you want to go to the bus stop target. You can travel between bus stops by buses only.
Return the least number of buses you must take to travel from source to target. Return -1 if it is not possible.

Example 1:

Input: routes = [[1,2,7],[3,6,7]], source = 1, target = 6
Output: 2
Explanation: The best strategy is take the first bus to the bus stop 7, then take the second bus to the bus stop 6.

Example 2:

Input: routes = [[7,12],[4,5,15],[6],[15,19],[9,12,13]], source = 15, target = 12
Output: -1

Constraints:

1 <= routes.length <= 500.
1 <= routes[i].length <= 105
All the values of routes[i] are unique.
sum(routes[i].length) <= 105
0 <= routes[i][j] < 106
0 <= source, target < 106

---

816. Ambiguous Coordinates

We had some 2-dimensional coordinates, like "(1, 3)" or "(2, 0.5)". Then, we removed all commas, decimal points, a nd spaces and ended up with the string s.

For example, "(1, 3)" becomes s = "(13)" and "(2, 0.5)" becomes s = "(205)".

Return a list of strings representing all possibilities for what our original coordinates could have been.
Our original representation never had extraneous zeroes, so we never started with numbers like "00", "0.0", "0.00", " 1.0", "001", "00.01", or any other number that can be represented with fewer digits. Also, a decimal point within a n umber never occurs without at least one digit occurring before it, so we never started with numbers like ".1".
The final answer list can be returned in any order. All coordinates in the final answer have exactly one space betwee n them (occurring after the comma.)

Example 1:

Input: s = "(123)"
Output: ["(1, 2.3)","(1, 23)","(1.2, 3)","(12, 3)"]

Example 2:

Input: s = "(0123)"
Output: ["(0, 1.23)","(0, 12.3)","(0, 123)","(0.1, 2.3)","(0.1, 23)","(0.12, 3)"]
Explanation: 0.0, 00, 0001 or 00.01 are not allowed.

Example 3:

Input: s = "(00011)"
Output: ["(0, 0.011)","(0.001, 1)"]

Example 4:

Input: s = "(100)"
Output: ["(10, 0)"]
Explanation: 1.0 is not allowed.

Constraints:

4 <= s.length <= 12
s[0] == '(' and s[s.length - 1] == ')'.
The rest of s are digits.

---

817. Linked List Components

You are given the head of a linked list containing unique integer values and an integer array nums that is a subset of the linked list values.
Return the number of connected components in nums where two values are connected if they appear consecutively i n the linked list.

Example 1:

Input: head = [0,1,2,3], nums = [0,1,3]
Output: 2
Explanation: 0 and 1 are connected, so [0, 1] and [3] are the two connected components.

Example 2:

Input: head = [0,1,2,3,4], nums = [0,3,1,4]
Output: 2
Explanation: 0 and 1 are connected, 3 and 4 are connected, so [0, 1] and [3, 4] are the two connected components.

Constraints:

The number of nodes in the linked list is n.
1 <= n <= 104
0 <= Node.val < n
All the values Node.val are unique.
1 <= nums.length <= n
0 <= nums[i] < n
All the values of nums are unique.

---

818. Race Car

Your car starts at position 0 and speed +1 on an infinite number line. Your car can go into negative positions. Your c ar drives automatically according to a sequence of instructions 'A' (accelerate) and 'R' (reverse):

When you get an instruction 'A', your car does the following:

position += speed

speed *= 2

When you get an instruction 'R', your car does the following:

If your speed is positive then speed = -1
otherwise speed = 1

Your position stays the same.

For example, after commands "AAR", your car goes to positions 0 --> 1 --> 3 --> 3, and your speed goes to 1 --> 2 --> 4 --> -1.
Given a target position target, return the length of the shortest sequence of instructions to get there.

Example 1:

Input: target = 3
Output: 2
Explanation:
The shortest instruction sequence is "AA".
Your position goes from 0 --> 1 --> 3.

Example 2:

Input: target = 6
Output: 5
Explanation:
The shortest instruction sequence is "AAARA".
Your position goes from 0 --> 1 --> 3 --> 7 --> 7 --> 6.

Constraints:

1 <= target <= 104

---

819. Most Common Word

Given a string paragraph and a string array of the banned words banned, return the most frequent word that is not ba nned. It is guaranteed there is at least one word that is not banned, and that the answer is unique.
The words in paragraph are case-insensitive and the answer should be returned in lowercase.

Example 1:

Input: paragraph = "Bob hit a ball, the hit BALL flew far after it was hit.", banned = ["hit"]
Output: "ball"
Explanation:

"hit" occurs 3 times, but it is a banned word.
"ball" occurs twice (and no other word does), so it is the most frequent non-banned word in the paragraph.
Note that words in the paragraph are not case sensitive,
that punctuation is ignored (even if adjacent to words, such as "ball,"),
and that "hit" isn't the answer even though it occurs more because it is banned.

Example 2:

Input: paragraph = "a.", banned = []
Output: "a"

Constraints:

1 <= paragraph.length <= 1000
paragraph consists of English letters, space ' ', or one of the symbols: "!?',;.".
0 <= banned.length <= 100
1 <= banned[i].length <= 10
banned[i] consists of only lowercase English letters.

---

820. Short Encoding of Words

A valid encoding of an array of words is any reference string s and array of indices indices such that:

words.length == indices.length
The reference string s ends with the '#' character.
For each index indices[i], the substring of s starting from indices[i] and up to (but not including) the next '#' characte
r is equal to words[i].

Given an array of words, return the length of the shortest reference string s possible of any valid encoding of words.

Example 1:

Input: words = ["time", "me", "bell"]
Output: 10
Explanation: A valid encoding would be s = "time#bell#" and indices = [0, 2, 5].
words[0] = "time", the substring of s starting from indices[0] = 0 to the next '#' is underlined in "time#bell#"
words[1] = "me", the substring of s starting from indices[1] = 2 to the next '#' is underlined in "time#bell#"
words[2] = "bell", the substring of s starting from indices[2] = 5 to the next '#' is underlined in "time#bell#"

Example 2:

Input: words = ["t"]
Output: 2
Explanation: A valid encoding would be s = "t#" and indices = [0].

Constraints:

1 <= words.length <= 2000
1 <= words[i].length <= 7
words[i] consists of only lowercase letters.

---

821. Shortest Distance to a Character

Given a string s and a character c that occurs in s, return an array of integers answer where answer.length == s.length and answer[i] is the distance from index i to the closest occurrence of character c in s.
The distance between two indices i and j is abs(i - j), where abs is the absolute value function.

Example 1:

Input: s = "loveleetcode", c = "e"
Output: [3,2,1,0,1,0,0,1,2,2,1,0]
Explanation: The character 'e' appears at indices 3, 5, 6, and 11 (0-indexed).
The closest occurrence of 'e' for index 0 is at index 3, so the distance is abs(0 - 3) = 3.
The closest occurrence of 'e' for index 1 is at index 3, so the distance is abs(1 - 3) = 2.
For index 4, there is a tie between the 'e' at index 3 and the 'e' at index 5, but the distance is still the same: abs(4 - 3) == abs(4 - 5) = 1.
The closest occurrence of 'e' for index 8 is at index 6, so the distance is abs(8 - 6) = 2.

Example 2:

Input: s = "aaab", c = "b"
Output: [3,2,1,0]

Constraints:

1 <= s.length <= 104
s[i] and c are lowercase English letters.
It is guaranteed that c occurs at least once in s.

---

822. Card Flipping Game

You are given n cards, with a positive integer printed on the front and back of each card (possibly different). You ca
n flip any number of cards (possibly zero).
After choosing the front and the back of each card, you will pick each card, and if the integer printed on the back of t
his card is not printed on the front of any other card, then this integer is good.
You are given two integer array fronts and backs where fronts[i] and backs[i] are the integers printer on the front and
 the back of the ith card respectively.
Return the smallest good and integer after flipping the cards. If there are no good integers, return 0.
Note that a flip swaps the front and back numbers, so the value on the front is now on the back and vice versa.

Example 1:

Input: fronts = [1,2,4,4,7], backs = [1,3,4,1,3]
Output: 2
Explanation: If we flip the second card, the fronts are [1,3,4,4,7] and the backs are [1,2,4,1,3].
We choose the second card, which has the number 2 on the back, and it is not on the front of any card, so 2 is good.

Example 2:

Input: fronts = [1], backs = [1]
Output: 0

Constraints:

n == fronts.length
n == backs.length
1 <= n <= 1000
1 <= fronts[i], backs[i] <= 2000

---

823. Binary Trees With Factors

Given an array of unique integers, arr, where each integer arr[i] is strictly greater than 1.
We make a binary tree using these integers, and each number may be used for any number of times. Each non-leaf n
ode's value should be equal to the product of the values of its children.
Return the number of binary trees we can make. The answer may be too large so return the answer modulo 109 + 7.

Example 1:

Input: arr = [2,4]
Output: 3
Explanation: We can make these trees: [2], [4], [4, 2, 2]
Example 2:

Input: arr = [2,4,5,10]
Output: 7
Explanation: We can make these trees: [2], [4], [5], [10], [4, 2, 2], [10, 2, 5], [10, 5, 2].

Constraints:

1 <= arr.length <= 1000
2 <= arr[i] <= 109
All the values of arr are unique.

---

824. Goat Latin

You are given a string sentence that consist of words separated by spaces. Each word consists of lowercase and uppe rcase letters only.
We would like to convert the sentence to "Goat Latin" (a made-up language similar to Pig Latin.) The rules of Goat Latin are as follows:

If a word begins with a vowel ('a', 'e', 'i', 'o', or 'u'), append "ma" to the end of the word.

For example, the word "apple" becomes "applema".

If a word begins with a consonant (i.e., not a vowel), remove the first letter and append it to the end, then add "ma".

For example, the word "goat" becomes "oatgma".

Add one letter 'a' to the end of each word per its word index in the sentence, starting with 1.

For example, the first word gets "a" added to the end, the second word gets "aa" added to the end, and so on.

Return the final sentence representing the conversion from sentence to Goat Latin.

Example 1:
Input: sentence = "I speak Goat Latin"
Output: "Imaa peaksmaaa oatGmaaaa atinLmaaaaa"
Example 2:
Input: sentence = "The quick brown fox jumped over the lazy dog"
Output: "heTmaa uickqmaaa rownbmaaaa oxfmaaaaa umpedjmaaaaaa overmaaaaaaa hetmaaaaaaaa azylmaaaaaaaaa ogdmaaaaaaaaaa"

Constraints:

1 <= sentence.length <= 150
sentence consists of English letters and spaces.

sentence has no leading or trailing spaces.
All the words in sentence are separated by a single space.

---

825. Friends Of Appropriate Ages

There are n persons on a social media website. You are given an integer array ages where ages[i] is the age of the ith person.
A Person x will not send a friend request to a person y (x != y) if any of the following conditions is true:

age[y] <= 0.5 * age[x] + 7
age[y] > age[x]
age[y] > 100 && age[x] < 100

Otherwise, x will send a friend request to y.
Note that if x sends a request to y, y will not necessarily send a request to x. Also, a person will not send a friend req uest to themself.
Return the total number of friend requests made.

Example 1:

Input: ages = [16,16]
Output: 2
Explanation: 2 people friend request each other.

Example 2:

Input: ages = [16,17,18]
Output: 2
Explanation: Friend requests are made 17 -> 16, 18 -> 17.

Example 3:

Input: ages = [20,30,100,110,120]
Output: 3
Explanation: Friend requests are made 110 -> 100, 120 -> 110, 120 -> 100.

Constraints:

n == ages.length
1 <= n <= 2 * 104
1 <= ages[i] <= 120

---

826. Most Profit Assigning Work

You have n jobs and m workers. You are given three arrays: difficulty, profit, and worker where:

difficulty[i] and profit[i] are the difficulty and the profit of the ith job, and
worker[j] is the ability of jth worker (i.e., the jth worker can only complete a job with difficulty at most worker[j]).

Every worker can be assigned at most one job, but one job can be completed multiple times.

For example, if three workers attempt the same job that pays $1, then the total profit will be $3. If a worker cannot c
omplete any job, their profit is $0.

Return the maximum profit we can achieve after assigning the workers to the jobs.

Example 1:

Input: difficulty = [2,4,6,8,10], profit = [10,20,30,40,50], worker = [4,5,6,7]
Output: 100
Explanation: Workers are assigned jobs of difficulty [4,4,6,6] and they get a profit of [20,20,30,30] separately.

Example 2:

Input: difficulty = [85,47,57], profit = [24,66,99], worker = [40,25,25]
Output: 0

Constraints:

n == difficulty.length
n == profit.length
m == worker.length
1 <= n, m <= 104
1 <= difficulty[i], profit[i], worker[i] <= 105

---

827. Making A Large Island

You are given an n x n binary matrix grid. You are allowed to change at most one 0 to be 1.
Return the size of the largest island in grid after applying this operation.
An island is a 4-directionally connected group of 1s.

Example 1:

Input: grid = [[1,0],[0,1]]
Output: 3
Explanation: Change one 0 to 1 and connect two 1s, then we get an island with area = 3.

Example 2:

Input: grid = [[1,1],[1,0]]
Output: 4
Explanation: Change the 0 to 1 and make the island bigger, only one island with area = 4.
Example 3:

Input: grid = [[1,1],[1,1]]
Output: 4
Explanation: Can't change any 0 to 1, only one island with area = 4.

Constraints:

n == grid.length
n == grid[i].length
1 <= n <= 500
grid[i][j] is either 0 or 1.

---

828. Count Unique Characters of All Substrings of a Given String

Let's define a function countUniqueChars(s) that returns the number of unique characters on s.

For example if s = "LEETCODE" then "L", "T", "C", "O", "D" are the unique characters since they appear only once in s, therefore countUniqueChars(s) = 5.

Given a string s, return the sum of countUniqueChars(t) where t is a substring of s.
Notice that some substrings can be repeated so in this case you have to count the repeated ones too.

Example 1:

Input: s = "ABC"
Output: 10
Explanation: All possible substrings are: "A","B","C","AB","BC" and "ABC".
Evey substring is composed with only unique letters.
Sum of lengths of all substring is 1 + 1 + 1 + 2 + 2 + 3 = 10

Example 2:

Input: s = "ABA"
Output: 8
Explanation: The same as example 1, except countUniqueChars("ABA") = 1.

Example 3:

Input: s = "LEETCODE"
Output: 92

Constraints:

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

---

829. Consecutive Numbers Sum

Given an integer n, return the number of ways you can write n as the sum of consecutive positive integers.

Example 1:

Input: n = 5
Output: 2
Explanation: 5 = 2 + 3

Example 2:

Input: n = 9
Output: 3
Explanation: 9 = 4 + 5 = 2 + 3 + 4

Example 3:

Input: n = 15
Output: 4
Explanation: 15 = 8 + 7 = 4 + 5 + 6 = 1 + 2 + 3 + 4 + 5

Constraints:

1 <= n <= 109

---

830. Positions of Large Groups

In a string s of lowercase letters, these letters form consecutive groups of the same character.
For example, a string like s = "abbxxxxzyy" has the groups "a", "bb", "xxxx", "z", and "yy".
A group is identified by an interval [start, end], where start and end denote the start and end indices (inclusive) of the group. In the above example, "xxxx" has the interval [3,6].
A group is considered large if it has 3 or more characters.
Return the intervals of every large group sorted in increasing order by start index.

Example 1:

Input: s = "abbxxxxzzy"
Output: [[3,6]]
Explanation: "xxxx" is the only large group with start index 3 and end index 6.

Example 2:

Input: s = "abc"
Output: []
Explanation: We have groups "a", "b", and "c", none of which are large groups.

Example 3:

Input: s = "abcdddeeeeaabbbcd"
Output: [[3,5],[6,9],[12,14]]
Explanation: The large groups are "ddd", "eeee", and "bbb".

Example 4:

Input: s = "aba"
Output: []

Constraints:

1 <= s.length <= 1000
s contains lower-case English letters only.

---

831. Masking Personal Information

You are given a personal information string s, representing either an email address or a phone number. Return the m asked personal information using the below rules.
Email address:
An email address is:

A name consisting of uppercase and lowercase English letters, followed by
The '@' symbol, followed by
The domain consisting of uppercase and lowercase English letters with a dot '.' somewhere in the middle (not the firs t or last character).

To mask an email:

The uppercase letters in the name and domain must be converted to lowercase letters.
The middle letters of the name (i.e., all but the first and last letters) must be replaced by 5 asterisks "*****".

Phone number:
A phone number is formatted as follows:

The phone number contains 10-13 digits.
The last 10 digits make up the local number.
The remaining 0-3 digits, in the beginning, make up the country code.
Separation characters from the set {'+', '-', '(', ')', ' '} separate the above digits in some way.

To mask a phone number:

Remove all separation characters.
The masked phone number should have the form:

"***-***-XXXX" if the country code has 0 digits.
"+*-***-***-XXXX" if the country code has 1 digit.
"+**-***-***-XXXX" if the country code has 2 digits.
"+***-***-***-XXXX" if the country code has 3 digits.

"XXXX" is the last 4 digits of the local number.

Example 1:

Input: s = "LeetCode@LeetCode.com"
Output: "l*****e@leetcode.com"
Explanation: s is an email address.
The name and domain are converted to lowercase, and the middle of the name is replaced by 5 asterisks.

Example 2:

Input: s = "AB@qq.com"
Output: "a*****b@qq.com"
Explanation: s is an email address.
The name and domain are converted to lowercase, and the middle of the name is replaced by 5 asterisks.
Note that even though "ab" is 2 characters, it still must have 5 asterisks in the middle.

Example 3:

Input: s = "1(234)567-890"
Output: "***-***-7890"
Explanation: s is a phone number.
There are 10 digits, so the local number is 10 digits and the country code is 0 digits.
Thus, the resulting masked number is "***-***-7890".

Example 4:

Input: s = "86-(10)12345678"
Output: "+**-***-***-5678"
Explanation: s is a phone number.
There are 12 digits, so the local number is 10 digits and the country code is 2 digits.
Thus, the resulting masked number is "+**-***-***-7890".

Constraints:

s is either a valid email or a phone number.
If s is an email:

8 <= s.length <= 40
s consists of uppercase and lowercase English letters and exactly one '@' symbol and '.' symbol.

If s is a phone number:

10 <= s.length <= 20
s consists of digits, spaces, and the symbols '(', ')', '-', and '+'.

---

832. Flipping an Image

Given an n x n binary matrix image, flip the image horizontally, then invert it, and return the resulting image.
To flip an image horizontally means that each row of the image is reversed.

For example, flipping [1,1,0] horizontally results in [0,1,1].

To invert an image means that each 0 is replaced by 1, and each 1 is replaced by 0.

For example, inverting [0,1,1] results in [1,0,0].

Example 1:

Input: image = [[1,1,0],[1,0,1],[0,0,0]]
Output: [[1,0,0],[0,1,0],[1,1,1]]
Explanation: First reverse each row: [[0,1,1],[1,0,1],[0,0,0]].
Then, invert the image: [[1,0,0],[0,1,0],[1,1,1]]

Example 2:

Input: image = [[1,1,0,0],[1,0,0,1],[0,1,1,1],[1,0,1,0]]
Output: [[1,1,0,0],[0,1,1,0],[0,0,0,1],[1,0,1,0]]
Explanation: First reverse each row: [[0,0,1,1],[1,0,0,1],[1,1,1,0],[0,1,0,1]].
Then invert the image: [[1,1,0,0],[0,1,1,0],[0,0,0,1],[1,0,1,0]]

Constraints:

n == image.length
n == image[i].length
1 <= n <= 20
images[i][j] is either 0 or 1.

---

833. Find And Replace in String

You are given a 0-indexed string s that you must perform k replacement operations on. The replacement operations a re given as three 0-indexed parallel arrays, indices, sources, and targets, all of length k.
To complete the ith replacement operation:

Check if the substring sources[i] occurs at index indices[i] in the original string s.
If it does not occur, do nothing.
Otherwise if it does occur, replace that substring with targets[i].

For example, if s = "abcd", indices[i] = 0, sources[i] = "ab", and targets[i] = "eee", then the result of this replacement will be "eeecd".
All replacement operations must occur simultaneously, meaning the replacement operations should not affect the ind exing of each other. The testcases will be generated such that the replacements will not overlap.

For example, a testcase with s = "abc", indices = [0, 1], and sources = ["ab","bc"] will not be generated because the " ab" and "bc" replacements overlap.

Return the resulting string after performing all replacement operations on s.
A substring is a contiguous sequence of characters in a string.

Example 1:

Input: s = "abcd", indices = [0, 2], sources = ["a", "cd"], targets = ["eee", "ffff"]
Output: "eeebffff"
Explanation:
"a" occurs at index 0 in s, so we replace it with "eee".
"cd" occurs at index 2 in s, so we replace it with "ffff".

Example 2:

Input: s = "abcd", indices = [0, 2], sources = ["ab","ec"], targets = ["eee","ffff"]
Output: "eeecd"
Explanation:
"ab" occurs at index 0 in s, so we replace it with "eee".
"ec" does not occur at index 2 in s, so we do nothing.

Constraints:

1 <= s.length <= 1000
k == indices.length == sources.length == targets.length
1 <= k <= 100
0 <= indexes[i] < s.length
1 <= sources[i].length, targets[i].length <= 50
s consists of only lowercase English letters.
sources[i] and targets[i] consist of only lowercase English letters.

---

834. Sum of Distances in Tree

There is an undirected connected tree with n nodes labeled from 0 to n - 1 and n - 1 edges.
You are given the integer n and the array edges where edges[i] = [ai, bi] indicates that there is an edge between node s ai and bi in the tree.
Return an array answer of length n where answer[i] is the sum of the distances between the ith node in the tree and al l other nodes.

Example 1:

Input: n = 6, edges = [[0,1],[0,2],[2,3],[2,4],[2,5]]
Output: [8,12,6,10,10,10]
Explanation: The tree is shown above.
We can see that dist(0,1) + dist(0,2) + dist(0,3) + dist(0,4) + dist(0,5)
equals 1 + 1 + 2 + 2 + 2 = 8.
Hence, answer[0] = 8, and so on.

Example 2:

Input: n = 1, edges = []
Output: [0]

Example 3:

Input: n = 2, edges = [[1,0]]

Output: [1,1]

Constraints:

1 <= n <= 3 * 104
edges.length == n - 1
edges[i].length == 2
0 <= ai, bi < n
ai != bi
The given input represents a valid tree.

---

835. Image Overlap

You are given two images, img1 and img2, represented as binary, square matrices of size n x n. A binary matrix has only 0s and 1s as values.
We translate one image however we choose by sliding all the 1 bits left, right, up, and/or down any number of units. We then place it on top of the other image. We can then calculate the overlap by counting the number of positions th at have a 1 in both images.
Note also that a translation does not include any kind of rotation. Any 1 bits that are translated outside of the matrix borders are erased.
Return the largest possible overlap.

Example 1:

Input: img1 = [[1,1,0],[0,1,0],[0,1,0]], img2 = [[0,0,0],[0,1,1],[0,0,1]]
Output: 3
Explanation: We translate img1 to right by 1 unit and down by 1 unit.

The number of positions that have a 1 in both images is 3 (shown in red).

Example 2:

Input: img1 = [[1]], img2 = [[1]]
Output: 1

Example 3:

Input: img1 = [[0]], img2 = [[0]]
Output: 0

Constraints:

n == img1.length == img1[i].length
n == img2.length == img2[i].length
1 <= n <= 30
img1[i][j] is either 0 or 1.
img2[i][j] is either 0 or 1.

---

836. Rectangle Overlap

An axis-aligned rectangle is represented as a list [x1, y1, x2, y2], where (x1, y1) is the coordinate of its bottom-left c orner, and (x2, y2) is the coordinate of its top-right corner. Its top and bottom edges are parallel to the X-axis, and its left and right edges are parallel to the Y-axis.
Two rectangles overlap if the area of their intersection is positive. To be clear, two rectangles that only touch at the c orner or edges do not overlap.
Given two axis-aligned rectangles rec1 and rec2, return true if they overlap, otherwise return false.

Example 1:
Input: rec1 = [0,0,2,2], rec2 = [1,1,3,3]
Output: true
Example 2:
Input: rec1 = [0,0,1,1], rec2 = [1,0,2,1]
Output: false
Example 3:
Input: rec1 = [0,0,1,1], rec2 = [2,2,3,3]
Output: false

Constraints:

rect1.length == 4
rect2.length == 4
-109 <= rec1[i], rec2[i] <= 109
rec1 and rec2 represent a valid rectangle with a non-zero area.

---

837. New 21 Game

Alice plays the following game, loosely based on the card game "21".
Alice starts with 0 points and draws numbers while she has less than k points. During each draw, she gains an intege r number of points randomly from the range [1, maxPts], where maxPts is an integer. Each draw is independent and t

he outcomes have equal probabilities.
Alice stops drawing numbers when she gets k or more points.
Return the probability that Alice has n or fewer points.
Answers within 10-5 of the actual answer are considered accepted.

Example 1:

Input: n = 10, k = 1, maxPts = 10
Output: 1.00000
Explanation: Alice gets a single card, then stops.

Example 2:

Input: n = 6, k = 1, maxPts = 10
Output: 0.60000
Explanation: Alice gets a single card, then stops.
In 6 out of 10 possibilities, she is at or below 6 points.

Example 3:

Input: n = 21, k = 17, maxPts = 10
Output: 0.73278

Constraints:

0 <= k <= n <= 104
1 <= maxPts <= 104

---

838. Push Dominoes

There are n dominoes in a line, and we place each domino vertically upright. In the beginning, we simultaneously pu sh some of the dominoes either to the left or to the right.
After each second, each domino that is falling to the left pushes the adjacent domino on the left. Similarly, the domin oes falling to the right push their adjacent dominoes standing on the right.
When a vertical domino has dominoes falling on it from both sides, it stays still due to the balance of the forces.
For the purposes of this question, we will consider that a falling domino expends no additional force to a falling or al ready fallen domino.
You are given a string dominoes representing the initial state where:

dominoes[i] = 'L', if the ith domino has been pushed to the left,
dominoes[i] = 'R', if the ith domino has been pushed to the right, and
dominoes[i] = '.', if the ith domino has not been pushed.

Return a string representing the final state.

Example 1:

Input: dominoes = "RR.L"
Output: "RR.L"
Explanation: The first domino expends no additional force on the second domino.

Example 2:

Input: dominoes = ".L.R...LR..L.."
Output: "LL.RR.LLRRLL.."

Constraints:

n == dominoes.length
1 <= n <= 105
dominoes[i] is either 'L', 'R', or '.'.

---

839. Similar String Groups

Two strings X and Y are similar if we can swap two letters (in different positions) of X, so that it equals Y. Also two strings X and Y are similar if they are equal.
For example, "tars" and "rats" are similar (swapping at positions 0 and 2), and "rats" and "arts" are similar, but "star" is not similar to "tars", "rats", or "arts".
Together, these form two connected groups by similarity: {"tars", "rats", "arts"} and {"star"}. Notice that "tars" and "arts" are in the same group even though they are not similar. Formally, each group is such that a word is in the grou p if and only if it is similar to at least one other word in the group.
We are given a list strs of strings where every string in strs is an anagram of every other string in strs. How many gro ups are there?

Example 1:

Input: strs = ["tars","rats","arts","star"]
Output: 2

Example 2:

Input: strs = ["omv","ovm"]
Output: 1

Constraints:

1 <= strs.length <= 300
1 <= strs[i].length <= 300

strs[i] consists of lowercase letters only.
All words in strs have the same length and are anagrams of each other.

---

840. Magic Squares In Grid

A 3 x 3 magic square is a 3 x 3 grid filled with distinct numbers from 1 to 9 such that each row, column, and both di agonals all have the same sum.
Given a row x col grid of integers, how many 3 x 3 "magic square" subgrids are there? (Each subgrid is contiguous).

Example 1:

Input: grid = [[4,3,8,4],[9,5,1,9],[2,7,6,2]]
Output: 1
Explanation:
The following subgrid is a 3 x 3 magic square:

while this one is not:

In total, there is only one magic square inside the given grid.

Example 2:

Input: grid = [[8]]
Output: 0

Example 3:

Input: grid = [[4,4],[3,3]]
Output: 0

Example 4:

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

Constraints:

row == grid.length
col == grid[i].length
1 <= row, col <= 10
0 <= grid[i][j] <= 15

---

841. Keys and Rooms

There are n rooms labeled from 0 to n - 1 and all the rooms are locked except for room 0. Your goal is to visit all the rooms. However, you cannot enter a locked room without having its key.
When you visit a room, you may find a set of distinct keys in it. Each key has a number on it, denoting which room i t unlocks, and you can take all of them with you to unlock the other rooms.
Given an array rooms where rooms[i] is the set of keys that you can obtain if you visited room i, return true if you ca n visit all the rooms, or false otherwise.

Example 1:

Input: rooms = [[1],[2],[3],[]]
Output: true
Explanation:
We visit room 0 and pick up key 1.
We then visit room 1 and pick up key 2.
We then visit room 2 and pick up key 3.
We then visit room 3.
Since we were able to visit every room, we return true.

Example 2:

Input: rooms = [[1,3],[3,0,1],[2],[0]]
Output: false
Explanation: We can not enter room number 2 since the only key that unlocks it is in that room.

Constraints:

n == rooms.length
2 <= n <= 1000
0 <= rooms[i].length <= 1000
1 <= sum(rooms[i].length) <= 3000
0 <= rooms[i][j] < n
All the values of rooms[i] are unique.

---

842. Split Array into Fibonacci Sequence

You are given a string of digits num, such as "123456579". We can split it into a Fibonacci-like sequence [123, 456, 579].

Formally, a Fibonacci-like sequence is a list f of non-negative integers such that:

0 <= f[i] < 231, (that is, each integer fits in a 32-bit signed integer type),
f.length >= 3, and
f[i] + f[i + 1] == f[i + 2] for all 0 <= i < f.length - 2.

Note that when splitting the string into pieces, each piece must not have extra leading zeroes, except if the piece is th e number 0 itself.
Return any Fibonacci-like sequence split from num, or return [] if it cannot be done.

Example 1:

Input: num = "123456579"
Output: [123,456,579]

Example 2:

Input: num = "11235813"
Output: [1,1,2,3,5,8,13]

Example 3:

Input: num = "112358130"
Output: []
Explanation: The task is impossible.

Example 4:

Input: num = "0123"
Output: []
Explanation: Leading zeroes are not allowed, so "01", "2", "3" is not valid.

Example 5:

Input: num = "1101111"
Output: [11,0,11,11]
Explanation: The output [11, 0, 11, 11] would also be accepted.

Constraints:

1 <= num.length <= 200
num contains only digits.

---

843. Guess the Word

This is an interactive problem.
You are given an array of unique strings wordlist where wordlist[i] is 6 letters long, and one word in this list is chose n as secret.
You may call Master.guess(word) to guess a word. The guessed word should have type string and must be from the original list with 6 lowercase letters.
This function returns an integer type, representing the number of exact matches (value and position) of your guess to the secret word. Also, if your guess is not in the given wordlist, it will return -1 instead.
For each test case, you have exactly 10 guesses to guess the word. At the end of any number of calls, if you have ma de 10 or fewer calls to Master.guess and at least one of these guesses was secret, then you pass the test case.

Example 1:

Input: secret = "acckzz", wordlist = ["acckzz","ccbazz","eiowzz","abcczz"], numguesses = 10
Output: You guessed the secret word correctly.
Explanation:
master.guess("aaaaaa") returns -1, because "aaaaaa" is not in wordlist.
master.guess("acckzz") returns 6, because "acckzz" is secret and has all 6 matches.
master.guess("ccbazz") returns 3, because "ccbazz" has 3 matches.
master.guess("eiowzz") returns 2, because "eiowzz" has 2 matches.
master.guess("abcczz") returns 4, because "abcczz" has 4 matches.
We made 5 calls to master.guess and one of them was the secret, so we pass the test case.

Example 2:

Input: secret = "hamada", wordlist = ["hamada","khaled"], numguesses = 10
Output: You guessed the secret word correctly.

Constraints:

1 <= wordlist.length <= 100
wordlist[i].length == 6
wordlist[i] consist of lowercase English letters.
All the strings of wordlist are unique.
secret exists in wordlist.
numguesses == 10

---

844. Backspace String Compare

Given two strings s and t, return true if they are equal when both are typed into empty text editors. '#' means a backs pace character.
Note that after backspacing an empty text, the text will continue empty.

Example 1:

Input: s = "ab#c", t = "ad#c"

Output: true
Explanation: Both s and t become "ac".

Example 2:

Input: s = "ab##", t = "c#d#"
Output: true
Explanation: Both s and t become "".

Example 3:

Input: s = "a##c", t = "#a#c"
Output: true
Explanation: Both s and t become "c".

Example 4:

Input: s = "a#c", t = "b"
Output: false
Explanation: s becomes "c" while t becomes "b".

Constraints:

1 <= s.length, t.length <= 200
s and t only contain lowercase letters and '#' characters.

Follow up: Can you solve it in O(n) time and O(1) space?

---

845. Longest Mountain in Array

You may recall that an array arr is a mountain array if and only if:

arr.length >= 3
There exists some index i (0-indexed) with 0 < i < arr.length - 1 such that:

arr[0] < arr[1] < ... < arr[i - 1] < arr[i]
arr[i] > arr[i + 1] > ... > arr[arr.length - 1]

Given an integer array arr, return the length of the longest subarray, which is a mountain. Return 0 if there is no mou ntain subarray.

Example 1:

Input: arr = [2,1,4,7,3,2,5]
Output: 5
Explanation: The largest mountain is [1,4,7,3,2] which has length 5.

Example 2:

Input: arr = [2,2,2]
Output: 0
Explanation: There is no mountain.

Constraints:

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

Follow up:

Can you solve it using only one pass?
Can you solve it in O(1) space?

---

846. Hand of Straights

Alice has some number of cards and she wants to rearrange the cards into groups so that each group is of size groupSize, and consists of groupSize consecutive cards.
Given an integer array hand where hand[i] is the value written on the ith card and an integer groupSize, return true if she can rearrange the cards, or false otherwise.

Example 1:

Input: hand = [1,2,3,6,2,3,4,7,8], groupSize = 3
Output: true
Explanation: Alice's hand can be rearranged as [1,2,3],[2,3,4],[6,7,8]

Example 2:

Input: hand = [1,2,3,4,5], groupSize = 4
Output: false
Explanation: Alice's hand can not be rearranged into groups of 4.

Constraints:

1 <= hand.length <= 104

0 <= hand[i] <= 109
1 <= groupSize <= hand.length

Note: This question is the same as 1296: https://leetcode.com/problems/divide-array-in-sets-of-k-consecutive-numbers/

---

847. Shortest Path Visiting All Nodes

You have an undirected, connected graph of n nodes labeled from 0 to n - 1. You are given an array graph where graph[i] is a list of all the nodes connected with node i by an edge.
Return the length of the shortest path that visits every node. You may start and stop at any node, you may revisit nodes multiple times, and you may reuse edges.

Example 1:

Input: graph = [[1,2,3],[0],[0],[0]]
Output: 4
Explanation: One possible path is [1,0,2,0,3]

Example 2:

Input: graph = [[1],[0,2,4],[1,3,4],[2],[1,2]]
Output: 4
Explanation: One possible path is [0,1,4,2,3]

Constraints:

n == graph.length
1 <= n <= 12
0 <= graph[i].length < n
graph[i] does not contain i.
If graph[a] contains b, then graph[b] contains a.
The input graph is always connected.

---

848. Shifting Letters

You are given a string s of lowercase English letters and an integer array shifts of the same length.
Call the shift() of a letter, the next letter in the alphabet, (wrapping around so that 'z' becomes 'a').

For example, shift('a') = 'b', shift('t') = 'u', and shift('z') = 'a'.

Now for each shifts[i] = x, we want to shift the first i + 1 letters of s, x times.
Return the final string after all such shifts to s are applied.

Example 1:

Input: s = "abc", shifts = [3,5,9]
Output: "rpl"
Explanation: We start with "abc".
After shifting the first 1 letters of s by 3, we have "dbc".
After shifting the first 2 letters of s by 5, we have "igc".
After shifting the first 3 letters of s by 9, we have "rpl", the answer.

Example 2:

Input: s = "aaa", shifts = [1,2,3]
Output: "gfd"

Constraints:

1 <= s.length <= 105
s consists of lowercase English letters.
shifts.length == s.length
0 <= shifts[i] <= 109

---

849. Maximize Distance to Closest Person

You are given an array representing a row of seats where seats[i] = 1 represents a person sitting in the ith seat, and se ats[i] = 0 represents that the ith seat is empty (0-indexed).
There is at least one empty seat, and at least one person sitting.
Alex wants to sit in the seat such that the distance between him and the closest person to him is maximized.
Return that maximum distance to the closest person.

Example 1:

Input: seats = [1,0,0,0,1,0,1]
Output: 2
Explanation:
If Alex sits in the second open seat (i.e. seats[2]), then the closest person has distance 2.

If Alex sits in any other open seat, the closest person has distance 1.
Thus, the maximum distance to the closest person is 2.

Example 2:

Input: seats = [1,0,0,0]
Output: 3
Explanation:
If Alex sits in the last seat (i.e. seats[3]), the closest person is 3 seats away.
This is the maximum distance possible, so the answer is 3.

Example 3:

Input: seats = [0,1]
Output: 1

Constraints:

2 <= seats.length <= 2 * 104
seats[i] is 0 or 1.
At least one seat is empty.
At least one seat is occupied.

---

850. Rectangle Area II

You are given a 2D array of axis-aligned rectangles. Each rectangle[i] = [xi1, yi1, xi2, yi2] denotes the ith rectangle where (xi1, yi1) are the coordinates of the bottom-left corner, and (xi2, yi2) are the coordinates of the top-right corne r.
Calculate the total area covered by all rectangles in the plane. Any area covered by two or more rectangles should on ly be counted once.
Return the total area. Since the answer may be too large, return it modulo 109 + 7.

Example 1:

Input: rectangles = [[0,0,2,2],[1,0,2,3],[1,0,3,1]]
Output: 6
Explanation: A total area of 6 is covered by all three rectangales, as illustrated in the picture.
From (1,1) to (2,2), the green and red rectangles overlap.
From (1,0) to (2,3), all three rectangles overlap.

Example 2:

Input: rectangles = [[0,0,1000000000,1000000000]]
Output: 49

Explanation: The answer is 1018 modulo (109 + 7), which is 49.

Constraints:

1 <= rectangles.length <= 200
rectanges[i].length == 4
0 <= xi1, yi1, xi2, yi2 <= 109

---

851. Loud and Rich

There is a group of n people labeled from 0 to n - 1 where each person has a different amount of money and a differe nt level of quietness.
You are given an array richer where richer[i] = [ai, bi] indicates that ai has more money than bi and an integer array quiet where quiet[i] is the quietness of the ith person. All the given data in richer are logically correct (i.e., the data will not lead you to a situation where x is richer than y and y is richer than x at the same time).
Return an integer array answer where answer[x] = y if y is the least quiet person (that is, the person y with the smalle st value of quiet[y]) among all people who definitely have equal to or more money than the person x.

Example 1:

Input: richer = [[1,0],[2,1],[3,1],[3,7],[4,3],[5,3],[6,3]], quiet = [3,2,5,4,6,1,7,0]
Output: [5,5,2,5,4,5,6,7]
Explanation:
answer[0] = 5.
Person 5 has more money than 3, which has more money than 1, which has more money than 0.
The only person who is quieter (has lower quiet[x]) is person 7, but it is not clear if they have more money than pers on 0.
answer[7] = 7.
Among all people that definitely have equal to or more money than person 7 (which could be persons 3, 4, 5, 6, or 7) , the person who is the quietest (has lower quiet[x]) is person 7.
The other answers can be filled out with similar reasoning.

Example 2:

Input: richer = [], quiet = [0]
Output: [0]

Constraints:

n == quiet.length
1 <= n <= 500
0 <= quiet[i] < n
All the values of quiet are unique.
0 <= richer.length <= n * (n - 1) / 2

0 <= ai, bi < n
ai != bi
All the pairs of richer are unique.
The observations in richer are all logically consistent.

---

852. Peak Index in a Mountain Array

Let's call an array arr a mountain if the following properties hold:

arr.length >= 3
There exists some i with 0 < i < arr.length - 1 such that:

arr[0] < arr[1] < ... arr[i-1] < arr[i]
arr[i] > arr[i+1] > ... > arr[arr.length - 1]

Given an integer array arr that is guaranteed to be a mountain, return any i such that arr[0] < arr[1] < ... arr[i - 1] < ar r[i] > arr[i + 1] > ... > arr[arr.length - 1].

Example 1:
Input: arr = [0,1,0]
Output: 1
Example 2:
Input: arr = [0,2,1,0]
Output: 1
Example 3:
Input: arr = [0,10,5,2]
Output: 1
Example 4:
Input: arr = [3,4,5,1]
Output: 2
Example 5:
Input: arr = [24,69,100,99,79,78,67,36,26,19]
Output: 2

Constraints:

3 <= arr.length <= 104
0 <= arr[i] <= 106
arr is guaranteed to be a mountain array.

Follow up: Finding the O(n) is straightforward, could you find an O(log(n)) solution?

---

853. Car Fleet

There are n cars going to the same destination along a one-lane road. The destination is target miles away.
You are given two integer array position and speed, both of length n, where position[i] is the position of the ith car a nd speed[i] is the speed of the ith car (in miles per hour).
A car can never pass another car ahead of it, but it can catch up to it and drive bumper to bumper at the same speed. The faster car will slow down to match the slower car's speed. The distance between these two cars is ignored (i.e., t hey are assumed to have the same position).
A car fleet is some non-empty set of cars driving at the same position and same speed. Note that a single car is also a car fleet.
If a car catches up to a car fleet right at the destination point, it will still be considered as one car fleet.
Return the number of car fleets that will arrive at the destination.

Example 1:

Input: target = 12, position = [10,8,0,5,3], speed = [2,4,1,1,3]
Output: 3
Explanation:
The cars starting at 10 and 8 become a fleet, meeting each other at 12.
The car starting at 0 doesn't catch up to any other car, so it is a fleet by itself.
The cars starting at 5 (speed 3) and 3 (speed 1) become a fleet, meeting each other at 6. The fleet moves at speed 1 u ntil it reaches target.
Note that no other cars meet these fleets before the destination, so the answer is 3.

Example 2:

Input: target = 10, position = [3], speed = [3]
Output: 1
Explanation: There is only one car, hence there is only one fleet.

Example 3:

Input: target = 100, position = [0,2,4], speed = [4,2,1]
Output: 1
Explanation:
The cars starting at 0 (speed 4) and 2 (speed 2) become a fleet, meeting each other at 4. The fleet moves at speed 2.
Then, the fleet (speed 2) and the car starting at 4 (speed 1) become one fleet, meeting each other at 6. The fleet move s at speed 1 until it reaches target.

Constraints:

n == position.length == speed.length
1 <= n <= 105
0 < target <= 106
0 <= position[i] < target
All the values of position are unique.
0 < speed[i] <= 106

---

854. K-Similar Strings

Strings s1 and s2 are k-similar (for some non-negative integer k) if we can swap the positions of two letters in s1 exa ctly k times so that the resulting string equals s2.
Given two anagrams s1 and s2, return the smallest k for which s1 and s2 are k-similar.

Example 1:
Input: s1 = "ab", s2 = "ba"
Output: 1
Example 2:
Input: s1 = "abc", s2 = "bca"
Output: 2
Example 3:
Input: s1 = "abac", s2 = "baca"
Output: 2
Example 4:
Input: s1 = "aabc", s2 = "abca"
Output: 2

Constraints:

1 <= s1.length <= 20
s2.length == s1.length
s1 and s2 contain only lowercase letters from the set {'a', 'b', 'c', 'd', 'e', 'f'}.
s2 is an anagram of s1.

---

855. Exam Room

There is an exam room with n seats in a single row labeled from 0 to n - 1.
When a student enters the room, they must sit in the seat that maximizes the distance to the closest person. If there ar e multiple such seats, they sit in the seat with the lowest number. If no one is in the room, then the student sits at seat number 0.
Design a class that simulates the mentioned exam room.
Implement the ExamRoom class:

ExamRoom(int n) Initializes the object of the exam room with the number of the seats n.

int seat() Returns the label of the seat at which the next student will set.
void leave(int p) Indicates that the student sitting at seat p will leave the room. It is guaranteed that there will be a stu dent sitting at seat p.

Example 1:

Input
["ExamRoom", "seat", "seat", "seat", "seat", "leave", "seat"]
[[10], [], [], [], [], [4], []]
Output
[null, 0, 9, 4, 2, null, 5]

Explanation
ExamRoom examRoom = new ExamRoom(10);
examRoom.seat(); // return 0, no one is in the room, then the student sits at seat number 0.
examRoom.seat(); // return 9, the student sits at the last seat number 9.
examRoom.seat(); // return 4, the student sits at the last seat number 4.
examRoom.seat(); // return 2, the student sits at the last seat number 2.
examRoom.leave(4);
examRoom.seat(); // return 5, the student sits at the last seat number 5.

Constraints:

1 <= n <= 109
It is guaranteed that there is a student sitting at seat p.
At most 104 calls will be made to seat and leave.

---

856. Score of Parentheses

Given a balanced parentheses string s, return the score of the string.
The score of a balanced parentheses string is based on the following rule:

"()" has score 1.
AB has score A + B, where A and B are balanced parentheses strings.
(A) has score 2 * A, where A is a balanced parentheses string.

Example 1:
Input: s = "()"
Output: 1
Example 2:
Input: s = "(())"
Output: 2

Example 3:
Input: s = "()()"
Output: 2
Example 4:
Input: s = "((()()))"
Output: 6

Constraints:

2 <= s.length <= 50
s consists of only '(' and ')'.
s is a balanced parentheses string.

---

857. Minimum Cost to Hire K Workers

There are n workers. You are given two integer arrays quality and wage where quality[i] is the quality of the ith wor ker and wage[i] is the minimum wage expectation for the ith worker.
We want to hire exactly k workers to form a paid group. To hire a group of k workers, we must pay them according t o the following rules:

Every worker in the paid group should be paid in the ratio of their quality compared to other workers in the paid gro up.
Every worker in the paid group must be paid at least their minimum wage expectation.

Given the integer k, return the least amount of money needed to form a paid group satisfying the above conditions. Answers within 10-5 of the actual answer will be accepted.

Example 1:

Input: quality = [10,20,5], wage = [70,50,30], k = 2
Output: 105.00000
Explanation: We pay 70 to 0th worker and 35 to 2nd worker.

Example 2:

Input: quality = [3,1,10,10,1], wage = [4,8,2,2,7], k = 3
Output: 30.66667
Explanation: We pay 4 to 0th worker, 13.33333 to 2nd and 3rd workers separately.

Constraints:

n == quality.length == wage.length
1 <= k <= n <= 104
1 <= quality[i], wage[i] <= 104

---

858. Mirror Reflection

There is a special square room with mirrors on each of the four walls. Except for the southwest corner, there are rece ptors on each of the remaining corners, numbered 0, 1, and 2.
The square room has walls of length p and a laser ray from the southwest corner first meets the east wall at a distanc e q from the 0th receptor.
Given the two integers p and q, return the number of the receptor that the ray meets first.
The test cases are guaranteed so that the ray will meet a receptor eventually.

Example 1:

Input: p = 2, q = 1
Output: 2
Explanation: The ray meets receptor 2 the first time it gets reflected back to the left wall.

Example 2:

Input: p = 3, q = 1
Output: 1

Constraints:

1 <= q <= p <= 1000

---

859. Buddy Strings

Given two strings s and goal, return true if you can swap two letters in s so the result is equal to goal, otherwise, retu rn false.
Swapping letters is defined as taking two indices i and j (0-indexed) such that i != j and swapping the characters at s[i] and s[j].

For example, swapping at indices 0 and 2 in "abcd" results in "cbad".

Example 1:

Input: s = "ab", goal = "ba"
Output: true
Explanation: You can swap s[0] = 'a' and s[1] = 'b' to get "ba", which is equal to goal.

Example 2:

Input: s = "ab", goal = "ab"
Output: false
Explanation: The only letters you can swap are s[0] = 'a' and s[1] = 'b', which results in "ba" != goal.

Example 3:

Input: s = "aa", goal = "aa"
Output: true
Explanation: You can swap s[0] = 'a' and s[1] = 'a' to get "aa", which is equal to goal.

Example 4:

Input: s = "aaaaaaabc", goal = "aaaaaaacb"
Output: true

Constraints:

1 <= s.length, goal.length <= 2 * 104
s and goal consist of lowercase letters.

---

860. Lemonade Change

At a lemonade stand, each lemonade costs $5. Customers are standing in a queue to buy from you, and order one at a time (in the order specified by bills). Each customer will only buy one lemonade and pay with either a $5, $10, or $20 bill. You must provide the correct change to each customer so that the net transaction is that the customer pays $5.
Note that you don't have any change in hand at first.
Given an integer array bills where bills[i] is the bill the ith customer pays, return true if you can provide every customer with correct change, or false otherwise.

Example 1:

Input: bills = [5,5,5,10,20]
Output: true
Explanation:
From the first 3 customers, we collect three $5 bills in order.
From the fourth customer, we collect a $10 bill and give back a $5.
From the fifth customer, we give a $10 bill and a $5 bill.

Since all customers got correct change, we output true.

Example 2:

Input: bills = [5,5,10,10,20]
Output: false
Explanation:
From the first two customers in order, we collect two $5 bills.
For the next two customers in order, we collect a $10 bill and give back a $5 bill.
For the last customer, we can not give change of $15 back because we only have two $10 bills.
Since not every customer received correct change, the answer is false.

Example 3:

Input: bills = [5,5,10]
Output: true

Example 4:

Input: bills = [10,10]
Output: false

Constraints:

1 <= bills.length <= 105
bills[i] is either 5, 10, or 20.

---

861. Score After Flipping Matrix

You are given an m x n binary matrix grid.
A move consists of choosing any row or column and toggling each value in that row or column (i.e., changing all 0's to 1's, and all 1's to 0's).
Every row of the matrix is interpreted as a binary number, and the score of the matrix is the sum of these numbers.
Return the highest possible score after making any number of moves (including zero moves).

Example 1:

Input: grid = [[0,0,1,1],[1,0,1,0],[1,1,0,0]]
Output: 39
Explanation: 0b1111 + 0b1001 + 0b1111 = 15 + 9 + 15 = 39

Example 2:

Input: grid = [[0]]

Output: 1

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 20
grid[i][j] is either 0 or 1.

---

862. Shortest Subarray with Sum at Least K

Given an integer array nums and an integer k, return the length of the shortest non-empty subarray of nums with a su m of at least k. If there is no such subarray, return -1.
A subarray is a contiguous part of an array.

Example 1:
Input: nums = [1], k = 1
Output: 1
Example 2:
Input: nums = [1,2], k = 4
Output: -1
Example 3:
Input: nums = [2,-1,2], k = 3
Output: 3

Constraints:

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

---

863. All Nodes Distance K in Binary Tree

Given the root of a binary tree, the value of a target node target, and an integer k, return an array of the values of all nodes that have a distance k from the target node.
You can return the answer in any order.

Example 1:

Input: root = [3,5,1,6,2,0,8,null,null,7,4], target = 5, k = 2
Output: [7,4,1]
Explanation: The nodes that are a distance 2 from the target node (with value 5) have values 7, 4, and 1.

Example 2:

Input: root = [1], target = 1, k = 3
Output: []

Constraints:

The number of nodes in the tree is in the range [1, 500].
0 <= Node.val <= 500
All the values Node.val are unique.
target is the value of one of the nodes in the tree.
0 <= k <= 1000

---

864. Shortest Path to Get All Keys

You are given an m x n grid grid where:

'.' is an empty cell.
'#' is a wall.
'@' is the starting point.
Lowercase letters represent keys.
Uppercase letters represent locks.

You start at the starting point and one move consists of walking one space in one of the four cardinal directions. You cannot walk outside the grid, or walk into a wall.
If you walk over a key, you can pick it up and you cannot walk over a lock unless you have its corresponding key.
For some 1 <= k <= 6, there is exactly one lowercase and one uppercase letter of the first k letters of the English alphabet in the grid. This means that there is exactly one key for each lock, and one lock for each key; and also that the letters used to represent the keys and locks were chosen in the same order as the English alphabet.
Return the lowest number of moves to acquire all keys. If it is impossible, return -1.

Example 1:

Input: grid = ["@.a.#","###.#","b.A.B"]
Output: 8
Explanation: Note that the goal is to obtain all the keys not to open all the locks.

Example 2:

Input: grid = ["@..aA","..B#.","....b"]
Output: 6

Example 3:

Input: grid = ["@Aa"]
Output: -1

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 30
grid[i][j] is either an English letter, '.', '#', or '@'.
The number of keys in the grid is in the range [1, 6].
Each key in the grid is unique.
Each key in the grid has a matching lock.

---

865. Smallest Subtree with all the Deepest Nodes

Given the root of a binary tree, the depth of each node is the shortest distance to the root.
Return the smallest subtree such that it contains all the deepest nodes in the original tree.
A node is called the deepest if it has the largest depth possible among any node in the entire tree.
The subtree of a node is a tree consisting of that node, plus the set of all descendants of that node.

Example 1:

Input: root = [3,5,1,6,2,0,8,null,null,7,4]
Output: [2,7,4]
Explanation: We return the node with value 2, colored in yellow in the diagram.
The nodes coloured in blue are the deepest nodes of the tree.
Notice that nodes 5, 3 and 2 contain the deepest nodes in the tree but node 2 is the smallest subtree among them, so we return it.

Example 2:

Input: root = [1]
Output: [1]
Explanation: The root is the deepest node in the tree.

Example 3:

Input: root = [0,1,3,null,2]
Output: [2]
Explanation: The deepest node in the tree is 2, the valid subtrees are the subtrees of nodes 2, 1 and 0 but the subtree of node 2 is the smallest.

Constraints:

The number of nodes in the tree will be in the range [1, 500].
0 <= Node.val <= 500
The values of the nodes in the tree are unique.

Note: This question is the same as 1123: https://leetcode.com/problems/lowest-common-ancestor-of-deepest-leaves/

---

866. Prime Palindrome

Given an integer n, return the smallest prime palindrome greater than or equal to n.
An integer is prime if it has exactly two divisors: 1 and itself. Note that 1 is not a prime number.

For example, 2, 3, 5, 7, 11, and 13 are all primes.

An integer is a palindrome if it reads the same from left to right as it does from right to left.

For example, 101 and 12321 are palindromes.

The test cases are generated so that the answer always exists and is in the range [2, 2 * 108].

Example 1:
Input: n = 6
Output: 7
Example 2:
Input: n = 8
Output: 11
Example 3:
Input: n = 13
Output: 101

Constraints:

1 <= n <= 108

---

867. Transpose Matrix

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

---

868. Binary Gap

Given a positive integer n, find and return the longest distance between any two adjacent 1's in the binary representat
ion of n. If there are no two adjacent 1's, return 0.
Two 1's are adjacent if there are only 0's separating them (possibly no 0's). The distance between two 1's is the absol
ute difference between their bit positions. For example, the two 1's in "1001" have a distance of 3.

Example 1:

Input: n = 22
Output: 2
Explanation: 22 in binary is "10110".
The first adjacent pair of 1's is "10110" with a distance of 2.

The second adjacent pair of 1's is "10110" with a distance of 1.
The answer is the largest of these two distances, which is 2.
Note that "10110" is not a valid pair since there is a 1 separating the two 1's underlined.

Example 2:

Input: n = 5
Output: 2
Explanation: 5 in binary is "101".

Example 3:

Input: n = 6
Output: 1
Explanation: 6 in binary is "110".

Example 4:

Input: n = 8
Output: 0
Explanation: 8 in binary is "1000".
There aren't any adjacent pairs of 1's in the binary representation of 8, so we return 0.

Example 5:

Input: n = 1
Output: 0

Constraints:

1 <= n <= 109

---

869. Reordered Power of 2

You are given an integer n. We reorder the digits in any order (including the original order) such that the leading digi t is not zero.
Return true if and only if we can do this so that the resulting number is a power of two.

Example 1:

Input: n = 1
Output: true

Example 2:

Input: n = 10
Output: false

Example 3:

Input: n = 16
Output: true

Example 4:

Input: n = 24
Output: false

Example 5:

Input: n = 46
Output: true

Constraints:

1 <= n <= 109

---

870. Advantage Shuffle

You are given two integer arrays nums1 and nums2 both of the same length. The advantage of nums1 with respect to nums2 is the number of indices i for which nums1[i] > nums2[i].
Return any permutation of nums1 that maximizes its advantage with respect to nums2.

Example 1:
Input: nums1 = [2,7,11,15], nums2 = [1,10,4,11]
Output: [2,11,7,15]
Example 2:
Input: nums1 = [12,24,8,32], nums2 = [13,25,32,11]
Output: [24,32,8,12]

Constraints:

1 <= nums1.length <= 105
nums2.length == nums1.length
0 <= nums1[i], nums2[i] <= 109

---

871. Minimum Number of Refueling Stops

A car travels from a starting position to a destination which is target miles east of the starting position.
There are gas stations along the way. The gas stations are represented as an array stations where stations[i] = [positio ni, fueli] indicates that the ith gas station is positioni miles east of the starting position and has fueli liters of gas.
The car starts with an infinite tank of gas, which initially has startFuel liters of fuel in it. It uses one liter of gas per o ne mile that it drives. When the car reaches a gas station, it may stop and refuel, transferring all the gas from the stati on into the car.
Return the minimum number of refueling stops the car must make in order to reach its destination. If it cannot reach the destination, return -1.
Note that if the car reaches a gas station with 0 fuel left, the car can still refuel there. If the car reaches the destinatio n with 0 fuel left, it is still considered to have arrived.

Example 1:

Input: target = 1, startFuel = 1, stations = []
Output: 0
Explanation: We can reach the target without refueling.

Example 2:

Input: target = 100, startFuel = 1, stations = [[10,100]]
Output: -1
Explanation: We can not reach the target (or even the first gas station).

Example 3:

Input: target = 100, startFuel = 10, stations = [[10,60],[20,30],[30,30],[60,40]]
Output: 2
Explanation: We start with 10 liters of fuel.
We drive to position 10, expending 10 liters of fuel. We refuel from 0 liters to 60 liters of gas.
Then, we drive from position 10 to position 60 (expending 50 liters of fuel),
and refuel from 10 liters to 50 liters of gas. We then drive to and reach the target.
We made 2 refueling stops along the way, so we return 2.

Constraints:

1 <= target, startFuel <= 109
0 <= stations.length <= 500
0 <= positioni <= positioni+1 < target
1 <= fueli < 109

---

872. Leaf-Similar Trees

Consider all the leaves of a binary tree, from left to right order, the values of those leaves form a leaf value sequence
.

For example, in the given tree above, the leaf value sequence is (6, 7, 4, 9, 8).
Two binary trees are considered leaf-similar if their leaf value sequence is the same.
Return true if and only if the two given trees with head nodes root1 and root2 are leaf-similar.

Example 1:

Input: root1 = [3,5,1,6,2,9,8,null,null,7,4], root2 = [3,5,1,6,7,4,2,null,null,null,null,null,null,9,8]
Output: true

Example 2:

Input: root1 = [1], root2 = [1]
Output: true

Example 3:

Input: root1 = [1], root2 = [2]
Output: false

Example 4:

Input: root1 = [1,2], root2 = [2,2]
Output: true

Example 5:

Input: root1 = [1,2,3], root2 = [1,3,2]
Output: false

Constraints:

The number of nodes in each tree will be in the range [1, 200].
Both of the given trees will have values in the range [0, 200].

---

873. Length of Longest Fibonacci Subsequence

A sequence x1, x2, ..., xn is Fibonacci-like if:

n >= 3
xi + xi+1 == xi+2 for all i + 2 <= n

Given a strictly increasing array arr of positive integers forming a sequence, return the length of the longest Fibonac ci-like subsequence of arr. If one does not exist, return 0.
A subsequence is derived from another sequence arr by deleting any number of elements (including none) from arr, without changing the order of the remaining elements. For example, [3, 5, 8] is a subsequence of [3, 4, 5, 6, 7, 8].

Example 1:

Input: arr = [1,2,3,4,5,6,7,8]
Output: 5
Explanation: The longest subsequence that is fibonacci-like: [1,2,3,5,8].
Example 2:

Input: arr = [1,3,7,11,12,14,18]
Output: 3
Explanation: The longest subsequence that is fibonacci-like: [1,11,12], [3,11,14] or [7,11,18].

Constraints:

3 <= arr.length <= 1000
1 <= arr[i] < arr[i + 1] <= 109

---

874. Walking Robot Simulation

A robot on an infinite XY-plane starts at point (0, 0) facing north. The robot can receive a sequence of these three po ssible types of commands:

-2: Turn left 90 degrees.
-1: Turn right 90 degrees.
1 <= k <= 9: Move forward k units, one unit at a time.

Some of the grid squares are obstacles. The ith obstacle is at grid point obstacles[i] = (xi, yi). If the robot runs into a n obstacle, then it will instead stay in its current location and move on to the next command.
Return the maximum Euclidean distance that the robot ever gets from the origin squared (i.e. if the distance is 5, retu rn 25).
Note:

North means +Y direction.
East means +X direction.
South means -Y direction.
West means -X direction.

Example 1:

Input: commands = [4,-1,3], obstacles = []
Output: 25
Explanation: The robot starts at (0, 0):
1. Move north 4 units to (0, 4).
2. Turn right.
3. Move east 3 units to (3, 4).
The furthest point the robot ever gets from the origin is (3, 4), which squared is 32 + 42 = 25 units away.

Example 2:

Input: commands = [4,-1,4,-2,4], obstacles = [[2,4]]
Output: 65
Explanation: The robot starts at (0, 0):
1. Move north 4 units to (0, 4).
2. Turn right.
3. Move east 1 unit and get blocked by the obstacle at (2, 4), robot is at (1, 4).
4. Turn left.
5. Move north 4 units to (1, 8).
The furthest point the robot ever gets from the origin is (1, 8), which squared is 12 + 82 = 65 units away.

Example 3:

Input: commands = [6,-1,-1,6], obstacles = []
Output: 36
Explanation: The robot starts at (0, 0):
1. Move north 6 units to (0, 6).
2. Turn right.
3. Turn right.
4. Move south 6 units to (0, 0).
The furthest point the robot ever gets from the origin is (0, 6), which squared is 62 = 36 units away.

Constraints:

1 <= commands.length <= 104
commands[i] is either -2, -1, or an integer in the range [1, 9].
0 <= obstacles.length <= 104
-3 * 104 <= xi, yi <= 3 * 104
The answer is guaranteed to be less than 231.

---

875. Koko Eating Bananas

Koko loves to eat bananas. There are n piles of bananas, the ith pile has piles[i] bananas. The guards have gone and

will come back in h hours.
Koko can decide her bananas-per-hour eating speed of k. Each hour, she chooses some pile of bananas and eats k ba
nanas from that pile. If the pile has less than k bananas, she eats all of them instead and will not eat any more banana
s during this hour.
Koko likes to eat slowly but still wants to finish eating all the bananas before the guards return.
Return the minimum integer k such that she can eat all the bananas within h hours.

Example 1:

Input: piles = [3,6,7,11], h = 8
Output: 4

Example 2:

Input: piles = [30,11,23,4,20], h = 5
Output: 30

Example 3:

Input: piles = [30,11,23,4,20], h = 6
Output: 23

Constraints:

1 <= piles.length <= 104
piles.length <= h <= 109
1 <= piles[i] <= 109

---

876. Middle of the Linked List

Given the head of a singly linked list, return the middle node of the linked list.
If there are two middle nodes, return the second middle node.

Example 1:

Input: head = [1,2,3,4,5]
Output: [3,4,5]
Explanation: The middle node of the list is node 3.

Example 2:

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

Explanation: Since the list has two middle nodes with values 3 and 4, we return the second one.

Constraints:

The number of nodes in the list is in the range [1, 100].
1 <= Node.val <= 100

---

877. Stone Game

Alice and Bob play a game with piles of stones. There are an even number of piles arranged in a row, and each pile h as a positive integer number of stones piles[i].
The objective of the game is to end with the most stones. The total number of stones across all the piles is odd, so th ere are no ties.
Alice and Bob take turns, with Alice starting first. Each turn, a player takes the entire pile of stones either from the b eginning or from the end of the row. This continues until there are no more piles left, at which point the person with the most stones wins.
Assuming Alice and Bob play optimally, return true if Alice wins the game, or false if Bob wins.

Example 1:

Input: piles = [5,3,4,5]
Output: true
Explanation:
Alice starts first, and can only take the first 5 or the last 5.
Say she takes the first 5, so that the row becomes [3, 4, 5].
If Bob takes 3, then the board is [4, 5], and Alice takes 5 to win with 10 points.
If Bob takes the last 5, then the board is [3, 4], and Alice takes 4 to win with 9 points.
This demonstrated that taking the first 5 was a winning move for Alice, so we return true.

Example 2:

Input: piles = [3,7,2,3]
Output: true

Constraints:

2 <= piles.length <= 500
piles.length is even.
1 <= piles[i] <= 500
sum(piles[i]) is odd.

---

878. Nth Magical Number

A positive integer is magical if it is divisible by either a or b.
Given the three integers n, a, and b, return the nth magical number. Since the answer may be very large, return it mo dulo 109 + 7.

Example 1:
Input: n = 1, a = 2, b = 3
Output: 2
Example 2:
Input: n = 4, a = 2, b = 3
Output: 6
Example 3:
Input: n = 5, a = 2, b = 4
Output: 10
Example 4:
Input: n = 3, a = 6, b = 4
Output: 8

Constraints:

1 <= n <= 109
2 <= a, b <= 4 * 104

---

879. Profitable Schemes

There is a group of n members, and a list of various crimes they could commit. The ith crime generates a profit[i] an d requires group[i] members to participate in it. If a member participates in one crime, that member can't participate i n another crime.
Let's call a profitable scheme any subset of these crimes that generates at least minProfit profit, and the total number of members participating in that subset of crimes is at most n.
Return the number of schemes that can be chosen. Since the answer may be very large, return it modulo 109 + 7.

Example 1:

Input: n = 5, minProfit = 3, group = [2,2], profit = [2,3]
Output: 2
Explanation: To make a profit of at least 3, the group could either commit crimes 0 and 1, or just crime 1.
In total, there are 2 schemes.
Example 2:

Input: n = 10, minProfit = 5, group = [2,3,5], profit = [6,7,8]
Output: 7
Explanation: To make a profit of at least 5, the group could commit any crimes, as long as they commit one.
There are 7 possible schemes: (0), (1), (2), (0,1), (0,2), (1,2), and (0,1,2).

Constraints:

1 <= n <= 100
0 <= minProfit <= 100
1 <= group.length <= 100
1 <= group[i] <= 100
profit.length == group.length
0 <= profit[i] <= 100

---

880. Decoded String at Index

You are given an encoded string s. To decode the string to a tape, the encoded string is read one character at a time a nd 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.

---

881. Boats to Save People

You are given an array people where people[i] is the weight of the ith person, and an infinite number of boats where each boat can carry a maximum weight of limit. Each boat carries at most two people at the same time, provided the sum of the weight of those people is at most limit.
Return the minimum number of boats to carry every given person.

Example 1:

Input: people = [1,2], limit = 3
Output: 1
Explanation: 1 boat (1, 2)

Example 2:

Input: people = [3,2,2,1], limit = 3
Output: 3
Explanation: 3 boats (1, 2), (2) and (3)

Example 3:

Input: people = [3,5,3,4], limit = 5
Output: 4
Explanation: 4 boats (3), (3), (4), (5)

Constraints:

1 <= people.length <= 5 * 104
1 <= people[i] <= limit <= 3 * 104

---

882. Reachable Nodes In Subdivided Graph

You are given an undirected graph (the "original graph") with n nodes labeled from 0 to n - 1. You decide to subdivide each edge in the graph into a chain of nodes, with the number of new nodes varying between each edge.
The graph is given as a 2D array of edges where edges[i] = [ui, vi, cnti] indicates that there is an edge between nodes ui and vi in the original graph, and cnti is the total number of new nodes that you will subdivide the edge into. Note that cnti == 0 means you will not subdivide the edge.
To subdivide the edge [ui, vi], replace it with (cnti + 1) new edges and cnti new nodes. The new nodes are x1, x2, ..., xcnti, and the new edges are [ui, x1], [x1, x2], [x2, x3], ..., [xcnti-1, xcnti], [xcnti, vi].
In this new graph, you want to know how many nodes are reachable from the node 0, where a node is reachable if the distance is maxMoves or less.
Given the original graph and maxMoves, return the number of nodes that are reachable from node 0 in the new graph.

Example 1:

Input: edges = [[0,1,10],[0,2,1],[1,2,2]], maxMoves = 6, n = 3
Output: 13
Explanation: The edge subdivisions are shown in the image above.
The nodes that are reachable are highlighted in yellow.

Example 2:

Input: edges = [[0,1,4],[1,2,6],[0,2,8],[1,3,1]], maxMoves = 10, n = 4
Output: 23

Example 3:

Input: edges = [[1,2,4],[1,4,5],[1,3,1],[2,3,4],[3,4,5]], maxMoves = 17, n = 5
Output: 1
Explanation: Node 0 is disconnected from the rest of the graph, so only node 0 is reachable.

Constraints:

0 <= edges.length <= min(n * (n - 1) / 2, 104)
edges[i].length == 3
0 <= ui < vi < n
There are no multiple edges in the graph.
0 <= cnti <= 104
0 <= maxMoves <= 109
1 <= n <= 3000

---

883. Projection Area of 3D Shapes

You are given an n x n grid where we place some 1 x 1 x 1 cubes that are axis-aligned with the x, y, and z axes.
Each value v = grid[i][j] represents a tower of v cubes placed on top of the cell (i, j).
We view the projection of these cubes onto the xy, yz, and zx planes.
A projection is like a shadow, that maps our 3-dimensional figure to a 2-dimensional plane. We are viewing the "sha dow" when looking at the cubes from the top, the front, and the side.
Return the total area of all three projections.

Example 1:

Input: grid = [[1,2],[3,4]]
Output: 17
Explanation: Here are the three projections ("shadows") of the shape made with each axis-aligned plane.

Example 2:

Input: grid = [[2]]
Output: 5

Example 3:

Input: grid = [[1,0],[0,2]]
Output: 8

Example 4:

Input: grid = [[1,1,1],[1,0,1],[1,1,1]]
Output: 14

Example 5:

Input: grid = [[2,2,2],[2,1,2],[2,2,2]]
Output: 21

Constraints:

n == grid.length
n == grid[i].length
1 <= n <= 50
0 <= grid[i][j] <= 50

---

884. Uncommon Words from Two Sentences

A sentence is a string of single-space separated words where each word consists only of lowercase letters.
A word is uncommon if it appears exactly once in one of the sentences, and does not appear in the other sentence.
Given two sentences s1 and s2, return a list of all the uncommon words. You may return the answer in any order.

Example 1:
Input: s1 = "this apple is sweet", s2 = "this apple is sour"
Output: ["sweet","sour"]
Example 2:
Input: s1 = "apple apple", s2 = "banana"
Output: ["banana"]

Constraints:

1 <= s1.length, s2.length <= 200
s1 and s2 consist of lowercase English letters and spaces.
s1 and s2 do not have leading or trailing spaces.
All the words in s1 and s2 are separated by a single space.

---

885. Spiral Matrix III

You start at the cell (rStart, cStart) of an rows x cols grid facing east. The northwest corner is at the first row and col umn in the grid, and the southeast corner is at the last row and column.
You will walk in a clockwise spiral shape to visit every position in this grid. Whenever you move outside the grid's b oundary, we continue our walk outside the grid (but may return to the grid boundary later.). Eventually, we reach all rows * cols spaces of the grid.
Return an array of coordinates representing the positions of the grid in the order you visited them.

Example 1:

Input: rows = 1, cols = 4, rStart = 0, cStart = 0
Output: [[0,0],[0,1],[0,2],[0,3]]

Example 2:

Input: rows = 5, cols = 6, rStart = 1, cStart = 4
Output: [[1,4],[1,5],[2,5],[2,4],[2,3],[1,3],[0,3],[0,4],[0,5],[3,5],[3,4],[3,3],[3,2],[2,2],[1,2],[0,2],[4,5],[4,4],[4,3],[4,2] ,[4,1],[3,1],[2,1],[1,1],[0,1],[4,0],[3,0],[2,0],[1,0],[0,0]]

Constraints:

1 <= rows, cols <= 100

0 <= rStart < rows
0 <= cStart < cols

---

886. Possible Bipartition

We want to split a group of n people (labeled from 1 to n) into two groups of any size. Each person may dislike som
e other people, and they should not go into the same group.
Given the integer n and the array dislikes where dislikes[i] = [ai, bi] indicates that the person labeled ai does not like the person labeled bi, return true if it is possible to split everyone into two groups in this way.

Example 1:

Input: n = 4, dislikes = [[1,2],[1,3],[2,4]]
Output: true
Explanation: group1 [1,4] and group2 [2,3].

Example 2:

Input: n = 3, dislikes = [[1,2],[1,3],[2,3]]
Output: false

Example 3:

Input: n = 5, dislikes = [[1,2],[2,3],[3,4],[4,5],[1,5]]
Output: false

Constraints:

1 <= n <= 2000
0 <= dislikes.length <= 104
dislikes[i].length == 2
1 <= dislikes[i][j] <= n
ai < bi
All the pairs of dislikes are unique.

---

887. Super Egg Drop

You are given k identical eggs and you have access to a building with n floors labeled from 1 to n.
You know that there exists a floor f where 0 <= f <= n such that any egg dropped at a floor higher than f will break, and any egg dropped at or below floor f will not break.
Each move, you may take an unbroken egg and drop it from any floor x (where 1 <= x <= n). If the egg breaks, you can no longer use it. However, if the egg does not break, you may reuse it in future moves.
Return the minimum number of moves that you need to determine with certainty what the value of f is.

Example 1:

Input: k = 1, n = 2
Output: 2
Explanation:
Drop the egg from floor 1. If it breaks, we know that f = 0.
Otherwise, drop the egg from floor 2. If it breaks, we know that f = 1.
If it does not break, then we know f = 2.
Hence, we need at minimum 2 moves to determine with certainty what the value of f is.

Example 2:

Input: k = 2, n = 6
Output: 3

Example 3:

Input: k = 3, n = 14
Output: 4

Constraints:

1 <= k <= 100
1 <= n <= 104

---

888. Fair Candy Swap

Alice and Bob have a different total number of candies. You are given two integer arrays aliceSizes and bobSizes wh ere aliceSizes[i] is the number of candies of the ith box of candy that Alice has and bobSizes[j] is the number of can dies of the jth box of candy that Bob has.
Since they are friends, they would like to exchange one candy box each so that after the exchange, they both have th e same total amount of candy. The total amount of candy a person has is the sum of the number of candies in each bo x they have.
Return an integer array answer where answer[0] is the number of candies in the box that Alice must exchange, and a nswer[1] is the number of candies in the box that Bob must exchange. If there are multiple answers, you may return any one of them. It is guaranteed that at least one answer exists.

Example 1:

Input: aliceSizes = [1,1], bobSizes = [2,2]
Output: [1,2]

Example 2:

Input: aliceSizes = [1,2], bobSizes = [2,3]
Output: [1,2]

Example 3:

Input: aliceSizes = [2], bobSizes = [1,3]
Output: [2,3]

Example 4:

Input: aliceSizes = [1,2,5], bobSizes = [2,4]
Output: [5,4]

Constraints:

1 <= aliceSizes.length, bobSizes.length <= 104
1 <= aliceSizes[i], bobSizes[j] <= 105
Alice and Bob have a different total number of candies.
There will be at least one valid answer for the given input.

---

889. Construct Binary Tree from Preorder and Postorder Traversal

Given two integer arrays, preorder and postorder where preorder is the preorder traversal of a binary tree of distinct values and postorder is the postorder traversal of the same tree, reconstruct and return the binary tree.
If there exist multiple answers, you can return any of them.

Example 1:

Input: preorder = [1,2,4,5,3,6,7], postorder = [4,5,2,6,7,3,1]
Output: [1,2,3,4,5,6,7]

Example 2:

Input: preorder = [1], postorder = [1]
Output: [1]

Constraints:

1 <= preorder.length <= 30
1 <= preorder[i] <= preorder.length
All the values of preorder are unique.
postorder.length == preorder.length
1 <= postorder[i] <= postorder.length
All the values of postorder are unique.
It is guaranteed that preorder and postorder are the preorder traversal and postorder traversal of the same binary tree.

---

890. Find and Replace Pattern

Given a list of strings words and a string pattern, return a list of words[i] that match pattern. You may return the ans
wer in any order.
A word matches the pattern if there exists a permutation of letters p so that after replacing every letter x in the patter
n with p(x), we get the desired word.
Recall that a permutation of letters is a bijection from letters to letters: every letter maps to another letter, and no two
letters map to the same letter.

Example 1:

Input: words = ["abc","deq","mee","aqq","dkd","ccc"], pattern = "abb"
Output: ["mee","aqq"]
Explanation: "mee" matches the pattern because there is a permutation {a -> m, b -> e, ...}.
"ccc" does not match the pattern because {a -> c, b -> c, ...} is not a permutation, since a and b map to the same lette
r.

Example 2:

Input: words = ["a","b","c"], pattern = "a"
Output: ["a","b","c"]

Constraints:

1 <= pattern.length <= 20
1 <= words.length <= 50
words[i].length == pattern.length
pattern and words[i] are lowercase English letters.

---

891. Sum of Subsequence Widths

The width of a sequence is the difference between the maximum and minimum elements in the sequence.
Given an array of integers nums, return the sum of the widths of all the non-empty subsequences of nums. Since the answer may be very large, return it modulo 109 + 7.
A subsequence is a sequence that can be derived from an array by deleting some or no elements without changing th e order of the remaining elements. For example, [3,6,2,7] is a subsequence of the array [0,3,1,6,2,2,7].

Example 1:

Input: nums = [2,1,3]
Output: 6
Explanation: The subsequences are [1], [2], [3], [2,1], [2,3], [1,3], [2,1,3].
The corresponding widths are 0, 0, 0, 1, 1, 2, 2.
The sum of these widths is 6.

Example 2:

Input: nums = [2]
Output: 0

Constraints:

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

---

892. Surface Area of 3D Shapes

You are given an n x n grid where you have placed some 1 x 1 x 1 cubes. Each value v = grid[i][j] represents a tower of v cubes placed on top of cell (i, j).
After placing these cubes, you have decided to glue any directly adjacent cubes to each other, forming several irregul ar 3D shapes.
Return the total surface area of the resulting shapes.
Note: The bottom face of each shape counts toward its surface area.

Example 1:

Input: grid = [[2]]
Output: 10

Example 2:

Input: grid = [[1,2],[3,4]]
Output: 34

Example 3:

Input: grid = [[1,0],[0,2]]
Output: 16

Example 4:

Input: grid = [[1,1,1],[1,0,1],[1,1,1]]
Output: 32

Example 5:

Input: grid = [[2,2,2],[2,1,2],[2,2,2]]
Output: 46

Constraints:

n == grid.length
n == grid[i].length
1 <= n <= 50
0 <= grid[i][j] <= 50

---

893. Groups of Special-Equivalent Strings

You are given an array of strings of the same length words.
In one move, you can swap any two even indexed characters or any two odd indexed characters of a string words[i].
Two strings words[i] and words[j] are special-equivalent if after any number of moves, words[i] == words[j].

For example, words[i] = "zzxy" and words[j] = "xyzz" are special-equivalent because we may make the moves "zzxy " -> "xzzy" -> "xyzz".

A group of special-equivalent strings from words is a non-empty subset of words such that:

Every pair of strings in the group are special equivalent, and
The group is the largest size possible (i.e., there is not a string words[i] not in the group such that words[i] is special-equivalent to every string in the group).

Return the number of groups of special-equivalent strings from words.

Example 1:

Input: words = ["abcd","cdab","cbad","xyzz","zzxy","zzyx"]
Output: 3
Explanation:
One group is ["abcd", "cdab", "cbad"], since they are all pairwise special equivalent, and none of the other strings is all pairwise special equivalent to these.
The other two groups are ["xyzz", "zzxy"] and ["zzyx"].
Note that in particular, "zzxy" is not special equivalent to "zzyx".

Example 2:

Input: words = ["abc","acb","bac","bca","cab","cba"]
Output: 3

Constraints:

1 <= words.length <= 1000
1 <= words[i].length <= 20
words[i] consist of lowercase English letters.
All the strings are of the same length.

---

894. All Possible Full Binary Trees

Given an integer n, return a list of all possible full binary trees with n nodes. Each node of each tree in the answer m ust have Node.val == 0.
Each element of the answer is the root node of one possible tree. You may return the final list of trees in any order.
A full binary tree is a binary tree where each node has exactly 0 or 2 children.

Example 1:

Input: n = 7
Output: [[0,0,0,null,null,0,0,null,null,0,0],[0,0,0,null,null,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,null,null,null,null,0,0],[0,0,0,0,0,null,null,0,0]]

Example 2:

Input: n = 3
Output: [[0,0,0]]

Constraints:

1 <= n <= 20

---

895. Maximum Frequency Stack

Design a stack-like data structure to push elements to the stack and pop the most frequent element from the stack.
Implement the FreqStack class:

FreqStack() constructs an empty frequency stack.
void push(int val) pushes an integer val onto the top of the stack.
int pop() removes and returns the most frequent element in the stack.

If there is a tie for the most frequent element, the element closest to the stack's top is removed and returned.

Example 1:

Input
["FreqStack", "push", "push", "push", "push", "push", "push", "pop", "pop", "pop", "pop"]
[[], [5], [7], [5], [7], [4], [5], [], [], [], []]
Output
[null, null, null, null, null, null, null, 5, 7, 5, 4]

Explanation
FreqStack freqStack = new FreqStack();
freqStack.push(5); // The stack is [5]
freqStack.push(7); // The stack is [5,7]
freqStack.push(5); // The stack is [5,7,5]
freqStack.push(7); // The stack is [5,7,5,7]
freqStack.push(4); // The stack is [5,7,5,7,4]
freqStack.push(5); // The stack is [5,7,5,7,4,5]
freqStack.pop(); // return 5, as 5 is the most frequent. The stack becomes [5,7,5,7,4].
freqStack.pop(); // return 7, as 5 and 7 is the most frequent, but 7 is closest to the top. The stack becomes [5,7,5,4].
freqStack.pop(); // return 5, as 5 is the most frequent. The stack becomes [5,7,4].
freqStack.pop(); // return 4, as 4, 5 and 7 is the most frequent, but 4 is closest to the top. The stack becomes [5,7].

Constraints:

0 <= val <= 109
At most 2 * 104 calls will be made to push and pop.
It is guaranteed that there will be at least one element in the stack before calling pop.

---

896. Monotonic Array

An array is monotonic if it is either monotone increasing or monotone decreasing.
An array nums is monotone increasing if for all i <= j, nums[i] <= nums[j]. An array nums is monotone decreasing if for all i <= j, nums[i] >= nums[j].
Given an integer array nums, return true if the given array is monotonic, or false otherwise.

Example 1:
Input: nums = [1,2,2,3]
Output: true
Example 2:
Input: nums = [6,5,4,4]
Output: true
Example 3:
Input: nums = [1,3,2]
Output: false
Example 4:
Input: nums = [1,2,4,5]
Output: true
Example 5:
Input: nums = [1,1,1]
Output: true

Constraints:

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

---

897. Increasing Order Search Tree

Given the root of a binary search tree, rearrange the tree in in-order so that the leftmost node in the tree is now the ro ot of the tree, and every node has no left child and only one right child.

Example 1:

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

Example 2:

Input: root = [5,1,7]
Output: [1,null,5,null,7]

Constraints:

The number of nodes in the given tree will be in the range [1, 100].
0 <= Node.val <= 1000

---

898. Bitwise ORs of Subarrays

We have an array arr of non-negative integers.
For every (contiguous) subarray sub = [arr[i], arr[i + 1], ..., arr[j]] (with i <= j), we take the bitwise OR of all the ele ments in sub, obtaining a result arr[i] | arr[i + 1] | ... | arr[j].
Return the number of possible results. Results that occur more than once are only counted once in the final answer

Example 1:

Input: arr = [0]
Output: 1
Explanation: There is only one possible result: 0.

Example 2:

Input: arr = [1,1,2]
Output: 3
Explanation: The possible subarrays are [1], [1], [2], [1, 1], [1, 2], [1, 1, 2].
These yield the results 1, 1, 2, 1, 3, 3.
There are 3 unique values, so the answer is 3.

Example 3:

Input: arr = [1,2,4]
Output: 6
Explanation: The possible results are 1, 2, 3, 4, 6, and 7.

Constraints:

1 <= nums.length <= 5 * 104
0 <= nums[i] <= 109

---

899. Orderly Queue

You are given a string s and an integer k. You can choose one of the first k letters of s and append it at the end of the string..
Return the lexicographically smallest string you could have after applying the mentioned step any number of moves.

Example 1:

Input: s = "cba", k = 1
Output: "acb"
Explanation:
In the first move, we move the 1st character 'c' to the end, obtaining the string "bac".
In the second move, we move the 1st character 'b' to the end, obtaining the final result "acb".

Example 2:

Input: s = "baaca", k = 3
Output: "aaabc"
Explanation:
In the first move, we move the 1st character 'b' to the end, obtaining the string "aacab".
In the second move, we move the 3rd character 'c' to the end, obtaining the final result "aaabc".

Constraints:

1 <= k <= s.length <= 1000
s consist of lowercase English letters.

---

900. RLE Iterator

We can use run-length encoding (i.e., RLE) to encode a sequence of integers. In a run-length encoded array of even l ength encoding (0-indexed), for all even i, encoding[i] tells us the number of times that the non-negative integer valu e encoding[i + 1] is repeated in the sequence.

For example, the sequence arr = [8,8,8,5,5] can be encoded to be encoding = [3,8,2,5]. encoding = [3,8,0,9,2,5] and e ncoding = [2,8,1,8,2,5] are also valid RLE of arr.

Given a run-length encoded array, design an iterator that iterates through it.
Implement the RLEIterator class:

RLEIterator(int[] encoded) Initializes the object with the encoded array encoded.

int next(int n) Exhausts the next n elements and returns the last element exhausted in this way. If there is no element left to exhaust, return -1 instead.

Example 1:

Input
["RLEIterator", "next", "next", "next", "next"]
[[[3, 8, 0, 9, 2, 5]], [2], [1], [1], [2]]
Output
[null, 8, 8, 5, -1]

Explanation
RLEIterator rLEIterator = new RLEIterator([3, 8, 0, 9, 2, 5]); // This maps to the sequence [8,8,8,5,5].
rLEIterator.next(2); // exhausts 2 terms of the sequence, returning 8. The remaining sequence is now [8, 5, 5].
rLEIterator.next(1); // exhausts 1 term of the sequence, returning 8. The remaining sequence is now [5, 5].
rLEIterator.next(1); // exhausts 1 term of the sequence, returning 5. The remaining sequence is now [5].
rLEIterator.next(2); // exhausts 2 terms, returning -1. This is because the first term exhausted was 5,
but the second term did not exist. Since the last term exhausted does not exist, we return -1.

Constraints:

2 <= encoding.length <= 1000
encoding.length is even.
0 <= encoding[i] <= 109
1 <= n <= 109
At most 1000 calls will be made to next.

---

901. Online Stock Span

Design an algorithm that collects daily price quotes for some stock and returns the span of that stock's price for the c urrent day.
The span of the stock's price today is defined as the maximum number of consecutive days (starting from today and going backward) for which the stock price was less than or equal to today's price.

For example, if the price of a stock over the next 7 days were [100,80,60,70,60,75,85], then the stock spans would b e [1,1,1,2,1,4,6].

Implement the StockSpanner class:

StockSpanner() Initializes the object of the class.
int next(int price) Returns the span of the stock's price given that today's price is price.

Example 1:

Input
["StockSpanner", "next", "next", "next", "next", "next", "next", "next"]
[[], [100], [80], [60], [70], [60], [75], [85]]
Output
[null, 1, 1, 1, 2, 1, 4, 6]

Explanation
StockSpanner stockSpanner = new StockSpanner();
stockSpanner.next(100); // return 1
stockSpanner.next(80); // return 1
stockSpanner.next(60); // return 1
stockSpanner.next(70); // return 2
stockSpanner.next(60); // return 1
stockSpanner.next(75); // return 4, because the last 4 prices (including today's price of 75) were less than or equal to today's price.
stockSpanner.next(85); // return 6

Constraints:

1 <= price <= 105
At most 104 calls will be made to next.

---

902. Numbers At Most N Given Digit Set

Given an array of digits which is sorted in non-decreasing order. You can write numbers using each digits[i] as many times as we want. For example, if digits = ['1','3','5'], we may write numbers such as '13', '551', and '1351315'. Return the number of positive integers that can be generated that are less than or equal to a given integer n.

Example 1:

Input: digits = ["1","3","5","7"], n = 100
Output: 20
Explanation:
The 20 numbers that can be written are:
1, 3, 5, 7, 11, 13, 15, 17, 31, 33, 35, 37, 51, 53, 55, 57, 71, 73, 75, 77.

Example 2:

Input: digits = ["1","4","9"], n = 1000000000
Output: 29523
Explanation:
We can write 3 one digit numbers, 9 two digit numbers, 27 three digit numbers,
81 four digit numbers, 243 five digit numbers, 729 six digit numbers,
2187 seven digit numbers, 6561 eight digit numbers, and 19683 nine digit numbers.

In total, this is 29523 integers that can be written using the digits array.

Example 3:

Input: digits = ["7"], n = 8
Output: 1

Constraints:

1 <= digits.length <= 9
digits[i].length == 1
digits[i] is a digit from '1' to '9'.
All the values in digits are unique.
digits is sorted in non-decreasing order.
1 <= n <= 109

---

903. Valid Permutations for DI Sequence

You are given a string s of length n where s[i] is either:

'D' means decreasing, or
'I' means increasing.

A permutation perm of n + 1 integers of all the integers in the range [0, n] is called a valid permutation if for all vali d i:

If s[i] == 'D', then perm[i] > perm[i + 1], and
If s[i] == 'I', then perm[i] < perm[i + 1].

Return the number of valid permutations perm. Since the answer may be large, return it modulo 109 + 7.

Example 1:

Input: s = "DID"
Output: 5
Explanation: The 5 valid permutations of (0, 1, 2, 3) are:
(1, 0, 3, 2)
(2, 0, 3, 1)
(2, 1, 3, 0)
(3, 0, 2, 1)
(3, 1, 2, 0)

Example 2:

Input: s = "D"

Output: 1

Constraints:

n == s.length
1 <= n <= 200
s[i] is either 'I' or 'D'.

---

904. Fruit Into Baskets

You are visiting a farm that has a single row of fruit trees arranged from left to right. The trees are represented by an integer array fruits where fruits[i] is the type of fruit the ith tree produces.
You want to collect as much fruit as possible. However, the owner has some strict rules that you must follow:

You only have two baskets, and each basket can only hold a single type of fruit. There is no limit on the amount of fr uit each basket can hold.
Starting from any tree of your choice, you must pick exactly one fruit from every tree (including the start tree) while moving to the right. The picked fruits must fit in one of your baskets.
Once you reach a tree with fruit that cannot fit in your baskets, you must stop.

Given the integer array fruits, return the maximum number of fruits you can pick.

Example 1:

Input: fruits = [1,2,1]
Output: 3
Explanation: We can pick from all 3 trees.

Example 2:

Input: fruits = [0,1,2,2]
Output: 3
Explanation: We can pick from trees [1,2,2].
If we had started at the first tree, we would only pick from trees [0,1].

Example 3:

Input: fruits = [1,2,3,2,2]
Output: 4
Explanation: We can pick from trees [2,3,2,2].
If we had started at the first tree, we would only pick from trees [1,2].

Example 4:

Input: fruits = [3,3,3,1,2,1,1,2,3,3,4]

Output: 5
Explanation: We can pick from trees [1,2,1,1,2].

Constraints:

1 <= fruits.length <= 105
0 <= fruits[i] < fruits.length

---

905. Sort Array By Parity

Given an integer array nums, move all the even integers at the beginning of the array followed by all the odd integers
.
Return any array that satisfies this condition.

Example 1:

Input: nums = [3,1,2,4]
Output: [2,4,3,1]
Explanation: The outputs [4,2,3,1], [2,4,1,3], and [4,2,1,3] would also be accepted.

Example 2:

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

Constraints:

1 <= nums.length <= 5000
0 <= nums[i] <= 5000

---

906. Super Palindromes

Let's say a positive integer is a super-palindrome if it is a palindrome, and it is also the square of a palindrome.
Given two positive integers left and right represented as strings, return the number of super-palindromes integers in t
he inclusive range [left, right].

Example 1:

Input: left = "4", right = "1000"
Output: 4
Explanation: 4, 9, 121, and 484 are superpalindromes.
Note that 676 is not a superpalindrome: 26 * 26 = 676, but 26 is not a palindrome.

Example 2:

Input: left = "1", right = "2"
Output: 1

Constraints:

1 <= left.length, right.length <= 18
left and right consist of only digits.
left and right cannot have leading zeros.
left and right represent integers in the range [1, 1018 - 1].
left is less than or equal to right.

---

907. Sum of Subarray Minimums

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

---

908. Smallest Range I

You are given an integer array nums and an integer k.
In one operation, you can choose any index i where 0 <= i < nums.length and change nums[i] to nums[i] + x where x is an integer from the range [-k, k]. You can apply this operation at most once for each index i.
The score of nums is the difference between the maximum and minimum elements in nums.
Return the minimum score of nums after applying the mentioned operation at most once for each index in it.

Example 1:

Input: nums = [1], k = 0
Output: 0
Explanation: The score is max(nums) - min(nums) = 1 - 1 = 0.

Example 2:

Input: nums = [0,10], k = 2
Output: 6
Explanation: Change nums to be [2, 8]. The score is max(nums) - min(nums) = 8 - 2 = 6.

Example 3:

Input: nums = [1,3,6], k = 3
Output: 0
Explanation: Change nums to be [4, 4, 4]. The score is max(nums) - min(nums) = 4 - 4 = 0.

Constraints:

1 <= nums.length <= 104
0 <= nums[i] <= 104
0 <= k <= 104

---

909. Snakes and Ladders

You are given an n x n integer matrix board where the cells are labeled from 1 to n2 in a Boustrophedon style startin g from the bottom left of the board (i.e. board[n - 1][0]) and alternating direction each row.
You start on square 1 of the board. In each move, starting from square curr, do the following:

Choose a destination square next with a label in the range [curr + 1, min(curr + 6, n2)].

This choice simulates the result of a standard 6-sided die roll: i.e., there are always at most 6 destinations, regardless of the size of the board.

If next has a snake or ladder, you must move to the destination of that snake or ladder. Otherwise, you move to next.
The game ends when you reach the square n2.

A board square on row r and column c has a snake or ladder if board[r][c] != -1. The destination of that snake or ladd er is board[r][c]. Squares 1 and n2 do not have a snake or ladder.
Note that you only take a snake or ladder at most once per move. If the destination to a snake or ladder is the start of another snake or ladder, you do not follow the subsequent snake or ladder.

For example, suppose the board is [[-1,4],[-1,3]], and on the first move, your destination square is 2. You follow the ladder to square 3, but do not follow the subsequent ladder to 4.

Return the least number of moves required to reach the square n2. If it is not possible to reach the square, return -1.

Example 1:

Input: board = [[-1,-1,-1,-1,-1,-1],[-1,-1,-1,-1,-1,-1],[-1,-1,-1,-1,-1,-1],[-1,35,-1,-1,13,-1],[-1,-1,-1,-1,-1,-1],[-1,15,-1,-1,-1,-1]]
Output: 4
Explanation:
In the beginning, you start at square 1 (at row 5, column 0).
You decide to move to square 2 and must take the ladder to square 15.
You then decide to move to square 17 and must take the snake to square 13.
You then decide to move to square 14 and must take the ladder to square 35.
You then decide to move to square 36, ending the game.
This is the lowest possible number of moves to reach the last square, so return 4.

Example 2:

Input: board = [[-1,-1],[-1,3]]
Output: 1

Constraints:

n == board.length == board[i].length
2 <= n <= 20
grid[i][j] is either -1 or in the range [1, n2].
The squares labeled 1 and n2 do not have any ladders or snakes.

---

910. Smallest Range II

You are given an integer array nums and an integer k.
For each index i where 0 <= i < nums.length, change nums[i] to be either nums[i] + k or nums[i] - k.
The score of nums is the difference between the maximum and minimum elements in nums.
Return the minimum score of nums after changing the values at each index.

Example 1:

Input: nums = [1], k = 0
Output: 0
Explanation: The score is max(nums) - min(nums) = 1 - 1 = 0.

Example 2:

Input: nums = [0,10], k = 2
Output: 6
Explanation: Change nums to be [2, 8]. The score is max(nums) - min(nums) = 8 - 2 = 6.

Example 3:

Input: nums = [1,3,6], k = 3
Output: 3
Explanation: Change nums to be [4, 6, 3]. The score is max(nums) - min(nums) = 6 - 3 = 3.

Constraints:

1 <= nums.length <= 104
0 <= nums[i] <= 104
0 <= k <= 104

---

911. Online Election

You are given two integer arrays persons and times. In an election, the ith vote was cast for persons[i] at time times[i].
For each query at a time t, find the person that was leading the election at time t. Votes cast at time t will count towar ds our query. In the case of a tie, the most recent vote (among tied candidates) wins.
Implement the TopVotedCandidate class:

TopVotedCandidate(int[] persons, int[] times) Initializes the object with the persons and times arrays.
int q(int t) Returns the number of the person that was leading the election at time t according to the mentioned rules.

Example 1:

Input
["TopVotedCandidate", "q", "q", "q", "q", "q", "q"]
[[[0, 1, 1, 0, 0, 1, 0], [0, 5, 10, 15, 20, 25, 30]], [3], [12], [25], [15], [24], [8]]
Output
[null, 0, 1, 1, 0, 0, 1]

Explanation
TopVotedCandidate topVotedCandidate = new TopVotedCandidate([0, 1, 1, 0, 0, 1, 0], [0, 5, 10, 15, 20, 25, 30]);
topVotedCandidate.q(3); // return 0, At time 3, the votes are [0], and 0 is leading.
topVotedCandidate.q(12); // return 1, At time 12, the votes are [0,1,1], and 1 is leading.
topVotedCandidate.q(25); // return 1, At time 25, the votes are [0,1,1,0,0,1], and 1 is leading (as ties go to the most r
ecent vote.)
topVotedCandidate.q(15); // return 0
topVotedCandidate.q(24); // return 0
topVotedCandidate.q(8); // return 1

Constraints:

1 <= persons.length <= 5000
times.length == persons.length
0 <= persons[i] < persons.length
0 <= times[i] <= 109
times is sorted in a strictly increasing order.
times[0] <= t <= 109
At most 104 calls will be made to q.

---

912. Sort an Array

Given an array of integers nums, sort the array in ascending order.

Example 1:
Input: nums = [5,2,3,1]
Output: [1,2,3,5]
Example 2:
Input: nums = [5,1,1,2,0,0]
Output: [0,0,1,1,2,5]

Constraints:

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

---

913. Cat and Mouse

A game on an undirected graph is played by two players, Mouse and Cat, who alternate turns.
The graph is given as follows: graph[a] is a list of all nodes b such that ab is an edge of the graph.
The mouse starts at node 1 and goes first, the cat starts at node 2 and goes second, and there is a hole at node 0.
During each player's turn, they must travel along one edge of the graph that meets where they are. For example, if th e Mouse is at node 1, it must travel to any node in graph[1].
Additionally, it is not allowed for the Cat to travel to the Hole (node 0.)
Then, the game can end in three ways:

If ever the Cat occupies the same node as the Mouse, the Cat wins.
If ever the Mouse reaches the Hole, the Mouse wins.
If ever a position is repeated (i.e., the players are in the same position as a previous turn, and it is the same player's t urn to move), the game is a draw.

Given a graph, and assuming both players play optimally, return

1 if the mouse wins the game,
2 if the cat wins the game, or
0 if the game is a draw.

Example 1:

Input: graph = [[2,5],[3],[0,4,5],[1,4,5],[2,3],[0,2,3]]
Output: 0

Example 2:

Input: graph = [[1,3],[0],[3],[0,2]]
Output: 1

Constraints:

3 <= graph.length <= 50
1 <= graph[i].length < graph.length
0 <= graph[i][j] < graph.length
graph[i][j] != i
graph[i] is unique.
The mouse and the cat can always move.

---

914. X of a Kind in a Deck of Cards

In a deck of cards, each card has an integer written on it.
Return true if and only if you can choose X >= 2 such that it is possible to split the entire deck into 1 or more groups of cards, where:

Each group has exactly X cards.
All the cards in each group have the same integer.

Example 1:

Input: deck = [1,2,3,4,4,3,2,1]
Output: true
Explanation: Possible partition [1,1],[2,2],[3,3],[4,4].

Example 2:

Input: deck = [1,1,1,2,2,2,3,3]
Output: false
Explanation: No possible partition.

Example 3:

Input: deck = [1]
Output: false
Explanation: No possible partition.

Example 4:

Input: deck = [1,1]
Output: true
Explanation: Possible partition [1,1].

Example 5:

Input: deck = [1,1,2,2,2,2]
Output: true
Explanation: Possible partition [1,1],[2,2],[2,2].

Constraints:

1 <= deck.length <= 104
0 <= deck[i] < 104

---

915. Partition Array into Disjoint Intervals

Given an integer array nums, partition it into two (contiguous) subarrays left and right so that:

Every element in left is less than or equal to every element in right.
left and right are non-empty.
left has the smallest possible size.

Return the length of left after such a partitioning.
Test cases are generated such that partitioning exists.

Example 1:

Input: nums = [5,0,3,8,6]
Output: 3
Explanation: left = [5,0,3], right = [8,6]

Example 2:

Input: nums = [1,1,1,0,6,12]
Output: 4
Explanation: left = [1,1,1,0], right = [6,12]

Constraints:

2 <= nums.length <= 105
0 <= nums[i] <= 106
There is at least one valid answer for the given input.

---

916. Word Subsets

You are given two string arrays words1 and words2.
A string b is a subset of string a if every letter in b occurs in a including multiplicity.

For example, "wrr" is a subset of "warrior" but is not a subset of "world".

A string a from words1 is universal if for every string b in words2, b is a subset of a.
Return an array of all the universal strings in words1. You may return the answer in any order.

Example 1:
Input: words1 = ["amazon","apple","facebook","google","leetcode"], words2 = ["e","o"]
Output: ["facebook","google","leetcode"]
Example 2:
Input: words1 = ["amazon","apple","facebook","google","leetcode"], words2 = ["l","e"]
Output: ["apple","google","leetcode"]
Example 3:
Input: words1 = ["amazon","apple","facebook","google","leetcode"], words2 = ["e","oo"]
Output: ["facebook","google"]
Example 4:
Input: words1 = ["amazon","apple","facebook","google","leetcode"], words2 = ["lo","eo"]
Output: ["google","leetcode"]
Example 5:
Input: words1 = ["amazon","apple","facebook","google","leetcode"], words2 = ["ec","oc","ceo"]
Output: ["facebook","leetcode"]

Constraints:

1 <= words1.length, words2.length <= 104
1 <= words1[i].length, words2[i].length <= 10
words1[i] and words2[i] consist only of lowercase English letters.
All the strings of words1 are unique.

---

917. Reverse Only Letters

Given a string s, reverse the string according to the following rules:

All the characters that are not English letters remain in the same position.
All the English letters (lowercase or uppercase) should be reversed.

Return s after reversing it.

Example 1:
Input: s = "ab-cd"
Output: "dc-ba"
Example 2:
Input: s = "a-bC-dEf-ghIj"
Output: "j-Ih-gfE-dCba"
Example 3:
Input: s = "Test1ng-Leet=code-Q!"
Output: "Qedo1ct-eeLg=ntse-T!"

Constraints:

1 <= s.length <= 100
s consists of characters with ASCII values in the range [33, 122].
s does not contain '\"' or '\\'.

---

918. Maximum Sum Circular Subarray

Given a circular integer array nums of length n, return the maximum possible sum of a non-empty subarray of nums.
A circular array means the end of the array connects to the beginning of the array. Formally, the next element of nu
ms[i] is nums[(i + 1) % n] and the previous element of nums[i] is nums[(i - 1 + n) % n].
A subarray may only include each element of the fixed buffer nums at most once. Formally, for a subarray nums[i],
nums[i + 1], ..., nums[j], there does not exist i <= k1, k2 <= j with k1 % n == k2 % n.

Example 1:

Input: nums = [1,-2,3,-2]
Output: 3
Explanation: Subarray [3] has maximum sum 3

Example 2:

Input: nums = [5,-3,5]
Output: 10
Explanation: Subarray [5,5] has maximum sum 5 + 5 = 10

Example 3:

Input: nums = [3,-1,2,-1]
Output: 4
Explanation: Subarray [2,-1,3] has maximum sum 2 + (-1) + 3 = 4

Example 4:

Input: nums = [3,-2,2,-3]
Output: 3
Explanation: Subarray [3] and [3,-2,2] both have maximum sum 3

Example 5:

Input: nums = [-2,-3,-1]
Output: -1
Explanation: Subarray [-1] has maximum sum -1

Constraints:

n == nums.length
1 <= n <= 3 * 104
-3 * 104 <= nums[i] <= 3 * 104

---

919. Complete Binary Tree Inserter

A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled, and all nod es are as far left as possible.
Design an algorithm to insert a new node to a complete binary tree keeping it complete after the insertion.
Implement the CBTInserter class:

CBTInserter(TreeNode root) Initializes the data structure with the root of the complete binary tree.
int insert(int v) Inserts a TreeNode into the tree with value Node.val == val so that the tree remains complete, and ret urns the value of the parent of the inserted TreeNode.
TreeNode get_root() Returns the root node of the tree.

Example 1:

Input
["CBTInserter", "insert", "insert", "get_root"]
[[[1, 2]], [3], [4], []]
Output
[null, 1, 2, [1, 2, 3, 4]]

Explanation
CBTInserter cBTInserter = new CBTInserter([1, 2]);
cBTInserter.insert(3); // return 1
cBTInserter.insert(4); // return 2
cBTInserter.get_root(); // return [1, 2, 3, 4]

Constraints:

The number of nodes in the tree will be in the range [1, 1000].
0 <= Node.val <= 5000
root is a complete binary tree.
0 <= val <= 5000
At most 104 calls will be made to insert and get_root.

---

920. Number of Music Playlists

Your music player contains n different songs. You want to listen to goal songs (not necessarily different) during you r trip. To avoid boredom, you will create a playlist so that:

Every song is played at least once.
A song can only be played again only if k other songs have been played.

Given n, goal, and k, return the number of possible playlists that you can create. Since the answer can be very large, return it modulo 109 + 7.

Example 1:

Input: n = 3, goal = 3, k = 1
Output: 6
Explanation: There are 6 possible playlists: [1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], and [3, 2, 1].

Example 2:

Input: n = 2, goal = 3, k = 0
Output: 6
Explanation: There are 6 possible playlists: [1, 1, 2], [1, 2, 1], [2, 1, 1], [2, 2, 1], [2, 1, 2], and [1, 2, 2].

Example 3:

Input: n = 2, goal = 3, k = 1
Output: 2
Explanation: There are 2 possible playlists: [1, 2, 1] and [2, 1, 2].

Constraints:

0 <= k < n <= goal <= 100

---

921. Minimum Add to Make Parentheses Valid

A parentheses string is valid if and only if:

It is the empty string,
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.

You are given a parentheses string s. In one move, you can insert a parenthesis at any position of the string.

For example, if s = "()))", you can insert an opening parenthesis to be "(()))" or a closing parenthesis to be "())))".

Return the minimum number of moves required to make s valid.

Example 1:

Input: s = "())"
Output: 1

Example 2:

Input: s = "((("
Output: 3

Example 3:

Input: s = "()"
Output: 0

Example 4:

Input: s = "()))(("
Output: 4

Constraints:

1 <= s.length <= 1000
s[i] is either '(' or ')'.

---

922. Sort Array By Parity II

Given an array of integers nums, half of the integers in nums are odd, and the other half are even.
Sort the array so that whenever nums[i] is odd, i is odd, and whenever nums[i] is even, i is even.
Return any answer array that satisfies this condition.

Example 1:

Input: nums = [4,2,5,7]
Output: [4,5,2,7]
Explanation: [4,7,2,5], [2,5,4,7], [2,7,4,5] would also have been accepted.

Example 2:

Input: nums = [2,3]

Output: [2,3]

Constraints:

2 <= nums.length <= 2 * 104
nums.length is even.
Half of the integers in nums are even.
0 <= nums[i] <= 1000

Follow Up: Could you solve it in-place?

---

923. 3Sum With Multiplicity

Given an integer array arr, and an integer target, return the number of tuples i, j, k such that i < j < k and arr[i] + arr[j] + arr[k] == target.
As the answer can be very large, return it modulo 109 + 7.

Example 1:

Input: arr = [1,1,2,2,3,3,4,4,5,5], target = 8
Output: 20
Explanation:
Enumerating by the values (arr[i], arr[j], arr[k]):
(1, 2, 5) occurs 8 times;
(1, 3, 4) occurs 8 times;
(2, 2, 4) occurs 2 times;
(2, 3, 3) occurs 2 times.

Example 2:

Input: arr = [1,1,2,2,2,2], target = 5
Output: 12
Explanation:
arr[i] = 1, arr[j] = arr[k] = 2 occurs 12 times:
We choose one 1 from [1,1] in 2 ways,
and two 2s from [2,2,2,2] in 6 ways.

Constraints:

3 <= arr.length <= 3000
0 <= arr[i] <= 100
0 <= target <= 300

---

924. Minimize Malware Spread

You are given a network of n nodes represented as an n x n adjacency matrix graph, where the ith node is directly co nnected to the jth node if graph[i][j] == 1.
Some nodes initial are initially infected by malware. Whenever two nodes are directly connected, and at least one of those two nodes is infected by malware, both nodes will be infected by malware. This spread of malware will contin ue until no more nodes can be infected in this manner.
Suppose M(initial) is the final number of nodes infected with malware in the entire network after the spread of malw are stops. We will remove exactly one node from initial.
Return the node that, if removed, would minimize M(initial). If multiple nodes could be removed to minimize M(init ial), return such a node with the smallest index.
Note that if a node was removed from the initial list of infected nodes, it might still be infected later due to the malw are spread.

Example 1:
Input: graph = [[1,1,0],[1,1,0],[0,0,1]], initial = [0,1]
Output: 0
Example 2:
Input: graph = [[1,0,0],[0,1,0],[0,0,1]], initial = [0,2]
Output: 0
Example 3:
Input: graph = [[1,1,1],[1,1,1],[1,1,1]], initial = [1,2]
Output: 1

Constraints:

n == graph.length
n == graph[i].length
2 <= n <= 300
graph[i][j] is 0 or 1.
graph[i][j] == graph[j][i]
graph[i][i] == 1
1 <= initial.length <= n
0 <= initial[i] <= n - 1
All the integers in initial are unique.

---

925. Long Pressed Name

Your friend is typing his name into a keyboard. Sometimes, when typing a character c, the key might get long presse d, and the character will be typed 1 or more times.
You examine the typed characters of the keyboard. Return True if it is possible that it was your friends name, with s ome characters (possibly none) being long pressed.

Example 1:

Input: name = "alex", typed = "aaleex"
Output: true
Explanation: 'a' and 'e' in 'alex' were long pressed.

Example 2:

Input: name = "saeed", typed = "ssaaedd"
Output: false
Explanation: 'e' must have been pressed twice, but it wasn't in the typed output.

Example 3:

Input: name = "leelee", typed = "lleeelee"
Output: true

Example 4:

Input: name = "laiden", typed = "laiden"
Output: true
Explanation: It's not necessary to long press any character.

Constraints:

1 <= name.length <= 1000
1 <= typed.length <= 1000
name and typed contain only lowercase English letters.

---

926. Flip String to Monotone Increasing

A binary string is monotone increasing if it consists of some number of 0's (possibly none), followed by some numb er of 1's (also possibly none).
You are given a binary string s. You can flip s[i] changing it from 0 to 1 or from 1 to 0.
Return the minimum number of flips to make s monotone increasing.

Example 1:

Input: s = "00110"
Output: 1

Explanation: We flip the last digit to get 00111.

Example 2:

Input: s = "010110"
Output: 2
Explanation: We flip to get 011111, or alternatively 000111.

Example 3:

Input: s = "00011000"
Output: 2
Explanation: We flip to get 00000000.

Constraints:

1 <= s.length <= 105
s[i] is either '0' or '1'.

---

927. Three Equal Parts

You are given an array arr which consists of only zeros and ones, divide the array into three non-empty parts such th at all of these parts represent the same binary value.
If it is possible, return any [i, j] with i + 1 < j, such that:

arr[0], arr[1], ..., arr[i] is the first part,
arr[i + 1], arr[i + 2], ..., arr[j - 1] is the second part, and
arr[j], arr[j + 1], ..., arr[arr.length - 1] is the third part.
All three parts have equal binary values.

If it is not possible, return [-1, -1].
Note that the entire part is used when considering what binary value it represents. For example, [1,1,0] represents 6 i n decimal, not 3. Also, leading zeros are allowed, so [0,1,1] and [1,1] represent the same value.

Example 1:
Input: arr = [1,0,1,0,1]
Output: [0,3]
Example 2:
Input: arr = [1,1,0,1,1]
Output: [-1,-1]
Example 3:
Input: arr = [1,1,0,0,1]
Output: [0,2]

Constraints:

3 <= arr.length <= 3 * 104
arr[i] is 0 or 1

---

928. Minimize Malware Spread II

You are given a network of n nodes represented as an n x n adjacency matrix graph, where the ith node is directly co nnected to the jth node if graph[i][j] == 1.
Some nodes initial are initially infected by malware. Whenever two nodes are directly connected, and at least one of those two nodes is infected by malware, both nodes will be infected by malware. This spread of malware will contin ue until no more nodes can be infected in this manner.
Suppose M(initial) is the final number of nodes infected with malware in the entire network after the spread of malw are stops.
We will remove exactly one node from initial, completely removing it and any connections from this node to any oth er node.
Return the node that, if removed, would minimize M(initial). If multiple nodes could be removed to minimize M(init ial), return such a node with the smallest index.

Example 1:
Input: graph = [[1,1,0],[1,1,0],[0,0,1]], initial = [0,1]
Output: 0
Example 2:
Input: graph = [[1,1,0],[1,1,1],[0,1,1]], initial = [0,1]
Output: 1
Example 3:
Input: graph = [[1,1,0,0],[1,1,1,0],[0,1,1,1],[0,0,1,1]], initial = [0,1]
Output: 1

Constraints:

n == graph.length
n == graph[i].length
2 <= n <= 300
graph[i][j] is 0 or 1.
graph[i][j] == graph[j][i]
graph[i][i] == 1
1 <= initial.length < n
0 <= initial[i] <= n - 1
All the integers in initial are unique.

---

929. Unique Email Addresses

Every valid email consists of a local name and a domain name, separated by the '@' sign. Besides lowercase letters, t he email may contain one or more '.' or '+'.

For example, in "alice@leetcode.com", "alice" is the local name, and "leetcode.com" is the domain name.

If you add periods '.' between some characters in the local name part of an email address, mail sent there will be forw arded to the same address without dots in the local name. Note that this rule does not apply to domain names.

For example, "alice.z@leetcode.com" and "alicez@leetcode.com" forward to the same email address.

If you add a plus '+' in the local name, everything after the first plus sign will be ignored. This allows certain emails t o be filtered. Note that this rule does not apply to domain names.

For example, "m.y+name@email.com" will be forwarded to "my@email.com".

It is possible to use both of these rules at the same time.
Given an array of strings emails where we send one email to each email[i], return the number of different addresses t hat actually receive mails.

Example 1:

Input: emails = ["test.email+alex@leetcode.com","test.e.mail+bob.cathy@leetcode.com","testemail+david@lee.tcod e.com"]
Output: 2
Explanation: "testemail@leetcode.com" and "testemail@lee.tcode.com" actually receive mails.

Example 2:

Input: emails = ["a@leetcode.com","b@leetcode.com","c@leetcode.com"]
Output: 3

Constraints:

1 <= emails.length <= 100
1 <= emails[i].length <= 100
email[i] consist of lowercase English letters, '+', '.' and '@'.
Each emails[i] contains exactly one '@' character.
All local and domain names are non-empty.
Local names do not start with a '+' character.

---

930. Binary Subarrays With Sum

Given a binary array nums and an integer goal, return the number of non-empty subarrays with a sum goal.
A subarray is a contiguous part of the array.

Example 1:

Input: nums = [1,0,1,0,1], goal = 2
Output: 4
Explanation: The 4 subarrays are bolded and underlined below:
[1,0,1,0,1]
[1,0,1,0,1]
[1,0,1,0,1]
[1,0,1,0,1]

Example 2:

Input: nums = [0,0,0,0,0], goal = 0
Output: 15

Constraints:

1 <= nums.length <= 3 * 104
nums[i] is either 0 or 1.
0 <= goal <= nums.length

---

931. Minimum Falling Path Sum

Given an n x n array of integers matrix, return the minimum sum of any falling path through matrix.
A falling path starts at any element in the first row and chooses the element in the next row that is either directly bel ow or diagonally left/right. Specifically, the next element from position (row, col) will be (row + 1, col - 1), (row + 1 , col), or (row + 1, col + 1).

Example 1:

Input: matrix = [[2,1,3],[6,5,4],[7,8,9]]
Output: 13
Explanation: There are two falling paths with a minimum sum as shown.

Example 2:

Input: matrix = [[-19,57],[-40,-5]]
Output: -59

Explanation: The falling path with a minimum sum is shown.

Constraints:

n == matrix.length == matrix[i].length
1 <= n <= 100
-100 <= matrix[i][j] <= 100

---

932. Beautiful Array

An array nums of length n is beautiful if:

nums is a permutation of the integers in the range [1, n].
For every 0 <= i < j < n, there is no index k with i < k < j where 2 * nums[k] == nums[i] + nums[j].

Given the integer n, return any beautiful array nums of length n. There will be at least one valid answer for the given n.

Example 1:
Input: n = 4
Output: [2,1,4,3]
Example 2:
Input: n = 5
Output: [3,1,2,5,4]

Constraints:

1 <= n <= 1000

---

933. Number of Recent Calls

You have a RecentCounter class which counts the number of recent requests within a certain time frame.
Implement the RecentCounter class:

RecentCounter() Initializes the counter with zero recent requests.
int ping(int t) Adds a new request at time t, where t represents some time in milliseconds, and returns the number of r

equests that has happened in the past 3000 milliseconds (including the new request). Specifically, return the number of requests that have happened in the inclusive range [t - 3000, t].

It is guaranteed that every call to ping uses a strictly larger value of t than the previous call.

Example 1:

Input
["RecentCounter", "ping", "ping", "ping", "ping"]
[[], [1], [100], [3001], [3002]]
Output
[null, 1, 2, 3, 3]

Explanation
RecentCounter recentCounter = new RecentCounter();
recentCounter.ping(1);   // requests = [1], range is [-2999,1], return 1
recentCounter.ping(100);   // requests = [1, 100], range is [-2900,100], return 2
recentCounter.ping(3001); // requests = [1, 100, 3001], range is [1,3001], return 3
recentCounter.ping(3002); // requests = [1, 100, 3001, 3002], range is [2,3002], return 3

Constraints:

1 <= t <= 109
Each test case will call ping with strictly increasing values of t.
At most 104 calls will be made to ping.

---

934. Shortest Bridge

You are given an n x n binary matrix grid where 1 represents land and 0 represents water.
An island is a 4-directionally connected group of 1's not connected to any other 1's. There are exactly two islands in grid.
You may change 0's to 1's to connect the two islands to form one island.
Return the smallest number of 0's you must flip to connect the two islands.

Example 1:

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

Example 2:

Input: grid = [[0,1,0],[0,0,0],[0,0,1]]
Output: 2

Example 3:

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

Constraints:

n == grid.length == grid[i].length
2 <= n <= 100
grid[i][j] is either 0 or 1.
There are exactly two islands in grid.

---

935. Knight Dialer

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

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, 8 3, 92, 94]

Example 3:

Input: n = 3
Output: 46

Example 4:

Input: n = 4
Output: 104

Example 5:

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

Constraints:

1 <= n <= 5000

---

936. Stamping The Sequence

You are given two strings stamp and target. Initially, there is a string s of length target.length with all s[i] == '?'.
In one turn, you can place stamp over s and replace every letter in the s with the corresponding letter from stamp.

For example, if stamp = "abc" and target = "abcba", then s is "?????" initially. In one turn you can:

place stamp at index 0 of s to obtain "abc??",
place stamp at index 1 of s to obtain "?abc?", or
place stamp at index 2 of s to obtain "??abc".

Note that stamp must be fully contained in the boundaries of s in order to stamp (i.e., you cannot place stamp at inde x 3 of s).

We want to convert s to target using at most 10 * target.length turns.
Return an array of the index of the left-most letter being stamped at each turn. If we cannot obtain target from s withi n 10 * target.length turns, return an empty array.

Example 1:

Input: stamp = "abc", target = "ababc"
Output: [0,2]
Explanation: Initially s = "?????".
- Place stamp at index 0 to get "abc??".
- Place stamp at index 2 to get "ababc".
[1,0,2] would also be accepted as an answer, as well as some other answers.

Example 2:

Input: stamp = "abca", target = "aabcaca"

Output: [3,0,1]
Explanation: Initially s = "???????".
- Place stamp at index 3 to get "???abca".
- Place stamp at index 0 to get "abcabca".
- Place stamp at index 1 to get "aabcaca".

Constraints:

1 <= stamp.length <= target.length <= 1000
stamp and target consist of lowercase English letters.

---

937. Reorder Data in Log Files

You are given an array of logs. Each log is a space-delimited string of words, where the first word is the identifier.
There are two types of logs:

Letter-logs: All words (except the identifier) consist of lowercase English letters.
Digit-logs: All words (except the identifier) consist of digits.

Reorder these logs so that:

The letter-logs come before all digit-logs.
The letter-logs are sorted lexicographically by their contents. If their contents are the same, then sort them lexicogra phically by their identifiers.
The digit-logs maintain their relative ordering.

Return the final order of the logs.

Example 1:

Input: logs = ["dig1 8 1 5 1","let1 art can","dig2 3 6","let2 own kit dig","let3 art zero"]
Output: ["let1 art can","let3 art zero","let2 own kit dig","dig1 8 1 5 1","dig2 3 6"]
Explanation:
The letter-log contents are all different, so their ordering is "art can", "art zero", "own kit dig".
The digit-logs have a relative order of "dig1 8 1 5 1", "dig2 3 6".

Example 2:

Input: logs = ["a1 9 2 3 1","g1 act car","zo4 4 7","ab1 off key dog","a8 act zoo"]
Output: ["g1 act car","a8 act zoo","ab1 off key dog","a1 9 2 3 1","zo4 4 7"]

Constraints:

1 <= logs.length <= 100

3 <= logs[i].length <= 100
All the tokens of logs[i] are separated by a single space.
logs[i] is guaranteed to have an identifier and at least one word after the identifier.

---

938. Range Sum of BST

Given the root node of a binary search tree and two integers low and high, return the sum of values of all nodes with a value in the inclusive range [low, high].

Example 1:

Input: root = [10,5,15,3,7,null,18], low = 7, high = 15
Output: 32
Explanation: Nodes 7, 10, and 15 are in the range [7, 15]. 7 + 10 + 15 = 32.

Example 2:

Input: root = [10,5,15,3,7,13,18,1,null,6], low = 6, high = 10
Output: 23
Explanation: Nodes 6, 7, and 10 are in the range [6, 10]. 6 + 7 + 10 = 23.

Constraints:

The number of nodes in the tree is in the range [1, 2 * 104].
1 <= Node.val <= 105
1 <= low <= high <= 105
All Node.val are unique.

---

939. Minimum Area Rectangle

You are given an array of points in the X-Y plane points where points[i] = [xi, yi].
Return the minimum area of a rectangle formed from these points, with sides parallel to the X and Y axes. If there is not any such rectangle, return 0.

Example 1:

Input: points = [[1,1],[1,3],[3,1],[3,3],[2,2]]
Output: 4

Example 2:

Input: points = [[1,1],[1,3],[3,1],[3,3],[4,1],[4,3]]
Output: 2

Constraints:

1 <= points.length <= 500
points[i].length == 2
0 <= xi, yi <= 4 * 104
All the given points are unique.

---

940. Distinct Subsequences II

Given a string s, return the number of distinct non-empty subsequences of s. Since the answer may be very large, ret urn it modulo 109 + 7.
A subsequence of a string is a new string that is formed from the original string by deleting some (can be none) of th e characters without disturbing the relative positions of the remaining characters. (i.e., "ace" is a subsequence of "abc de" while "aec" is not.

Example 1:

Input: s = "abc"
Output: 7
Explanation: The 7 distinct subsequences are "a", "b", "c", "ab", "ac", "bc", and "abc".

Example 2:

Input: s = "aba"
Output: 6
Explanation: The 6 distinct subsequences are "a", "b", "ab", "aa", "ba", and "aba".

Example 3:

Input: s = "aaa"
Output: 3
Explanation: The 3 distinct subsequences are "a", "aa" and "aaa".

Constraints:

1 <= s.length <= 2000
s consists of lowercase English letters.

---

941. Valid Mountain Array

Given an array of integers arr, return true if and only if it is a valid mountain array.
Recall that arr is a mountain array if and only if:

arr.length >= 3
There exists some i with 0 < i < arr.length - 1 such that:

arr[0] < arr[1] < ... < arr[i - 1] < arr[i]
arr[i] > arr[i + 1] > ... > arr[arr.length - 1]

Example 1:
Input: arr = [2,1]
Output: false
Example 2:
Input: arr = [3,5,5]
Output: false
Example 3:
Input: arr = [0,3,2,1]
Output: true

Constraints:

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

---

942. DI String Match

A permutation perm of n + 1 integers of all the integers in the range [0, n] can be represented as a string s of length n where:

s[i] == 'I' if perm[i] < perm[i + 1], and
s[i] == 'D' if perm[i] > perm[i + 1].

Given a string s, reconstruct the permutation perm and return it. If there are multiple valid permutations perm, return any of them.

Example 1:
Input: s = "IDID"
Output: [0,4,1,3,2]
Example 2:
Input: s = "III"
Output: [0,1,2,3]
Example 3:
Input: s = "DDI"
Output: [3,2,0,1]

Constraints:

1 <= s.length <= 105
s[i] is either 'I' or 'D'.

---

943. Find the Shortest Superstring

Given an array of strings words, return the smallest string that contains each string in words as a substring. If there ar e multiple valid strings of the smallest length, return any of them.
You may assume that no string in words is a substring of another string in words.

Example 1:

Input: words = ["alex","loves","leetcode"]
Output: "alexlovesleetcode"
Explanation: All permutations of "alex","loves","leetcode" would also be accepted.

Example 2:

Input: words = ["catg","ctaagt","gcta","ttca","atgcatc"]
Output: "gctaagttcatgcatc"

Constraints:

1 <= words.length <= 12

1 <= words[i].length <= 20
words[i] consists of lowercase English letters.
All the strings of words are unique.

---

944. Delete Columns to Make Sorted

You are given an array of n strings strs, all of the same length.
The strings can be arranged such that there is one on each line, making a grid. For example, strs = ["abc", "bce", "cae"] can be arranged as:

abc
bce
cae

You want to delete the columns that are not sorted lexicographically. In the above example (0-indexed), columns 0 ('a', 'b', 'c') and 2 ('c', 'e', 'e') are sorted while column 1 ('b', 'c', 'a') is not, so you would delete column 1.
Return the number of columns that you will delete.

Example 1:

Input: strs = ["cba","daf","ghi"]
Output: 1
Explanation: The grid looks as follows:
  cba
  daf
  ghi
Columns 0 and 2 are sorted, but column 1 is not, so you only need to delete 1 column.

Example 2:

Input: strs = ["a","b"]
Output: 0
Explanation: The grid looks as follows:
  a
  b
Column 0 is the only column and is sorted, so you will not delete any columns.

Example 3:

Input: strs = ["zyx","wvu","tsr"]
Output: 3
Explanation: The grid looks as follows:
  zyx
  wvu
  tsr
All 3 columns are not sorted, so you will delete all 3.

Constraints:

n == strs.length
1 <= n <= 100
1 <= strs[i].length <= 1000
strs[i] consists of lowercase English letters.

---

945. Minimum Increment to Make Array Unique

You are given an integer array nums. In one move, you can pick an index i where 0 <= i < nums.length and increme nt nums[i] by 1.
Return the minimum number of moves to make every value in nums unique.

Example 1:

Input: nums = [1,2,2]
Output: 1
Explanation: After 1 move, the array could be [1, 2, 3].

Example 2:

Input: nums = [3,2,1,2,1,7]
Output: 6
Explanation: After 6 moves, the array could be [3, 4, 1, 2, 5, 7].
It can be shown with 5 or less moves that it is impossible for the array to have all unique values.

Constraints:

1 <= nums.length <= 105
0 <= nums[i] <= 105

---

946. Validate Stack Sequences

Given two integer arrays pushed and popped each with distinct values, return true if this could have been the result o f a sequence of push and pop operations on an initially empty stack, or false otherwise.

Example 1:

Input: pushed = [1,2,3,4,5], popped = [4,5,3,2,1]
Output: true
Explanation: We might do the following sequence:
push(1), push(2), push(3), push(4),
pop() -> 4,
push(5),
pop() -> 5, pop() -> 3, pop() -> 2, pop() -> 1

Example 2:

Input: pushed = [1,2,3,4,5], popped = [4,3,5,1,2]
Output: false
Explanation: 1 cannot be popped before 2.

Constraints:

1 <= pushed.length <= 1000
0 <= pushed[i] <= 1000
All the elements of pushed are unique.
popped.length == pushed.length
popped is a permutation of pushed.

---

947. Most Stones Removed with Same Row or Column

On a 2D plane, we place n stones at some integer coordinate points. Each coordinate point may have at most one sto ne.
A stone can be removed if it shares either the same row or the same column as another stone that has not been remov ed.
Given an array stones of length n where stones[i] = [xi, yi] represents the location of the ith stone, return the largest possible number of stones that can be removed.

Example 1:

Input: stones = [[0,0],[0,1],[1,0],[1,2],[2,1],[2,2]]
Output: 5
Explanation: One way to remove 5 stones is as follows:
1. Remove stone [2,2] because it shares the same row as [2,1].
2. Remove stone [2,1] because it shares the same column as [0,1].
3. Remove stone [1,2] because it shares the same row as [1,0].
4. Remove stone [1,0] because it shares the same column as [0,0].
5. Remove stone [0,1] because it shares the same row as [0,0].
Stone [0,0] cannot be removed since it does not share a row/column with another stone still on the plane.

Example 2:

Input: stones = [[0,0],[0,2],[1,1],[2,0],[2,2]]
Output: 3
Explanation: One way to make 3 moves is as follows:
1. Remove stone [2,2] because it shares the same row as [2,0].
2. Remove stone [2,0] because it shares the same column as [0,0].
3. Remove stone [0,2] because it shares the same row as [0,0].
Stones [0,0] and [1,1] cannot be removed since they do not share a row/column with another stone still on the plane.

Example 3:

Input: stones = [[0,0]]
Output: 0
Explanation: [0,0] is the only stone on the plane, so you cannot remove it.

Constraints:

1 <= stones.length <= 1000
0 <= xi, yi <= 104
No two stones are at the same coordinate point.

---

948. Bag of Tokens

You have an initial power of power, an initial score of 0, and a bag of tokens where tokens[i] is the value of the ith t oken (0-indexed).
Your goal is to maximize your total score by potentially playing each token in one of two ways:

If your current power is at least tokens[i], you may play the ith token face up, losing tokens[i] power and gaining 1 s core.
If your current score is at least 1, you may play the ith token face down, gaining tokens[i] power and losing 1 score.

Each token may be played at most once and in any order. You do not have to play all the tokens.
Return the largest possible score you can achieve after playing any number of tokens.

Example 1:

Input: tokens = [100], power = 50
Output: 0
Explanation: Playing the only token in the bag is impossible because you either have too little power or too little scor e.

Example 2:

Input: tokens = [100,200], power = 150
Output: 1
Explanation: Play the 0th token (100) face up, your power becomes 50 and score becomes 1.
There is no need to play the 1st token since you cannot play it face up to add to your score.

Example 3:

Input: tokens = [100,200,300,400], power = 200
Output: 2
Explanation: Play the tokens in this order to get a score of 2:
1. Play the 0th token (100) face up, your power becomes 100 and score becomes 1.
2. Play the 3rd token (400) face down, your power becomes 500 and score becomes 0.
3. Play the 1st token (200) face up, your power becomes 300 and score becomes 1.
4. Play the 2nd token (300) face up, your power becomes 0 and score becomes 2.

Constraints:

0 <= tokens.length <= 1000
0 <= tokens[i], power < 104

---

949. Largest Time for Given Digits

Given an array arr of 4 digits, find the latest 24-hour time that can be made using each digit exactly once.
24-hour times are formatted as "HH:MM", where HH is between 00 and 23, and MM is between 00 and 59. The earl iest 24-hour time is 00:00, and the latest is 23:59.
Return the latest 24-hour time in "HH:MM" format. If no valid time can be made, return an empty string.

Example 1:

Input: arr = [1,2,3,4]
Output: "23:41"
Explanation: The valid 24-hour times are "12:34", "12:43", "13:24", "13:42", "14:23", "14:32", "21:34", "21:43", "2 3:14", and "23:41". Of these times, "23:41" is the latest.

Example 2:

Input: arr = [5,5,5,5]
Output: ""
Explanation: There are no valid 24-hour times as "55:55" is not valid.

Example 3:

Input: arr = [0,0,0,0]
Output: "00:00"

Example 4:

Input: arr = [0,0,1,0]
Output: "10:00"

Constraints:

arr.length == 4
0 <= arr[i] <= 9

---

950. Reveal Cards In Increasing Order

You are given an integer array deck. There is a deck of cards where every card has a unique integer. The integer on t he ith card is deck[i].
You can order the deck in any order you want. Initially, all the cards start face down (unrevealed) in one deck.
You will do the following steps repeatedly until all cards are revealed:

Take the top card of the deck, reveal it, and take it out of the deck.
If there are still cards in the deck then put the next top card of the deck at the bottom of the deck.
If there are still unrevealed cards, go back to step 1. Otherwise, stop.

Return an ordering of the deck that would reveal the cards in increasing order.
Note that the first entry in the answer is considered to be the top of the deck.

Example 1:

Input: deck = [17,13,11,2,3,5,7]
Output: [2,13,3,11,5,17,7]
Explanation:
We get the deck in the order [17,13,11,2,3,5,7] (this order does not matter), and reorder it.
After reordering, the deck starts as [2,13,3,11,5,17,7], where 2 is the top of the deck.
We reveal 2, and move 13 to the bottom. The deck is now [3,11,5,17,7,13].
We reveal 3, and move 11 to the bottom. The deck is now [5,17,7,13,11].
We reveal 5, and move 17 to the bottom. The deck is now [7,13,11,17].
We reveal 7, and move 13 to the bottom. The deck is now [11,17,13].
We reveal 11, and move 17 to the bottom. The deck is now [13,17].
We reveal 13, and move 17 to the bottom. The deck is now [17].
We reveal 17.
Since all the cards revealed are in increasing order, the answer is correct.

Example 2:

Input: deck = [1,1000]
Output: [1,1000]

Constraints:

1 <= deck.length <= 1000
1 <= deck[i] <= 106
All the values of deck are unique.

---

951. Flip Equivalent Binary Trees

For a binary tree T, we can define a flip operation as follows: choose any node, and swap the left and right child subt rees.
A binary tree X is flip equivalent to a binary tree Y if and only if we can make X equal to Y after some number of fli p operations.
Given the roots of two binary trees root1 and root2, return true if the two trees are flip equivalent or false otherwise.

Example 1:

Input: root1 = [1,2,3,4,5,6,null,null,null,7,8], root2 = [1,3,2,null,6,4,5,null,null,null,null,8,7]
Output: true
Explanation: We flipped at nodes with values 1, 3, and 5.

Example 2:

Input: root1 = [], root2 = []
Output: true

Example 3:

Input: root1 = [], root2 = [1]
Output: false

Example 4:

Input: root1 = [0,null,1], root2 = []
Output: false

Example 5:

Input: root1 = [0,null,1], root2 = [0,1]
Output: true

Constraints:

The number of nodes in each tree is in the range [0, 100].

Each tree will have unique node values in the range [0, 99].

---

952. Largest Component Size by Common Factor

You are given an integer array of unique positive integers nums. Consider the following graph:

There are nums.length nodes, labeled nums[0] to nums[nums.length - 1],
There is an undirected edge between nums[i] and nums[j] if nums[i] and nums[j] share a common factor greater than 1.

Return the size of the largest connected component in the graph.

Example 1:

Input: nums = [4,6,15,35]
Output: 4

Example 2:

Input: nums = [20,50,9,63]
Output: 2

Example 3:

Input: nums = [2,3,6,7,4,12,21,39]
Output: 8

Constraints:

1 <= nums.length <= 2 * 104
1 <= nums[i] <= 105
All the values of nums are unique.

---

953. Verifying an Alien Dictionary

In an alien language, surprisingly, they also use English lowercase letters, but possibly in a different order. The order of the alphabet is some permutation of lowercase letters.
Given a sequence of words written in the alien language, and the order of the alphabet, return true if and only if the g iven words are sorted lexicographically in this alien language.

Example 1:

Input: words = ["hello","leetcode"], order = "hlabcdefgijkmnopqrstuvwxyz"
Output: true
Explanation: As 'h' comes before 'l' in this language, then the sequence is sorted.

Example 2:

Input: words = ["word","world","row"], order = "worldabcefghijkmnpqstuvxyz"
Output: false
Explanation: As 'd' comes after 'l' in this language, then words[0] > words[1], hence the sequence is unsorted.

Example 3:

Input: words = ["apple","app"], order = "abcdefghijklmnopqrstuvwxyz"
Output: false
Explanation: The first three characters "app" match, and the second string is shorter (in size.) According to lexicogra phical rules "apple" > "app", because 'l' > '□ ', where '□ ' is defined as the blank character which is less than any othe r character (More info).

Constraints:

1 <= words.length <= 100
1 <= words[i].length <= 20
order.length == 26
All characters in words[i] and order are English lowercase letters.

---

954. Array of Doubled Pairs

Given an integer array of even length arr, return true if it is possible to reorder arr such that arr[2 * i + 1] = 2 * arr[2 * i] for every 0 <= i < len(arr) / 2, or false otherwise.

Example 1:

Input: arr = [3,1,3,6]
Output: false

Example 2:

Input: arr = [2,1,2,6]
Output: false

Example 3:

Input: arr = [4,-2,2,-4]
Output: true
Explanation: We can take two groups, [-2,-4] and [2,4] to form [-2,-4,2,4] or [2,4,-2,-4].

Example 4:

Input: arr = [1,2,4,16,8,4]
Output: false

Constraints:

2 <= arr.length <= 3 * 104
arr.length is even.
-105 <= arr[i] <= 105

---

955. Delete Columns to Make Sorted II

You are given an array of n strings strs, all of the same length.
We may choose any deletion indices, and we delete all the characters in those indices for each string.
For example, if we have strs = ["abcdef","uvwxyz"] and deletion indices {0, 2, 3}, then the final array after deletions is ["bef", "vyz"].
Suppose we chose a set of deletion indices answer such that after deletions, the final array has its elements in lexicographic order (i.e., strs[0] <= strs[1] <= strs[2] <= ... <= strs[n - 1]). Return the minimum possible value of answer.length.

Example 1:

Input: strs = ["ca","bb","ac"]
Output: 1
Explanation:
After deleting the first column, strs = ["a", "b", "c"].
Now strs is in lexicographic order (ie. strs[0] <= strs[1] <= strs[2]).
We require at least 1 deletion since initially strs was not in lexicographic order, so the answer is 1.

Example 2:

Input: strs = ["xc","yb","za"]
Output: 0
Explanation:
strs is already in lexicographic order, so we do not need to delete anything.

Note that the rows of strs are not necessarily in lexicographic order:
i.e., it is NOT necessarily true that (strs[0][0] <= strs[0][1] <= ...)

Example 3:

Input: strs = ["zyx","wvu","tsr"]
Output: 3
Explanation: We have to delete every column.

Constraints:

n == strs.length
1 <= n <= 100
1 <= strs[i].length <= 100
strs[i] consists of lowercase English letters.

---

956. Tallest Billboard

You are installing a billboard and want it to have the largest height. The billboard will have two steel supports, one o n each side. Each steel support must be an equal height.
You are given a collection of rods that can be welded together. For example, if you have rods of lengths 1, 2, and 3, you can weld them together to make a support of length 6.
Return the largest possible height of your billboard installation. If you cannot support the billboard, return 0.

Example 1:

Input: rods = [1,2,3,6]
Output: 6
Explanation: We have two disjoint subsets {1,2,3} and {6}, which have the same sum = 6.

Example 2:

Input: rods = [1,2,3,4,5,6]
Output: 10
Explanation: We have two disjoint subsets {2,3,5} and {4,6}, which have the same sum = 10.

Example 3:

Input: rods = [1,2]
Output: 0
Explanation: The billboard cannot be supported, so we return 0.

Constraints:

1 <= rods.length <= 20
1 <= rods[i] <= 1000
sum(rods[i]) <= 5000

---

957. Prison Cells After N Days

There are 8 prison cells in a row and each cell is either occupied or vacant.
Each day, whether the cell is occupied or vacant changes according to the following rules:

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

Note that because the prison is a row, the first and the last cells in the row can't have two adjacent neighbors.
You are given an integer array cells where cells[i] == 1 if the ith cell is occupied and cells[i] == 0 if the ith cell is va cant, and you are given an integer n.
Return the state of the prison after n days (i.e., n such changes described above).

Example 1:

Input: cells = [0,1,0,1,1,0,0,1], n = 7
Output: [0,0,1,1,0,0,0,0]
Explanation: The following table summarizes the state of the prison on each day:
Day 0: [0, 1, 0, 1, 1, 0, 0, 1]
Day 1: [0, 1, 1, 0, 0, 0, 0, 0]
Day 2: [0, 0, 0, 0, 1, 1, 1, 0]
Day 3: [0, 1, 1, 0, 0, 1, 0, 0]
Day 4: [0, 0, 0, 0, 0, 1, 0, 0]
Day 5: [0, 1, 1, 1, 0, 1, 0, 0]
Day 6: [0, 0, 1, 0, 1, 1, 0, 0]
Day 7: [0, 0, 1, 1, 0, 0, 0, 0]

Example 2:

Input: cells = [1,0,0,1,0,0,1,0], n = 1000000000
Output: [0,0,1,1,1,1,1,0]

Constraints:

cells.length == 8
cells[i] is either 0 or 1.
1 <= n <= 109

---

958. Check Completeness of a Binary Tree

Given the root of a binary tree, determine if it is a complete binary tree.
In a complete binary tree, every level, except possibly the last, is completely filled, and all nodes in the last level are as far left as possible. It can have between 1 and 2h nodes inclusive at the last level h.

Example 1:

Input: root = [1,2,3,4,5,6]
Output: true
Explanation: Every level before the last is full (ie. levels with node-values {1} and {2, 3}), and all nodes in the last l evel ({4, 5, 6}) are as far left as possible.

Example 2:

Input: root = [1,2,3,4,5,null,7]
Output: false
Explanation: The node with value 7 isn't as far left as possible.

Constraints:

The number of nodes in the tree is in the range [1, 100].
1 <= Node.val <= 1000

---

959. Regions Cut By Slashes

An n x n grid is composed of 1 x 1 squares where each 1 x 1 square consists of a '/', '\', or blank space ' '. These chara cters divide the square into contiguous regions.
Given the grid grid represented as a string array, return the number of regions.
Note that backslash characters are escaped, so a '\' is represented as '\\'.

Example 1:

Input: grid = [" /","/ "]
Output: 2

Example 2:

Input: grid = [" /"," "]
Output: 1

Example 3:

Input: grid = ["\\/","/\\"]
Output: 4
Explanation: (Recall that because \ characters are escaped, "\\/" refers to \/, and "/\\" refers to /\.)

Example 4:

Input: grid = ["/\\","\\/"]
Output: 5
Explanation: (Recall that because \ characters are escaped, "\\/" refers to \/, and "/\\" refers to /\.)

Example 5:

Input: grid = ["//","/ "]
Output: 3

Constraints:

n == grid.length
n == grid[i].length
1 <= n <= 30
grid[i][j] is either '/', '\', or ' '.

---

960. Delete Columns to Make Sorted III

You are given an array of n strings strs, all of the same length.
We may choose any deletion indices, and we delete all the characters in those indices for each string.
For example, if we have strs = ["abcdef","uvwxyz"] and deletion indices {0, 2, 3}, then the final array after deletions is ["bef", "vyz"].
Suppose we chose a set of deletion indices answer such that after deletions, the final array has every string (row) in l exicographic order. (i.e., (strs[0][0] <= strs[0][1] <= ... <= strs[0][strs[0].length - 1]), and (strs[1][0] <= strs[1][1] <= ... <= strs[1][strs[1].length - 1]), and so on). Return the minimum possible value of answer.length.

Example 1:

Input: strs = ["babca","bbazb"]

Output: 3
Explanation: After deleting columns 0, 1, and 4, the final array is strs = ["bc", "az"].
Both these rows are individually in lexicographic order (ie. strs[0][0] <= strs[0][1] and strs[1][0] <= strs[1][1]).
Note that strs[0] > strs[1] - the array strs is not necessarily in lexicographic order.
Example 2:

Input: strs = ["edcba"]
Output: 4
Explanation: If we delete less than 4 columns, the only row will not be lexicographically sorted.

Example 3:

Input: strs = ["ghi","def","abc"]
Output: 0
Explanation: All rows are already lexicographically sorted.

Constraints:

n == strs.length
1 <= n <= 100
1 <= strs[i].length <= 100
strs[i] consists of lowercase English letters.

---

961. N-Repeated Element in Size 2N Array

You are given an integer array nums with the following properties:

nums.length == 2 * n.
nums contains n + 1 unique elements.
Exactly one element of nums is repeated n times.

Return the element that is repeated n times.

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

Constraints:

2 <= n <= 5000
nums.length == 2 * n
0 <= nums[i] <= 104
nums contains n + 1 unique elements and one of them is repeated exactly n times.

---

962. Maximum Width Ramp

A ramp in an integer array nums is a pair (i, j) for which i < j and nums[i] <= nums[j]. The width of such a ramp is j - i.
Given an integer array nums, return the maximum width of a ramp in nums. If there is no ramp in nums, return 0.

Example 1:

Input: nums = [6,0,8,2,1,5]
Output: 4
Explanation: The maximum width ramp is achieved at (i, j) = (1, 5): nums[1] = 0 and nums[5] = 5.

Example 2:

Input: nums = [9,8,1,0,1,9,4,0,4,1]
Output: 7
Explanation: The maximum width ramp is achieved at (i, j) = (2, 9): nums[2] = 1 and nums[9] = 1.

Constraints:

2 <= nums.length <= 5 * 104
0 <= nums[i] <= 5 * 104

---

963. Minimum Area Rectangle II

You are given an array of points in the X-Y plane points where points[i] = [xi, yi].
Return the minimum area of any rectangle formed from these points, with sides not necessarily parallel to the X and Y axes. If there is not any such rectangle, return 0.

Answers within 10-5 of the actual answer will be accepted.

Example 1:

Input: points = [[1,2],[2,1],[1,0],[0,1]]
Output: 2.00000
Explanation: The minimum area rectangle occurs at [1,2],[2,1],[1,0],[0,1], with an area of 2.

Example 2:

Input: points = [[0,1],[2,1],[1,1],[1,0],[2,0]]
Output: 1.00000
Explanation: The minimum area rectangle occurs at [1,0],[1,1],[2,1],[2,0], with an area of 1.

Example 3:

Input: points = [[0,3],[1,2],[3,1],[1,3],[2,1]]
Output: 0
Explanation: There is no possible rectangle to form from these points.

Example 4:

Input: points = [[3,1],[1,1],[0,1],[2,1],[3,3],[3,2],[0,2],[2,3]]
Output: 2.00000
Explanation: The minimum area rectangle occurs at [2,1],[2,3],[3,3],[3,1], with an area of 2.

Constraints:

1 <= points.length <= 50
points[i].length == 2
0 <= xi, yi <= 4 * 104
All the given points are unique.

---

964. Least Operators to Express Number

Given a single positive integer x, we will write an expression of the form x (op1) x (op2) x (op3) x ... where each op erator op1, op2, etc. is either addition, subtraction, multiplication, or division (+, -, *, or /). For example, with x = 3, we might write 3 * 3 / 3 + 3 - 3 which is a value of 3.
When writing such an expression, we adhere to the following conventions:

The division operator (/) returns rational numbers.

There are no parentheses placed anywhere.
We use the usual order of operations: multiplication and division happen before addition and subtraction.
It is not allowed to use the unary negation operator (-). For example, "x - x" is a valid expression as it only uses subtr action, but "-x + x" is not because it uses negation.

We would like to write an expression with the least number of operators such that the expression equals the given tar get. Return the least number of operators used.

Example 1:

Input: x = 3, target = 19
Output: 5
Explanation: 3 * 3 + 3 * 3 + 3 / 3.
The expression contains 5 operations.

Example 2:

Input: x = 5, target = 501
Output: 8
Explanation: 5 * 5 * 5 * 5 - 5 * 5 * 5 + 5 / 5.
The expression contains 8 operations.

Example 3:

Input: x = 100, target = 100000000
Output: 3
Explanation: 100 * 100 * 100 * 100.
The expression contains 3 operations.

Constraints:

2 <= x <= 100
1 <= target <= 2 * 108

---

965. Univalued Binary Tree

A binary tree is uni-valued if every node in the tree has the same value.
Given the root of a binary tree, return true if the given tree is uni-valued, or false otherwise.

Example 1:

Input: root = [1,1,1,1,1,null,1]
Output: true

Example 2:

Input: root = [2,2,2,5,2]
Output: false

Constraints:

The number of nodes in the tree is in the range [1, 100].
0 <= Node.val < 100

---

966. Vowel Spellchecker

Given a wordlist, we want to implement a spellchecker that converts a query word into a correct word.
For a given query word, the spell checker handles two categories of spelling mistakes:

Capitalization: If the query matches a word in the wordlist (case-insensitive), then the query word is returned with th e same case as the case in the wordlist.

Example: wordlist = ["yellow"], query = "YellOw": correct = "yellow"
Example: wordlist = ["Yellow"], query = "yellow": correct = "Yellow"
Example: wordlist = ["yellow"], query = "yellow": correct = "yellow"

Vowel Errors: If after replacing the vowels ('a', 'e', 'i', 'o', 'u') of the query word with any vowel individually, it match es a word in the wordlist (case-insensitive), then the query word is returned with the same case as the match in the w ordlist.

Example: wordlist = ["YellOw"], query = "yollow": correct = "YellOw"
Example: wordlist = ["YellOw"], query = "yeellow": correct = "" (no match)
Example: wordlist = ["YellOw"], query = "yllw": correct = "" (no match)

In addition, the spell checker operates under the following precedence rules:

When the query exactly matches a word in the wordlist (case-sensitive), you should return the same word back.
When the query matches a word up to capitlization, you should return the first such match in the wordlist.
When the query matches a word up to vowel errors, you should return the first such match in the wordlist.
If the query has no matches in the wordlist, you should return the empty string.

Given some queries, return a list of words answer, where answer[i] is the correct word for query = queries[i].

Example 1:

Input: wordlist = ["KiTe","kite","hare","Hare"], queries = ["kite","Kite","KiTe","Hare","HARE","Hear","hear","keti","keet","keto"]
Output: ["kite","KiTe","KiTe","Hare","hare","","","KiTe","","KiTe"]
Example 2:
Input: wordlist = ["yellow"], queries = ["YellOw"]
Output: ["yellow"]

Constraints:

1 <= wordlist.length, queries.length <= 5000
1 <= wordlist[i].length, queries[i].length <= 7
wordlist[i] and queries[i] consist only of only English letters.

---

967. Numbers With Same Consecutive Differences

Return all non-negative integers of length n such that the absolute difference between every two consecutive digits is k.
Note that every number in the answer must not have leading zeros. For example, 01 has one leading zero and is invalid.
You may return the answer in any order.

Example 1:

Input: n = 3, k = 7
Output: [181,292,707,818,929]
Explanation: Note that 070 is not a valid number, because it has leading zeroes.

Example 2:

Input: n = 2, k = 1
Output: [10,12,21,23,32,34,43,45,54,56,65,67,76,78,87,89,98]

Example 3:

Input: n = 2, k = 0
Output: [11,22,33,44,55,66,77,88,99]

Example 4:

Input: n = 2, k = 2
Output: [13,20,24,31,35,42,46,53,57,64,68,75,79,86,97]

Constraints:

2 <= n <= 9
0 <= k <= 9

---

968. Binary Tree Cameras

You are given the root of a binary tree. We install cameras on the tree nodes where each camera at a node can monit or its parent, itself, and its immediate children.
Return the minimum number of cameras needed to monitor all nodes of the tree.

Example 1:

Input: root = [0,0,null,0,0]
Output: 1
Explanation: One camera is enough to monitor all nodes if placed as shown.

Example 2:

Input: root = [0,0,null,0,null,0,null,null,0]
Output: 2
Explanation: At least two cameras are needed to monitor all nodes of the tree. The above image shows one of the val id configurations of camera placement.

Constraints:

The number of nodes in the tree is in the range [1, 1000].
Node.val == 0

---

969. Pancake Sorting

Given an array of integers arr, sort the array by performing a series of pancake flips.
In one pancake flip we do the following steps:

Choose an integer k where 1 <= k <= arr.length.
Reverse the sub-array arr[0...k-1] (0-indexed).

For example, if arr = [3,2,1,4] and we performed a pancake flip choosing k = 3, we reverse the sub-array [3,2,1], so a
rr = [1,2,3,4] after the pancake flip at k = 3.
Return an array of the k-values corresponding to a sequence of pancake flips that sort arr. Any valid answer that sort
s the array within 10 * arr.length flips will be judged as correct.

Example 1:

Input: arr = [3,2,4,1]
Output: [4,2,4,3]
Explanation:
We perform 4 pancake flips, with k values 4, 2, 4, and 3.
Starting state: arr = [3, 2, 4, 1]
After 1st flip (k = 4): arr = [1, 4, 2, 3]
After 2nd flip (k = 2): arr = [4, 1, 2, 3]
After 3rd flip (k = 4): arr = [3, 2, 1, 4]
After 4th flip (k = 3): arr = [1, 2, 3, 4], which is sorted.

Example 2:

Input: arr = [1,2,3]
Output: []
Explanation: The input is already sorted, so there is no need to flip anything.
Note that other answers, such as [3, 3], would also be accepted.

Constraints:

1 <= arr.length <= 100
1 <= arr[i] <= arr.length
All integers in arr are unique (i.e. arr is a permutation of the integers from 1 to arr.length).

---

970. Powerful Integers

Given three integers x, y, and bound, return a list of all the powerful integers that have a value less than or equal to b
ound.
An integer is powerful if it can be represented as xi + yj for some integers i >= 0 and j >= 0.
You may return the answer in any order. In your answer, each value should occur at most once.

Example 1:

Input: x = 2, y = 3, bound = 10
Output: [2,3,4,5,7,9,10]
Explanation:
2 = 20 + 30
3 = 21 + 30
4 = 20 + 31

5 = 21 + 31
7 = 22 + 31
9 = 23 + 30
10 = 20 + 32

Example 2:

Input: x = 3, y = 5, bound = 15
Output: [2,4,6,8,10,14]

Constraints:

1 <= x, y <= 100
0 <= bound <= 106

---

971. Flip Binary Tree To Match Preorder Traversal

You are given the root of a binary tree with n nodes, where each node is uniquely assigned a value from 1 to n. You are also given a sequence of n values voyage, which is the desired pre-order traversal of the binary tree.
Any node in the binary tree can be flipped by swapping its left and right subtrees. For example, flipping node 1 will have the following effect:

Flip the smallest number of nodes so that the pre-order traversal of the tree matches voyage.
Return a list of the values of all flipped nodes. You may return the answer in any order. If it is impossible to flip the nodes in the tree to make the pre-order traversal match voyage, return the list [-1].

Example 1:

Input: root = [1,2], voyage = [2,1]
Output: [-1]
Explanation: It is impossible to flip the nodes such that the pre-order traversal matches voyage.

Example 2:

Input: root = [1,2,3], voyage = [1,3,2]
Output: [1]
Explanation: Flipping node 1 swaps nodes 2 and 3, so the pre-order traversal matches voyage.
Example 3:

Input: root = [1,2,3], voyage = [1,2,3]
Output: []
Explanation: The tree's pre-order traversal already matches voyage, so no nodes need to be flipped.

Constraints:

The number of nodes in the tree is n.
n == voyage.length
1 <= n <= 100
1 <= Node.val, voyage[i] <= n
All the values in the tree are unique.
All the values in voyage are unique.

---

972. Equal Rational Numbers

Given two strings s and t, each of which represents a non-negative rational number, return true if and only if they rep resent the same number. The strings may use parentheses to denote the repeating part of the rational number.
A rational number can be represented using up to three parts: <IntegerPart>, <NonRepeatingPart>, and a <Repeating Part>. The number will be represented in one of the following three ways:

<IntegerPart>

For example, 12, 0, and 123.

<IntegerPart><.><NonRepeatingPart>

For example, 0.5, 1., 2.12, and 123.0001.

<IntegerPart><.><NonRepeatingPart><(><RepeatingPart><)>

For example, 0.1(6), 1.(9), 123.00(1212).

The repeating portion of a decimal expansion is conventionally denoted within a pair of round brackets. For example :

1/6 = 0.16666666... = 0.1(6) = 0.1666(6) = 0.166(66).

Example 1:

Input: s = "0.(52)", t = "0.5(25)"
Output: true
Explanation: Because "0.(52)" represents 0.52525252..., and "0.5(25)" represents 0.52525252525..... , the strings rep resent the same number.

Example 2:

Input: s = "0.1666(6)", t = "0.166(66)"
Output: true

Example 3:

Input: s = "0.9(9)", t = "1."
Output: true
Explanation: "0.9(9)" represents 0.999999999... repeated forever, which equals 1. [See this link for an explanation.]
"1." represents the number 1, which is formed correctly: (IntegerPart) = "1" and (NonRepeatingPart) = "".

Constraints:

Each part consists only of digits.
The <IntegerPart> does not have leading zeros (except for the zero itself).
1 <= <IntegerPart>.length <= 4
0 <= <NonRepeatingPart>.length <= 4
1 <= <RepeatingPart>.length <= 4

---

973. K Closest Points to Origin

Given an array of points where points[i] = [xi, yi] represents a point on the X-Y plane and an integer k, return the k c losest points to the origin (0, 0).
The distance between two points on the X-Y plane is the Euclidean distance (i.e., √(x1 - x2)2 + (y1 - y2)2).
You may return the answer in any order. The answer is guaranteed to be unique (except for the order that it is in).

Example 1:

Input: points = [[1,3],[-2,2]], k = 1
Output: [[-2,2]]
Explanation:
The distance between (1, 3) and the origin is sqrt(10).
The distance between (-2, 2) and the origin is sqrt(8).
Since sqrt(8) < sqrt(10), (-2, 2) is closer to the origin.
We only want the closest k = 1 points from the origin, so the answer is just [[-2,2]].

Example 2:

Input: points = [[3,3],[5,-1],[-2,4]], k = 2
Output: [[3,3],[-2,4]]
Explanation: The answer [[-2,4],[3,3]] would also be accepted.

Constraints:

1 <= k <= points.length <= 104
-104 < xi, yi < 104

---

974. Subarray Sums Divisible by K

Given an integer array nums and an integer k, return the number of non-empty subarrays that have a sum divisible b y k.
A subarray is a contiguous part of an array.

Example 1:

Input: nums = [4,5,0,-2,-3,1], k = 5
Output: 7
Explanation: There are 7 subarrays with a sum divisible by k = 5:
[4, 5, 0, -2, -3, 1], [5], [5, 0], [5, 0, -2, -3], [0], [0, -2, -3], [-2, -3]

Example 2:

Input: nums = [5], k = 9
Output: 0

Constraints:

1 <= nums.length <= 3 * 104
-104 <= nums[i] <= 104
2 <= k <= 104

---

975. Odd Even Jump

You are given an integer array arr. From some starting index, you can make a series of jumps. The (1st, 3rd, 5th, ...) jumps in the series are called odd-numbered jumps, and the (2nd, 4th, 6th, ...) jumps in the series are called even-nu mbered jumps. Note that the jumps are numbered, not the indices.
You may jump forward from index i to index j (with i < j) in the following way:

During odd-numbered jumps (i.e., jumps 1, 3, 5, ...), you jump to the index j such that arr[i] <= arr[j] and arr[j] is the smallest possible value. If there are multiple such indices j, you can only jump to the smallest such index j.
During even-numbered jumps (i.e., jumps 2, 4, 6, ...), you jump to the index j such that arr[i] >= arr[j] and arr[j] is th e largest possible value. If there are multiple such indices j, you can only jump to the smallest such index j.
It may be the case that for some index i, there are no legal jumps.

A starting index is good if, starting from that index, you can reach the end of the array (index arr.length - 1) by jump ing some number of times (possibly 0 or more than once).
Return the number of good starting indices.

Example 1:

Input: arr = [10,13,12,14,15]
Output: 2
Explanation:
From starting index i = 0, we can make our 1st jump to i = 2 (since arr[2] is the smallest among arr[1], arr[2], arr[3], arr[4] that is greater or equal to arr[0]), then we cannot jump any more.
From starting index i = 1 and i = 2, we can make our 1st jump to i = 3, then we cannot jump any more.
From starting index i = 3, we can make our 1st jump to i = 4, so we have reached the end.
From starting index i = 4, we have reached the end already.
In total, there are 2 different starting indices i = 3 and i = 4, where we can reach the end with some number of jumps.

Example 2:

Input: arr = [2,3,1,1,4]
Output: 3
Explanation:
From starting index i = 0, we make jumps to i = 1, i = 2, i = 3:
During our 1st jump (odd-numbered), we first jump to i = 1 because arr[1] is the smallest value in [arr[1], arr[2], arr[3], arr[4]] that is greater than or equal to arr[0].
During our 2nd jump (even-numbered), we jump from i = 1 to i = 2 because arr[2] is the largest value in [arr[2], arr[3], arr[4]] that is less than or equal to arr[1]. arr[3] is also the largest value, but 2 is a smaller index, so we can only j ump to i = 2 and not i = 3
During our 3rd jump (odd-numbered), we jump from i = 2 to i = 3 because arr[3] is the smallest value in [arr[3], arr[4]] that is greater than or equal to arr[2].
We can't jump from i = 3 to i = 4, so the starting index i = 0 is not good.
In a similar manner, we can deduce that:
From starting index i = 1, we jump to i = 4, so we reach the end.
From starting index i = 2, we jump to i = 3, and then we can't jump anymore.
From starting index i = 3, we jump to i = 4, so we reach the end.
From starting index i = 4, we are already at the end.
In total, there are 3 different starting indices i = 1, i = 3, and i = 4, where we can reach the end with some number of jumps.

Example 3:

Input: arr = [5,1,3,4,2]
Output: 3
Explanation: We can reach the end from starting indices 1, 2, and 4.

Constraints:

1 <= arr.length <= 2 * 104
0 <= arr[i] < 105

---

976. Largest Perimeter Triangle

Given an integer array nums, return the largest perimeter of a triangle with a non-zero area, formed from three of the se lengths. If it is impossible to form any triangle of a non-zero area, return 0.

Example 1:
Input: nums = [2,1,2]
Output: 5
Example 2:
Input: nums = [1,2,1]
Output: 0
Example 3:
Input: nums = [3,2,3,4]
Output: 10
Example 4:
Input: nums = [3,6,2,3]
Output: 8

Constraints:

3 <= nums.length <= 104
1 <= nums[i] <= 106

---

977. Squares of a Sorted Array

Given an integer array nums sorted in non-decreasing order, return an array of the squares of each number sorted in non-decreasing order.

Example 1:

Input: nums = [-4,-1,0,3,10]
Output: [0,1,9,16,100]
Explanation: After squaring, the array becomes [16,1,0,9,100].
After sorting, it becomes [0,1,9,16,100].

Example 2:

Input: nums = [-7,-3,2,3,11]
Output: [4,9,9,49,121]

Constraints:

1 <= nums.length <= 104
-104 <= nums[i] <= 104
nums is sorted in non-decreasing order.

Follow up: Squaring each element and sorting the new array is very trivial, could you find an O(n) solution using a d ifferent approach?

---

978. Longest Turbulent Subarray

Given an integer array arr, return the length of a maximum size turbulent subarray of arr.
A subarray is turbulent if the comparison sign flips between each adjacent pair of elements in the subarray.
More formally, a subarray [arr[i], arr[i + 1], ..., arr[j]] of arr is said to be turbulent if and only if:

For i <= k < j:

arr[k] > arr[k + 1] when k is odd, and
arr[k] < arr[k + 1] when k is even.

Or, for i <= k < j:

arr[k] > arr[k + 1] when k is even, and
arr[k] < arr[k + 1] when k is odd.

Example 1:

Input: arr = [9,4,2,10,7,8,8,1,9]
Output: 5
Explanation: arr[1] > arr[2] < arr[3] > arr[4] < arr[5]

Example 2:

Input: arr = [4,8,12,16]
Output: 2

Example 3:

Input: arr = [100]
Output: 1

Constraints:

1 <= arr.length <= 4 * 104
0 <= arr[i] <= 109

---

979. Distribute Coins in Binary Tree

You are given the root of a binary tree with n nodes where each node in the tree has node.val coins. There are n coin s in total throughout the whole tree.
In one move, we may choose two adjacent nodes and move one coin from one node to another. A move may be from parent to child, or from child to parent.
Return the minimum number of moves required to make every node have exactly one coin.

Example 1:

Input: root = [3,0,0]
Output: 2
Explanation: From the root of the tree, we move one coin to its left child, and one coin to its right child.

Example 2:

Input: root = [0,3,0]
Output: 3
Explanation: From the left child of the root, we move two coins to the root [taking two moves]. Then, we move one coin from the root of the tree to the right child.

Example 3:

Input: root = [1,0,2]
Output: 2

Example 4:

Input: root = [1,0,0,null,3]
Output: 4

Constraints:

The number of nodes in the tree is n.
1 <= n <= 100
0 <= Node.val <= n
The sum of all Node.val is n.

---

980. Unique Paths III

You are given an m x n integer array grid where grid[i][j] could be:

1 representing the starting square. There is exactly one starting square.
2 representing the ending square. There is exactly one ending square.
0 representing empty squares we can walk over.
-1 representing obstacles that we cannot walk over.

Return the number of 4-directional walks from the starting square to the ending square, that walk over every non-obs tacle square exactly once.

Example 1:

Input: grid = [[1,0,0,0],[0,0,0,0],[0,0,2,-1]]
Output: 2
Explanation: We have the following two paths:
1. (0,0),(0,1),(0,2),(0,3),(1,3),(1,2),(1,1),(1,0),(2,0),(2,1),(2,2)
2. (0,0),(1,0),(2,0),(2,1),(1,1),(0,1),(0,2),(0,3),(1,3),(1,2),(2,2)

Example 2:

Input: grid = [[1,0,0,0],[0,0,0,0],[0,0,0,2]]
Output: 4
Explanation: We have the following four paths:
1. (0,0),(0,1),(0,2),(0,3),(1,3),(1,2),(1,1),(1,0),(2,0),(2,1),(2,2),(2,3)
2. (0,0),(0,1),(1,1),(1,0),(2,0),(2,1),(2,2),(1,2),(0,2),(0,3),(1,3),(2,3)
3. (0,0),(1,0),(2,0),(2,1),(2,2),(1,2),(1,1),(0,1),(0,2),(0,3),(1,3),(2,3)
4. (0,0),(1,0),(2,0),(2,1),(1,1),(0,1),(0,2),(0,3),(1,3),(1,2),(2,2),(2,3)

Example 3:

Input: grid = [[0,1],[2,0]]
Output: 0

Explanation: There is no path that walks over every empty square exactly once.
Note that the starting and ending square can be anywhere in the grid.

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 20
1 <= m * n <= 20
-1 <= grid[i][j] <= 2
There is exactly one starting cell and one ending cell.

---

981. Time Based Key-Value Store

Design a time-based key-value data structure that can store multiple values for the same key at different time stamps and retrieve the key's value at a certain timestamp.
Implement the TimeMap class:

TimeMap() Initializes the object of the data structure.
void set(String key, String value, int timestamp) Stores the key key with the value value at the given time timestamp.
String get(String key, int timestamp) Returns a value such that set was called previously, with timestamp_prev <= ti
mestamp. If there are multiple such values, it returns the value associated with the largest timestamp_prev. If there ar
e no values, it returns "".

Example 1:

Input
["TimeMap", "set", "get", "get", "set", "get", "get"]
[[], ["foo", "bar", 1], ["foo", 1], ["foo", 3], ["foo", "bar2", 4], ["foo", 4], ["foo", 5]]
Output
[null, null, "bar", "bar", null, "bar2", "bar2"]

Explanation
TimeMap timeMap = new TimeMap();
timeMap.set("foo", "bar", 1); // store the key "foo" and value "bar" along with timestamp = 1.
timeMap.get("foo", 1);   // return "bar"
timeMap.get("foo", 3);   // return "bar", since there is no value corresponding to foo at timestamp 3 and timestam
p 2, then the only value is at timestamp 1 is "bar".
timeMap.set("foo", "bar2", 4); // store the key "foo" and value "bar2" along with timestamp = 4.
timeMap.get("foo", 4);   // return "bar2"
timeMap.get("foo", 5);   // return "bar2"

Constraints:

1 <= key.length, value.length <= 100
key and value consist of lowercase English letters and digits.
1 <= timestamp <= 107
All the timestamps timestamp of set are strictly increasing.
At most 2 * 105 calls will be made to set and get.

---

982. Triples with Bitwise AND Equal To Zero

Given an integer array nums, return the number of AND triples.
An AND triple is a triple of indices (i, j, k) such that:

0 <= i < nums.length
0 <= j < nums.length
0 <= k < nums.length
nums[i] & nums[j] & nums[k] == 0, where & represents the bitwise-AND operator.

Example 1:

Input: nums = [2,1,3]
Output: 12
Explanation: We could choose the following i, j, k triples:
(i=0, j=0, k=1) : 2 & 2 & 1
(i=0, j=1, k=0) : 2 & 1 & 2
(i=0, j=1, k=1) : 2 & 1 & 1
(i=0, j=1, k=2) : 2 & 1 & 3
(i=0, j=2, k=1) : 2 & 3 & 1
(i=1, j=0, k=0) : 1 & 2 & 2
(i=1, j=0, k=1) : 1 & 2 & 1
(i=1, j=0, k=2) : 1 & 2 & 3
(i=1, j=1, k=0) : 1 & 1 & 2
(i=1, j=2, k=0) : 1 & 3 & 2
(i=2, j=0, k=1) : 3 & 2 & 1
(i=2, j=1, k=0) : 3 & 1 & 2

Example 2:

Input: nums = [0,0,0]
Output: 27

Constraints:

1 <= nums.length <= 1000
0 <= nums[i] < 216

---

983. Minimum Cost For Tickets

You have planned some train traveling one year in advance. The days of the year in which you will travel are given a s an integer array days. Each day is an integer from 1 to 365.
Train tickets are sold in three different ways:

a 1-day pass is sold for costs[0] dollars,
a 7-day pass is sold for costs[1] dollars, and
a 30-day pass is sold for costs[2] dollars.

The passes allow that many days of consecutive travel.

For example, if we get a 7-day pass on day 2, then we can travel for 7 days: 2, 3, 4, 5, 6, 7, and 8.

Return the minimum number of dollars you need to travel every day in the given list of days.

Example 1:

Input: days = [1,4,6,7,8,20], costs = [2,7,15]
Output: 11
Explanation: For example, here is one way to buy passes that lets you travel your travel plan:
On day 1, you bought a 1-day pass for costs[0] = $2, which covered day 1.
On day 3, you bought a 7-day pass for costs[1] = $7, which covered days 3, 4, ..., 9.
On day 20, you bought a 1-day pass for costs[0] = $2, which covered day 20.
In total, you spent $11 and covered all the days of your travel.

Example 2:

Input: days = [1,2,3,4,5,6,7,8,9,10,30,31], costs = [2,7,15]
Output: 17
Explanation: For example, here is one way to buy passes that lets you travel your travel plan:
On day 1, you bought a 30-day pass for costs[2] = $15 which covered days 1, 2, ..., 30.
On day 31, you bought a 1-day pass for costs[0] = $2 which covered day 31.
In total, you spent $17 and covered all the days of your travel.

Constraints:

1 <= days.length <= 365
1 <= days[i] <= 365
days is in strictly increasing order.
costs.length == 3
1 <= costs[i] <= 1000

---

984. String Without AAA or BBB

Given two integers a and b, return any string s such that:

s has length a + b and contains exactly a 'a' letters, and exactly b 'b' letters,
The substring 'aaa' does not occur in s, and
The substring 'bbb' does not occur in s.

Example 1:

Input: a = 1, b = 2
Output: "abb"
Explanation: "abb", "bab" and "bba" are all correct answers.

Example 2:

Input: a = 4, b = 1
Output: "aabaa"

Constraints:

0 <= a, b <= 100
It is guaranteed such an s exists for the given a and b.

---

985. Sum of Even Numbers After Queries

You are given an integer array nums and an array queries where queries[i] = [vali, indexi].
For each query i, first, apply nums[indexi] = nums[indexi] + vali, then print the sum of the even values of nums.
Return an integer array answer where answer[i] is the answer to the ith query.

Example 1:

Input: nums = [1,2,3,4], queries = [[1,0],[-3,1],[-4,0],[2,3]]
Output: [8,6,2,4]
Explanation: At the beginning, the array is [1,2,3,4].
After adding 1 to nums[0], the array is [2,2,3,4], and the sum of even values is 2 + 2 + 4 = 8.
After adding -3 to nums[1], the array is [2,-1,3,4], and the sum of even values is 2 + 4 = 6.

After adding -4 to nums[0], the array is [-2,-1,3,4], and the sum of even values is -2 + 4 = 2.
After adding 2 to nums[3], the array is [-2,-1,3,6], and the sum of even values is -2 + 6 = 4.

Example 2:

Input: nums = [1], queries = [[4,0]]
Output: [0]

Constraints:

1 <= nums.length <= 104
-104 <= nums[i] <= 104
1 <= queries.length <= 104
-104 <= vali <= 104
0 <= indexi < nums.length

---

986. Interval List Intersections

You are given two lists of closed intervals, firstList and secondList, where firstList[i] = [starti, endi] and secondList[j] = [startj, endj]. Each list of intervals is pairwise disjoint and in sorted order.
Return the intersection of these two interval lists.
A closed interval [a, b] (with a <= b) denotes the set of real numbers x with a <= x <= b.
The intersection of two closed intervals is a set of real numbers that are either empty or represented as a closed interval. For example, the intersection of [1, 3] and [2, 4] is [2, 3].

Example 1:

Input: firstList = [[0,2],[5,10],[13,23],[24,25]], secondList = [[1,5],[8,12],[15,24],[25,26]]
Output: [[1,2],[5,5],[8,10],[15,23],[24,24],[25,25]]

Example 2:

Input: firstList = [[1,3],[5,9]], secondList = []
Output: []

Example 3:

Input: firstList = [], secondList = [[4,8],[10,12]]
Output: []

Example 4:

Input: firstList = [[1,7]], secondList = [[3,10]]
Output: [[3,7]]

Constraints:

0 <= firstList.length, secondList.length <= 1000
firstList.length + secondList.length >= 1
0 <= starti < endi <= 109
endi < starti+1
0 <= startj < endj <= 109
endj < startj+1

---

987. Vertical Order Traversal of a Binary Tree

Given the root of a binary tree, calculate the vertical order traversal of the binary tree.
For each node at position (row, col), its left and right children will be at positions (row + 1, col - 1) and (row + 1, col + 1) respectively. The root of the tree is at (0, 0).
The vertical order traversal of a binary tree is a list of top-to-bottom orderings for each column index starting from th e leftmost column and ending on the rightmost column. There may be multiple nodes in the same row and same colu mn. In such a case, sort these nodes by their values.
Return the vertical order traversal of the binary tree.

Example 1:

Input: root = [3,9,20,null,null,15,7]
Output: [[9],[3,15],[20],[7]]
Explanation:
Column -1: Only node 9 is in this column.
Column 0: Nodes 3 and 15 are in this column in that order from top to bottom.
Column 1: Only node 20 is in this column.
Column 2: Only node 7 is in this column.
Example 2:

Input: root = [1,2,3,4,5,6,7]
Output: [[4],[2],[1,5,6],[3],[7]]
Explanation:
Column -2: Only node 4 is in this column.
Column -1: Only node 2 is in this column.
Column 0: Nodes 1, 5, and 6 are in this column.
    1 is at the top, so it comes first.
    5 and 6 are at the same position (2, 0), so we order them by their value, 5 before 6.
Column 1: Only node 3 is in this column.
Column 2: Only node 7 is in this column.

Example 3:

Input: root = [1,2,3,4,6,5,7]
Output: [[4],[2],[1,5,6],[3],[7]]
Explanation:
This case is the exact same as example 2, but with nodes 5 and 6 swapped.
Note that the solution remains the same since 5 and 6 are in the same location and should be ordered by their values.

Constraints:

The number of nodes in the tree is in the range [1, 1000].
0 <= Node.val <= 1000

---

988. Smallest String Starting From Leaf

You are given the root of a binary tree where each node has a value in the range [0, 25] representing the letters 'a' to 'z'.
Return the lexicographically smallest string that starts at a leaf of this tree and ends at the root.
As a reminder, any shorter prefix of a string is lexicographically smaller.

For example, "ab" is lexicographically smaller than "aba".

A leaf of a node is a node that has no children.

Example 1:

Input: root = [0,1,2,3,4,3,4]
Output: "dba"

Example 2:

Input: root = [25,1,3,1,3,0,2]
Output: "adz"

Example 3:

Input: root = [2,2,1,null,1,0,null,0]
Output: "abc"

Constraints:

The number of nodes in the tree is in the range [1, 8500].
0 <= Node.val <= 25

---

989. Add to Array-Form of Integer

The array-form of an integer num is an array representing its digits in left to right order.

For example, for num = 1321, the array form is [1,3,2,1].

Given num, the array-form of an integer, and an integer k, return the array-form of the integer num + k.

Example 1:

Input: num = [1,2,0,0], k = 34
Output: [1,2,3,4]
Explanation: 1200 + 34 = 1234

Example 2:

Input: num = [2,7,4], k = 181
Output: [4,5,5]
Explanation: 274 + 181 = 455

Example 3:

Input: num = [2,1,5], k = 806
Output: [1,0,2,1]
Explanation: 215 + 806 = 1021

Example 4:

Input: num = [9,9,9,9,9,9,9,9,9,9], k = 1
Output: [1,0,0,0,0,0,0,0,0,0,0]
Explanation: 9999999999 + 1 = 10000000000

Constraints:

1 <= num.length <= 104
0 <= num[i] <= 9
num does not contain any leading zeros except for the zero itself.
1 <= k <= 104

---

990. Satisfiability of Equality Equations

You are given an array of strings equations that represent relationships between variables where each string equation s[i] is of length 4 and takes one of two different forms: "xi==yi" or "xi!=yi".Here, xi and yi are lowercase letters (not necessarily different) that represent one-letter variable names.
Return true if it is possible to assign integers to variable names so as to satisfy all the given equations, or false other wise.

Example 1:

Input: equations = ["a==b","b!=a"]
Output: false
Explanation: If we assign say, a = 1 and b = 1, then the first equation is satisfied, but not the second.
There is no way to assign the variables to satisfy both equations.

Example 2:

Input: equations = ["b==a","a==b"]
Output: true
Explanation: We could assign a = 1 and b = 1 to satisfy both equations.

Example 3:

Input: equations = ["a==b","b==c","a==c"]
Output: true

Example 4:

Input: equations = ["a==b","b!=c","c==a"]
Output: false

Example 5:

Input: equations = ["c==c","b==d","x!=z"]
Output: true

Constraints:

1 <= equations.length <= 500
equations[i].length == 4
equations[i][0] is a lowercase letter.
equations[i][1] is either '=' or '!'.
equations[i][2] is '='.
equations[i][3] is a lowercase letter.

---

991. Broken Calculator

There is a broken calculator that has the integer startValue on its display initially. In one operation, you can:

multiply the number on display by 2, or
subtract 1 from the number on display.

Given two integers startValue and target, return the minimum number of operations needed to display target on the c
alculator.

Example 1:

Input: startValue = 2, target = 3
Output: 2
Explanation: Use double operation and then decrement operation {2 -> 4 -> 3}.

Example 2:

Input: startValue = 5, target = 8
Output: 2
Explanation: Use decrement and then double {5 -> 4 -> 8}.

Example 3:

Input: startValue = 3, target = 10
Output: 3
Explanation: Use double, decrement and double {3 -> 6 -> 5 -> 10}.

Example 4:

Input: startValue = 1024, target = 1
Output: 1023
Explanation: Use decrement operations 1023 times.

Constraints:

1 <= x, y <= 109

---

992. Subarrays with K Different Integers

Given an integer array nums and an integer k, return the number of good subarrays of nums.
A good array is an array where the number of different integers in that array is exactly k.

For example, [1,2,3,1,2] has 3 different integers: 1, 2, and 3.

A subarray is a contiguous part of an array.

Example 1:

Input: nums = [1,2,1,2,3], k = 2
Output: 7
Explanation: Subarrays formed with exactly 2 different integers: [1,2], [2,1], [1,2], [2,3], [1,2,1], [2,1,2], [1,2,1,2]

Example 2:

Input: nums = [1,2,1,3,4], k = 3
Output: 3
Explanation: Subarrays formed with exactly 3 different integers: [1,2,1,3], [2,1,3], [1,3,4].

Constraints:

1 <= nums.length <= 2 * 104
1 <= nums[i], k <= nums.length

---

993. Cousins in Binary Tree

Given the root of a binary tree with unique values and the values of two different nodes of the tree x and y, return tru e if the nodes corresponding to the values x and y in the tree are cousins, or false otherwise.
Two nodes of a binary tree are cousins if they have the same depth with different parents.
Note that in a binary tree, the root node is at the depth 0, and children of each depth k node are at the depth k + 1.

Example 1:

Input: root = [1,2,3,4], x = 4, y = 3
Output: false

Example 2:

Input: root = [1,2,3,null,4,null,5], x = 5, y = 4
Output: true

Example 3:

Input: root = [1,2,3,null,4], x = 2, y = 3
Output: false

Constraints:

The number of nodes in the tree is in the range [2, 100].
1 <= Node.val <= 100
Each node has a unique value.
x != y
x and y are exist in the tree.

---

994. Rotting Oranges

You are given an m x n grid where each cell can have one of three values:

0 representing an empty cell,
1 representing a fresh orange, or
2 representing a rotten orange.

Every minute, any fresh orange that is 4-directionally adjacent to a rotten orange becomes rotten.
Return the minimum number of minutes that must elapse until no cell has a fresh orange. If this is impossible, return -1.

Example 1:

Input: grid = [[2,1,1],[1,1,0],[0,1,1]]
Output: 4

Example 2:

Input: grid = [[2,1,1],[0,1,1],[1,0,1]]
Output: -1
Explanation: The orange in the bottom left corner (row 2, column 0) is never rotten, because rotting only happens 4-directionally.

Example 3:

Input: grid = [[0,2]]
Output: 0
Explanation: Since there are already no fresh oranges at minute 0, the answer is just 0.

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 10
grid[i][j] is 0, 1, or 2.

---

995. Minimum Number of K Consecutive Bit Flips

You are given a binary array nums and an integer k.
A k-bit flip is choosing a subarray of length k from nums and simultaneously changing every 0 in the subarray to 1, and every 1 in the subarray to 0.
Return the minimum number of k-bit flips required so that there is no 0 in the array. If it is not possible, return -1.
A subarray is a contiguous part of an array.

Example 1:

Input: nums = [0,1,0], k = 1
Output: 2
Explanation: Flip nums[0], then flip nums[2].

Example 2:

Input: nums = [1,1,0], k = 2
Output: -1
Explanation: No matter how we flip subarrays of size 2, we cannot make the array become [1,1,1].

Example 3:

Input: nums = [0,0,0,1,0,1,1,0], k = 3
Output: 3
Explanation:
Flip nums[0],nums[1],nums[2]: nums becomes [1,1,1,1,0,1,1,0]
Flip nums[4],nums[5],nums[6]: nums becomes [1,1,1,1,1,0,0,0]
Flip nums[5],nums[6],nums[7]: nums becomes [1,1,1,1,1,1,1,1]

Constraints:

1 <= nums.length <= 105
1 <= k <= nums.length

---

996. Number of Squareful Arrays

An array is squareful if the sum of every pair of adjacent elements is a perfect square.
Given an integer array nums, return the number of permutations of nums that are squareful.
Two permutations perm1 and perm2 are different if there is some index i such that perm1[i] != perm2[i].

Example 1:

Input: nums = [1,17,8]
Output: 2
Explanation: [1,8,17] and [17,8,1] are the valid permutations.

Example 2:

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

Constraints:

1 <= nums.length <= 12
0 <= nums[i] <= 109

---

997. Find the Town Judge

In a town, there are n people labeled from 1 to n. There is a rumor that one of these people is secretly the town judge.
If the town judge exists, then:

The town judge trusts nobody.
Everybody (except for the town judge) trusts the town judge.
There is exactly one person that satisfies properties 1 and 2.

You are given an array trust where trust[i] = [ai, bi] representing that the person labeled ai trusts the person labeled b i.
Return the label of the town judge if the town judge exists and can be identified, or return -1 otherwise.

Example 1:

Input: n = 2, trust = [[1,2]]
Output: 2

Example 2:

Input: n = 3, trust = [[1,3],[2,3]]
Output: 3

Example 3:

Input: n = 3, trust = [[1,3],[2,3],[3,1]]
Output: -1

Example 4:

Input: n = 3, trust = [[1,2],[2,3]]
Output: -1

Example 5:

Input: n = 4, trust = [[1,3],[1,4],[2,3],[2,4],[4,3]]
Output: 3

Constraints:

1 <= n <= 1000
0 <= trust.length <= 104
trust[i].length == 2
All the pairs of trust are unique.
ai != bi
1 <= ai, bi <= n

---

998. Maximum Binary Tree II

A maximum tree is a tree where every node has a value greater than any other value in its subtree.
You are given the root of a maximum binary tree and an integer val.
Just as in the previous problem, the given tree was constructed from a list a (root = Construct(a)) recursively with the following Construct(a) routine:

If a is empty, return null.
Otherwise, let a[i] be the largest element of a. Create a root node with the value a[i].
The left child of root will be Construct([a[0], a[1], ..., a[i - 1]]).
The right child of root will be Construct([a[i + 1], a[i + 2], ..., a[a.length - 1]]).
Return root.

Note that we were not given a directly, only a root node root = Construct(a).
Suppose b is a copy of a with the value val appended to it. It is guaranteed that b has unique values.
Return Construct(b).

Example 1:

Input: root = [4,1,3,null,null,2], val = 5
Output: [5,4,null,1,3,null,null,2]
Explanation: a = [1,4,2,3], b = [1,4,2,3,5]

Example 2:

Input: root = [5,2,4,null,1], val = 3
Output: [5,2,4,null,1,null,3]
Explanation: a = [2,1,5,4], b = [2,1,5,4,3]

Example 3:

Input: root = [5,2,3,null,1], val = 4
Output: [5,2,4,null,1,3]
Explanation: a = [2,1,5,3], b = [2,1,5,3,4]

Constraints:

The number of nodes in the tree is in the range [1, 100].
1 <= Node.val <= 100
All the values of the tree are unique.
1 <= val <= 100

----

999. Available Captures for Rook

On an 8 x 8 chessboard, there is exactly one white rook 'R' and some number of white bishops 'B', black pawns 'p', a
nd empty squares '.'.
When the rook moves, it chooses one of four cardinal directions (north, east, south, or west), then moves in that dire
ction until it chooses to stop, reaches the edge of the board, captures a black pawn, or is blocked by a white bishop.
A rook is considered attacking a pawn if the rook can capture the pawn on the rook's turn. The number of available c
aptures for the white rook is the number of pawns that the rook is attacking.
Return the number of available captures for the white rook.

Example 1:

Input: board = [[".",".",".",".",".",".",".","."],[".",".",".","p",".",".",".","."],[".",".",".","R",".",".",".","p"],[".",".",".",".","
.",".",".","."],[".",".",".",".",".",".",".","."],[".",".",".","p",".",".",".","."],[".",".",".",".",".",".",".","."],[".",".",".",".",".",".
",".","."]]
Output: 3
Explanation: In this example, the rook is attacking all the pawns.

Example 2:

Input: board = [[".",".",".",".",".",".",".","."],[".","p","p","p","p","p",".","."],[".","p","p","B","p","p",".","."],[".","p","B","R","B","p",".","."],[".","p","p","B","p","p",".","."],[".","p","p","p","p","p",".","."],[".",".",".",".",".",".",".","."],[".",".",".",".",".",".",".","."]]
Output: 0
Explanation: The bishops are blocking the rook from attacking any of the pawns.

Example 3:

Input: board = [[".",".",".",".",".",".",".","."],[".",".",".","p",".",".",".","."],[".",".",".","p",".",".",".","."],["p","p",".","R",".","p","B","."],[".",".",".",".",".",".",".","."],[".",".",".","B",".",".",".","."],[".",".",".","p",".",".",".","."],[".",".",".",".",".",".",".","."]]
Output: 3
Explanation: The rook is attacking the pawns at positions b5, d6, and f5.

Constraints:

board.length == 8
board[i].length == 8
board[i][j] is either 'R', '.', 'B', or 'p'
There is exactly one cell with board[i][j] == 'R'

1000. Minimum Cost to Merge Stones

There are n piles of stones arranged in a row. The ith pile has stones[i] stones.
A move consists of merging exactly k consecutive piles into one pile, and the cost of this move is equal to the total number of stones in these k piles.
Return the minimum cost to merge all piles of stones into one pile. If it is impossible, return -1.

Example 1:

Input: stones = [3,2,4,1], k = 2
Output: 20
Explanation: We start with [3, 2, 4, 1].
We merge [3, 2] for a cost of 5, and we are left with [5, 4, 1].
We merge [4, 1] for a cost of 5, and we are left with [5, 5].
We merge [5, 5] for a cost of 10, and we are left with [10].
The total cost was 20, and this is the minimum possible.

Example 2:

Input: stones = [3,2,4,1], k = 3
Output: -1
Explanation: After any merge operation, there are 2 piles left, and we can't merge anymore. So the task is impossibl
e.

Example 3:

Input: stones = [3,5,1,2,6], k = 3
Output: 25
Explanation: We start with [3, 5, 1, 2, 6].
We merge [5, 1, 2] for a cost of 8, and we are left with [3, 8, 6].
We merge [3, 8, 6] for a cost of 17, and we are left with [17].
The total cost was 25, and this is the minimum possible.

Constraints:

n == stones.length
1 <= n <= 30
1 <= stones[i] <= 100
2 <= k <= 30

1001. Grid Illumination

There is a 2D grid of size n x n where each cell of this grid has a lamp that is initially turned off.
You are given a 2D array of lamp positions lamps, where lamps[i] = [rowi, coli] indicates that the lamp at grid[rowi]
[coli] is turned on. Even if the same lamp is listed more than once, it is turned on.
When a lamp is turned on, it illuminates its cell and all other cells in the same row, column, or diagonal.
You are also given another 2D array queries, where queries[j] = [rowj, colj]. For the jth query, determine whether gri
d[rowj][colj] is illuminated or not. After answering the jth query, turn off the lamp at grid[rowj][colj] and its 8 adjac
ent lamps if they exist. A lamp is adjacent if its cell shares either a side or corner with grid[rowj][colj].
Return an array of integers ans, where ans[j] should be 1 if the cell in the jth query was illuminated, or 0 if the lamp
was not.

Example 1:

Input: n = 5, lamps = [[0,0],[4,4]], queries = [[1,1],[1,0]]
Output: [1,0]
Explanation: We have the initial grid with all lamps turned off. In the above picture we see the grid after turning on t
he lamp at grid[0][0] then turning on the lamp at grid[4][4].
The 0th query asks if the lamp at grid[1][1] is illuminated or not (the blue square). It is illuminated, so set ans[0] = 1.
 Then, we turn off all lamps in the red square.

The 1st query asks if the lamp at grid[1][0] is illuminated or not (the blue square). It is not illuminated, so set ans[1]
= 0. Then, we turn off all lamps in the red rectangle.

Example 2:

Input: n = 5, lamps = [[0,0],[4,4]], queries = [[1,1],[1,1]]
Output: [1,1]

Example 3:

Input: n = 5, lamps = [[0,0],[0,4]], queries = [[0,4],[0,1],[1,4]]
Output: [1,1,0]

Constraints:

1 <= n <= 109
0 <= lamps.length <= 20000
0 <= queries.length <= 20000
lamps[i].length == 2
0 <= rowi, coli < n
queries[j].length == 2
0 <= rowj, colj < n

1002. Find Common Characters

Given a string array words, return an array of all characters that show up in all strings within the words (including du plicates). You may return the answer in any order.

Example 1:
Input: words = ["bella","label","roller"]
Output: ["e","l","l"]
Example 2:
Input: words = ["cool","lock","cook"]
Output: ["c","o"]

Constraints:

1 <= words.length <= 100
1 <= words[i].length <= 100
words[i] consists of lowercase English letters.

1003. Check If Word Is Valid After Substitutions

Given a string s, determine if it is valid.
A string s is valid if, starting with an empty string t = "", you can transform t into s after performing the following op
eration any number of times:

Insert string "abc" into any position in t. More formally, t becomes tleft + "abc" + tright, where t == tleft + tright. No
te that tleft and tright may be empty.

Return true if s is a valid string, otherwise, return false.

Example 1:

Input: s = "aabcbc"
Output: true
Explanation:
"" -> "abc" -> "aabcbc"
Thus, "aabcbc" is valid.
Example 2:

Input: s = "abcabcababcc"
Output: true
Explanation:
"" -> "abc" -> "abcabc" -> "abcabcabc" -> "abcabcababcc"
Thus, "abcabcababcc" is valid.

Example 3:

Input: s = "abccba"
Output: false
Explanation: It is impossible to get "abccba" using the operation.

Example 4:

Input: s = "cababc"
Output: false
Explanation: It is impossible to get "cababc" using the operation.

Constraints:

1 <= s.length <= 2 * 104
s consists of letters 'a', 'b', and 'c'

1004. Max Consecutive Ones III

Given a binary array nums and an integer k, return the maximum number of consecutive 1's in the array if you can fli p at most k 0's.

Example 1:

Input: nums = [1,1,1,0,0,0,1,1,1,1,0], k = 2
Output: 6
Explanation: [1,1,1,0,0,1,1,1,1,1,1]
Bolded numbers were flipped from 0 to 1. The longest subarray is underlined.
Example 2:

Input: nums = [0,0,1,1,0,0,1,1,1,0,1,1,0,0,0,1,1,1,1], k = 3
Output: 10
Explanation: [0,0,1,1,1,1,1,1,1,1,1,1,0,0,0,1,1,1,1]
Bolded numbers were flipped from 0 to 1. The longest subarray is underlined.

Constraints:

1 <= nums.length <= 105
nums[i] is either 0 or 1.
0 <= k <= nums.length

1005. Maximize Sum Of Array After K Negations

Given an integer array nums and an integer k, modify the array in the following way:

choose an index i and replace nums[i] with -nums[i].

You should apply this process exactly k times. You may choose the same index i multiple times.
Return the largest possible sum of the array after modifying it in this way.

Example 1:

Input: nums = [4,2,3], k = 1
Output: 5
Explanation: Choose index 1 and nums becomes [4,-2,3].

Example 2:

Input: nums = [3,-1,0,2], k = 3
Output: 6

Explanation: Choose indices (1, 2, 2) and nums becomes [3,1,0,2].

Example 3:

Input: nums = [2,-3,-1,5,-4], k = 2
Output: 13
Explanation: Choose indices (1, 4) and nums becomes [2,3,-1,5,4].

Constraints:

1 <= nums.length <= 104
-100 <= nums[i] <= 100
1 <= k <= 104

1006. Clumsy Factorial

The factorial of a positive integer n is the product of all positive integers less than or equal to n.

For example, factorial(10) = 10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1.

We make a clumsy factorial using the integers in decreasing order by swapping out the multiply operations for a fixe d rotation of operations with multiply '*', divide '/', add '+', and subtract '-' in this order.

For example, clumsy(10) = 10 * 9 / 8 + 7 - 6 * 5 / 4 + 3 - 2 * 1.

However, these operations are still applied using the usual order of operations of arithmetic. We do all multiplication and division steps before any addition or subtraction steps, and multiplication and division steps are processed left t o right.
Additionally, the division that we use is floor division such that 10 * 9 / 8 = 90 / 8 = 11.
Given an integer n, return the clumsy factorial of n.

Example 1:

Input: n = 4
Output: 7
Explanation: 7 = 4 * 3 / 2 + 1

Example 2:

Input: n = 10
Output: 12
Explanation: 12 = 10 * 9 / 8 + 7 - 6 * 5 / 4 + 3 - 2 * 1

Constraints:

1 <= n <= 104

1007. Minimum Domino Rotations For Equal Row

In a row of dominoes, tops[i] and bottoms[i] represent the top and bottom halves of the ith domino. (A domino is a ti le with two numbers from 1 to 6 - one on each half of the tile.)
We may rotate the ith domino, so that tops[i] and bottoms[i] swap values.
Return the minimum number of rotations so that all the values in tops are the same, or all the values in bottoms are t he same.
If it cannot be done, return -1.

Example 1:

Input: tops = [2,1,2,4,2,2], bottoms = [5,2,6,2,3,2]
Output: 2
Explanation:
The first figure represents the dominoes as given by tops and bottoms: before we do any rotations.
If we rotate the second and fourth dominoes, we can make every value in the top row equal to 2, as indicated by the s econd figure.

Example 2:

Input: tops = [3,5,1,2,3], bottoms = [3,6,3,3,4]
Output: -1
Explanation:
In this case, it is not possible to rotate the dominoes to make one row of values equal.

Constraints:

2 <= tops.length <= 2 * 104
bottoms.length == tops.length
1 <= tops[i], bottoms[i] <= 6

1008. Construct Binary Search Tree from Preorder Traversal

Given an array of integers preorder, which represents the preorder traversal of a BST (i.e., binary search tree), constr uct the tree and return its root.
It is guaranteed that there is always possible to find a binary search tree with the given requirements for the given tes t cases.
A binary search tree is a binary tree where for every node, any descendant of Node.left has a value strictly less than Node.val, and any descendant of Node.right has a value strictly greater than Node.val.
A preorder traversal of a binary tree displays the value of the node first, then traverses Node.left, then traverses Node .right.

Example 1:

Input: preorder = [8,5,1,7,10,12]
Output: [8,5,10,1,7,null,12]

Example 2:

Input: preorder = [1,3]
Output: [1,null,3]

Constraints:

1 <= preorder.length <= 100
1 <= preorder[i] <= 1000
All the values of preorder are unique.

1009. Complement of Base 10 Integer

The complement of an integer is the integer you get when you flip all the 0's to 1's and all the 1's to 0's in its binary r epresentation.

For example, The integer 5 is "101" in binary and its complement is "010" which is the integer 2.

Given an integer n, return its complement.

Example 1:

Input: n = 5
Output: 2
Explanation: 5 is "101" in binary, with complement "010" in binary, which is 2 in base-10.

Example 2:

Input: n = 7
Output: 0

Explanation: 7 is "111" in binary, with complement "000" in binary, which is 0 in base-10.

Example 3:

Input: n = 10
Output: 5
Explanation: 10 is "1010" in binary, with complement "0101" in binary, which is 5 in base-10.

Constraints:

0 <= n < 109

Note: This question is the same as 476: https://leetcode.com/problems/number-complement/

1010. Pairs of Songs With Total Durations Divisible by 60

You are given a list of songs where the ith song has a duration of time[i] seconds.
Return the number of pairs of songs for which their total duration in seconds is divisible by 60. Formally, we want th e number of indices i, j such that i < j with (time[i] + time[j]) % 60 == 0.

Example 1:

Input: time = [30,20,150,100,40]
Output: 3
Explanation: Three pairs have a total duration divisible by 60:
(time[0] = 30, time[2] = 150): total duration 180
(time[1] = 20, time[3] = 100): total duration 120
(time[1] = 20, time[4] = 40): total duration 60

Example 2:

Input: time = [60,60,60]
Output: 3
Explanation: All three pairs have a total duration of 120, which is divisible by 60.

Constraints:

1 <= time.length <= 6 * 104
1 <= time[i] <= 500

1011. Capacity To Ship Packages Within D Days

A conveyor belt has packages that must be shipped from one port to another within days days.
The ith package on the conveyor belt has a weight of weights[i]. Each day, we load the ship with packages on the co nveyor belt (in the order given by weights). We may not load more weight than the maximum weight capacity of the ship.
Return the least weight capacity of the ship that will result in all the packages on the conveyor belt being shipped wit hin days days.

Example 1:

Input: weights = [1,2,3,4,5,6,7,8,9,10], days = 5
Output: 15
Explanation: A ship capacity of 15 is the minimum to ship all the packages in 5 days like this:
1st day: 1, 2, 3, 4, 5
2nd day: 6, 7
3rd day: 8
4th day: 9
5th day: 10

Note that the cargo must be shipped in the order given, so using a ship of capacity 14 and splitting the packages into parts like (2, 3, 4, 5), (1, 6, 7), (8), (9), (10) is not allowed.

Example 2:

Input: weights = [3,2,2,4,1,4], days = 3
Output: 6
Explanation: A ship capacity of 6 is the minimum to ship all the packages in 3 days like this:
1st day: 3, 2
2nd day: 2, 4
3rd day: 1, 4

Example 3:

Input: weights = [1,2,3,1,1], days = 4
Output: 3
Explanation:
1st day: 1
2nd day: 2
3rd day: 3
4th day: 1, 1

Constraints:

1 <= days <= weights.length <= 5 * 104
1 <= weights[i] <= 500

1012. Numbers With Repeated Digits

Given an integer n, return the number of positive integers in the range [1, n] that have at least one repeated digit.

Example 1:

Input: n = 20
Output: 1
Explanation: The only positive number (<= 20) with at least 1 repeated digit is 11.

Example 2:

Input: n = 100
Output: 10
Explanation: The positive numbers (<= 100) with atleast 1 repeated digit are 11, 22, 33, 44, 55, 66, 77, 88, 99, and 1 00.

Example 3:

Input: n = 1000
Output: 262

Constraints:

1 <= n <= 109

1013. Partition Array Into Three Parts With Equal Sum

Given an array of integers arr, return true if we can partition the array into three non-empty parts with equal sums. Formally, we can partition the array if we can find indexes i + 1 < j with (arr[0] + arr[1] + ... + arr[i] == arr[i + 1] + a rr[i + 2] + ... + arr[j - 1] == arr[j] + arr[j + 1] + ... + arr[arr.length - 1])

Example 1:

Input: arr = [0,2,1,-6,6,-7,9,1,2,0,1]
Output: true
Explanation: 0 + 2 + 1 = -6 + 6 - 7 + 9 + 1 = 2 + 0 + 1

Example 2:

Input: arr = [0,2,1,-6,6,7,9,-1,2,0,1]
Output: false

Example 3:

Input: arr = [3,3,6,5,-2,2,5,1,-9,4]
Output: true
Explanation: 3 + 3 = 6 = 5 - 2 + 2 + 5 + 1 - 9 + 4

Constraints:

3 <= arr.length <= 5 * 104
-104 <= arr[i] <= 104

1014. Best Sightseeing Pair

You are given an integer array values where values[i] represents the value of the ith sightseeing spot. Two sightseein g spots i and j have a distance j - i between them.
The score of a pair (i < j) of sightseeing spots is values[i] + values[j] + i - j: the sum of the values of the sightseeing s pots, minus the distance between them.
Return the maximum score of a pair of sightseeing spots.

Example 1:

Input: values = [8,1,5,2,6]
Output: 11
Explanation: i = 0, j = 2, values[i] + values[j] + i - j = 8 + 5 + 0 - 2 = 11

Example 2:

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

Constraints:

2 <= values.length <= 5 * 104
1 <= values[i] <= 1000

1015. Smallest Integer Divisible by K

Given a positive integer k, you need to find the length of the smallest positive integer n such that n is divisible by k, and n only contains the digit 1.
Return the length of n. If there is no such n, return -1.
Note: n may not fit in a 64-bit signed integer.

Example 1:

Input: k = 1
Output: 1
Explanation: The smallest answer is n = 1, which has length 1.

Example 2:

Input: k = 2
Output: -1
Explanation: There is no such positive integer n divisible by 2.

Example 3:

Input: k = 3
Output: 3
Explanation: The smallest answer is n = 111, which has length 3.

Constraints:

1 <= k <= 105

1016. Binary String With Substrings Representing 1 To N

Given a binary string s and a positive integer n, return true if the binary representation of all the integers in the range [1, n] are substrings of s, or false otherwise.
A substring is a contiguous sequence of characters within a string.

Example 1:
Input: s = "0110", n = 3
Output: true
Example 2:
Input: s = "0110", n = 4
Output: false

Constraints:

1 <= s.length <= 1000
s[i] is either '0' or '1'.
1 <= n <= 109

1017. Convert to Base -2

Given an integer n, return a binary string representing its representation in base -2.
Note that the returned string should not have leading zeros unless the string is "0".

Example 1:

Input: n = 2
Output: "110"
Explantion: (-2)2 + (-2)1 = 2

Example 2:

Input: n = 3
Output: "111"
Explantion: (-2)2 + (-2)1 + (-2)0 = 3

Example 3:

Input: n = 4
Output: "100"
Explantion: (-2)2 = 4

Constraints:

0 <= n <= 109

1018. Binary Prefix Divisible By 5

You are given a binary array nums (0-indexed).
We define xi as the number whose binary representation is the subarray nums[0..i] (from most-significant-bit to least

-significant-bit).

For example, if nums = [1,0,1], then x0 = 1, x1 = 2, and x2 = 5.

Return an array of booleans answer where answer[i] is true if xi is divisible by 5.

Example 1:

Input: nums = [0,1,1]
Output: [true,false,false]
Explanation: The input numbers in binary are 0, 01, 011; which are 0, 1, and 3 in base-10.
Only the first number is divisible by 5, so answer[0] is true.

Example 2:

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

Example 3:

Input: nums = [0,1,1,1,1,1]
Output: [true,false,false,false,true,false]

Example 4:

Input: nums = [1,1,1,0,1]
Output: [false,false,false,false,false]

Constraints:

1 <= nums.length <= 105
nums[i] is 0 or 1.

1019. Next Greater Node In Linked List

You are given the head of a linked list with n nodes.
For each node in the list, find the value of the next greater node. That is, for each node, find the value of the first nod e that is next to it and has a strictly larger value than it.
Return an integer array answer where answer[i] is the value of the next greater node of the ith node (1-indexed). If th e ith node does not have a next greater node, set answer[i] = 0.

Example 1:

Input: head = [2,1,5]

Output: [5,5,0]

Example 2:

Input: head = [2,7,4,3,5]
Output: [7,0,5,5,0]

Constraints:

The number of nodes in the list is n.
1 <= n <= 104
1 <= Node.val <= 109

1020. Number of Enclaves

You are given an m x n binary matrix grid, where 0 represents a sea cell and 1 represents a land cell.
A move consists of walking from one land cell to another adjacent (4-directionally) land cell or walking off the boun dary of the grid.
Return the number of land cells in grid for which we cannot walk off the boundary of the grid in any number of mov es.

Example 1:

Input: grid = [[0,0,0,0],[1,0,1,0],[0,1,1,0],[0,0,0,0]]
Output: 3
Explanation: There are three 1s that are enclosed by 0s, and one 1 that is not enclosed because its on the boundary.

Example 2:

Input: grid = [[0,1,1,0],[0,0,1,0],[0,0,1,0],[0,0,0,0]]
Output: 0
Explanation: All 1s are either on the boundary or can reach the boundary.

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 500
grid[i][j] is either 0 or 1.

1021. Remove Outermost Parentheses

A valid parentheses string is either empty "", "(" + A + ")", or A + B, where A and B are valid parentheses strings, a nd + represents string concatenation.

For example, "", "()", "(())()", and "(()(()))" are all valid parentheses strings.

A valid parentheses string s is primitive if it is nonempty, and there does not exist a way to split it into s = A + B, wit h A and B nonempty valid parentheses strings.
Given a valid parentheses string s, consider its primitive decomposition: s = P1 + P2 + ... + Pk, where Pi are primitiv e valid parentheses strings.
Return s after removing the outermost parentheses of every primitive string in the primitive decomposition of s.

Example 1:

Input: s = "(()())(())"
Output: "()()()"
Explanation:
The input string is "(()())(())", with primitive decomposition "(()())" + "(())".
After removing outer parentheses of each part, this is "()()" + "()" = "()()()".

Example 2:

Input: s = "(()())(())(()(()))"
Output: "()()()()(())"
Explanation:
The input string is "(()())(())(()(()))", with primitive decomposition "(()())" + "(())" + "(()(()))".
After removing outer parentheses of each part, this is "()()" + "()" + "()(())" = "()()()()(())".

Example 3:

Input: s = "()()"
Output: ""
Explanation:
The input string is "()()", with primitive decomposition "()" + "()".
After removing outer parentheses of each part, this is "" + "" = "".

Constraints:

1 <= s.length <= 105
s[i] is either '(' or ')'.
s is a valid parentheses string.

1022. Sum of Root To Leaf Binary Numbers

You are given the root of a binary tree where each node has a value 0 or 1. Each root-to-leaf path represents a binar y number starting with the most significant bit. For example, if the path is 0 -> 1 -> 1 -> 0 -> 1, then this could repre sent 01101 in binary, which is 13.
For all leaves in the tree, consider the numbers represented by the path from the root to that leaf.
Return the sum of these numbers. The answer is guaranteed to fit in a 32-bits integer.

Example 1:

Input: root = [1,0,1,0,1,0,1]
Output: 22
Explanation: (100) + (101) + (110) + (111) = 4 + 5 + 6 + 7 = 22

Example 2:

Input: root = [0]
Output: 0

Example 3:

Input: root = [1]
Output: 1

Example 4:

Input: root = [1,1]
Output: 3

Constraints:

The number of nodes in the tree is in the range [1, 1000].
Node.val is 0 or 1.

1023. Camelcase Matching

Given an array of strings queries and a string pattern, return a boolean array answer where answer[i] is true if queries [i] matches pattern, and false otherwise.
A query word queries[i] matches pattern if you can insert lowercase English letters pattern so that it equals the query

. You may insert each character at any position and you may not insert any characters.

Example 1:

Input: queries = ["FooBar","FooBarTest","FootBall","FrameBuffer","ForceFeedBack"], pattern = "FB"
Output: [true,false,true,true,false]
Explanation: "FooBar" can be generated like this "F" + "oo" + "B" + "ar".
"FootBall" can be generated like this "F" + "oot" + "B" + "all".
"FrameBuffer" can be generated like this "F" + "rame" + "B" + "uffer".

Example 2:

Input: queries = ["FooBar","FooBarTest","FootBall","FrameBuffer","ForceFeedBack"], pattern = "FoBa"
Output: [true,false,true,false,false]
Explanation: "FooBar" can be generated like this "Fo" + "o" + "Ba" + "r".
"FootBall" can be generated like this "Fo" + "ot" + "Ba" + "ll".

Example 3:

Input: queries = ["FooBar","FooBarTest","FootBall","FrameBuffer","ForceFeedBack"], pattern = "FoBaT"
Output: [false,true,false,false,false]
Explanation: "FooBarTest" can be generated like this "Fo" + "o" + "Ba" + "r" + "T" + "est".

Constraints:

1 <= pattern.length, queries.length <= 100
1 <= queries[i].length <= 100
queries[i] and pattern consist of English letters.

1024. Video Stitching

You are given a series of video clips from a sporting event that lasted time seconds. These video clips can be overlap ping with each other and have varying lengths.
Each video clip is described by an array clips where clips[i] = [starti, endi] indicates that the ith clip started at starti a nd ended at endi.
We can cut these clips into segments freely.

For example, a clip [0, 7] can be cut into segments [0, 1] + [1, 3] + [3, 7].

Return the minimum number of clips needed so that we can cut the clips into segments that cover the entire sporting event [0, time]. If the task is impossible, return -1.

Example 1:

Input: clips = [[0,2],[4,6],[8,10],[1,9],[1,5],[5,9]], time = 10

Output: 3
Explanation:
We take the clips [0,2], [8,10], [1,9]; a total of 3 clips.
Then, we can reconstruct the sporting event as follows:
We cut [1,9] into segments [1,2] + [2,8] + [8,9].
Now we have segments [0,2] + [2,8] + [8,10] which cover the sporting event [0, 10].

Example 2:

Input: clips = [[0,1],[1,2]], time = 5
Output: -1
Explanation: We can't cover [0,5] with only [0,1] and [1,2].

Example 3:

Input: clips = [[0,1],[6,8],[0,2],[5,6],[0,4],[0,3],[6,7],[1,3],[4,7],[1,4],[2,5],[2,6],[3,4],[4,5],[5,7],[6,9]], time = 9
Output: 3
Explanation: We can take clips [0,4], [4,7], and [6,9].

Example 4:

Input: clips = [[0,4],[2,8]], time = 5
Output: 2
Explanation: Notice you can have extra video after the event ends.

Constraints:

1 <= clips.length <= 100
0 <= starti <= endi <= 100
1 <= time <= 100

1025. Divisor Game

Alice and Bob take turns playing a game, with Alice starting first.
Initially, there is a number n on the chalkboard. On each player's turn, that player makes a move consisting of:

Choosing any x with 0 < x < n and n % x == 0.
Replacing the number n on the chalkboard with n - x.

Also, if a player cannot make a move, they lose the game.
Return true if and only if Alice wins the game, assuming both players play optimally.

Example 1:

Input: n = 2

Output: true
Explanation: Alice chooses 1, and Bob has no more moves.

Example 2:

Input: n = 3
Output: false
Explanation: Alice chooses 1, Bob chooses 1, and Alice has no more moves.

Constraints:

1 <= n <= 1000

1026. Maximum Difference Between Node and Ancestor

Given the root of a binary tree, find the maximum value v for which there exist different nodes a and b where v = |a.val - b.val| and a is an ancestor of b.
A node a is an ancestor of b if either: any child of a is equal to b or any child of a is an ancestor of b.

Example 1:

Input: root = [8,3,10,1,6,null,14,null,null,4,7,13]
Output: 7
Explanation: We have various ancestor-node differences, some of which are given below :
|8 - 3| = 5
|3 - 7| = 4
|8 - 1| = 7
|10 - 13| = 3
Among all possible differences, the maximum value of 7 is obtained by |8 - 1| = 7.
Example 2:

Input: root = [1,null,2,null,0,3]
Output: 3

Constraints:

The number of nodes in the tree is in the range [2, 5000].
0 <= Node.val <= 105

1027. Longest Arithmetic Subsequence

Given an array nums of integers, return the length of the longest arithmetic subsequence in nums.
Recall that a subsequence of an array nums is a list nums[i1], nums[i2], ..., nums[ik] with 0 <= i1 < i2 < ... < ik <= n ums.length - 1, and that a sequence seq is arithmetic if seq[i+1] - seq[i] are all the same value (for 0 <= i < seq.length - 1).

Example 1:

Input: nums = [3,6,9,12]
Output: 4
Explanation:
The whole array is an arithmetic sequence with steps of length = 3.

Example 2:

Input: nums = [9,4,7,2,10]
Output: 3
Explanation:
The longest arithmetic subsequence is [4,7,10].

Example 3:

Input: nums = [20,1,15,3,10,5,8]
Output: 4
Explanation:
The longest arithmetic subsequence is [20,15,10,5].

Constraints:

2 <= nums.length <= 1000
0 <= nums[i] <= 500

1028. Recover a Tree From Preorder Traversal

We run a preorder depth-first search (DFS) on the root of a binary tree.
At each node in this traversal, we output D dashes (where D is the depth of this node), then we output the value of th is node. If the depth of a node is D, the depth of its immediate child is D + 1. The depth of the root node is 0.
If a node has only one child, that child is guaranteed to be the left child.
Given the output traversal of this traversal, recover the tree and return its root.

Example 1:

Input: traversal = "1-2--3--4-5--6--7"
Output: [1,2,5,3,4,6,7]

Example 2:

Input: traversal = "1-2--3---4-5--6---7"
Output: [1,2,5,3,null,6,null,4,null,7]

Example 3:

Input: traversal = "1-401--349---90--88"
Output: [1,401,null,349,88,90]

Constraints:

The number of nodes in the original tree is in the range [1, 1000].
1 <= Node.val <= 109

1029. Two City Scheduling

A company is planning to interview 2n people. Given the array costs where costs[i] = [aCosti, bCosti], the cost of fly ing the ith person to city a is aCosti, and the cost of flying the ith person to city b is bCosti.
Return the minimum cost to fly every person to a city such that exactly n people arrive in each city.

Example 1:

Input: costs = [[10,20],[30,200],[400,50],[30,20]]
Output: 110
Explanation:
The first person goes to city A for a cost of 10.
The second person goes to city A for a cost of 30.
The third person goes to city B for a cost of 50.
The fourth person goes to city B for a cost of 20.

The total minimum cost is 10 + 30 + 50 + 20 = 110 to have half the people interviewing in each city.

Example 2:

Input: costs = [[259,770],[448,54],[926,667],[184,139],[840,118],[577,469]]

Output: 1859

Example 3:

Input: costs = [[515,563],[451,713],[537,709],[343,819],[855,779],[457,60],[650,359],[631,42]]
Output: 3086

Constraints:

2 * n == costs.length
2 <= costs.length <= 100
costs.length is even.
1 <= aCosti, bCosti <= 1000

1030. Matrix Cells in Distance Order

You are given four integers row, cols, rCenter, and cCenter. There is a rows x cols matrix and you are on the cell wit h the coordinates (rCenter, cCenter).
Return the coordinates of all cells in the matrix, sorted by their distance from (rCenter, cCenter) from the smallest di stance to the largest distance. You may return the answer in any order that satisfies this condition.
The distance between two cells (r1, c1) and (r2, c2) is |r1 - r2| + |c1 - c2|.

Example 1:

Input: rows = 1, cols = 2, rCenter = 0, cCenter = 0
Output: [[0,0],[0,1]]
Explanation: The distances from (0, 0) to other cells are: [0,1]

Example 2:

Input: rows = 2, cols = 2, rCenter = 0, cCenter = 1
Output: [[0,1],[0,0],[1,1],[1,0]]
Explanation: The distances from (0, 1) to other cells are: [0,1,1,2]
The answer [[0,1],[1,1],[0,0],[1,0]] would also be accepted as correct.

Example 3:

Input: rows = 2, cols = 3, rCenter = 1, cCenter = 2
Output: [[1,2],[0,2],[1,1],[0,1],[1,0],[0,0]]
Explanation: The distances from (1, 2) to other cells are: [0,1,1,2,2,3]
There are other answers that would also be accepted as correct, such as [[1,2],[1,1],[0,2],[1,0],[0,1],[0,0]].

Constraints:

1 <= rows, cols <= 100
0 <= rCenter < rows
0 <= cCenter < cols

1031. Maximum Sum of Two Non-Overlapping Subarrays

Given an integer array nums and two integers firstLen and secondLen, return the maximum sum of elements in two non-overlapping subarrays with lengths firstLen and secondLen.
The array with length firstLen could occur before or after the array with length secondLen, but they have to be non-overlapping.
A subarray is a contiguous part of an array.

Example 1:

Input: nums = [0,6,5,2,2,5,1,9,4], firstLen = 1, secondLen = 2
Output: 20
Explanation: One choice of subarrays is [9] with length 1, and [6,5] with length 2.

Example 2:

Input: nums = [3,8,1,3,2,1,8,9,0], firstLen = 3, secondLen = 2
Output: 29
Explanation: One choice of subarrays is [3,8,1] with length 3, and [8,9] with length 2.

Example 3:

Input: nums = [2,1,5,6,0,9,5,0,3,8], firstLen = 4, secondLen = 3
Output: 31
Explanation: One choice of subarrays is [5,6,0,9] with length 4, and [3,8] with length 3.

Constraints:

1 <= firstLen, secondLen <= 1000
2 <= firstLen + secondLen <= 1000
firstLen + secondLen <= nums.length <= 1000
0 <= nums[i] <= 1000

1032. Stream of Characters

Design an algorithm that accepts a stream of characters and checks if a suffix of these characters is a string of a give n array of strings words.
For example, if words = ["abc", "xyz"] and the stream added the four characters (one by one) 'a', 'x', 'y', and 'z', your algorithm should detect that the suffix "xyz" of the characters "axyz" matches "xyz" from words.
Implement the StreamChecker class:

StreamChecker(String[] words) Initializes the object with the strings array words.
boolean query(char letter) Accepts a new character from the stream and returns true if any non-empty suffix from the stream forms a word that is in words.

Example 1:

Input
["StreamChecker", "query", "query", "query", "query", "query", "query", "query", "query", "query", "query", "query", "query"]
[[["cd", "f", "kl"]], ["a"], ["b"], ["c"], ["d"], ["e"], ["f"], ["g"], ["h"], ["i"], ["j"], ["k"], ["l"]]
Output
[null, false, false, false, true, false, true, false, false, false, false, false, true]

Explanation
StreamChecker streamChecker = new StreamChecker(["cd", "f", "kl"]);
streamChecker.query("a"); // return False
streamChecker.query("b"); // return False
streamChecker.query("c"); // return False
streamChecker.query("d"); // return True, because 'cd' is in the wordlist
streamChecker.query("e"); // return False
streamChecker.query("f"); // return True, because 'f' is in the wordlist
streamChecker.query("g"); // return False
streamChecker.query("h"); // return False
streamChecker.query("i"); // return False
streamChecker.query("j"); // return False
streamChecker.query("k"); // return False
streamChecker.query("l"); // return True, because 'kl' is in the wordlist

Constraints:

1 <= words.length <= 2000
1 <= words[i].length <= 2000
words[i] consists of lowercase English letters.
letter is a lowercase English letter.
At most 4 * 104 calls will be made to query.

1033. Moving Stones Until Consecutive

There are three stones in different positions on the X-axis. You are given three integers a, b, and c, the positions of t
he stones.
In one move, you pick up a stone at an endpoint (i.e., either the lowest or highest position stone), and move it to an u
noccupied position between those endpoints. Formally, let's say the stones are currently at positions x, y, and z with
x < y < z. You pick up the stone at either position x or position z, and move that stone to an integer position k, with x
< k < z and k != y.
The game ends when you cannot make any more moves (i.e., the stones are in three consecutive positions).
Return an integer array answer of length 2 where:

answer[0] is the minimum number of moves you can play, and
answer[1] is the maximum number of moves you can play.

Example 1:

Input: a = 1, b = 2, c = 5
Output: [1,2]
Explanation: Move the stone from 5 to 3, or move the stone from 5 to 4 to 3.

Example 2:

Input: a = 4, b = 3, c = 2
Output: [0,0]
Explanation: We cannot make any moves.

Example 3:

Input: a = 3, b = 5, c = 1
Output: [1,2]
Explanation: Move the stone from 1 to 4; or move the stone from 1 to 2 to 4.

Constraints:

1 <= a, b, c <= 100
a, b, and c have different values.

1034. Coloring A Border

You are given an m x n integer matrix grid, and three integers row, col, and color. Each value in the grid represents t
he color of the grid square at that location.
Two squares belong to the same connected component if they have the same color and are next to each other in any
of the 4 directions.
The border of a connected component is all the squares in the connected component that are either 4-directionally ad

jacent to a square not in the component, or on the boundary of the grid (the first or last row or column).
You should color the border of the connected component that contains the square grid[row][col] with color.
Return the final grid.

Example 1:
Input: grid = [[1,1],[1,2]], row = 0, col = 0, color = 3
Output: [[3,3],[3,2]]
Example 2:
Input: grid = [[1,2,2],[2,3,2]], row = 0, col = 1, color = 3
Output: [[1,3,3],[2,3,3]]
Example 3:
Input: grid = [[1,1,1],[1,1,1],[1,1,1]], row = 1, col = 1, color = 2
Output: [[2,2,2],[2,1,2],[2,2,2]]

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 50
1 <= grid[i][j], color <= 1000
0 <= row < m
0 <= col < n

1035. Uncrossed Lines

You are given two integer arrays nums1 and nums2. We write the integers of nums1 and nums2 (in the order they ar e given) on two separate horizontal lines.
We may draw connecting lines: a straight line connecting two numbers nums1[i] and nums2[j] such that:

nums1[i] == nums2[j], and
the line we draw does not intersect any other connecting (non-horizontal) line.

Note that a connecting line cannot intersect even at the endpoints (i.e., each number can only belong to one connecti ng line).
Return the maximum number of connecting lines we can draw in this way.

Example 1:

Input: nums1 = [1,4,2], nums2 = [1,2,4]
Output: 2
Explanation: We can draw 2 uncrossed lines as in the diagram.
We cannot draw 3 uncrossed lines, because the line from nums1[1] = 4 to nums2[2] = 4 will intersect the line from n ums1[2]=2 to nums2[1]=2.

Example 2:

Input: nums1 = [2,5,1,2,5], nums2 = [10,5,2,1,5,2]
Output: 3

Example 3:

Input: nums1 = [1,3,7,1,7,5], nums2 = [1,9,2,5,1]
Output: 2

Constraints:

1 <= nums1.length, nums2.length <= 500
1 <= nums1[i], nums2[j] <= 2000

1036. Escape a Large Maze

There is a 1 million by 1 million grid on an XY-plane, and the coordinates of each grid square are (x, y).
We start at the source = [sx, sy] square and want to reach the target = [tx, ty] square. There is also an array of blocke d squares, where each blocked[i] = [xi, yi] represents a blocked square with coordinates (xi, yi).
Each move, we can walk one square north, east, south, or west if the square is not in the array of blocked squares. W e are also not allowed to walk outside of the grid.
Return true if and only if it is possible to reach the target square from the source square through a sequence of valid moves.

Example 1:

Input: blocked = [[0,1],[1,0]], source = [0,0], target = [0,2]
Output: false
Explanation: The target square is inaccessible starting from the source square because we cannot move.
We cannot move north or east because those squares are blocked.
We cannot move south or west because we cannot go outside of the grid.

Example 2:

Input: blocked = [], source = [0,0], target = [999999,999999]
Output: true
Explanation: Because there are no blocked cells, it is possible to reach the target square.

Constraints:

0 <= blocked.length <= 200
blocked[i].length == 2
0 <= xi, yi < 106

source.length == target.length == 2
0 <= sx, sy, tx, ty < 106
source != target
It is guaranteed that source and target are not blocked.

1037. Valid Boomerang

Given an array points where points[i] = [xi, yi] represents a point on the X-Y plane, return true if these points are a b oomerang.
A boomerang is a set of three points that are all distinct and not in a straight line.

Example 1:
Input: points = [[1,1],[2,3],[3,2]]
Output: true
Example 2:
Input: points = [[1,1],[2,2],[3,3]]
Output: false

Constraints:

points.length == 3
points[i].length == 2
0 <= xi, yi <= 100

1038. Binary Search Tree to Greater Sum Tree

Given the root of a Binary Search Tree (BST), convert it to a Greater Tree such that every key of the original BST is changed to the original key plus the sum of all keys greater than the original key in BST.
As a reminder, a binary search tree is a tree that satisfies these constraints:

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

Example 1:

Input: root = [4,1,6,0,2,5,7,null,null,null,3,null,null,null,8]
Output: [30,36,21,36,35,26,15,null,null,null,33,null,null,null,8]

Example 2:

Input: root = [0,null,1]
Output: [1,null,1]

Example 3:

Input: root = [1,0,2]
Output: [3,3,2]

Example 4:

Input: root = [3,2,4,1]
Output: [7,9,4,10]

Constraints:

The number of nodes in the tree is in the range [1, 100].
0 <= Node.val <= 100
All the values in the tree are unique.
root is guaranteed to be a valid binary search tree.

Note: This question is the same as 538: https://leetcode.com/problems/convert-bst-to-greater-tree/

1039. Minimum Score Triangulation of Polygon

You have a convex n-sided polygon where each vertex has an integer value. You are given an integer array values w here values[i] is the value of the ith vertex (i.e., clockwise order).
You will triangulate the polygon into n - 2 triangles. For each triangle, the value of that triangle is the product of the values of its vertices, and the total score of the triangulation is the sum of these values over all n - 2 triangles in the tr iangulation.
Return the smallest possible total score that you can achieve with some triangulation of the polygon.

Example 1:

Input: values = [1,2,3]
Output: 6
Explanation: The polygon is already triangulated, and the score of the only triangle is 6.

Example 2:

Input: values = [3,7,4,5]
Output: 144
Explanation: There are two triangulations, with possible scores: 3*7*5 + 4*5*7 = 245, or 3*4*5 + 3*4*7 = 144.
The minimum score is 144.

Example 3:

Input: values = [1,3,1,4,1,5]
Output: 13
Explanation: The minimum score triangulation has score 1*1*3 + 1*1*4 + 1*1*5 + 1*1*1 = 13.

Constraints:

n == values.length
3 <= n <= 50
1 <= values[i] <= 100

1040. Moving Stones Until Consecutive II

There are some stones in different positions on the X-axis. You are given an integer array stones, the positions of the stones.
Call a stone an endpoint stone if it has the smallest or largest position. In one move, you pick up an endpoint stone a nd move it to an unoccupied position so that it is no longer an endpoint stone.

In particular, if the stones are at say, stones = [1,2,5], you cannot move the endpoint stone at position 5, since movin g it to any position (such as 0, or 3) will still keep that stone as an endpoint stone.

The game ends when you cannot make any more moves (i.e., the stones are in three consecutive positions).
Return an integer array answer of length 2 where:

answer[0] is the minimum number of moves you can play, and
answer[1] is the maximum number of moves you can play.

Example 1:

Input: stones = [7,4,9]
Output: [1,2]
Explanation: We can move 4 -> 8 for one move to finish the game.
Or, we can move 9 -> 5, 4 -> 6 for two moves to finish the game.

Example 2:

Input: stones = [6,5,4,3,10]
Output: [2,3]
Explanation: We can move 3 -> 8 then 10 -> 7 to finish the game.
Or, we can move 3 -> 7, 4 -> 8, 5 -> 9 to finish the game.
Notice we cannot move 10 -> 2 to finish the game, because that would be an illegal move.

Constraints:

3 <= stones.length <= 104
1 <= stones[i] <= 109
All the values of stones are unique.

1041. Robot Bounded In Circle

On an infinite plane, a robot initially stands at (0, 0) and faces north. The robot can receive one of three instructions:

"G": go straight 1 unit;
"L": turn 90 degrees to the left;
"R": turn 90 degrees to the right.

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 moves from (0,0) to (0,2), turns 180 degrees, and then returns to (0,0).
When repeating these instructions, the robot remains in the circle of radius 2 centered at the origin.
Example 2:

Input: instructions = "GG"
Output: false
Explanation: The robot moves north indefinitely.
Example 3:

Input: instructions = "GL"
Output: true
Explanation: The robot moves from (0, 0) -> (0, 1) -> (-1, 1) -> (-1, 0) -> (0, 0) -> ...

Constraints:

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

1042. Flower Planting With No Adjacent

You have n gardens, labeled from 1 to n, and an array paths where paths[i] = [xi, yi] describes a bidirectional path be tween garden xi to garden yi. In each garden, you want to plant one of 4 types of flowers.
All gardens have at most 3 paths coming into or leaving it.
Your task is to choose a flower type for each garden such that, for any two gardens connected by a path, they have di fferent types of flowers.
Return any such a choice as an array answer, where answer[i] is the type of flower planted in the (i+1)th garden. The flower types are denoted 1, 2, 3, or 4. It is guaranteed an answer exists.

Example 1:

Input: n = 3, paths = [[1,2],[2,3],[3,1]]
Output: [1,2,3]
Explanation:
Gardens 1 and 2 have different types.
Gardens 2 and 3 have different types.
Gardens 3 and 1 have different types.
Hence, [1,2,3] is a valid answer. Other valid answers include [1,2,4], [1,4,2], and [3,2,1].

Example 2:

Input: n = 4, paths = [[1,2],[3,4]]
Output: [1,2,1,2]

Example 3:

Input: n = 4, paths = [[1,2],[2,3],[3,4],[4,1],[1,3],[2,4]]
Output: [1,2,3,4]

Constraints:

1 <= n <= 104
0 <= paths.length <= 2 * 104
paths[i].length == 2
1 <= xi, yi <= n
xi != yi
Every garden has at most 3 paths coming into or leaving it.

1043. Partition Array for Maximum Sum

Given an integer array arr, partition the array into (contiguous) subarrays of length at most k. After partitioning, each subarray has their values changed to become the maximum value of that subarray.
Return the largest sum of the given array after partitioning. Test cases are generated so that the answer fits in a 32-bit integer.

Example 1:

Input: arr = [1,15,7,9,2,5,10], k = 3
Output: 84
Explanation: arr becomes [15,15,15,9,10,10,10]

Example 2:

Input: arr = [1,4,1,5,7,3,6,1,9,9,3], k = 4
Output: 83

Example 3:

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

Constraints:

1 <= arr.length <= 500
0 <= arr[i] <= 109
1 <= k <= arr.length

1044. Longest Duplicate Substring

Given a string s, consider all duplicated substrings: (contiguous) substrings of s that occur 2 or more times. The occu rrences may overlap.
Return any duplicated substring that has the longest possible length. If s does not have a duplicated substring, the ans wer is "".

Example 1:
Input: s = "banana"
Output: "ana"
Example 2:
Input: s = "abcd"
Output: ""

Constraints:

2 <= s.length <= 3 * 104
s consists of lowercase English letters.

1046. Last Stone Weight

You are given an array of integers stones where stones[i] is the weight of the ith stone.
We are playing a game with the stones. On each turn, we choose the heaviest two stones and smash them together. S uppose the heaviest two stones have weights x and y with x <= y. The result of this smash is:

If x == y, both stones are destroyed, and
If x != y, the stone of weight x is destroyed, and the stone of weight y has new weight y - x.

At the end of the game, there is at most one stone left.
Return the smallest possible weight of the left stone. If there are no stones left, return 0.

Example 1:

Input: stones = [2,7,4,1,8,1]
Output: 1
Explanation:
We combine 7 and 8 to get 1 so the array converts to [2,4,1,1,1] then,
we combine 2 and 4 to get 2 so the array converts to [2,1,1,1] then,
we combine 2 and 1 to get 1 so the array converts to [1,1,1] then,
we combine 1 and 1 to get 0 so the array converts to [1] then that's the value of the last stone.

Example 2:

Input: stones = [1]
Output: 1

Constraints:

1 <= stones.length <= 30
1 <= stones[i] <= 1000

1047. Remove All Adjacent Duplicates In String

You are given a string s consisting of lowercase English letters. A duplicate removal consists of choosing two adjace nt and equal letters and removing them.
We repeatedly make duplicate removals on s until we no longer can.
Return the final string after all such duplicate removals have been made. It can be proven that the answer is unique.

Example 1:

Input: s = "abbaca"
Output: "ca"
Explanation:
For example, in "abbaca" we could remove "bb" since the letters are adjacent and equal, and this is the only possible move. The result of this move is that the string is "aaca", of which only "aa" is possible, so the final string is "ca".

Example 2:

Input: s = "azxxzy"
Output: "ay"

Constraints:

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

1048. Longest String Chain

You are given an array of words where each word consists of lowercase English letters.
wordA is a predecessor of wordB if and only if we can insert exactly one letter anywhere in wordA without changin g the order of the other characters to make it equal to wordB.

For example, "abc" is a predecessor of "abac", while "cba" is not a predecessor of "bcad".

A word chain is a sequence of words [word1, word2, ..., wordk] with k >= 1, where word1 is a predecessor of word2 , word2 is a predecessor of word3, and so on. A single word is trivially a word chain with k == 1.
Return the length of the longest possible word chain with words chosen from the given list of words.

Example 1:

Input: words = ["a","b","ba","bca","bda","bdca"]
Output: 4
Explanation: One of the longest word chains is ["a","ba","bda","bdca"].

Example 2:

Input: words = ["xbc","pcxbcf","xb","cxbc","pcxbc"]
Output: 5
Explanation: All the words can be put in a word chain ["xb", "xbc", "cxbc", "pcxbc", "pcxbcf"].

Example 3:

Input: words = ["abcd","dbqca"]
Output: 1
Explanation: The trivial word chain ["abcd"] is one of the longest word chains.
["abcd","dbqca"] is not a valid word chain because the ordering of the letters is changed.

Constraints:

1 <= words.length <= 1000
1 <= words[i].length <= 16
words[i] only consists of lowercase English letters.

1049. Last Stone Weight II

You are given an array of integers stones where stones[i] is the weight of the ith stone.
We are playing a game with the stones. On each turn, we choose any two stones and smash them together. Suppose t he stones have weights x and y with x <= y. The result of this smash is:

If x == y, both stones are destroyed, and
If x != y, the stone of weight x is destroyed, and the stone of weight y has new weight y - x.

At the end of the game, there is at most one stone left.
Return the smallest possible weight of the left stone. If there are no stones left, return 0.

Example 1:

Input: stones = [2,7,4,1,8,1]
Output: 1
Explanation:
We can combine 2 and 4 to get 2, so the array converts to [2,7,1,8,1] then,
we can combine 7 and 8 to get 1, so the array converts to [2,1,1,1] then,
we can combine 2 and 1 to get 1, so the array converts to [1,1,1] then,
we can combine 1 and 1 to get 0, so the array converts to [1], then that's the optimal value.

Example 2:

Input: stones = [31,26,33,21,40]
Output: 5

Example 3:

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

Constraints:

1 <= stones.length <= 30
1 <= stones[i] <= 100

1051. Height Checker

A school is trying to take an annual photo of all the students. The students are asked to stand in a single file line in n on-decreasing order by height. Let this ordering be represented by the integer array expected where expected[i] is the expected height of the ith student in line.
You are given an integer array heights representing the current order that the students are standing in. Each heights[i] is the height of the ith student in line (0-indexed).
Return the number of indices where heights[i] != expected[i].

Example 1:

Input: heights = [1,1,4,2,1,3]
Output: 3
Explanation:
heights: [1,1,4,2,1,3]
expected: [1,1,1,2,3,4]
Indices 2, 4, and 5 do not match.

Example 2:

Input: heights = [5,1,2,3,4]
Output: 5
Explanation:
heights: [5,1,2,3,4]
expected: [1,2,3,4,5]
All indices do not match.

Example 3:

Input: heights = [1,2,3,4,5]
Output: 0
Explanation:
heights: [1,2,3,4,5]
expected: [1,2,3,4,5]

All indices match.

Constraints:

1 <= heights.length <= 100
1 <= heights[i] <= 100

1052. Grumpy Bookstore Owner

There is a bookstore owner that has a store open for n minutes. Every minute, some number of customers enter the st ore. You are given an integer array customers of length n where customers[i] is the number of the customer that ente rs the store at the start of the ith minute and all those customers leave after the end of that minute.
On some minutes, the bookstore owner is grumpy. You are given a binary array grumpy where grumpy[i] is 1 if the bookstore owner is grumpy during the ith minute, and is 0 otherwise.
When the bookstore owner is grumpy, the customers of that minute are not satisfied, otherwise, they are satisfied.
The bookstore owner knows a secret technique to keep themselves not grumpy for minutes consecutive minutes, but can only use it once.
Return the maximum number of customers that can be satisfied throughout the day.

Example 1:

Input: customers = [1,0,1,2,1,1,7,5], grumpy = [0,1,0,1,0,1,0,1], minutes = 3
Output: 16
Explanation: The bookstore owner keeps themselves not grumpy for the last 3 minutes.
The maximum number of customers that can be satisfied = 1 + 1 + 1 + 1 + 7 + 5 = 16.

Example 2:

Input: customers = [1], grumpy = [0], minutes = 1
Output: 1

Constraints:

n == customers.length == grumpy.length
1 <= minutes <= n <= 2 * 104
0 <= customers[i] <= 1000
grumpy[i] is either 0 or 1.

1053. Previous Permutation With One Swap

Given an array of positive integers arr (not necessarily distinct), return the lexicographically largest permutation that is smaller than arr, that can be made with exactly one swap (A swap exchanges the positions of two numbers arr[i] a nd arr[j]). If it cannot be done, then return the same array.

Example 1:

Input: arr = [3,2,1]
Output: [3,1,2]
Explanation: Swapping 2 and 1.

Example 2:

Input: arr = [1,1,5]
Output: [1,1,5]
Explanation: This is already the smallest permutation.

Example 3:

Input: arr = [1,9,4,6,7]
Output: [1,7,4,6,9]
Explanation: Swapping 9 and 7.

Example 4:

Input: arr = [3,1,1,3]
Output: [1,3,1,3]
Explanation: Swapping 1 and 3.

Constraints:

1 <= arr.length <= 104
1 <= arr[i] <= 104

1054. Distant Barcodes

In a warehouse, there is a row of barcodes, where the ith barcode is barcodes[i].
Rearrange the barcodes so that no two adjacent barcodes are equal. You may return any answer, and it is guaranteed an answer exists.

Example 1:
Input: barcodes = [1,1,1,2,2,2]

Output: [2,1,2,1,2,1]
Example 2:
Input: barcodes = [1,1,1,1,2,2,3,3]
Output: [1,3,1,3,1,2,1,2]

Constraints:

1 <= barcodes.length <= 10000
1 <= barcodes[i] <= 10000

1071. Greatest Common Divisor of Strings

For two strings s and t, we say "t divides s" if and only if s = t + ... + t (t concatenated with itself 1 or more times)
Given two strings str1 and str2, return the largest string x such that x divides both str1 and str2.

Example 1:
Input: str1 = "ABCABC", str2 = "ABC"
Output: "ABC"
Example 2:
Input: str1 = "ABABAB", str2 = "ABAB"
Output: "AB"
Example 3:
Input: str1 = "LEET", str2 = "CODE"
Output: ""
Example 4:
Input: str1 = "ABCDEF", str2 = "ABC"
Output: ""

Constraints:

1 <= str1.length <= 1000
1 <= str2.length <= 1000
str1 and str2 consist of English uppercase letters.

1072. Flip Columns For Maximum Number of Equal Rows

You are given an m x n binary matrix matrix.
You can choose any number of columns in the matrix and flip every cell in that column (i.e., Change the value of the cell from 0 to 1 or vice versa).
Return the maximum number of rows that have all values equal after some number of flips.

Example 1:

Input: matrix = [[0,1],[1,1]]
Output: 1
Explanation: After flipping no values, 1 row has all values equal.

Example 2:

Input: matrix = [[0,1],[1,0]]
Output: 2
Explanation: After flipping values in the first column, both rows have equal values.

Example 3:

Input: matrix = [[0,0,0],[0,0,1],[1,1,0]]
Output: 2
Explanation: After flipping values in the first two columns, the last two rows have equal values.

Constraints:

m == matrix.length
n == matrix[i].length
1 <= m, n <= 300
matrix[i][j] is either 0 or 1.

1073. Adding Two Negabinary Numbers

Given two numbers arr1 and arr2 in base -2, return the result of adding them together.
Each number is given in array format: as an array of 0s and 1s, from most significant bit to least significant bit. For example, arr = [1,1,0,1] represents the number (-2)^3 + (-2)^2 + (-2)^0 = -3. A number arr in array, format is also g uaranteed to have no leading zeros: either arr == [0] or arr[0] == 1.
Return the result of adding arr1 and arr2 in the same format: as an array of 0s and 1s with no leading zeros.

Example 1:

Input: arr1 = [1,1,1,1,1], arr2 = [1,0,1]
Output: [1,0,0,0,0]
Explanation: arr1 represents 11, arr2 represents 5, the output represents 16.

Example 2:

Input: arr1 = [0], arr2 = [0]
Output: [0]

Example 3:

Input: arr1 = [0], arr2 = [1]
Output: [1]

Constraints:

1 <= arr1.length, arr2.length <= 1000
arr1[i] and arr2[i] are 0 or 1
arr1 and arr2 have no leading zeros

1074. Number of Submatrices That Sum to Target

Given a matrix and a target, return the number of non-empty submatrices that sum to target.
A submatrix x1, y1, x2, y2 is the set of all cells matrix[x][y] with x1 <= x <= x2 and y1 <= y <= y2.
Two submatrices (x1, y1, x2, y2) and (x1', y1', x2', y2') are different if they have some coordinate that is different: fo
r example, if x1 != x1'.

Example 1:

Input: matrix = [[0,1,0],[1,1,1],[0,1,0]], target = 0
Output: 4
Explanation: The four 1x1 submatrices that only contain 0.

Example 2:

Input: matrix = [[1,-1],[-1,1]], target = 0
Output: 5
Explanation: The two 1x2 submatrices, plus the two 2x1 submatrices, plus the 2x2 submatrix.

Example 3:

Input: matrix = [[904]], target = 0
Output: 0

Constraints:

1 <= matrix.length <= 100
1 <= matrix[0].length <= 100

-1000 <= matrix[i] <= 1000
-10^8 <= target <= 10^8

1078. Occurrences After Bigram

Given two strings first and second, consider occurrences in some text of the form "first second third", where second comes immediately after first, and third comes immediately after second.
Return an array of all the words third for each occurrence of "first second third".

Example 1:
Input: text = "alice is a good girl she is a good student", first = "a", second = "good"
Output: ["girl","student"]
Example 2:
Input: text = "we will we will rock you", first = "we", second = "will"
Output: ["we","rock"]

Constraints:

1 <= text.length <= 1000
text consists of lowercase English letters and spaces.
All the words in text a separated by a single space.
1 <= first.length, second.length <= 10
first and second consist of lowercase English letters.

1079. Letter Tile Possibilities

You have n tiles, where each tile has one letter tiles[i] printed on it.
Return the number of possible non-empty sequences of letters you can make using the letters printed on those tiles.

Example 1:

Input: tiles = "AAB"
Output: 8
Explanation: The possible sequences are "A", "B", "AA", "AB", "BA", "AAB", "ABA", "BAA".

Example 2:

Input: tiles = "AAABBC"
Output: 188

Example 3:

Input: tiles = "V"
Output: 1

Constraints:

1 <= tiles.length <= 7
tiles consists of uppercase English letters.

1080. Insufficient Nodes in Root to Leaf Paths

Given the root of a binary tree and an integer limit, delete all insufficient nodes in the tree simultaneously, and return the root of the resulting binary tree.
A node is insufficient if every root to leaf path intersecting this node has a sum strictly less than limit.
A leaf is a node with no children.

Example 1:

Input: root = [1,2,3,4,-99,-99,7,8,9,-99,-99,12,13,-99,14], limit = 1
Output: [1,2,3,4,null,null,7,8,9,null,14]

Example 2:

Input: root = [5,4,8,11,null,17,4,7,1,null,null,5,3], limit = 22
Output: [5,4,8,11,null,17,4,7,null,null,null,5]

Example 3:

Input: root = [1,2,-3,-5,null,4,null], limit = -1
Output: [1,null,-3,4]

Constraints:

The number of nodes in the tree is in the range [1, 5000].
-105 <= Node.val <= 105
-109 <= limit <= 109

1081. Smallest Subsequence of Distinct Characters

Given a string s, return the lexicographically smallest subsequence of s that contains all the distinct characters of s ex
actly once.

Example 1:

Input: s = "bcabc"
Output: "abc"

Example 2:

Input: s = "cbacdcbc"
Output: "acdb"

Constraints:

1 <= s.length <= 1000
s consists of lowercase English letters.

Note: This question is the same as 316: https://leetcode.com/problems/remove-duplicate-letters/

1089. Duplicate Zeros

Given a fixed-length integer array arr, duplicate each occurrence of zero, shifting the remaining elements to the right
.
Note that elements beyond the length of the original array are not written. Do the above modifications to the input ar
ray in place and do not return anything.

Example 1:

Input: arr = [1,0,2,3,0,4,5,0]
Output: [1,0,0,2,3,0,0,4]
Explanation: After calling your function, the input array is modified to: [1,0,0,2,3,0,0,4]

Example 2:

Input: arr = [1,2,3]

Output: [1,2,3]
Explanation: After calling your function, the input array is modified to: [1,2,3]

Constraints:

1 <= arr.length <= 104
0 <= arr[i] <= 9

1090. Largest Values From Labels

There is a set of n items. You are given two integer arrays values and labels where the value and the label of the ith e lement are values[i] and labels[i] respectively. You are also given two integers numWanted and useLimit.
Choose a subset s of the n elements such that:

The size of the subset s is less than or equal to numWanted.
There are at most useLimit items with the same label in s.

The score of a subset is the sum of the values in the subset.
Return the maximum score of a subset s.

Example 1:

Input: values = [5,4,3,2,1], labels = [1,1,2,2,3], numWanted = 3, useLimit = 1
Output: 9
Explanation: The subset chosen is the first, third, and fifth items.

Example 2:

Input: values = [5,4,3,2,1], labels = [1,3,3,3,2], numWanted = 3, useLimit = 2
Output: 12
Explanation: The subset chosen is the first, second, and third items.

Example 3:

Input: values = [9,8,8,7,6], labels = [0,0,0,1,1], numWanted = 3, useLimit = 1
Output: 16
Explanation: The subset chosen is the first and fourth items.

Example 4:

Input: values = [9,8,8,7,6], labels = [0,0,0,1,1], numWanted = 3, useLimit = 2
Output: 24
Explanation: The subset chosen is the first, second, and fourth items.

Constraints:

n == values.length == labels.length
1 <= n <= 2 * 104
0 <= values[i], labels[i] <= 2 * 104
1 <= numWanted, useLimit <= n

1091. Shortest Path in Binary Matrix

Given an n x n binary matrix grid, return the length of the shortest clear path in the matrix. If there is no clear path, r eturn -1.
A clear path in a binary matrix is a path from the top-left cell (i.e., (0, 0)) to the bottom-right cell (i.e., (n - 1, n - 1)) s uch that:

All the visited cells of the path are 0.
All the adjacent cells of the path are 8-directionally connected (i.e., they are different and they share an edge or a cor ner).

The length of a clear path is the number of visited cells of this path.

Example 1:

Input: grid = [[0,1],[1,0]]
Output: 2

Example 2:

Input: grid = [[0,0,0],[1,1,0],[1,1,0]]
Output: 4

Example 3:

Input: grid = [[1,0,0],[1,1,0],[1,1,0]]
Output: -1

Constraints:

n == grid.length
n == grid[i].length
1 <= n <= 100
grid[i][j] is 0 or 1

1092. Shortest Common Supersequence

Given two strings str1 and str2, return the shortest string that has both str1 and str2 as subsequences. If there are mult iple valid strings, return any of them.
A string s is a subsequence of string t if deleting some number of characters from t (possibly 0) results in the string s.

Example 1:

Input: str1 = "abac", str2 = "cab"
Output: "cabac"
Explanation:
str1 = "abac" is a subsequence of "cabac" because we can delete the first "c".
str2 = "cab" is a subsequence of "cabac" because we can delete the last "ac".
The answer provided is the shortest such string that satisfies these properties.

Example 2:

Input: str1 = "aaaaaaaa", str2 = "aaaaaaaa"
Output: "aaaaaaaa"

Constraints:

1 <= str1.length, str2.length <= 1000
str1 and str2 consist of lowercase English letters.

1093. Statistics from a Large Sample

You are given a large sample of integers in the range [0, 255]. Since the sample is so large, it is represented by an arr ay count where count[k] is the number of times that k appears in the sample.
Calculate the following statistics:

minimum: The minimum element in the sample.
maximum: The maximum element in the sample.
mean: The average of the sample, calculated as the total sum of all elements divided by the total number of elements.
median:

If the sample has an odd number of elements, then the median is the middle element once the sample is sorted.
If the sample has an even number of elements, then the median is the average of the two middle elements once the sa

mple is sorted.

mode: The number that appears the most in the sample. It is guaranteed to be unique.

Return the statistics of the sample as an array of floating-point numbers [minimum, maximum, mean, median, mode] . Answers within 10-5 of the actual answer will be accepted.

Example 1:

Input: count = [0,1,3,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
Output: [1.00000,3.00000,2.37500,2.50000,3.00000]
Explanation: The sample represented by count is [1,2,2,2,3,3,3,3].
The minimum and maximum are 1 and 3 respectively.
The mean is (1+2+2+2+3+3+3+3) / 8 = 19 / 8 = 2.375.
Since the size of the sample is even, the median is the average of the two middle elements 2 and 3, which is 2.5.
The mode is 3 as it appears the most in the sample.

Example 2:

Input: count = [0,4,3,2,2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
Output: [1.00000,4.00000,2.18182,2.00000,1.00000]
Explanation: The sample represented by count is [1,1,1,1,2,2,2,3,3,4,4].
The minimum and maximum are 1 and 4 respectively.
The mean is (1+1+1+1+2+2+2+3+3+4+4) / 11 = 24 / 11 = 2.18181818... (for display purposes, the output shows the rounded number 2.18182).
Since the size of the sample is odd, the median is the middle element 2.
The mode is 1 as it appears the most in the sample.

Constraints:

count.length == 256
0 <= count[i] <= 109
1 <= sum(count) <= 109
The mode of the sample that count represents is unique.

1094. Car Pooling

There is a car with capacity empty seats. The vehicle only drives east (i.e., it cannot turn around and drive west).
You are given the integer capacity and an array trips where trip[i] = [numPassengersi, fromi, toi] indicates that the it h trip has numPassengersi passengers and the locations to pick them up and drop them off are fromi and toi respectiv ely. The locations are given as the number of kilometers due east from the car's initial location.
Return true if it is possible to pick up and drop off all passengers for all the given trips, or false otherwise.

Example 1:

Input: trips = [[2,1,5],[3,3,7]], capacity = 4
Output: false

Example 2:

Input: trips = [[2,1,5],[3,3,7]], capacity = 5
Output: true

Example 3:

Input: trips = [[2,1,5],[3,5,7]], capacity = 3
Output: true

Example 4:

Input: trips = [[3,2,7],[3,7,9],[8,3,9]], capacity = 11
Output: true

Constraints:

1 <= trips.length <= 1000
trips[i].length == 3
1 <= numPassengersi <= 100
0 <= fromi < toi <= 1000
1 <= capacity <= 105

1095. Find in Mountain Array

(This problem is an interactive problem.)
You may recall that an array arr is a mountain array if and only if:

arr.length >= 3
There exists some i with 0 < i < arr.length - 1 such that:

arr[0] < arr[1] < ... < arr[i - 1] < arr[i]
arr[i] > arr[i + 1] > ... > arr[arr.length - 1]

Given a mountain array mountainArr, return the minimum index such that mountainArr.get(index) == target. If such an index does not exist, return -1.
You cannot access the mountain array directly. You may only access the array using a MountainArray interface:

MountainArray.get(k) returns the element of the array at index k (0-indexed).
MountainArray.length() returns the length of the array.

Submissions making more than 100 calls to MountainArray.get will be judged Wrong Answer. Also, any solutions t hat attempt to circumvent the judge will result in disqualification.

Example 1:

Input: array = [1,2,3,4,5,3,1], target = 3
Output: 2
Explanation: 3 exists in the array, at index=2 and index=5. Return the minimum index, which is 2.
Example 2:

Input: array = [0,1,2,4,2,1], target = 3
Output: -1
Explanation: 3 does not exist in the array, so we return -1.

Constraints:

3 <= mountain_arr.length() <= 104
0 <= target <= 109
0 <= mountain_arr.get(index) <= 109

1096. Brace Expansion II

Under the grammar given below, strings can represent a set of lowercase words. Let R(expr) denote the set of words the expression represents.
The grammar can best be understood through simple examples:

Single letters represent a singleton set containing that word.

R("a") = {"a"}
R("w") = {"w"}

When we take a comma-delimited list of two or more expressions, we take the union of possibilities.

R("{a,b,c}") = {"a","b","c"}

R("{{a,b},{b,c}}") = {"a","b","c"} (notice the final set only contains each word at most once)

When we concatenate two expressions, we take the set of possible concatenations between two words where the first word comes from the first expression and the second word comes from the second expression.

R("{a,b}{c,d}") = {"ac","ad","bc","bd"}
R("a{b,c}{d,e}f{g,h}") = {"abdfg", "abdfh", "abefg", "abefh", "acdfg", "acdfh", "acefg", "acefh"}

Formally, the three rules for our grammar:

For every lowercase letter x, we have R(x) = {x}.
For expressions e1, e2, ... , ek with k >= 2, we have R({e1, e2, ...}) = R(e1) □ R(e2) □ ...
For expressions e1 and e2, we have R(e1 + e2) = {a + b for (a, b) in R(e1) × R(e2)}, where + denotes concatenation, and × denotes the cartesian product.

Given an expression representing a set of words under the given grammar, return the sorted list of words that the exp ression represents.

Example 1:

Input: expression = "{a,b}{c,{d,e}}"
Output: ["ac","ad","ae","bc","bd","be"]

Example 2:

Input: expression = "{{a,z},a{b,c},{ab,z}}"
Output: ["a","ab","ac","z"]
Explanation: Each distinct word is written only once in the final answer.

Constraints:

1 <= expression.length <= 60
expression[i] consists of '{', '}', ','or lowercase English letters.
The given expression represents a set of words based on the grammar given in the description.

1103. Distribute Candies to People

We distribute some number of candies, to a row of n = num_people people in the following way:
We then give 1 candy to the first person, 2 candies to the second person, and so on until we give n candies to the last person.
Then, we go back to the start of the row, giving n + 1 candies to the first person, n + 2 candies to the second person, and so on until we give 2 * n candies to the last person.

This process repeats (with us giving one more candy each time, and moving to the start of the row after we reach the end) until we run out of candies. The last person will receive all of our remaining candies (not necessarily one more than the previous gift).
Return an array (of length num_people and sum candies) that represents the final distribution of candies.

Example 1:

Input: candies = 7, num_people = 4
Output: [1,2,3,1]
Explanation:
On the first turn, ans[0] += 1, and the array is [1,0,0,0].
On the second turn, ans[1] += 2, and the array is [1,2,0,0].
On the third turn, ans[2] += 3, and the array is [1,2,3,0].
On the fourth turn, ans[3] += 1 (because there is only one candy left), and the final array is [1,2,3,1].

Example 2:

Input: candies = 10, num_people = 3
Output: [5,2,3]
Explanation:
On the first turn, ans[0] += 1, and the array is [1,0,0].
On the second turn, ans[1] += 2, and the array is [1,2,0].
On the third turn, ans[2] += 3, and the array is [1,2,3].
On the fourth turn, ans[0] += 4, and the final array is [5,2,3].

Constraints:

1 <= candies <= 10^9
1 <= num_people <= 1000

1104. Path In Zigzag Labelled Binary Tree

In an infinite binary tree where every node has two children, the nodes are labelled in row order.
In the odd numbered rows (ie., the first, third, fifth,...), the labelling is left to right, while in the even numbered rows (second, fourth, sixth,...), the labelling is right to left.

Given the label of a node in this tree, return the labels in the path from the root of the tree to the node with that label.

Example 1:

Input: label = 14
Output: [1,3,4,14]

Example 2:

Input: label = 26
Output: [1,2,6,10,26]

Constraints:

1 <= label <= 10^6

1105. Filling Bookcase Shelves

You are given an array books where books[i] = [thicknessi, heighti] indicates the thickness and height of the ith boo k. You are also given an integer shelfWidth.
We want to place these books in order onto bookcase shelves that have a total width shelfWidth.
We choose some of the books to place on this shelf such that the sum of their thickness is less than or equal to shelf Width, then build another level of the shelf of the bookcase so that the total height of the bookcase has increased by t he maximum height of the books we just put down. We repeat this process until there are no more books to place.
Note that at each step of the above process, the order of the books we place is the same order as the given sequence o f books.

For example, if we have an ordered list of 5 books, we might place the first and second book onto the first shelf, the t hird book on the second shelf, and the fourth and fifth book on the last shelf.

Return the minimum possible height that the total bookshelf can be after placing shelves in this manner.

Example 1:

Input: books = [[1,1],[2,3],[2,3],[1,1],[1,1],[1,1],[1,2]], shelf_width = 4
Output: 6
Explanation:
The sum of the heights of the 3 shelves is 1 + 3 + 2 = 6.
Notice that book number 2 does not have to be on the first shelf.

Example 2:

Input: books = [[1,3],[2,4],[3,2]], shelfWidth = 6
Output: 4

Constraints:

1 <= books.length <= 1000
1 <= thicknessi <= shelfWidth <= 1000
1 <= heighti <= 1000

1106. Parsing A Boolean Expression

Return the result of evaluating a given boolean expression, represented as a string.
An expression can either be:

"t", evaluating to True;
"f", evaluating to False;
"!(expr)", evaluating to the logical NOT of the inner expression expr;
"&(expr1,expr2,...)", evaluating to the logical AND of 2 or more inner expressions expr1, expr2, ...;
"|(expr1,expr2,...)", evaluating to the logical OR of 2 or more inner expressions expr1, expr2, ...

Example 1:

Input: expression = "!(f)"
Output: true

Example 2:

Input: expression = "|(f,t)"
Output: true

Example 3:

Input: expression = "&(t,f)"
Output: false

Example 4:

Input: expression = "|(&(t,f,t),!(t))"
Output: false

Constraints:

1 <= expression.length <= 20000
expression[i] consists of characters in {'(', ')', '&', '|', '!', 't', 'f', ','}.
expression is a valid expression representing a boolean, as given in the description.

1108. Defanging an IP Address

Given a valid (IPv4) IP address, return a defanged version of that IP address.
A defanged IP address replaces every period "." with "[.]".

Example 1:
Input: address = "1.1.1.1"
Output: "1[.]1[.]1[.]1"
Example 2:
Input: address = "255.100.50.0"
Output: "255[.]100[.]50[.]0"

Constraints:

The given address is a valid IPv4 address.

1109. Corporate Flight Bookings

There are n flights that are labeled from 1 to n.
You are given an array of flight bookings bookings, where bookings[i] = [firsti, lasti, seatsi] represents a booking for flights firsti through lasti (inclusive) with seatsi seats reserved for each flight in the range.
Return an array answer of length n, where answer[i] is the total number of seats reserved for flight i.

Example 1:

Input: bookings = [[1,2,10],[2,3,20],[2,5,25]], n = 5
Output: [10,55,45,25,25]
Explanation:
Flight labels: 1 2 3 4 5
Booking 1 reserved: 10 10
Booking 2 reserved: 20 20
Booking 3 reserved: 25 25 25 25
Total seats: 10 55 45 25 25
Hence, answer = [10,55,45,25,25]

Example 2:

Input: bookings = [[1,2,10],[2,2,15]], n = 2
Output: [10,25]
Explanation:
Flight labels: 1 2
Booking 1 reserved: 10 10
Booking 2 reserved: 15
Total seats: 10 25
Hence, answer = [10,25]

Constraints:

1 <= n <= 2 * 104
1 <= bookings.length <= 2 * 104
bookings[i].length == 3
1 <= firsti <= lasti <= n
1 <= seatsi <= 104

1110. Delete Nodes And Return Forest

Given the root of a binary tree, each node in the tree has a distinct value.
After deleting all nodes with a value in to_delete, we are left with a forest (a disjoint union of trees).
Return the roots of the trees in the remaining forest. You may return the result in any order.

Example 1:

Input: root = [1,2,3,4,5,6,7], to_delete = [3,5]
Output: [[1,2,null,4],[6],[7]]

Example 2:

Input: root = [1,2,4,null,3], to_delete = [3]
Output: [[1,2,4]]

Constraints:

The number of nodes in the given tree is at most 1000.
Each node has a distinct value between 1 and 1000.
to_delete.length <= 1000
to_delete contains distinct values between 1 and 1000.

1111. Maximum Nesting Depth of Two Valid Parentheses Strings

A string is a valid parentheses string (denoted VPS) if and only if it consists of "(" and ")" characters only, and:

It is the empty string, or
It can be written as AB (A concatenated with B), where A and B are VPS's, or
It can be written as (A), where A is a VPS.

We can similarly define the nesting depth depth(S) of any VPS S as follows:

depth("") = 0
depth(A + B) = max(depth(A), depth(B)), where A and B are VPS's
depth("(" + A + ")") = 1 + depth(A), where A is a VPS.

For example, "", "()()", and "()(()())" are VPS's (with nesting depths 0, 1, and 2), and ")(" and "(()" are not VPS's.

Given a VPS seq, split it into two disjoint subsequences A and B, such that A and B are VPS's (and A.length + B.len
gth = seq.length).
Now choose any such A and B such that max(depth(A), depth(B)) is the minimum possible value.
Return an answer array (of length seq.length) that encodes such a choice of A and B: answer[i] = 0 if seq[i] is part o
f A, else answer[i] = 1. Note that even though multiple answers may exist, you may return any of them.

Example 1:

Input: seq = "(()())"
Output: [0,1,1,1,1,0]

Example 2:

Input: seq = "()(())()"
Output: [0,0,0,1,1,0,1,1]

Constraints:

1 <= seq.size <= 10000

1114. Print in Order

Suppose we have a class:

```
public class Foo {
  public void first() { print("first"); }
  public void second() { print("second"); }
  public void third() { print("third"); }
}
```

The same instance of Foo will be passed to three different threads. Thread A will call first(), thread B will call secon
d(), and thread C will call third(). Design a mechanism and modify the program to ensure that second() is executed a

fter first(), and third() is executed after second().
Note:
We do not know how the threads will be scheduled in the operating system, even though the numbers in the input se em to imply the ordering. The input format you see is mainly to ensure our tests' comprehensiveness.

Example 1:

Input: nums = [1,2,3]
Output: "firstsecondthird"
Explanation: There are three threads being fired asynchronously. The input [1,2,3] means thread A calls first(), threa d B calls second(), and thread C calls third(). "firstsecondthird" is the correct output.

Example 2:

Input: nums = [1,3,2]
Output: "firstsecondthird"
Explanation: The input [1,3,2] means thread A calls first(), thread B calls third(), and thread C calls second(). "firstse condthird" is the correct output.

Constraints:

nums is a permutation of [1, 2, 3].

1115. Print FooBar Alternately

Suppose you are given the following code:

```
class FooBar {
  public void foo() {
    for (int i = 0; i < n; i++) {
      print("foo");
    }
  }

  public void bar() {
    for (int i = 0; i < n; i++) {
      print("bar");
    }
  }
}
```

The same instance of FooBar will be passed to two different threads:

thread A will call foo(), while
thread B will call bar().

Modify the given program to output "foobar" n times.

Example 1:

Input: n = 1
Output: "foobar"
Explanation: There are two threads being fired asynchronously. One of them calls foo(), while the other calls bar(). "foobar" is being output 1 time.

Example 2:

Input: n = 2
Output: "foobarfoobar"
Explanation: "foobar" is being output 2 times.

Constraints:

1 <= n <= 1000

1116. Print Zero Even Odd

You have a function printNumber that can be called with an integer parameter and prints it to the console.

For example, calling printNumber(7) prints 7 to the console.

You are given an instance of the class ZeroEvenOdd that has three functions: zero, even, and odd. The same instance of ZeroEvenOdd will be passed to three different threads:

Thread A: calls zero() that should only output 0's.
Thread B: calls even() that should only output even numbers.
Thread C: calls odd() that should only output odd numbers.

Modify the given class to output the series "010203040506..." where the length of the series must be 2n.
Implement the ZeroEvenOdd class:

ZeroEvenOdd(int n) Initializes the object with the number n that represents the numbers that should be printed.
void zero(printNumber) Calls printNumber to output one zero.
void even(printNumber) Calls printNumber to output one even number.
void odd(printNumber) Calls printNumber to output one odd number.

Example 1:

Input: n = 2

Output: "0102"
Explanation: There are three threads being fired asynchronously.
One of them calls zero(), the other calls even(), and the last one calls odd().
"0102" is the correct output.

Example 2:

Input: n = 5
Output: "0102030405"

Constraints:

1 <= n <= 1000

1117. Building H2O

There are two kinds of threads: oxygen and hydrogen. Your goal is to group these threads to form water molecules.
There is a barrier where each thread has to wait until a complete molecule can be formed. Hydrogen and oxygen thre ads will be given releaseHydrogen and releaseOxygen methods respectively, which will allow them to pass the barri er. These threads should pass the barrier in groups of three, and they must immediately bond with each other to form a water molecule. You must guarantee that all the threads from one molecule bond before any other threads from th e next molecule do.
In other words:

If an oxygen thread arrives at the barrier when no hydrogen threads are present, it must wait for two hydrogen thread s.
If a hydrogen thread arrives at the barrier when no other threads are present, it must wait for an oxygen thread and an other hydrogen thread.

We do not have to worry about matching the threads up explicitly; the threads do not necessarily know which other t hreads they are paired up with. The key is that threads pass the barriers in complete sets; thus, if we examine the seq uence of threads that bind and divide them into groups of three, each group should contain one oxygen and two hydr ogen threads.
Write synchronization code for oxygen and hydrogen molecules that enforces these constraints.

Example 1:

Input: water = "HOH"
Output: "HHO"
Explanation: "HOH" and "OHH" are also valid answers.

Example 2:

Input: water = "OOHHHH"
Output: "HHOHHO"

Explanation: "HOHHHO", "OHHHHO", "HHOHOH", "HOHHOH", "OHHHOH", "HHOOHH", "HOHOHH" and "OHHOHH" are also valid answers.

Constraints:

3 * n == water.length
1 <= n <= 20
water[i] is either 'H' or 'O'.
There will be exactly 2 * n 'H' in water.
There will be exactly n 'O' in water.

1122. Relative Sort Array

Given two arrays arr1 and arr2, the elements of arr2 are distinct, and all elements in arr2 are also in arr1.
Sort the elements of arr1 such that the relative ordering of items in arr1 are the same as in arr2. Elements that do not appear in arr2 should be placed at the end of arr1 in ascending order.

Example 1:

Input: arr1 = [2,3,1,3,2,4,6,7,9,2,19], arr2 = [2,1,4,3,9,6]
Output: [2,2,2,1,4,3,3,9,6,7,19]

Example 2:

Input: arr1 = [28,6,22,8,44,17], arr2 = [22,28,8,6]
Output: [22,28,8,6,17,44]

Constraints:

1 <= arr1.length, arr2.length <= 1000
0 <= arr1[i], arr2[i] <= 1000
All the elements of arr2 are distinct.
Each arr2[i] is in arr1.

1123. Lowest Common Ancestor of Deepest Leaves

Given the root of a binary tree, return the lowest common ancestor of its deepest leaves.
Recall that:

The node of a binary tree is a leaf if and only if it has no children
The depth of the root of the tree is 0. if the depth of a node is d, the depth of each of its children is d + 1.
The lowest common ancestor of a set S of nodes, is the node A with the largest depth such that every node in S is in t
he subtree with root A.

Example 1:

Input: root = [3,5,1,6,2,0,8,null,null,7,4]
Output: [2,7,4]
Explanation: We return the node with value 2, colored in yellow in the diagram.
The nodes coloured in blue are the deepest leaf-nodes of the tree.
Note that nodes 6, 0, and 8 are also leaf nodes, but the depth of them is 2, but the depth of nodes 7 and 4 is 3.
Example 2:

Input: root = [1]
Output: [1]
Explanation: The root is the deepest node in the tree, and it's the lca of itself.

Example 3:

Input: root = [0,1,3,null,2]
Output: [2]
Explanation: The deepest leaf node in the tree is 2, the lca of one node is itself.

Constraints:

The number of nodes in the tree will be in the range [1, 1000].
0 <= Node.val <= 1000
The values of the nodes in the tree are unique.

Note: This question is the same as 865: https://leetcode.com/problems/smallest-subtree-with-all-the-deepest-nodes/

1124. Longest Well-Performing Interval

We are given hours, a list of the number of hours worked per day for a given employee.
A day is considered to be a tiring day if and only if the number of hours worked is (strictly) greater than 8.
A well-performing interval is an interval of days for which the number of tiring days is strictly larger than the numbe
r of non-tiring days.
Return the length of the longest well-performing interval.

Example 1:

Input: hours = [9,9,6,0,6,6,9]
Output: 3
Explanation: The longest well-performing interval is [9,9,6].

Example 2:

Input: hours = [6,6,6]
Output: 0

Constraints:

1 <= hours.length <= 104
0 <= hours[i] <= 16

1125. Smallest Sufficient Team

In a project, you have a list of required skills req_skills, and a list of people. The ith person people[i] contains a list o f skills that the person has.
Consider a sufficient team: a set of people such that for every required skill in req_skills, there is at least one person in the team who has that skill. We can represent these teams by the index of each person.

For example, team = [0, 1, 3] represents the people with skills people[0], people[1], and people[3].

Return any sufficient team of the smallest possible size, represented by the index of each person. You may return the answer in any order.
It is guaranteed an answer exists.

Example 1:
Input: req_skills = ["java","nodejs","reactjs"], people = [["java"],["nodejs"],["nodejs","reactjs"]]
Output: [0,2]
Example 2:
Input: req_skills = ["algorithms","math","java","reactjs","csharp","aws"], people = [["algorithms","math","java"],["a lgorithms","math","reactjs"],["java","csharp","aws"],["reactjs","csharp"],["csharp","math"],["aws","java"]]
Output: [1,2]

Constraints:

1 <= req_skills.length <= 16
1 <= req_skills[i].length <= 16
req_skills[i] consists of lowercase English letters.
All the strings of req_skills are unique.
1 <= people.length <= 60

0 <= people[i].length <= 16
1 <= people[i][j].length <= 16
people[i][j] consists of lowercase English letters.
All the strings of people[i] are unique.
Every skill in people[i] is a skill in req_skills.
It is guaranteed a sufficient team exists.

1128. Number of Equivalent Domino Pairs

Given a list of dominoes, dominoes[i] = [a, b] is equivalent to dominoes[j] = [c, d] if and only if either (a == c and b == d), or (a == d and b == c) - that is, one domino can be rotated to be equal to another domino.
Return the number of pairs (i, j) for which 0 <= i < j < dominoes.length, and dominoes[i] is equivalent to dominoes[j].

Example 1:

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

Example 2:

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

Constraints:

1 <= dominoes.length <= 4 * 104
dominoes[i].length == 2
1 <= dominoes[i][j] <= 9

1129. Shortest Path with Alternating Colors

Consider a directed graph, with nodes labelled 0, 1, ..., n-1. In this graph, each edge is either red or blue, and there c ould be self-edges or parallel edges.
Each [i, j] in red_edges denotes a red directed edge from node i to node j. Similarly, each [i, j] in blue_edges denote s a blue directed edge from node i to node j.
Return an array answer of length n, where each answer[X] is the length of the shortest path from node 0 to node X s

uch that the edge colors alternate along the path (or -1 if such a path doesn't exist).

Example 1:
Input: n = 3, red_edges = [[0,1],[1,2]], blue_edges = []
Output: [0,1,-1]
Example 2:
Input: n = 3, red_edges = [[0,1]], blue_edges = [[2,1]]
Output: [0,1,-1]
Example 3:
Input: n = 3, red_edges = [[1,0]], blue_edges = [[2,1]]
Output: [0,-1,-1]
Example 4:
Input: n = 3, red_edges = [[0,1]], blue_edges = [[1,2]]
Output: [0,1,2]
Example 5:
Input: n = 3, red_edges = [[0,1],[0,2]], blue_edges = [[1,0]]
Output: [0,1,1]

Constraints:

1 <= n <= 100
red_edges.length <= 400
blue_edges.length <= 400
red_edges[i].length == blue_edges[i].length == 2
0 <= red_edges[i][j], blue_edges[i][j] < n

1130. Minimum Cost Tree From Leaf Values

Given an array arr of positive integers, consider all binary trees such that:

Each node has either 0 or 2 children;
The values of arr correspond to the values of each leaf in an in-order traversal of the tree.
The value of each non-leaf node is equal to the product of the largest leaf value in its left and right subtree, respectiv ely.

Among all possible binary trees considered, return the smallest possible sum of the values of each non-leaf node. It i s guaranteed this sum fits into a 32-bit integer.
A node is a leaf if and only if it has zero children.

Example 1:

Input: arr = [6,2,4]
Output: 32
Explanation: There are two possible trees shown.
The first has a non-leaf node sum 36, and the second has non-leaf node sum 32.

Example 2:

Input: arr = [4,11]
Output: 44

Constraints:

2 <= arr.length <= 40
1 <= arr[i] <= 15
It is guaranteed that the answer fits into a 32-bit signed integer (i.e., it is less than 231).

1131. Maximum of Absolute Value Expression

Given two arrays of integers with equal lengths, return the maximum value of:
|arr1[i] - arr1[j]| + |arr2[i] - arr2[j]| + |i - j|
where the maximum is taken over all 0 <= i, j < arr1.length.

Example 1:

Input: arr1 = [1,2,3,4], arr2 = [-1,4,5,6]
Output: 13

Example 2:

Input: arr1 = [1,-2,-5,0,10], arr2 = [0,-2,-1,-7,-4]
Output: 20

Constraints:

2 <= arr1.length == arr2.length <= 40000
-10^6 <= arr1[i], arr2[i] <= 10^6

1137. N-th Tribonacci Number

The Tribonacci sequence Tn is defined as follows:
T0 = 0, T1 = 1, T2 = 1, and Tn+3 = Tn + Tn+1 + Tn+2 for n >= 0.
Given n, return the value of Tn.

Example 1:

Input: n = 4
Output: 4
Explanation:
T_3 = 0 + 1 + 1 = 2
T_4 = 1 + 1 + 2 = 4

Example 2:

Input: n = 25
Output: 1389537

Constraints:

0 <= n <= 37
The answer is guaranteed to fit within a 32-bit integer, ie. answer <= 2^31 - 1.

1138. Alphabet Board Path

On an alphabet board, we start at position (0, 0), corresponding to character board[0][0].
Here, board = ["abcde", "fghij", "klmno", "pqrst", "uvwxy", "z"], as shown in the diagram below.

We may make the following moves:

'U' moves our position up one row, if the position exists on the board;
'D' moves our position down one row, if the position exists on the board;
'L' moves our position left one column, if the position exists on the board;
'R' moves our position right one column, if the position exists on the board;
'!' adds the character board[r][c] at our current position (r, c) to the answer.

(Here, the only positions that exist on the board are positions with letters on them.)
Return a sequence of moves that makes our answer equal to target in the minimum number of moves. You may retu rn any path that does so.

Example 1:
Input: target = "leet"
Output: "DDR!UURRR!!DDD!"
Example 2:
Input: target = "code"
Output: "RR!DDRR!UUL!R!"

Constraints:

1 <= target.length <= 100
target consists only of English lowercase letters.

1139. Largest 1-Bordered Square

Given a 2D grid of 0s and 1s, return the number of elements in the largest square subgrid that has all 1s on its border , or 0 if such a subgrid doesn't exist in the grid.

Example 1:

Input: grid = [[1,1,1],[1,0,1],[1,1,1]]
Output: 9

Example 2:

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

Constraints:

1 <= grid.length <= 100
1 <= grid[0].length <= 100
grid[i][j] is 0 or 1

1140. Stone Game II

Alice and Bob continue their games with piles of stones. There are a number of piles arranged in a row, and each pil e has a positive integer number of stones piles[i]. The objective of the game is to end with the most stones.
Alice and Bob take turns, with Alice starting first. Initially, M = 1.
On each player's turn, that player can take all the stones in the first X remaining piles, where 1 <= X <= 2M. Then, we set M = max(M, X).
The game continues until all the stones have been taken.
Assuming Alice and Bob play optimally, return the maximum number of stones Alice can get.

Example 1:

Input: piles = [2,7,9,4,4]
Output: 10
Explanation: If Alice takes one pile at the beginning, Bob takes two piles, then Alice takes 2 piles again. Alice can g et 2 + 4 + 4 = 10 piles in total. If Alice takes two piles at the beginning, then Bob can take all three piles left. In this case, Alice get 2 + 7 = 9 piles in total. So we return 10 since it's larger.

Example 2:

Input: piles = [1,2,3,4,5,100]
Output: 104

Constraints:

1 <= piles.length <= 100
1 <= piles[i] <= 104

1143. Longest Common Subsequence

Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common sub sequence, return 0.
A subsequence of a string is a new string generated from the original string with some characters (can be none) delet ed without changing the relative order of the remaining characters.

For example, "ace" is a subsequence of "abcde".

A common subsequence of two strings is a subsequence that is common to both strings.

Example 1:

Input: text1 = "abcde", text2 = "ace"
Output: 3
Explanation: The longest common subsequence is "ace" and its length is 3.

Example 2:

Input: text1 = "abc", text2 = "abc"
Output: 3
Explanation: The longest common subsequence is "abc" and its length is 3.

Example 3:

Input: text1 = "abc", text2 = "def"
Output: 0
Explanation: There is no such common subsequence, so the result is 0.

Constraints:

1 <= text1.length, text2.length <= 1000
text1 and text2 consist of only lowercase English characters.

1144. Decrease Elements To Make Array Zigzag

Given an array nums of integers, a move consists of choosing any element and decreasing it by 1.
An array A is a zigzag array if either:

Every even-indexed element is greater than adjacent elements, ie. A[0] > A[1] < A[2] > A[3] < A[4] > ...
OR, every odd-indexed element is greater than adjacent elements, ie. A[0] < A[1] > A[2] < A[3] > A[4] < ...

Return the minimum number of moves to transform the given array nums into a zigzag array.

Example 1:

Input: nums = [1,2,3]
Output: 2
Explanation: We can decrease 2 to 0 or 3 to 1.

Example 2:

Input: nums = [9,6,1,6,2]
Output: 4

Constraints:

1 <= nums.length <= 1000
1 <= nums[i] <= 1000

1145. Binary Tree Coloring Game

Two players play a turn based game on a binary tree. We are given the root of this binary tree, and the number of no des n in the tree. n is odd, and each node has a distinct value from 1 to n.
Initially, the first player names a value x with 1 <= x <= n, and the second player names a value y with 1 <= y <= n a

nd y != x. The first player colors the node with value x red, and the second player colors the node with value y blue.
Then, the players take turns starting with the first player. In each turn, that player chooses a node of their color (red i f player 1, blue if player 2) and colors an uncolored neighbor of the chosen node (either the left child, right child, or parent of the chosen node.)
If (and only if) a player cannot choose such a node in this way, they must pass their turn. If both players pass their tu rn, the game ends, and the winner is the player that colored more nodes.
You are the second player. If it is possible to choose such a y to ensure you win the game, return true. If it is not poss ible, return false.

Example 1:

Input: root = [1,2,3,4,5,6,7,8,9,10,11], n = 11, x = 3
Output: true
Explanation: The second player can choose the node with value 2.

Example 2:

Input: root = [1,2,3], n = 3, x = 1
Output: false

Constraints:

The number of nodes in the tree is n.
1 <= x <= n <= 100
n is odd.
1 <= Node.val <= n
All the values of the tree are unique.

1146. Snapshot Array

Implement a SnapshotArray that supports the following interface:

SnapshotArray(int length) initializes an array-like data structure with the given length. Initially, each element equals 0.
void set(index, val) sets the element at the given index to be equal to val.
int snap() takes a snapshot of the array and returns the snap_id: the total number of times we called snap() minus 1.
int get(index, snap_id) returns the value at the given index, at the time we took the snapshot with the given snap_id

Example 1:

Input: ["SnapshotArray","set","snap","set","get"]
[[3],[0,5],[],[0,6],[0,0]]
Output: [null,null,0,null,5]

Explanation:
SnapshotArray snapshotArr = new SnapshotArray(3); // set the length to be 3
snapshotArr.set(0,5); // Set array[0] = 5
snapshotArr.snap(); // Take a snapshot, return snap_id = 0
snapshotArr.set(0,6);
snapshotArr.get(0,0); // Get the value of array[0] with snap_id = 0, return 5

Constraints:

1 <= length <= 50000
At most 50000 calls will be made to set, snap, and get.
0 <= index < length
0 <= snap_id < (the total number of times we call snap())
0 <= val <= 10^9

1147. Longest Chunked Palindrome Decomposition

You are given a string text. You should split it to k substrings (subtext1, subtext2, ..., subtextk) such that:

subtexti is a non-empty string.
The concatenation of all the substrings is equal to text (i.e., subtext1 + subtext2 + ... + subtextk == text).
subtexti == subtextk - i + 1 for all valid values of i (i.e., 1 <= i <= k).

Return the largest possible value of k.

Example 1:

Input: text = "ghiabcdefhelloadamhelloabcdefghi"
Output: 7
Explanation: We can split the string on "(ghi)(abcdef)(hello)(adam)(hello)(abcdef)(ghi)".

Example 2:

Input: text = "merchant"
Output: 1
Explanation: We can split the string on "(merchant)".

Example 3:

Input: text = "antaprezatepzapreanta"
Output: 11
Explanation: We can split the string on "(a)(nt)(a)(pre)(za)(tpe)(za)(pre)(a)(nt)(a)".

Example 4:

Input: text = "aaa"

Output: 3
Explanation: We can split the string on "(a)(a)(a)".

Constraints:

1 <= text.length <= 1000
text consists only of lowercase English characters.

1154. Day of the Year

Given a string date representing a Gregorian calendar date formatted as YYYY-MM-DD, return the day number of t he year.

Example 1:

Input: date = "2019-01-09"
Output: 9
Explanation: Given date is the 9th day of the year in 2019.

Example 2:

Input: date = "2019-02-10"
Output: 41

Example 3:

Input: date = "2003-03-01"
Output: 60

Example 4:

Input: date = "2004-03-01"
Output: 61

Constraints:

date.length == 10
date[4] == date[7] == '-', and all other date[i]'s are digits
date represents a calendar date between Jan 1st, 1900 and Dec 31, 2019.

1155. Number of Dice Rolls With Target Sum

You have d dice and each die has f faces numbered 1, 2, ..., f. You are given three integers d, f, and target.
Return the number of possible ways (out of fd total ways) modulo 109 + 7 to roll the dice so the sum of the face-up n umbers equals target.

Example 1:

Input: d = 1, f = 6, target = 3
Output: 1
Explanation:
You throw one die with 6 faces. There is only one way to get a sum of 3.

Example 2:

Input: d = 2, f = 6, target = 7
Output: 6
Explanation:
You throw two dice, each with 6 faces. There are 6 ways to get a sum of 7:
1+6, 2+5, 3+4, 4+3, 5+2, 6+1.

Example 3:

Input: d = 2, f = 5, target = 10
Output: 1
Explanation:
You throw two dice, each with 5 faces. There is only one way to get a sum of 10: 5+5.

Example 4:

Input: d = 1, f = 2, target = 3
Output: 0
Explanation:
You throw one die with 2 faces. There is no way to get a sum of 3.

Example 5:

Input: d = 30, f = 30, target = 500
Output: 222616187
Explanation:
The answer must be returned modulo 10^9 + 7.

Constraints:

1 <= d, f <= 30
1 <= target <= 1000

1156. Swap For Longest Repeated Character Substring

You are given a string text. You can swap two of the characters in the text.
Return the length of the longest substring with repeated characters.

Example 1:

Input: text = "ababa"
Output: 3
Explanation: We can swap the first 'b' with the last 'a', or the last 'b' with the first 'a'. Then, the longest repeated chara cter substring is "aaa", which its length is 3.

Example 2:

Input: text = "aaabaaa"
Output: 6
Explanation: Swap 'b' with the last 'a' (or the first 'a'), and we get longest repeated character substring "aaaaaa", whic h its length is 6.

Example 3:

Input: text = "aaabbaaa"
Output: 4

Example 4:

Input: text = "aaaaa"
Output: 5
Explanation: No need to swap, longest repeated character substring is "aaaaa", length is 5.

Example 5:

Input: text = "abcdef"
Output: 1

Constraints:

1 <= text.length <= 2 * 104
text consist of lowercase English characters only.

1157. Online Majority Element In Subarray

Design a data structure that efficiently finds the majority element of a given subarray.
The majority element of a subarray is an element that occurs threshold times or more in the subarray.
Implementing the MajorityChecker class:

MajorityChecker(int[] arr) Initializes the instance of the class with the given array arr.
int query(int left, int right, int threshold) returns the element in the subarray arr[left...right] that occurs at least thresh old times, or -1 if no such element exists.

Example 1:

Input
["MajorityChecker", "query", "query", "query"]
[[[1, 1, 2, 2, 1, 1]], [0, 5, 4], [0, 3, 3], [2, 3, 2]]
Output
[null, 1, -1, 2]

Explanation
MajorityChecker majorityChecker = new MajorityChecker([1, 1, 2, 2, 1, 1]);
majorityChecker.query(0, 5, 4); // return 1
majorityChecker.query(0, 3, 3); // return -1
majorityChecker.query(2, 3, 2); // return 2

Constraints:

1 <= arr.length <= 2 * 104
1 <= arr[i] <= 2 * 104
0 <= left <= right < arr.length
threshold <= right - left + 1
2 * threshold > right - left + 1
At most 104 calls will be made to query.

1160. Find Words That Can Be Formed by Characters

You are given an array of strings words and a string chars.
A string is good if it can be formed by characters from chars (each character can only be used once).
Return the sum of lengths of all good strings in words.

Example 1:

Input: words = ["cat","bt","hat","tree"], chars = "atach"
Output: 6

Explanation: The strings that can be formed are "cat" and "hat" so the answer is 3 + 3 = 6.

Example 2:

Input: words = ["hello","world","leetcode"], chars = "welldonehoneyr"
Output: 10
Explanation: The strings that can be formed are "hello" and "world" so the answer is 5 + 5 = 10.

Constraints:

1 <= words.length <= 1000
1 <= words[i].length, chars.length <= 100
words[i] and chars consist of lowercase English letters.

1161. Maximum Level Sum of a Binary Tree

Given the root of a binary tree, the level of its root is 1, the level of its children is 2, and so on.
Return the smallest level x such that the sum of all the values of nodes at level x is maximal.

Example 1:

Input: root = [1,7,0,7,-8,null,null]
Output: 2
Explanation:
Level 1 sum = 1.
Level 2 sum = 7 + 0 = 7.
Level 3 sum = 7 + -8 = -1.
So we return the level with the maximum sum which is level 2.

Example 2:

Input: root = [989,null,10250,98693,-89388,null,null,null,-32127]
Output: 2

Constraints:

The number of nodes in the tree is in the range [1, 104].
-105 <= Node.val <= 105

1162. As Far from Land as Possible

Given an n x n grid containing only values 0 and 1, where 0 represents water and 1 represents land, find a water cell such that its distance to the nearest land cell is maximized, and return the distance. If no land or water exists in the gr id, return -1.
The distance used in this problem is the Manhattan distance: the distance between two cells (x0, y0) and (x1, y1) is | x0 - x1| + |y0 - y1|.

Example 1:

Input: grid = [[1,0,1],[0,0,0],[1,0,1]]
Output: 2
Explanation: The cell (1, 1) is as far as possible from all the land with distance 2.

Example 2:

Input: grid = [[1,0,0],[0,0,0],[0,0,0]]
Output: 4
Explanation: The cell (2, 2) is as far as possible from all the land with distance 4.

Constraints:

n == grid.length
n == grid[i].length
1 <= n <= 100
grid[i][j] is 0 or 1

1163. Last Substring in Lexicographical Order

Given a string s, return the last substring of s in lexicographical order.

Example 1:

Input: s = "abab"
Output: "bab"
Explanation: The substrings are ["a", "ab", "aba", "abab", "b", "ba", "bab"]. The lexicographically maximum substrin g is "bab".

Example 2:

Input: s = "leetcode"
Output: "tcode"

Constraints:

1 <= s.length <= 4 * 105
s contains only lowercase English letters.

1169. Invalid Transactions

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 t he 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.

1170. Compare Strings by Frequency of the Smallest Character

Let the function f(s) be the frequency of the lexicographically smallest character in a non-empty string s. For exampl
e, if s = "dcce" then f(s) = 2 because the lexicographically smallest character is 'c', which has a frequency of 2.
You are given an array of strings words and another array of query strings queries. For each query queries[i], count t
he number of words in words such that f(queries[i]) < f(W) for each W in words.
Return an integer array answer, where each answer[i] is the answer to the ith query.

Example 1:

Input: queries = ["cbd"], words = ["zaaaz"]
Output: [1]
Explanation: On the first query we have f("cbd") = 1, f("zaaaz") = 3 so f("cbd") < f("zaaaz").

Example 2:

Input: queries = ["bbb","cc"], words = ["a","aa","aaa","aaaa"]
Output: [1,2]
Explanation: On the first query only f("bbb") < f("aaaa"). On the second query both f("aaa") and f("aaaa") are both >
f("cc").

Constraints:

1 <= queries.length <= 2000
1 <= words.length <= 2000
1 <= queries[i].length, words[i].length <= 10
queries[i][j], words[i][j] consist of lowercase English letters.

1171. Remove Zero Sum Consecutive Nodes from Linked List

Given the head of a linked list, we repeatedly delete consecutive sequences of nodes that sum to 0 until there are no s
uch sequences.
After doing so, return the head of the final linked list. You may return any such answer.

(Note that in the examples below, all sequences are serializations of ListNode objects.)
Example 1:

Input: head = [1,2,-3,3,1]
Output: [3,1]
Note: The answer [1,2,1] would also be accepted.

Example 2:

Input: head = [1,2,3,-3,4]
Output: [1,2,4]

Example 3:

Input: head = [1,2,3,-3,-2]
Output: [1]

Constraints:

The given linked list will contain between 1 and 1000 nodes.
Each node in the linked list has -1000 <= node.val <= 1000.

1172. Dinner Plate Stacks

You have an infinite number of stacks arranged in a row and numbered (left to right) from 0, each of the stacks has t he same maximum capacity.
Implement the DinnerPlates class:

DinnerPlates(int capacity) Initializes the object with the maximum capacity of the stacks capacity.
void push(int val) Pushes the given integer val into the leftmost stack with a size less than capacity.
int pop() Returns the value at the top of the rightmost non-empty stack and removes it from that stack, and returns -1 if all the stacks are empty.
int popAtStack(int index) Returns the value at the top of the stack with the given index index and removes it from th at stack or returns -1 if the stack with that given index is empty.

Example 1:

Input
["DinnerPlates", "push", "push", "push", "push", "push", "popAtStack", "push", "push", "popAtStack", "popAtStack ", "pop", "pop", "pop", "pop", "pop"]
[[2], [1], [2], [3], [4], [5], [0], [20], [21], [0], [2], [], [], [], [], []]
Output
[null, null, null, null, null, null, 2, null, null, 20, 21, 5, 4, 3, 1, -1]

Explanation:
DinnerPlates D = DinnerPlates(2); // Initialize with capacity = 2
D.push(1);

D.push(2);
D.push(3);
D.push(4);
D.push(5); // The stacks are now: 2 4
1 3 5
□ □ □
D.popAtStack(0); // Returns 2. The stacks are now: 4
1 3 5
□ □ □
D.push(20); // The stacks are now: 20 4
1 3 5
□ □ □
D.push(21); // The stacks are now: 20 4 21
1 3 5
□ □ □
D.popAtStack(0); // Returns 20. The stacks are now: 4 21
1 3 5
□ □ □
D.popAtStack(2); // Returns 21. The stacks are now: 4
1 3 5
□ □ □
D.pop() // Returns 5. The stacks are now: 4
1 3
□ □
D.pop() // Returns 4. The stacks are now: 1 3
□ □
D.pop() // Returns 3. The stacks are now: 1
□
D.pop() // Returns 1. There are no stacks.
D.pop() // Returns -1. There are still no stacks.

Constraints:

1 <= capacity <= 2 * 104
1 <= val <= 2 * 104
0 <= index <= 105
At most 2 * 105 calls will be made to push, pop, and popAtStack.

1175. Prime Arrangements

Return the number of permutations of 1 to n so that prime numbers are at prime indices (1-indexed.)
(Recall that an integer is prime if and only if it is greater than 1, and cannot be written as a product of two positive in tegers both smaller than it.)
Since the answer may be large, return the answer modulo 10^9 + 7.

Example 1:

Input: n = 5
Output: 12
Explanation: For example [1,2,5,4,3] is a valid permutation, but [5,2,3,4,1] is not because the prime number 5 is at in dex 1.

Example 2:

Input: n = 100
Output: 682289015

Constraints:

1 <= n <= 100

1177. Can Make Palindrome from Substring

You are given a string s and array queries where queries[i] = [lefti, righti, ki]. We may rearrange the substring s[lefti. ..righti] for each query and then choose up to ki of them to replace with any lowercase English letter.
If the substring is possible to be a palindrome string after the operations above, the result of the query is true. Otherw ise, the result is false.
Return a boolean array answer where answer[i] is the result of the ith query queries[i].
Note that each letter is counted individually for replacement, so if, for example s[lefti...righti] = "aaa", and ki = 2, we can only replace two of the letters. Also, note that no query modifies the initial string s.

Example :

Input: s = "abcda", queries = [[3,3,0],[1,2,0],[0,3,1],[0,3,2],[0,4,1]]
Output: [true,false,false,true,true]
Explanation:
queries[0]: substring = "d", is palidrome.
queries[1]: substring = "bc", is not palidrome.
queries[2]: substring = "abcd", is not palidrome after replacing only 1 character.
queries[3]: substring = "abcd", could be changed to "abba" which is palidrome. Also this can be changed to "baab" fi rst rearrange it "bacd" then replace "cd" with "ab".
queries[4]: substring = "abcda", could be changed to "abcba" which is palidrome.

Example 2:

Input: s = "lyb", queries = [[0,1,0],[2,2,1]]
Output: [false,true]

Constraints:

1 <= s.length, queries.length <= 105
0 <= lefti <= righti < s.length
0 <= ki <= s.length
s consists of lowercase English letters.

1178. Number of Valid Words for Each Puzzle

With respect to a given puzzle string, a word is valid if both the following conditions are satisfied:

word contains the first letter of puzzle.
For each letter in word, that letter is in puzzle.

For example, if the puzzle is "abcdefg", then valid words are "faced", "cabbage", and "baggage", while invalid words are "beefed" (does not include 'a') and "based" (includes 's' which is not in the puzzle).

Return an array answer, where answer[i] is the number of words in the given word list words that is valid with respect to the puzzle puzzles[i].

Example 1:

Input: words = ["aaaa","asas","able","ability","actt","actor","access"], puzzles = ["aboveyz","abrodyz","abslute","absoryz","actresz","gaswxyz"]
Output: [1,1,3,2,4,0]
Explanation:
1 valid word for "aboveyz" : "aaaa"
1 valid word for "abrodyz" : "aaaa"
3 valid words for "abslute" : "aaaa", "asas", "able"
2 valid words for "absoryz" : "aaaa", "asas"
4 valid words for "actresz" : "aaaa", "asas", "actt", "access"
There are no valid words for "gaswxyz" cause none of the words in the list contains letter 'g'.

Example 2:

Input: words = ["apple","pleas","please"], puzzles = ["aelwxyz","aelpxyz","aelpsxy","saelpxy","xaelpsy"]
Output: [0,1,3,2,0]

Constraints:

1 <= words.length <= 105
4 <= words[i].length <= 50
1 <= puzzles.length <= 104
puzzles[i].length == 7

words[i] and puzzles[i] consist of lowercase English letters.
Each puzzles[i] does not contain repeated characters.

1179. Reformat Department Table

Table: Department

```
+-------------+---------+
| Column Name | Type    |
+-------------+---------+
| id          | int     |
| revenue     | int     |
| month       | varchar |
+-------------+---------+
```

(id, month) is the primary key of this table.
The table has information about the revenue of each department per month.
The month has values in ["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"].

Write an SQL query to reformat the table such that there is a department id column and a revenue column for each m
onth.
Return the result table in any order.
The query result format is in the following example.

Example 1:

Input:
Department table:

```
+------+---------+-------+
| id   | revenue | month |
+------+---------+-------+
| 1    | 8000    | Jan   |
| 2    | 9000    | Jan   |
| 3    | 10000   | Feb   |
| 1    | 7000    | Feb   |
| 1    | 6000    | Mar   |
+------+---------+-------+
```

Output:

```
+------+-------------+-------------+-------------+-----+-------------+
| id   | Jan_Revenue | Feb_Revenue | Mar_Revenue | ... | Dec_Revenue |
+------+-------------+-------------+-------------+-----+-------------+
| 1    | 8000        | 7000        | 6000        | ... | null        |
| 2    | 9000        | null        | null        | ... | null        |
| 3    | null        | 10000       | null        | ... | null        |
+------+-------------+-------------+-------------+-----+-------------+
```

Explanation: The revenue from Apr to Dec is null.

Note that the result table has 13 columns (1 for the department id + 12 for the months).

1184. Distance Between Bus Stops

A bus has n stops numbered from 0 to n - 1 that form a circle. We know the distance between all pairs of neighborin g stops where distance[i] is the distance between the stops number i and (i + 1) % n.
The bus goes along both directions i.e. clockwise and counterclockwise.
Return the shortest distance between the given start and destination stops.

Example 1:

Input: distance = [1,2,3,4], start = 0, destination = 1
Output: 1
Explanation: Distance between 0 and 1 is 1 or 9, minimum is 1.

Example 2:

Input: distance = [1,2,3,4], start = 0, destination = 2
Output: 3
Explanation: Distance between 0 and 2 is 3 or 7, minimum is 3.

Example 3:

Input: distance = [1,2,3,4], start = 0, destination = 3
Output: 4
Explanation: Distance between 0 and 3 is 6 or 4, minimum is 4.

Constraints:

1 <= n <= 10^4
distance.length == n
0 <= start, destination < n
0 <= distance[i] <= 10^4

1185. Day of the Week

Given a date, return the corresponding day of the week for that date.
The input is given as three integers representing the day, month and year respectively.
Return the answer as one of the following values {"Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Fri
day", "Saturday"}.

Example 1:

Input: day = 31, month = 8, year = 2019
Output: "Saturday"

Example 2:

Input: day = 18, month = 7, year = 1999
Output: "Sunday"

Example 3:

Input: day = 15, month = 8, year = 1993
Output: "Sunday"

Constraints:

The given dates are valid dates between the years 1971 and 2100.

1186. Maximum Subarray Sum with One Deletion

Given an array of integers, return the maximum sum for a non-empty subarray (contiguous elements) with at most o
ne element deletion. In other words, you want to choose a subarray and optionally delete one element from it so that
there is still at least one element left and the sum of the remaining elements is maximum possible.
Note that the subarray needs to be non-empty after deleting one element.

Example 1:

Input: arr = [1,-2,0,3]
Output: 4
Explanation: Because we can choose [1, -2, 0, 3] and drop -2, thus the subarray [1, 0, 3] becomes the maximum valu
e.
Example 2:

Input: arr = [1,-2,-2,3]
Output: 3
Explanation: We just choose [3] and it's the maximum sum.

Example 3:

Input: arr = [-1,-1,-1,-1]
Output: -1
Explanation: The final subarray needs to be non-empty. You can't choose [-1] and delete -1 from it, then get an empt
y subarray to make the sum equals to 0.

Constraints:

1 <= arr.length <= 105
-104 <= arr[i] <= 104

1187. Make Array Strictly Increasing

Given two integer arrays arr1 and arr2, return the minimum number of operations (possibly zero) needed to make arr
1 strictly increasing.
In one operation, you can choose two indices 0 <= i < arr1.length and 0 <= j < arr2.length and do the assignment arr
1[i] = arr2[j].
If there is no way to make arr1 strictly increasing, return -1.

Example 1:

Input: arr1 = [1,5,3,6,7], arr2 = [1,3,2,4]
Output: 1
Explanation: Replace 5 with 2, then arr1 = [1, 2, 3, 6, 7].

Example 2:

Input: arr1 = [1,5,3,6,7], arr2 = [4,3,1]
Output: 2
Explanation: Replace 5 with 3 and then replace 3 with 4. arr1 = [1, 3, 4, 6, 7].

Example 3:

Input: arr1 = [1,5,3,6,7], arr2 = [1,6,3,3]
Output: -1
Explanation: You can't make arr1 strictly increasing.

Constraints:

1 <= arr1.length, arr2.length <= 2000
0 <= arr1[i], arr2[i] <= 10^9

1189. Maximum Number of Balloons

Given a string text, you want to use the characters of text to form as many instances of the word "balloon" as possibl
e.
You can use each character in text at most once. Return the maximum number of instances that can be formed.

Example 1:

Input: text = "nlaebolko"
Output: 1

Example 2:

Input: text = "loonbalxballpoon"
Output: 2

Example 3:

Input: text = "leetcode"
Output: 0

Constraints:

1 <= text.length <= 104
text consists of lower case English letters only.

1190. Reverse Substrings Between Each Pair of Parentheses

You are given a string s that consists of lower case English letters and brackets.
Reverse the strings in each pair of matching parentheses, starting from the innermost one.
Your result should not contain any brackets.

Example 1:

Input: s = "(abcd)"
Output: "dcba"

Example 2:

Input: s = "(u(love)i)"
Output: "iloveu"
Explanation: The substring "love" is reversed first, then the whole string is reversed.

Example 3:

Input: s = "(ed(et(oc))el)"
Output: "leetcode"
Explanation: First, we reverse the substring "oc", then "etco", and finally, the whole string.

Example 4:

Input: s = "a(bcdefghijkl(mno)p)q"
Output: "apmnolkjihgfedcbq"

Constraints:

0 <= s.length <= 2000
s only contains lower case English characters and parentheses.
It's guaranteed that all parentheses are balanced.

1191. K-Concatenation Maximum Sum

Given an integer array arr and an integer k, modify the array by repeating it k times.
For example, if arr = [1, 2] and k = 3 then the modified array will be [1, 2, 1, 2, 1, 2].
Return the maximum sub-array sum in the modified array. Note that the length of the sub-array can be 0 and its sum in that case is 0.
As the answer can be very large, return the answer modulo 109 + 7.

Example 1:

Input: arr = [1,2], k = 3
Output: 9

Example 2:

Input: arr = [1,-2,1], k = 5
Output: 2

Example 3:

Input: arr = [-1,-2], k = 7
Output: 0

Constraints:

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

1192. Critical Connections in a Network

There are n servers numbered from 0 to n - 1 connected by undirected server-to-server connections forming a networ
k where connections[i] = [ai, bi] represents a connection between servers ai and bi. Any server can reach other server
s directly or indirectly through the network.
A critical connection is a connection that, if removed, will make some servers unable to reach some other server.
Return all critical connections in the network in any order.

Example 1:

Input: n = 4, connections = [[0,1],[1,2],[2,0],[1,3]]
Output: [[1,3]]
Explanation: [[3,1]] is also accepted.

Example 2:

Input: n = 2, connections = [[0,1]]
Output: [[0,1]]

Constraints:

2 <= n <= 105
n - 1 <= connections.length <= 105
0 <= ai, bi <= n - 1
ai != bi
There are no repeated connections.

1195. Fizz Buzz Multithreaded

You have the four functions:

printFizz that prints the word "Fizz" to the console,
printBuzz that prints the word "Buzz" to the console,
printFizzBuzz that prints the word "FizzBuzz" to the console, and
printNumber that prints a given integer to the console.

You are given an instance of the class FizzBuzz that has four functions: fizz, buzz, fizzbuzz and number. The same i nstance of FizzBuzz will be passed to four different threads:

Thread A: calls fizz() that should output the word "Fizz".
Thread B: calls buzz() that should output the word "Buzz".
Thread C: calls fizzbuzz() that should output the word "FizzBuzz".
Thread D: calls number() that should only output the integers.

Modify the given class to output the series [1, 2, "Fizz", 4, "Buzz", ...] where the ith token (1-indexed) of the series i s:

"FizzBuzz" if i is divisible by 3 and 5,
"Fizz" if i is divisible by 3 and not 5,
"Buzz" if i is divisible by 5 and not 3, or
i if i is not divisible by 3 or 5.

Implement the FizzBuzz class:

FizzBuzz(int n) Initializes the object with the number n that represents the length of the sequence that should be prin ted.
void fizz(printFizz) Calls printFizz to output "Fizz".
void buzz(printBuzz) Calls printBuzz to output "Buzz".
void fizzbuzz(printFizzBuzz) Calls printFizzBuzz to output "FizzBuzz".
void number(printNumber) Calls printnumber to output the numbers.

Example 1:
Input: n = 15
Output: [1,2,"fizz",4,"buzz","fizz",7,8,"fizz","buzz",11,"fizz",13,14,"fizzbuzz"]
Example 2:
Input: n = 5
Output: [1,2,"fizz",4,"buzz"]

Constraints:

1 <= n <= 50

1200. Minimum Absolute Difference

Given an array of distinct integers arr, find all pairs of elements with the minimum absolute difference of any two el ements.
Return a list of pairs in ascending order(with respect to pairs), each pair [a, b] follows

a, b are from arr
a < b
b - a equals to the minimum absolute difference of any two elements in arr

Example 1:

Input: arr = [4,2,1,3]
Output: [[1,2],[2,3],[3,4]]
Explanation: The minimum absolute difference is 1. List all pairs with difference equal to 1 in ascending order.
Example 2:

Input: arr = [1,3,6,10,15]
Output: [[1,3]]

Example 3:

Input: arr = [3,8,-10,23,19,-4,-14,27]
Output: [[-14,-10],[19,23],[23,27]]

Constraints:

2 <= arr.length <= 10^5
-10^6 <= arr[i] <= 10^6

1201. Ugly Number III

An ugly number is a positive integer that is divisible by a, b, or c.
Given four integers n, a, b, and c, return the nth ugly number.

Example 1:

Input: n = 3, a = 2, b = 3, c = 5
Output: 4
Explanation: The ugly numbers are 2, 3, 4, 5, 6, 8, 9, 10... The 3rd is 4.

Example 2:

Input: n = 4, a = 2, b = 3, c = 4

Output: 6
Explanation: The ugly numbers are 2, 3, 4, 6, 8, 9, 10, 12... The 4th is 6.

Example 3:

Input: n = 5, a = 2, b = 11, c = 13
Output: 10
Explanation: The ugly numbers are 2, 4, 6, 8, 10, 11, 12, 13... The 5th is 10.

Example 4:

Input: n = 1000000000, a = 2, b = 217983653, c = 336916467
Output: 1999999984

Constraints:

1 <= n, a, b, c <= 109
1 <= a * b * c <= 1018
It is guaranteed that the result will be in range [1, 2 * 109].

1202. Smallest String With Swaps

You are given a string s, and an array of pairs of indices in the string pairs where pairs[i] = [a, b] indicates 2 indices(0-indexed) of the string.
You can swap the characters at any pair of indices in the given pairs any number of times.
Return the lexicographically smallest string that s can be changed to after using the swaps.

Example 1:

Input: s = "dcab", pairs = [[0,3],[1,2]]
Output: "bacd"
Explaination:
Swap s[0] and s[3], s = "bcad"
Swap s[1] and s[2], s = "bacd"

Example 2:

Input: s = "dcab", pairs = [[0,3],[1,2],[0,2]]
Output: "abcd"
Explaination:
Swap s[0] and s[3], s = "bcad"
Swap s[0] and s[2], s = "acbd"
Swap s[1] and s[2], s = "abcd"
Example 3:

Input: s = "cba", pairs = [[0,1],[1,2]]
Output: "abc"
Explaination:
Swap s[0] and s[1], s = "bca"
Swap s[1] and s[2], s = "bac"
Swap s[0] and s[1], s = "abc"

Constraints:

1 <= s.length <= 10^5
0 <= pairs.length <= 10^5
0 <= pairs[i][0], pairs[i][1] < s.length
s only contains lower case English letters.

1203. Sort Items by Groups Respecting Dependencies

There are n items each belonging to zero or one of m groups where group[i] is the group that the i-th item belongs to and it's equal to -1 if the i-th item belongs to no group. The items and the groups are zero indexed. A group can have no item belonging to it.
Return a sorted list of the items such that:

The items that belong to the same group are next to each other in the sorted list.
There are some relations between these items where beforeItems[i] is a list containing all the items that should come before the i-th item in the sorted array (to the left of the i-th item).

Return any solution if there is more than one solution and return an empty list if there is no solution.

Example 1:

Input: n = 8, m = 2, group = [-1,-1,1,0,0,1,0,-1], beforeItems = [[],[6],[5],[6],[3,6],[],[],[]]
Output: [6,3,4,1,5,2,0,7]

Example 2:

Input: n = 8, m = 2, group = [-1,-1,1,0,0,1,0,-1], beforeItems = [[],[6],[5],[6],[3],[],[4],[]]
Output: []
Explanation: This is the same as example 1 except that 4 needs to be before 6 in the sorted list.

Constraints:

1 <= m <= n <= 3 * 104
group.length == beforeItems.length == n
-1 <= group[i] <= m - 1

0 <= beforeItems[i].length <= n - 1
0 <= beforeItems[i][j] <= n - 1
i != beforeItems[i][j]
beforeItems[i] does not contain duplicates elements.

1206. Design Skiplist

Design a Skiplist without using any built-in libraries.
A skiplist is a data structure that takes O(log(n)) time to add, erase and search. Comparing with treap and red-black t ree which has the same function and performance, the code length of Skiplist can be comparatively short and the ide a behind Skiplists is just simple linked lists.
For example, we have a Skiplist containing [30,40,50,60,70,90] and we want to add 80 and 45 into it. The Skiplist w orks this way:

Artyom Kalinin [CC BY-SA 3.0], via Wikimedia Commons
You can see there are many layers in the Skiplist. Each layer is a sorted linked list. With the help of the top layers, a dd, erase and search can be faster than O(n). It can be proven that the average time complexity for each operation is O(log(n)) and space complexity is O(n).
See more about Skiplist: https://en.wikipedia.org/wiki/Skip_list
Implement the Skiplist class:

Skiplist() Initializes the object of the skiplist.
bool search(int target) Returns true if the integer target exists in the Skiplist or false otherwise.
void add(int num) Inserts the value num into the SkipList.
bool erase(int num) Removes the value num from the Skiplist and returns true. If num does not exist in the Skiplist, do nothing and return false. If there exist multiple num values, removing any one of them is fine.

Note that duplicates may exist in the Skiplist, your code needs to handle this situation.

Example 1:

Input
["Skiplist", "add", "add", "add", "search", "add", "search", "erase", "erase", "search"]
[[], [1], [2], [3], [0], [4], [1], [0], [1], [1]]
Output
[null, null, null, null, false, null, true, false, true, false]

Explanation
Skiplist skiplist = new Skiplist();
skiplist.add(1);
skiplist.add(2);
skiplist.add(3);
skiplist.search(0); // return False
skiplist.add(4);
skiplist.search(1); // return True
skiplist.erase(0); // return False, 0 is not in skiplist.

skiplist.erase(1); // return True
skiplist.search(1); // return False, 1 has already been erased.

Constraints:

0 <= num, target <= 2 * 104
At most 5 * 104 calls will be made to search, add, and erase.

1207. Unique Number of Occurrences

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

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 occurrence s.
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

1208. Get Equal Substrings Within Budget

You are given two strings s and t of the same length. You want to change s to t. Changing the i-th character of s to i-t

h character of t costs |s[i] - t[i]| that is, the absolute difference between the ASCII values of the characters.
You are also given an integer maxCost.
Return the maximum length of a substring of s that can be changed to be the same as the corresponding substring of twith a cost less than or equal to maxCost.
If there is no substring from s that can be changed to its corresponding substring from t, return 0.

Example 1:

Input: s = "abcd", t = "bcdf", maxCost = 3
Output: 3
Explanation: "abc" of s can change to "bcd". That costs 3, so the maximum length is 3.
Example 2:

Input: s = "abcd", t = "cdef", maxCost = 3
Output: 1
Explanation: Each character in s costs 2 to change to charactor in t, so the maximum length is 1.

Example 3:

Input: s = "abcd", t = "acde", maxCost = 0
Output: 1
Explanation: You can't make any change, so the maximum length is 1.

Constraints:

1 <= s.length, t.length <= 10^5
0 <= maxCost <= 10^6
s and t only contain lower case English letters.

1209. Remove All Adjacent Duplicates in String II

You are given a string s and an integer k, a k duplicate removal consists of choosing k adjacent and equal letters fro m s and removing them, causing the left and the right side of the deleted substring to concatenate together.
We repeatedly make k duplicate removals on s until we no longer can.
Return the final string after all such duplicate removals have been made. It is guaranteed that the answer is unique.

Example 1:

Input: s = "abcd", k = 2
Output: "abcd"
Explanation: There's nothing to delete.
Example 2:

Input: s = "deeedbbcccbdaa", k = 3
Output: "aa"

Explanation:
First delete "eee" and "ccc", get "ddbbbdaa"
Then delete "bbb", get "dddaa"
Finally delete "ddd", get "aa"
Example 3:

Input: s = "pbbcggttciiippooaais", k = 2
Output: "ps"

Constraints:

1 <= s.length <= 105
2 <= k <= 104
s only contains lower case English letters.

1210. Minimum Moves to Reach Target with Rotations

In an n*n grid, there is a snake that spans 2 cells and starts moving from the top left corner at (0, 0) and (0, 1). The g rid has empty cells represented by zeros and blocked cells represented by ones. The snake wants to reach the lower ri ght corner at (n-1, n-2) and (n-1, n-1).
In one move the snake can:

Move one cell to the right if there are no blocked cells there. This move keeps the horizontal/vertical position of the snake as it is.
Move down one cell if there are no blocked cells there. This move keeps the horizontal/vertical position of the snake as it is.
Rotate clockwise if it's in a horizontal position and the two cells under it are both empty. In that case the snake move s from (r, c) and (r, c+1) to (r, c) and (r+1, c).

Rotate counterclockwise if it's in a vertical position and the two cells to its right are both empty. In that case the snak e moves from (r, c) and (r+1, c) to (r, c) and (r, c+1).

Return the minimum number of moves to reach the target.
If there is no way to reach the target, return -1.

Example 1:

Input: grid = [[0,0,0,0,0,1],
[1,1,0,0,1,0],
[0,0,0,0,1,1],
[0,0,1,0,1,0],
[0,1,1,0,0,0],
[0,1,1,0,0,0]]

Output: 11
Explanation:
One possible solution is [right, right, rotate clockwise, right, down, down, down, down, rotate counterclockwise, rig
ht, down].

Example 2:

Input: grid = [[0,0,1,1,1,1],
[0,0,0,0,1,1],
[1,1,0,0,0,1],
[1,1,1,0,0,1],
[1,1,1,0,0,1],
[1,1,1,0,0,0]]
Output: 9

Constraints:

2 <= n <= 100
0 <= grid[i][j] <= 1
It is guaranteed that the snake starts at empty cells.

1217. Minimum Cost to Move Chips to The Same Position

We have n chips, where the position of the ith chip is position[i].
We need to move all the chips to the same position. In one step, we can change the position of the ith chip from posit
ion[i] to:

position[i] + 2 or position[i] - 2 with cost = 0.
position[i] + 1 or position[i] - 1 with cost = 1.

Return the minimum cost needed to move all the chips to the same position.

Example 1:

Input: position = [1,2,3]
Output: 1
Explanation: First step: Move the chip at position 3 to position 1 with cost = 0.
Second step: Move the chip at position 2 to position 1 with cost = 1.
Total cost is 1.

Example 2:

Input: position = [2,2,2,3,3]

Output: 2
Explanation: We can move the two chips at position 3 to position 2. Each move has cost = 1. The total cost = 2.

Example 3:

Input: position = [1,1000000000]
Output: 1

Constraints:

1 <= position.length <= 100
1 <= position[i] <= 10^9

1218. Longest Arithmetic Subsequence of Given Difference

Given an integer array arr and an integer difference, return the length of the longest subsequence in arr which is an ar
ithmetic sequence such that the difference between adjacent elements in the subsequence equals difference.
A subsequence is a sequence that can be derived from arr by deleting some or no elements without changing the orde
r of the remaining elements.

Example 1:

Input: arr = [1,2,3,4], difference = 1
Output: 4
Explanation: The longest arithmetic subsequence is [1,2,3,4].
Example 2:

Input: arr = [1,3,5,7], difference = 1
Output: 1
Explanation: The longest arithmetic subsequence is any single element.

Example 3:

Input: arr = [1,5,7,8,5,3,4,2,1], difference = -2
Output: 4
Explanation: The longest arithmetic subsequence is [7,5,3,1].

Constraints:

1 <= arr.length <= 105
-104 <= arr[i], difference <= 104

1219. Path with Maximum Gold

In a gold mine grid of size m x n, each cell in this mine has an integer representing the amount of gold in that cell, 0 if it is empty.
Return the maximum amount of gold you can collect under the conditions:

Every time you are located in a cell you will collect all the gold in that cell.
From your position, you can walk one step to the left, right, up, or down.
You can't visit the same cell more than once.
Never visit a cell with 0 gold.
You can start and stop collecting gold from any position in the grid that has some gold.

Example 1:

Input: grid = [[0,6,0],[5,8,7],[0,9,0]]
Output: 24
Explanation:
[[0,6,0],
 [5,8,7],
 [0,9,0]]
Path to get the maximum gold, 9 -> 8 -> 7.

Example 2:

Input: grid = [[1,0,7],[2,0,6],[3,4,5],[0,3,0],[9,0,20]]
Output: 28
Explanation:
[[1,0,7],
 [2,0,6],
 [3,4,5],
 [0,3,0],
 [9,0,20]]
Path to get the maximum gold, 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7.

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 15
0 <= grid[i][j] <= 100
There are at most 25 cells containing gold.

1220. Count Vowels Permutation

Given an integer n, your task is to count how many strings of length n can be formed under the following rules:

Each character is a lower case vowel ('a', 'e', 'i', 'o', 'u')
Each vowel 'a' may only be followed by an 'e'.
Each vowel 'e' may only be followed by an 'a' or an 'i'.
Each vowel 'i' may not be followed by another 'i'.
Each vowel 'o' may only be followed by an 'i' or a 'u'.
Each vowel 'u' may only be followed by an 'a'.

Since the answer may be too large, return it modulo 10^9 + 7.

Example 1:

Input: n = 1
Output: 5
Explanation: All possible strings are: "a", "e", "i" , "o" and "u".

Example 2:

Input: n = 2
Output: 10
Explanation: All possible strings are: "ae", "ea", "ei", "ia", "ie", "io", "iu", "oi", "ou" and "ua".

Example 3:

Input: n = 5
Output: 68

Constraints:

1 <= n <= 2 * 10^4

1221. Split a String in Balanced Strings

Balanced strings are those that have an equal quantity of 'L' and 'R' characters.
Given a balanced string s, split it in the maximum amount of balanced strings.
Return the maximum amount of split balanced strings.

Example 1:

Input: s = "RLRRLLRLRL"
Output: 4
Explanation: s can be split into "RL", "RRLL", "RL", "RL", each substring contains same number of 'L' and 'R'.

Example 2:

Input: s = "RLLLLRRRLR"
Output: 3
Explanation: s can be split into "RL", "LLLRRR", "LR", each substring contains same number of 'L' and 'R'.

Example 3:

Input: s = "LLLLRRRR"
Output: 1
Explanation: s can be split into "LLLLRRRR".

Example 4:

Input: s = "RLRRRLLRLL"
Output: 2
Explanation: s can be split into "RL", "RRRLLRLL", since each substring contains an equal number of 'L' and 'R'

Constraints:

1 <= s.length <= 1000
s[i] is either 'L' or 'R'.
s is a balanced string.

1222. Queens That Can Attack the King

On an 8x8 chessboard, there can be multiple Black Queens and one White King.
Given an array of integer coordinates queens that represents the positions of the Black Queens, and a pair of coordin ates king that represent the position of the White King, return the coordinates of all the queens (in any order) that can attack the King.

Example 1:

Input: queens = [[0,1],[1,0],[4,0],[0,4],[3,3],[2,4]], king = [0,0]
Output: [[0,1],[1,0],[3,3]]
Explanation:
The queen at [0,1] can attack the king cause they're in the same row.
The queen at [1,0] can attack the king cause they're in the same column.
The queen at [3,3] can attack the king cause they're in the same diagnal.
The queen at [0,4] can't attack the king cause it's blocked by the queen at [0,1].

The queen at [4,0] can't attack the king cause it's blocked by the queen at [1,0].
The queen at [2,4] can't attack the king cause it's not in the same row/column/diagnal as the king.

Example 2:

Input: queens = [[0,0],[1,1],[2,2],[3,4],[3,5],[4,4],[4,5]], king = [3,3]
Output: [[2,2],[3,4],[4,4]]

Example 3:

Input: queens = [[5,6],[7,7],[2,1],[0,7],[1,6],[5,1],[3,7],[0,3],[4,0],[1,2],[6,3],[5,0],[0,4],[2,2],[1,1],[6,4],[5,4],[0,0],[2,6],[4,5],[5,2],[1,4],[7,5],[2,3],[0,5],[4,2],[1,0],[2,7],[0,1],[4,6],[6,1],[0,6],[4,3],[1,7]], king = [3,4]
Output: [[2,3],[1,4],[1,6],[3,7],[4,3],[5,4],[4,5]]

Constraints:

1 <= queens.length <= 63
queens[i].length == 2
0 <= queens[i][j] < 8
king.length == 2
0 <= king[0], king[1] < 8
At most one piece is allowed in a cell.

1223. Dice Roll Simulation

A die simulator generates a random number from 1 to 6 for each roll. You introduced a constraint to the generator such that it cannot roll the number i more than rollMax[i] (1-indexed) consecutive times.
Given an array of integers rollMax and an integer n, return the number of distinct sequences that can be obtained with exact n rolls. Since the answer may be too large, return it modulo 109 + 7.
Two sequences are considered different if at least one element differs from each other.

Example 1:

Input: n = 2, rollMax = [1,1,2,2,2,3]
Output: 34
Explanation: There will be 2 rolls of die, if there are no constraints on the die, there are 6 * 6 = 36 possible combinations. In this case, looking at rollMax array, the numbers 1 and 2 appear at most once consecutively, therefore sequences (1,1) and (2,2) cannot occur, so the final answer is 36-2 = 34.

Example 2:

Input: n = 2, rollMax = [1,1,1,1,1,1]
Output: 30

Example 3:

Input: n = 3, rollMax = [1,1,1,2,2,3]
Output: 181

Constraints:

1 <= n <= 5000
rollMax.length == 6
1 <= rollMax[i] <= 15

1224. Maximum Equal Frequency

Given an array nums of positive integers, return the longest possible length of an array prefix of nums, such that it is possible to remove exactly one element from this prefix so that every number that has appeared in it will have the sa me number of occurrences.
If after removing one element there are no remaining elements, it's still considered that every appeared number has t he same number of ocurrences (0).

Example 1:

Input: nums = [2,2,1,1,5,3,3,5]
Output: 7
Explanation: For the subarray [2,2,1,1,5,3,3] of length 7, if we remove nums[4]=5, we will get [2,2,1,1,3,3], so that e ach number will appear exactly twice.

Example 2:

Input: nums = [1,1,1,2,2,2,3,3,3,4,4,4,5]
Output: 13

Example 3:

Input: nums = [1,1,1,2,2,2]
Output: 5

Example 4:

Input: nums = [10,2,8,9,3,8,1,5,2,3,7,6]
Output: 8

Constraints:

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

1226. The Dining Philosophers

Five silent philosophers sit at a round table with bowls of spaghetti. Forks are placed between each pair of adjacent p
hilosophers.
Each philosopher must alternately think and eat. However, a philosopher can only eat spaghetti when they have both
left and right forks. Each fork can be held by only one philosopher and so a philosopher can use the fork only if it is
not being used by another philosopher. After an individual philosopher finishes eating, they need to put down both f
orks so that the forks become available to others. A philosopher can take the fork on their right or the one on their lef
t as they become available, but cannot start eating before getting both forks.
Eating is not limited by the remaining amounts of spaghetti or stomach space; an infinite supply and an infinite dema
nd are assumed.
Design a discipline of behaviour (a concurrent algorithm) such that no philosopher will starve; i.e., each can forever
continue to alternate between eating and thinking, assuming that no philosopher can know when others may want to
eat or think.

The problem statement and the image above are taken from wikipedia.org

The philosophers' ids are numbered from 0 to 4 in a clockwise order. Implement the function void wantsToEat(philo
sopher, pickLeftFork, pickRightFork, eat, putLeftFork, putRightFork) where:

philosopher is the id of the philosopher who wants to eat.
pickLeftFork and pickRightFork are functions you can call to pick the corresponding forks of that philosopher.
eat is a function you can call to let the philosopher eat once he has picked both forks.
putLeftFork and putRightFork are functions you can call to put down the corresponding forks of that philosopher.
The philosophers are assumed to be thinking as long as they are not asking to eat (the function is not being called wit
h their number).

Five threads, each representing a philosopher, will simultaneously use one object of your class to simulate the proces
s. The function may be called for the same philosopher more than once, even before the last call ends.

Example 1:

Input: n = 1
Output: [[4,2,1],[4,1,1],[0,1,1],[2,2,1],[2,1,1],[2,0,3],[2,1,2],[2,2,2],[4,0,3],[4,1,2],[0,2,1],[4,2,2],[3,2,1],[3,1,1],[0,0,3
],[0,1,2],[0,2,2],[1,2,1],[1,1,1],[3,0,3],[3,1,2],[3,2,2],[1,0,3],[1,1,2],[1,2,2]]
Explanation:
n is the number of times each philosopher will call the function.
The output array describes the calls you made to the functions controlling the forks and the eat function, its format is
:
output[i] = [a, b, c] (three integers)
- a is the id of a philosopher.
- b specifies the fork: {1 : left, 2 : right}.
- c specifies the operation: {1 : pick, 2 : put, 3 : eat}.

Constraints:

1 <= n <= 60

1227. Airplane Seat Assignment Probability

n passengers board an airplane with exactly n seats. The first passenger has lost the ticket and picks a seat randomly. But after that, the rest of passengers will:

Take their own seat if it is still available,
Pick other seats randomly when they find their seat occupied

What is the probability that the n-th person can get his own seat?

Example 1:

Input: n = 1
Output: 1.00000
Explanation: The first person can only get the first seat.
Example 2:

Input: n = 2
Output: 0.50000
Explanation: The second person has a probability of 0.5 to get the second seat (when first person gets the first seat).

Constraints:

1 <= n <= 10^5

1232. Check If It Is a Straight Line

You are given an array coordinates, coordinates[i] = [x, y], where [x, y] represents the coordinate of a point. Check i f these points make a straight line in the XY plane.

Example 1:

Input: coordinates = [[1,2],[2,3],[3,4],[4,5],[5,6],[6,7]]
Output: true

Example 2:

Input: coordinates = [[1,1],[2,2],[3,4],[4,5],[5,6],[7,7]]
Output: false

Constraints:

2 <= coordinates.length <= 1000
coordinates[i].length == 2
-10^4 <= coordinates[i][0], coordinates[i][1] <= 10^4
coordinates contains no duplicate point.

1233. Remove Sub-Folders from the Filesystem

Given a list of folders folder, return the folders after removing all sub-folders in those folders. You may return the an swer in any order.
If a folder[i] is located within another folder[j], it is called a sub-folder of it.
The format of a path is one or more concatenated strings of the form: '/' followed by one or more lowercase English l etters.

For example, "/leetcode" and "/leetcode/problems" are valid paths while an empty string and "/" are not.

Example 1:

Input: folder = ["/a","/a/b","/c/d","/c/d/e","/c/f"]
Output: ["/a","/c/d","/c/f"]
Explanation: Folders "/a/b/" is a subfolder of "/a" and "/c/d/e" is inside of folder "/c/d" in our filesystem.

Example 2:

Input: folder = ["/a","/a/b/c","/a/b/d"]
Output: ["/a"]
Explanation: Folders "/a/b/c" and "/a/b/d/" will be removed because they are subfolders of "/a".

Example 3:

Input: folder = ["/a/b/c","/a/b/ca","/a/b/d"]
Output: ["/a/b/c","/a/b/ca","/a/b/d"]

Constraints:

1 <= folder.length <= 4 * 104
2 <= folder[i].length <= 100
folder[i] contains only lowercase letters and '/'.
folder[i] always starts with the character '/'.
Each folder name is unique.

1234. Replace the Substring for Balanced String

You are given a string containing only 4 kinds of characters 'Q', 'W', 'E' and 'R'.
A string is said to be balanced if each of its characters appears n/4 times where n is the length of the string.
Return the minimum length of the substring that can be replaced with any other string of the same length to make the original string s balanced.
Return 0 if the string is already balanced.

Example 1:

Input: s = "QWER"
Output: 0
Explanation: s is already balanced.
Example 2:

Input: s = "QQWE"
Output: 1
Explanation: We need to replace a 'Q' to 'R', so that "RQWE" (or "QRWE") is balanced.

Example 3:

Input: s = "QQQW"
Output: 2
Explanation: We can replace the first "QQ" to "ER".

Example 4:

Input: s = "QQQQ"
Output: 3
Explanation: We can replace the last 3 'Q' to make s = "QWER".

Constraints:

1 <= s.length <= 10^5
s.length is a multiple of 4
s contains only 'Q', 'W', 'E' and 'R'.

1235. Maximum Profit in Job Scheduling

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

1237. Find Positive Integer Solution for a Given Equation

Given a callable function f(x, y) with a hidden formula and a value z, reverse engineer the formula and return all pos itive integer pairs x and y where f(x,y) == z. You may return the pairs in any order.
While the exact formula is hidden, the function is monotonically increasing, i.e.:

f(x, y) < f(x + 1, y)
f(x, y) < f(x, y + 1)

The function interface is defined like this:

```
interface CustomFunction {
public:
  // Returns some positive integer f(x, y) for two positive integers x and y based on a formula.
  int f(int x, int y);
};
```

We will judge your solution as follows:

The judge has a list of 9 hidden implementations of CustomFunction, along with a way to generate an answer key of all valid pairs for a specific z.
The judge will receive two inputs: a function_id (to determine which implementation to test your code with), and the target z.
The judge will call your findSolution and compare your results with the answer key.
If your results match the answer key, your solution will be Accepted.

Example 1:

Input: function_id = 1, z = 5
Output: [[1,4],[2,3],[3,2],[4,1]]
Explanation: The hidden formula for function_id = 1 is f(x, y) = x + y.
The following positive integer values of x and y make f(x, y) equal to 5:
x=1, y=4 -> f(1, 4) = 1 + 4 = 5.
x=2, y=3 -> f(2, 3) = 2 + 3 = 5.
x=3, y=2 -> f(3, 2) = 3 + 2 = 5.
x=4, y=1 -> f(4, 1) = 4 + 1 = 5.

Example 2:

Input: function_id = 2, z = 5
Output: [[1,5],[5,1]]
Explanation: The hidden formula for function_id = 2 is f(x, y) = x * y.
The following positive integer values of x and y make f(x, y) equal to 5:
x=1, y=5 -> f(1, 5) = 1 * 5 = 5.
x=5, y=1 -> f(5, 1) = 5 * 1 = 5.

Constraints:

1 <= function_id <= 9
1 <= z <= 100
It is guaranteed that the solutions of f(x, y) == z will be in the range 1 <= x, y <= 1000.
It is also guaranteed that f(x, y) will fit in 32 bit signed integer if 1 <= x, y <= 1000.

1238. Circular Permutation in Binary Representation

Given 2 integers n and start. Your task is return any permutation p of (0,1,2.....,2^n -1) such that :

p[0] = start
p[i] and p[i+1] differ by only one bit in their binary representation.
p[0] and p[2^n -1] must also differ by only one bit in their binary representation.

Example 1:

Input: n = 2, start = 3
Output: [3,2,0,1]
Explanation: The binary representation of the permutation is (11,10,00,01).
All the adjacent element differ by one bit. Another valid permutation is [3,1,0,2]

Example 2:

Input: n = 3, start = 2
Output: [2,6,7,5,4,0,1,3]
Explanation: The binary representation of the permutation is (010,110,111,101,100,000,001,011).

Constraints:

1 <= n <= 16
0 <= start < 2 ^ n

1239. Maximum Length of a Concatenated String with Unique Characters

You are given an array of strings arr. A string s is formed by the concatenation of a subsequence of arr that has uniq ue characters.
Return the maximum possible length of s.
A subsequence is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements.

Example 1:

Input: arr = ["un","iq","ue"]

Output: 4
Explanation: All the valid concatenations are:
- ""
- "un"
- "iq"
- "ue"
- "uniq" ("un" + "iq")
- "ique" ("iq" + "ue")
Maximum length is 4.

Example 2:

Input: arr = ["cha","r","act","ers"]
Output: 6
Explanation: Possible longest valid concatenations are "chaers" ("cha" + "ers") and "acters" ("act" + "ers").

Example 3:

Input: arr = ["abcdefghijklmnopqrstuvwxyz"]
Output: 26
Explanation: The only string in arr has all 26 characters.

Example 4:

Input: arr = ["aa","bb"]
Output: 0
Explanation: Both strings in arr do not have unique characters, thus there are no valid concatenations.

Constraints:

1 <= arr.length <= 16
1 <= arr[i].length <= 26
arr[i] contains only lowercase English letters.

1240. Tiling a Rectangle with the Fewest Squares

Given a rectangle of size n x m, return the minimum number of integer-sided squares that tile the rectangle.

Example 1:

Input: n = 2, m = 3
Output: 3
Explanation: 3 squares are necessary to cover the rectangle.
2 (squares of 1x1)

1 (square of 2x2)
Example 2:

Input: n = 5, m = 8
Output: 5

Example 3:

Input: n = 11, m = 13
Output: 6

Constraints:

1 <= n, m <= 13

1247. Minimum Swaps to Make Strings Equal

You are given two strings s1 and s2 of equal length consisting of letters "x" and "y" only. Your task is to make these two strings equal to each other. You can swap any two characters that belong to different strings, which means: swap s1[i] and s2[j].
Return the minimum number of swaps required to make s1 and s2 equal, or return -1 if it is impossible to do so.

Example 1:

Input: s1 = "xx", s2 = "yy"
Output: 1
Explanation:
Swap s1[0] and s2[1], s1 = "yx", s2 = "yx".
Example 2:

Input: s1 = "xy", s2 = "yx"
Output: 2
Explanation:
Swap s1[0] and s2[0], s1 = "yy", s2 = "xx".
Swap s1[0] and s2[1], s1 = "xy", s2 = "xy".
Note that you can't swap s1[0] and s1[1] to make s1 equal to "yx", cause we can only swap chars in different strings.
Example 3:

Input: s1 = "xx", s2 = "xy"
Output: -1

Example 4:

Input: s1 = "xxyyxyxyxx", s2 = "xyyxyxxxyx"
Output: 4

Constraints:

1 <= s1.length, s2.length <= 1000
s1, s2 only contain 'x' or 'y'.

1248. Count Number of Nice Subarrays

Given an array of integers nums and an integer k. A continuous subarray is called nice if there are k odd numbers on it.
Return the number of nice sub-arrays.

Example 1:

Input: nums = [1,1,2,1,1], k = 3
Output: 2
Explanation: The only sub-arrays with 3 odd numbers are [1,1,2,1] and [1,2,1,1].

Example 2:

Input: nums = [2,4,6], k = 1
Output: 0
Explanation: There is no odd numbers in the array.

Example 3:

Input: nums = [2,2,2,1,2,2,1,2,2,2], k = 2
Output: 16

Constraints:

1 <= nums.length <= 50000
1 <= nums[i] <= 10^5
1 <= k <= nums.length

1249. Minimum Remove to Make Valid Parentheses

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 parenthe ses 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.

Example 4:

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

Constraints:

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

1250. Check If It Is a Good Array

Given an array nums of positive integers. Your task is to select some subset of nums, multiply each element by an in teger and add all these numbers. The array is said to be good if you can obtain a sum of 1 from the array by any poss ible subset and multiplicand.
Return True if the array is good otherwise return False.

Example 1:

Input: nums = [12,5,7,23]
Output: true
Explanation: Pick numbers 5 and 7.
5*3 + 7*(-2) = 1

Example 2:

Input: nums = [29,6,10]
Output: true
Explanation: Pick numbers 29, 6 and 10.
29*1 + 6*(-3) + 10*(-1) = 1

Example 3:

Input: nums = [3,6]
Output: false

Constraints:

1 <= nums.length <= 10^5
1 <= nums[i] <= 10^9

1252. Cells with Odd Values in a Matrix

There is an m x n matrix that is initialized to all 0's. There is also a 2D array indices where each indices[i] = [ri, ci] r epresents a 0-indexed location to perform some increment operations on the matrix.
For each location indices[i], do both of the following:

Increment all the cells on row ri.
Increment all the cells on column ci.

Given m, n, and indices, return the number of odd-valued cells in the matrix after applying the increment to all locati ons in indices.

Example 1:

Input: m = 2, n = 3, indices = [[0,1],[1,1]]
Output: 6
Explanation: Initial matrix = [[0,0,0],[0,0,0]].
After applying first increment it becomes [[1,2,1],[0,1,0]].
The final matrix is [[1,3,1],[1,3,1]], which contains 6 odd numbers.

Example 2:

Input: m = 2, n = 2, indices = [[1,1],[0,0]]
Output: 0
Explanation: Final matrix = [[2,2],[2,2]]. There are no odd numbers in the final matrix.

Constraints:

1 <= m, n <= 50
1 <= indices.length <= 100
0 <= ri < m
0 <= ci < n

Follow up: Could you solve this in O(n + m + indices.length) time with only O(n + m) extra space?

1253. Reconstruct a 2-Row Binary Matrix

Given the following details of a matrix with n columns and 2 rows :

The matrix is a binary matrix, which means each element in the matrix can be 0 or 1.
The sum of elements of the 0-th(upper) row is given as upper.
The sum of elements of the 1-st(lower) row is given as lower.
The sum of elements in the i-th column(0-indexed) is colsum[i], where colsum is given as an integer array with lengt
h n.

Your task is to reconstruct the matrix with upper, lower and colsum.
Return it as a 2-D integer array.
If there are more than one valid solution, any of them will be accepted.
If no valid solution exists, return an empty 2-D array.

Example 1:

Input: upper = 2, lower = 1, colsum = [1,1,1]
Output: [[1,1,0],[0,0,1]]
Explanation: [[1,0,1],[0,1,0]], and [[0,1,1],[1,0,0]] are also correct answers.

Example 2:

Input: upper = 2, lower = 3, colsum = [2,2,1,1]
Output: []

Example 3:

Input: upper = 5, lower = 5, colsum = [2,1,2,0,1,0,1,2,0,1]
Output: [[1,1,1,0,1,0,0,1,0,0],[1,0,1,0,0,0,1,1,0,1]]

Constraints:

1 <= colsum.length <= 10^5
0 <= upper, lower <= colsum.length
0 <= colsum[i] <= 2

1254. Number of Closed Islands

Given a 2D grid consists of 0s (land) and 1s (water). An island is a maximal 4-directionally connected group of 0s a nd a closed island is an island totally (all left, top, right, bottom) surrounded by 1s.
Return the number of closed islands.

Example 1:

Input: grid = [[1,1,1,1,1,1,1,0],[1,0,0,0,0,1,1,0],[1,0,1,0,1,1,1,0],[1,0,0,0,0,1,0,1],[1,1,1,1,1,1,1,0]]
Output: 2
Explanation:
Islands in gray are closed because they are completely surrounded by water (group of 1s).
Example 2:

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

Example 3:

Input: grid = [[1,1,1,1,1,1,1],
[1,0,0,0,0,0,1],
[1,0,1,1,1,0,1],
[1,0,1,0,1,0,1],
[1,0,1,1,1,0,1],
[1,0,0,0,0,0,1],
[1,1,1,1,1,1,1]]
Output: 2

Constraints:

1 <= grid.length, grid[0].length <= 100
0 <= grid[i][j] <=1

1255. Maximum Score Words Formed by Letters

Given a list of words, list of single letters (might be repeating) and score of every character.
Return the maximum score of any valid set of words formed by using the given letters (words[i] cannot be used two or more times).
It is not necessary to use all characters in letters and each letter can only be used once. Score of letters 'a', 'b', 'c', ... ,'z ' is given by score[0], score[1], ... , score[25] respectively.

Example 1:

Input: words = ["dog","cat","dad","good"], letters = ["a","a","c","d","d","d","g","o","o"], score = [1,0,9,5,0,0,3,0,0,0,0,0,0,0,2,0,0,0,0,0,0,0,0,0,0,0]
Output: 23
Explanation:
Score a=1, c=9, d=5, g=3, o=2
Given letters, we can form the words "dad" (5+1+5) and "good" (3+2+2+5) with a score of 23.
Words "dad" and "dog" only get a score of 21.
Example 2:

Input: words = ["xxxz","ax","bx","cx"], letters = ["z","a","b","c","x","x","x"], score = [4,4,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,10]
Output: 27
Explanation:
Score a=4, b=4, c=4, x=5, z=10
Given letters, we can form the words "ax" (4+5), "bx" (4+5) and "cx" (4+5) with a score of 27.
Word "xxxz" only get a score of 25.
Example 3:

Input: words = ["leetcode"], letters = ["l","e","t","c","o","d"], score = [0,0,1,1,1,0,0,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,0,0,0,0]
Output: 0
Explanation:
Letter "e" can only be used once.

Constraints:

1 <= words.length <= 14
1 <= words[i].length <= 15
1 <= letters.length <= 100
letters[i].length == 1
score.length == 26
0 <= score[i] <= 10
words[i], letters[i] contains only lower case English letters.

1260. Shift 2D Grid

Given a 2D grid of size m x n and an integer k. You need to shift the grid k times.
In one shift operation:

Element at grid[i][j] moves to grid[i][j + 1].
Element at grid[i][n - 1] moves to grid[i + 1][0].
Element at grid[m - 1][n - 1] moves to grid[0][0].

Return the 2D grid after applying shift operation k times.

Example 1:

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

Example 2:

Input: grid = [[3,8,1,9],[19,7,2,5],[4,6,11,10],[12,0,21,13]], k = 4
Output: [[12,0,21,13],[3,8,1,9],[19,7,2,5],[4,6,11,10]]

Example 3:

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

Constraints:

m == grid.length
n == grid[i].length
1 <= m <= 50
1 <= n <= 50
-1000 <= grid[i][j] <= 1000
0 <= k <= 100

1261. Find Elements in a Contaminated Binary Tree

Given a binary tree with the following rules:

root.val == 0
If treeNode.val == x and treeNode.left != null, then treeNode.left.val == 2 * x + 1
If treeNode.val == x and treeNode.right != null, then treeNode.right.val == 2 * x + 2

Now the binary tree is contaminated, which means all treeNode.val have been changed to -1.
Implement the FindElements class:

FindElements(TreeNode* root) Initializes the object with a contaminated binary tree and recovers it.
bool find(int target) Returns true if the target value exists in the recovered binary tree.

Example 1:

Input
["FindElements","find","find"]
[[[-1,null,-1]],[1],[2]]
Output
[null,false,true]
Explanation
FindElements findElements = new FindElements([-1,null,-1]);
findElements.find(1); // return False
findElements.find(2); // return True
Example 2:

Input
["FindElements","find","find","find"]
[[[-1,-1,-1,-1,-1]],[1],[3],[5]]
Output
[null,true,true,false]
Explanation
FindElements findElements = new FindElements([-1,-1,-1,-1,-1]);
findElements.find(1); // return True
findElements.find(3); // return True
findElements.find(5); // return False
Example 3:

Input
["FindElements","find","find","find","find"]
[[[-1,null,-1,-1,null,-1]],[2],[3],[4],[5]]
Output
[null,true,false,false,true]
Explanation
FindElements findElements = new FindElements([-1,null,-1,-1,null,-1]);
findElements.find(2); // return True
findElements.find(3); // return False
findElements.find(4); // return False
findElements.find(5); // return True

Constraints:

TreeNode.val == -1
The height of the binary tree is less than or equal to 20
The total number of nodes is between [1, 104]
Total calls of find() is between [1, 104]
0 <= target <= 106

1262. Greatest Sum Divisible by Three

Given an array nums of integers, we need to find the maximum possible sum of elements of the array such that it is d ivisible by three.

Example 1:

Input: nums = [3,6,5,1,8]
Output: 18
Explanation: Pick numbers 3, 6, 1 and 8 their sum is 18 (maximum sum divisible by 3).
Example 2:

Input: nums = [4]
Output: 0
Explanation: Since 4 is not divisible by 3, do not pick any number.

Example 3:

Input: nums = [1,2,3,4,4]
Output: 12
Explanation: Pick numbers 1, 3, 4 and 4 their sum is 12 (maximum sum divisible by 3).

Constraints:

1 <= nums.length <= 4 * 10^4
1 <= nums[i] <= 10^4

1263. Minimum Moves to Move a Box to Their Target Location

A storekeeper is a game in which the player pushes boxes around in a warehouse trying to get them to target location s.
The game is represented by an m x n grid of characters grid where each element is a wall, floor, or box.
Your task is to move the box 'B' to the target position 'T' under the following rules:

The character 'S' represents the player. The player can move up, down, left, right in grid if it is a floor (empty cell).
The character '.' represents the floor which means a free cell to walk.
The character '#' represents the wall which means an obstacle (impossible to walk there).
There is only one box 'B' and one target cell 'T' in the grid.
The box can be moved to an adjacent free cell by standing next to the box and then moving in the direction of the bo x. This is a push.
The player cannot walk through the box.

Return the minimum number of pushes to move the box to the target. If there is no way to reach the target, return -1.

Example 1:

Input: grid = [["#","#","#","#","#","#"],
["#","T","#","#","#","#"],
["#",".",".","B",".","#"],
["#",".","#","#",".","#"],
["#",".",".",".",".","S","#"],
["#","#","#","#","#","#"]]
Output: 3
Explanation: We return only the number of times the box is pushed.
Example 2:

Input: grid = [["#","#","#","#","#","#"],
["#","T","#","#","#","#"],
["#",".",".","B",".","#"],
["#","#","#","#",".","#"],
["#",".",".",".",".","S","#"],
["#","#","#","#","#","#"]]
Output: -1

Example 3:

Input: grid = [["#","#","#","#","#","#"],
["#","T",".",".","#","#"],
["#",".","#","B",".","#"],
["#",".",".",".",".",".","#"],
["#",".",".",".",".","S","#"],
["#","#","#","#","#","#"]]
Output: 5
Explanation: push the box down, left, left, up and up.

Example 4:

Input: grid = [["#","#","#","#","#","#","#"],
["#","S","#",".","B","T","#"],
["#","#","#","#","#","#","#"]]

Output: -1

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 20
grid contains only characters '.', '#', 'S', 'T', or 'B'.
There is only one character 'S', 'B', and 'T' in the grid.

1266. Minimum Time Visiting All Points

On a 2D plane, there are n points with integer coordinates points[i] = [xi, yi]. Return the minimum time in seconds t
o visit all the points in the order given by points.
You can move according to these rules:

In 1 second, you can either:

move vertically by one unit,
move horizontally by one unit, or
move diagonally sqrt(2) units (in other words, move one unit vertically then one unit horizontally in 1 second).

You have to visit the points in the same order as they appear in the array.
You are allowed to pass through points that appear later in the order, but these do not count as visits.

Example 1:

Input: points = [[1,1],[3,4],[-1,0]]
Output: 7
Explanation: One optimal path is [1,1] -> [2,2] -> [3,3] -> [3,4] -> [2,3] -> [1,2] -> [0,1] -> [-1,0]
Time from [1,1] to [3,4] = 3 seconds
Time from [3,4] to [-1,0] = 4 seconds
Total time = 7 seconds
Example 2:

Input: points = [[3,2],[-2,2]]
Output: 5

Constraints:

points.length == n
1 <= n <= 100
points[i].length == 2
-1000 <= points[i][0], points[i][1] <= 1000

1267. Count Servers that Communicate

You are given a map of a server center, represented as a m * n integer matrix grid, where 1 means that on that cell th ere is a server and 0 means that it is no server. Two servers are said to communicate if they are on the same row or o n the same column.

Return the number of servers that communicate with any other server.

Example 1:

Input: grid = [[1,0],[0,1]]
Output: 0
Explanation: No servers can communicate with others.
Example 2:

Input: grid = [[1,0],[1,1]]
Output: 3
Explanation: All three servers can communicate with at least one other server.

Example 3:

Input: grid = [[1,1,0,0],[0,0,1,0],[0,0,1,0],[0,0,0,1]]
Output: 4
Explanation: The two servers in the first row can communicate with each other. The two servers in the third column can communicate with each other. The server at right bottom corner can't communicate with any other server.

Constraints:

m == grid.length
n == grid[i].length
1 <= m <= 250
1 <= n <= 250
grid[i][j] == 0 or 1

1268. Search Suggestions System

Given an array of strings products and a string searchWord. We want to design a system that suggests at most three product names from products after each character of searchWord is typed. Suggested products should have common prefix with the searchWord. If there are more than three products with a common prefix return the three lexicographi cally minimums products.
Return list of lists of the suggested products after each character of searchWord is typed.

Example 1:

Input: products = ["mobile","mouse","moneypot","monitor","mousepad"], searchWord = "mouse"
Output: [
["mobile","moneypot","monitor"],
["mobile","moneypot","monitor"],
["mouse","mousepad"],
["mouse","mousepad"],
["mouse","mousepad"]
]
Explanation: products sorted lexicographically = ["mobile","moneypot","monitor","mouse","mousepad"]
After typing m and mo all products match and we show user ["mobile","moneypot","monitor"]
After typing mou, mous and mouse the system suggests ["mouse","mousepad"]

Example 2:

Input: products = ["havana"], searchWord = "havana"
Output: [["havana"],["havana"],["havana"],["havana"],["havana"],["havana"]]

Example 3:

Input: products = ["bags","baggage","banner","box","cloths"], searchWord = "bags"
Output: [["baggage","bags","banner"],["baggage","bags","banner"],["baggage","bags"],["bags"]]

Example 4:

Input: products = ["havana"], searchWord = "tatiana"
Output: [[],[],[],[],[],[],[]]

Constraints:

1 <= products.length <= 1000
There are no repeated elements in products.
1 <= Σ products[i].length <= 2 * 10^4
All characters of products[i] are lower-case English letters.
1 <= searchWord.length <= 1000
All characters of searchWord are lower-case English letters.

1269. Number of Ways to Stay in the Same Place After Some Steps

You have a pointer at index 0 in an array of size arrLen. At each step, you can move 1 position to the left, 1 position to the right in the array, or stay in the same place (The pointer should not be placed outside the array at any time). Given two integers steps and arrLen, return the number of ways such that your pointer still at index 0 after exactly st eps steps. Since the answer may be too large, return it modulo 109 + 7.

Example 1:

Input: steps = 3, arrLen = 2
Output: 4
Explanation: There are 4 differents ways to stay at index 0 after 3 steps.
Right, Left, Stay
Stay, Right, Left
Right, Stay, Left
Stay, Stay, Stay

Example 2:

Input: steps = 2, arrLen = 4
Output: 2
Explanation: There are 2 differents ways to stay at index 0 after 2 steps
Right, Left
Stay, Stay

Example 3:

Input: steps = 4, arrLen = 2
Output: 8

Constraints:

1 <= steps <= 500
1 <= arrLen <= 106

1275. Find Winner on a Tic Tac Toe Game

Tic-tac-toe is played by two players A and B on a 3 x 3 grid. The rules of Tic-Tac-Toe are:

Players take turns placing characters into empty squares ' '.
The first player A always places 'X' characters, while the second player B always places 'O' characters.
'X' and 'O' characters are always placed into empty squares, never on filled ones.
The game ends when there are three of the same (non-empty) character filling any row, column, or diagonal.
The game also ends if all squares are non-empty.
No more moves can be played if the game is over.

Given a 2D integer array moves where moves[i] = [rowi, coli] indicates that the ith move will be played on grid[rowi][coli]. return the winner of the game if it exists (A or B). In case the game ends in a draw return "Draw". If there are still movements to play return "Pending".
You can assume that moves is valid (i.e., it follows the rules of Tic-Tac-Toe), the grid is initially empty, and A will play first.

Example 1:

Input: moves = [[0,0],[2,0],[1,1],[2,1],[2,2]]
Output: "A"
Explanation: A wins, they always play first.

Example 2:

Input: moves = [[0,0],[1,1],[0,1],[0,2],[1,0],[2,0]]
Output: "B"
Explanation: B wins.

Example 3:

Input: moves = [[0,0],[1,1],[2,0],[1,0],[1,2],[2,1],[0,1],[0,2],[2,2]]
Output: "Draw"
Explanation: The game ends in a draw since there are no moves to make.

Example 4:

Input: moves = [[0,0],[1,1]]
Output: "Pending"
Explanation: The game has not finished yet.

Constraints:

1 <= moves.length <= 9
moves[i].length == 2
0 <= rowi, coli <= 2
There are no repeated elements on moves.
moves follow the rules of tic tac toe.

1276. Number of Burgers with No Waste of Ingredients

Given two integers tomatoSlices and cheeseSlices. The ingredients of different burgers are as follows:

Jumbo Burger: 4 tomato slices and 1 cheese slice.
Small Burger: 2 Tomato slices and 1 cheese slice.

Return [total_jumbo, total_small] so that the number of remaining tomatoSlices equal to 0 and the number of remain ing cheeseSlices equal to 0. If it is not possible to make the remaining tomatoSlices and cheeseSlices equal to 0 retur n [].

Example 1:

Input: tomatoSlices = 16, cheeseSlices = 7
Output: [1,6]
Explantion: To make one jumbo burger and 6 small burgers we need 4*1 + 2*6 = 16 tomato and 1 + 6 = 7 cheese. T here will be no remaining ingredients.

Example 2:

Input: tomatoSlices = 17, cheeseSlices = 4
Output: []
Explantion: There will be no way to use all ingredients to make small and jumbo burgers.

Example 3:

Input: tomatoSlices = 4, cheeseSlices = 17
Output: []
Explantion: Making 1 jumbo burger there will be 16 cheese remaining and making 2 small burgers there will be 15 c heese remaining.

Example 4:

Input: tomatoSlices = 0, cheeseSlices = 0
Output: [0,0]

Example 5:

Input: tomatoSlices = 2, cheeseSlices = 1
Output: [0,1]

Constraints:

0 <= tomatoSlices <= 10^7
0 <= cheeseSlices <= 10^7

1277. Count Square Submatrices with All Ones

Given a m * n matrix of ones and zeros, return how many square submatrices have all ones.

Example 1:

Input: matrix =
[
  [0,1,1,1],
  [1,1,1,1],
  [0,1,1,1]
]
Output: 15
Explanation:
There are 10 squares of side 1.
There are 4 squares of side 2.
There is 1 square of side 3.
Total number of squares = 10 + 4 + 1 = 15.

Example 2:

Input: matrix =
[
  [1,0,1],
  [1,1,0],
  [1,1,0]
]
Output: 7
Explanation:
There are 6 squares of side 1.
There is 1 square of side 2.
Total number of squares = 6 + 1 = 7.

Constraints:

1 <= arr.length <= 300
1 <= arr[0].length <= 300
0 <= arr[i][j] <= 1

1278. Palindrome Partitioning III

You are given a string s containing lowercase letters and an integer k. You need to :

First, change some characters of s to other lowercase English letters.
Then divide s into k non-empty disjoint substrings such that each substring is a palindrome.

Return the minimal number of characters that you need to change to divide the string.

Example 1:

Input: s = "abc", k = 2
Output: 1
Explanation: You can split the string into "ab" and "c", and change 1 character in "ab" to make it palindrome.

Example 2:

Input: s = "aabbc", k = 3
Output: 0
Explanation: You can split the string into "aa", "bb" and "c", all of them are palindrome.
Example 3:

Input: s = "leetcode", k = 8
Output: 0

Constraints:

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

1281. Subtract the Product and Sum of Digits of an Integer

Given an integer number n, return the difference between the product of its digits and the sum of its digits.

Example 1:

Input: n = 234
Output: 15
Explanation:
Product of digits = 2 * 3 * 4 = 24
Sum of digits = 2 + 3 + 4 = 9
Result = 24 - 9 = 15

Example 2:

Input: n = 4421
Output: 21

Explanation:
Product of digits = 4 * 4 * 2 * 1 = 32
Sum of digits = 4 + 4 + 2 + 1 = 11
Result = 32 - 11 = 21

Constraints:

1 <= n <= 10^5

1282. Group the People Given the Group Size They Belong To

There are n people that are split into some unknown number of groups. Each person is labeled with a unique ID from 0 to n - 1.
You are given an integer array groupSizes, where groupSizes[i] is the size of the group that person i is in. For example, if groupSizes[1] = 3, then person 1 must be in a group of size 3.
Return a list of groups such that each person i is in a group of size groupSizes[i].
Each person should appear in exactly one group, and every person must be in a group. If there are multiple answers, return any of them. It is guaranteed that there will be at least one valid solution for the given input.

Example 1:

Input: groupSizes = [3,3,3,3,3,1,3]
Output: [[5],[0,1,2],[3,4,6]]
Explanation:
The first group is [5]. The size is 1, and groupSizes[5] = 1.
The second group is [0,1,2]. The size is 3, and groupSizes[0] = groupSizes[1] = groupSizes[2] = 3.
The third group is [3,4,6]. The size is 3, and groupSizes[3] = groupSizes[4] = groupSizes[6] = 3.
Other possible solutions are [[2,1,6],[5],[0,4,3]] and [[5],[0,6,2],[4,3,1]].

Example 2:

Input: groupSizes = [2,1,3,3,3,2]
Output: [[1],[0,5],[2,3,4]]

Constraints:

groupSizes.length == n
1 <= n <= 500
1 <= groupSizes[i] <= n

1283. Find the Smallest Divisor Given a Threshold

Given an array of integers nums and an integer threshold, we will choose a positive integer divisor, divide all the arr ay by it, and sum the division's result. Find the smallest divisor such that the result mentioned above is less than or e qual to threshold.
Each result of the division is rounded to the nearest integer greater than or equal to that element. (For example: 7/3 = 3 and 10/2 = 5).
It is guaranteed that there will be an answer.

Example 1:

Input: nums = [1,2,5,9], threshold = 6
Output: 5
Explanation: We can get a sum to 17 (1+2+5+9) if the divisor is 1.
If the divisor is 4 we can get a sum of 7 (1+1+2+3) and if the divisor is 5 the sum will be 5 (1+1+1+2).

Example 2:

Input: nums = [44,22,33,11,1], threshold = 5
Output: 44

Example 3:

Input: nums = [21212,10101,12121], threshold = 1000000
Output: 1

Example 4:

Input: nums = [2,3,5,7,11], threshold = 11
Output: 3

Constraints:

1 <= nums.length <= 5 * 104
1 <= nums[i] <= 106
nums.length <= threshold <= 106

1284. Minimum Number of Flips to Convert Binary Matrix to Zero Matrix

Given a m x n binary matrix mat. In one step, you can choose one cell and flip it and all the four neighbors of it if th ey exist (Flip is changing 1 to 0 and 0 to 1). A pair of cells are called neighbors if they share one edge.

Return the minimum number of steps required to convert mat to a zero matrix or -1 if you cannot.
A binary matrix is a matrix with all cells equal to 0 or 1 only.
A zero matrix is a matrix with all cells equal to 0.

Example 1:

Input: mat = [[0,0],[0,1]]
Output: 3
Explanation: One possible solution is to flip (1, 0) then (0, 1) and finally (1, 1) as shown.

Example 2:

Input: mat = [[0]]
Output: 0
Explanation: Given matrix is a zero matrix. We don't need to change it.

Example 3:

Input: mat = [[1,1,1],[1,0,1],[0,0,0]]
Output: 6

Example 4:

Input: mat = [[1,0,0],[1,0,0]]
Output: -1
Explanation: Given matrix can't be a zero matrix

Constraints:

m == mat.length
n == mat[i].length
1 <= m, n <= 3
mat[i][j] is either 0 or 1.

1286. Iterator for Combination

Design the CombinationIterator class:

CombinationIterator(string characters, int combinationLength) Initializes the object with a string characters of sorted distinct lowercase English letters and a number combinationLength as arguments.
next() Returns the next combination of length combinationLength in lexicographical order.
hasNext() Returns true if and only if there exists a next combination.

Example 1:

Input
["CombinationIterator", "next", "hasNext", "next", "hasNext", "next", "hasNext"]
[["abc", 2], [], [], [], [], [], []]
Output
[null, "ab", true, "ac", true, "bc", false]

Explanation
CombinationIterator itr = new CombinationIterator("abc", 2);
itr.next();   // return "ab"
itr.hasNext(); // return True
itr.next();   // return "ac"
itr.hasNext(); // return True
itr.next();   // return "bc"
itr.hasNext(); // return False

Constraints:

1 <= combinationLength <= characters.length <= 15
All the characters of characters are unique.
At most 104 calls will be made to next and hasNext.
It is guaranteed that all calls of the function next are valid.

1287. Element Appearing More Than 25% In Sorted Array

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

1288. Remove Covered Intervals

Given an array intervals where intervals[i] = [li, ri] represent the interval [li, ri), remove all intervals that are covered by another interval in the list.
The interval [a, b) is covered by the interval [c, d) if and only if c <= a and b <= d.
Return the number of remaining intervals.

Example 1:

Input: intervals = [[1,4],[3,6],[2,8]]
Output: 2
Explanation: Interval [3,6] is covered by [2,8], therefore it is removed.

Example 2:

Input: intervals = [[1,4],[2,3]]
Output: 1

Example 3:

Input: intervals = [[0,10],[5,12]]
Output: 2

Example 4:

Input: intervals = [[3,10],[4,10],[5,11]]
Output: 2

Example 5:

Input: intervals = [[1,2],[1,4],[3,4]]
Output: 1

Constraints:

1 <= intervals.length <= 1000
intervals[i].length == 2
0 <= li <= ri <= 105
All the given intervals are unique.

1289. Minimum Falling Path Sum II

Given an n x n integer matrix grid, return the minimum sum of a falling path with non-zero shifts.
A falling path with non-zero shifts is a choice of exactly one element from each row of grid such that no two element s chosen in adjacent rows are in the same column.

Example 1:

Input: arr = [[1,2,3],[4,5,6],[7,8,9]]
Output: 13
Explanation:
The possible falling paths are:
[1,5,9], [1,5,7], [1,6,7], [1,6,8],
[2,4,8], [2,4,9], [2,6,7], [2,6,8],
[3,4,8], [3,4,9], [3,5,7], [3,5,9]
The falling path with the smallest sum is [1,5,7], so the answer is 13.

Example 2:

Input: grid = [[7]]
Output: 7

Constraints:

n == grid.length == grid[i].length
1 <= n <= 200
-99 <= grid[i][j] <= 99

1290. Convert Binary Number in a Linked List to Integer

Given head which is a reference node to a singly-linked list. The value of each node in the linked list is either 0 or 1. The linked list holds the binary representation of a number.
Return the decimal value of the number in the linked list.

Example 1:

Input: head = [1,0,1]
Output: 5
Explanation: (101) in base 2 = (5) in base 10

Example 2:

Input: head = [0]
Output: 0

Example 3:

Input: head = [1]
Output: 1

Example 4:

Input: head = [1,0,0,1,0,0,1,1,1,0,0,0,0,0,0]
Output: 18880

Example 5:

Input: head = [0,0]
Output: 0

Constraints:

The Linked List is not empty.
Number of nodes will not exceed 30.
Each node's value is either 0 or 1.

1291. Sequential Digits

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

1292. Maximum Side Length of a Square with Sum Less than or Equal to Threshold

Given a m x n matrix mat and an integer threshold. Return the maximum side-length of a square with a sum less than or equal to threshold or return 0 if there is no such square.

Example 1:

Input: mat = [[1,1,3,2,4,3,2],[1,1,3,2,4,3,2],[1,1,3,2,4,3,2]], threshold = 4
Output: 2
Explanation: The maximum side length of square with sum less than 4 is 2 as shown.

Example 2:

Input: mat = [[2,2,2,2,2],[2,2,2,2,2],[2,2,2,2,2],[2,2,2,2,2],[2,2,2,2,2]], threshold = 1
Output: 0

Example 3:

Input: mat = [[1,1,1,1],[1,0,0,0],[1,0,0,0],[1,0,0,0]], threshold = 6
Output: 3

Example 4:

Input: mat = [[18,70],[61,1],[25,85],[14,40],[11,96],[97,96],[63,45]], threshold = 40184
Output: 2

Constraints:

1 <= m, n <= 300
m == mat.length
n == mat[i].length
0 <= mat[i][j] <= 10000
0 <= threshold <= 10^5

1293. Shortest Path in a Grid with Obstacles Elimination

You are given an m x n integer matrix grid where each cell is either 0 (empty) or 1 (obstacle). You can move up, do wn, left, or right from and to an empty cell in one step.
Return the minimum number of steps to walk from the upper left corner (0, 0) to the lower right corner (m - 1, n - 1)

given that you can eliminate at most k obstacles. If it is not possible to find such walk return -1.

Example 1:

Input: grid = [[0,0,0],[1,1,0],[0,0,0],[0,1,1],[0,0,0]], k = 1
Output: 6
Explanation:
The shortest path without eliminating any obstacle is 10.
The shortest path with one obstacle elimination at position (3,2) is 6. Such path is (0,0) -> (0,1) -> (0,2) -> (1,2) -> (2,2) -> (3,2) -> (4,2).

Example 2:

Input: grid = [[0,1,1],[1,1,1],[1,0,0]], k = 1
Output: -1
Explanation: We need to eliminate at least two obstacles to find such a walk.

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 40
1 <= k <= m * n
grid[i][j] is either 0 or 1.
grid[0][0] == grid[m - 1][n - 1] == 0

1295. Find Numbers with Even Number of Digits

Given an array nums of integers, return how many of them contain an even number of digits.

Example 1:

Input: nums = [12,345,2,6,7896]
Output: 2
Explanation:
12 contains 2 digits (even number of digits).
345 contains 3 digits (odd number of digits).
2 contains 1 digit (odd number of digits).
6 contains 1 digit (odd number of digits).
7896 contains 4 digits (even number of digits).
Therefore only 12 and 7896 contain an even number of digits.

Example 2:

Input: nums = [555,901,482,1771]
Output: 1
Explanation:
Only 1771 contains an even number of digits.

Constraints:

1 <= nums.length <= 500
1 <= nums[i] <= 105

1296. Divide Array in Sets of K Consecutive Numbers

Given an array of integers nums and a positive integer k, find whether it is possible to divide this array into sets of k consecutive numbers.
Return true if it is possible. Otherwise, return false.

Example 1:

Input: nums = [1,2,3,3,4,4,5,6], k = 4
Output: true
Explanation: Array can be divided into [1,2,3,4] and [3,4,5,6].

Example 2:

Input: nums = [3,2,1,2,3,4,3,4,5,9,10,11], k = 3
Output: true
Explanation: Array can be divided into [1,2,3] , [2,3,4] , [3,4,5] and [9,10,11].

Example 3:

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

Example 4:

Input: nums = [1,2,3,4], k = 3
Output: false
Explanation: Each array should be divided in subarrays of size 3.

Constraints:

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

Note: This question is the same as 846: https://leetcode.com/problems/hand-of-straights/

1297. Maximum Number of Occurrences of a Substring

Given a string s, return the maximum number of ocurrences of any substring under the following rules:

The number of unique characters in the substring must be less than or equal to maxLetters.
The substring size must be between minSize and maxSize inclusive.

Example 1:

Input: s = "aababcaab", maxLetters = 2, minSize = 3, maxSize = 4
Output: 2
Explanation: Substring "aab" has 2 ocurrences in the original string.
It satisfies the conditions, 2 unique letters and size 3 (between minSize and maxSize).

Example 2:

Input: s = "aaaa", maxLetters = 1, minSize = 3, maxSize = 3
Output: 2
Explanation: Substring "aaa" occur 2 times in the string. It can overlap.

Example 3:

Input: s = "aabcabcab", maxLetters = 2, minSize = 2, maxSize = 3
Output: 3

Example 4:

Input: s = "abcde", maxLetters = 2, minSize = 3, maxSize = 3
Output: 0

Constraints:

1 <= s.length <= 10^5
1 <= maxLetters <= 26
1 <= minSize <= maxSize <= min(26, s.length)
s only contains lowercase English letters.

1298. Maximum Candies You Can Get from Boxes

Given n boxes, each box is given in the format [status, candies, keys, containedBoxes] where:

status[i]: an integer which is 1 if box[i] is open and 0 if box[i] is closed.
candies[i]: an integer representing the number of candies in box[i].
keys[i]: an array contains the indices of the boxes you can open with the key in box[i].
containedBoxes[i]: an array contains the indices of the boxes found in box[i].

You will start with some boxes given in initialBoxes array. You can take all the candies in any open box and you can use the keys in it to open new boxes and you also can use the boxes you find in it.
Return the maximum number of candies you can get following the rules above.

Example 1:

Input: status = [1,0,1,0], candies = [7,5,4,100], keys = [[],[],[1],[]], containedBoxes = [[1,2],[3],[],[]], initialBoxes = [0]
Output: 16
Explanation: You will be initially given box 0. You will find 7 candies in it and boxes 1 and 2. Box 1 is closed and y ou don't have a key for it so you will open box 2. You will find 4 candies and a key to box 1 in box 2.
In box 1, you will find 5 candies and box 3 but you will not find a key to box 3 so box 3 will remain closed.
Total number of candies collected = 7 + 4 + 5 = 16 candy.

Example 2:

Input: status = [1,0,0,0,0,0], candies = [1,1,1,1,1,1], keys = [[1,2,3,4,5],[],[],[],[],[]], containedBoxes = [[1,2,3,4,5],[],[],[],[],[]], initialBoxes = [0]
Output: 6
Explanation: You have initially box 0. Opening it you can find boxes 1,2,3,4 and 5 and their keys. The total number of candies will be 6.

Example 3:

Input: status = [1,1,1], candies = [100,1,100], keys = [[],[0,2],[]], containedBoxes = [[],[],[]], initialBoxes = [1]
Output: 1

Example 4:

Input: status = [1], candies = [100], keys = [[]], containedBoxes = [[]], initialBoxes = []
Output: 0

Example 5:

Input: status = [1,1,1], candies = [2,3,2], keys = [[],[],[]], containedBoxes = [[],[],[]], initialBoxes = [2,1,0]
Output: 7

Constraints:

1 <= status.length <= 1000
status.length == candies.length == keys.length == containedBoxes.length == n
status[i] is 0 or 1.

1 <= candies[i] <= 1000
0 <= keys[i].length <= status.length
0 <= keys[i][j] < status.length
All values in keys[i] are unique.
0 <= containedBoxes[i].length <= status.length
0 <= containedBoxes[i][j] < status.length
All values in containedBoxes[i] are unique.
Each box is contained in one box at most.
0 <= initialBoxes.length <= status.length
0 <= initialBoxes[i] < status.length

1299. Replace Elements with Greatest Element on Right Side

Given an array arr, replace every element in that array with the greatest element among the elements to its right, and replace the last element with -1.
After doing so, return the array.

Example 1:

Input: arr = [17,18,5,4,6,1]
Output: [18,6,6,6,1,-1]
Explanation:
- index 0 --> the greatest element to the right of index 0 is index 1 (18).
- index 1 --> the greatest element to the right of index 1 is index 4 (6).
- index 2 --> the greatest element to the right of index 2 is index 4 (6).
- index 3 --> the greatest element to the right of index 3 is index 4 (6).
- index 4 --> the greatest element to the right of index 4 is index 5 (1).
- index 5 --> there are no elements to the right of index 5, so we put -1.

Example 2:

Input: arr = [400]
Output: [-1]
Explanation: There are no elements to the right of index 0.

Constraints:

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

1300. Sum of Mutated Array Closest to Target

Given an integer array arr and a target value target, return the integer value such that when we change all the integers larger than value in the given array to be equal to value, the sum of the array gets as close as possible (in absolute difference) to target.
In case of a tie, return the minimum such integer.
Notice that the answer is not neccesarilly a number from arr.

Example 1:

Input: arr = [4,9,3], target = 10
Output: 3
Explanation: When using 3 arr converts to [3, 3, 3] which sums 9 and that's the optimal answer.

Example 2:

Input: arr = [2,3,5], target = 10
Output: 5

Example 3:

Input: arr = [60864,25176,27249,21296,20204], target = 56803
Output: 11361

Constraints:

1 <= arr.length <= 104
1 <= arr[i], target <= 105

1301. Number of Paths with Max Score

You are given a square board of characters. You can move on the board starting at the bottom right square marked with the character 'S'.
You need to reach the top left square marked with the character 'E'. The rest of the squares are labeled either with a numeric character 1, 2, ..., 9 or with an obstacle 'X'. In one move you can go up, left or up-left (diagonally) only if there is no obstacle there.
Return a list of two integers: the first integer is the maximum sum of numeric characters you can collect, and the second is the number of such paths that you can take to get that maximum sum, taken modulo 10^9 + 7.
In case there is no path, return [0, 0].

Example 1:
Input: board = ["E23","2X2","12S"]
Output: [7,1]

Example 2:
Input: board = ["E12","1X1","21S"]
Output: [4,2]
Example 3:
Input: board = ["E11","XXX","11S"]
Output: [0,0]

Constraints:

2 <= board.length == board[i].length <= 100

1302. Deepest Leaves Sum

Given the root of a binary tree, return the sum of values of its deepest leaves.

Example 1:

Input: root = [1,2,3,4,5,null,6,7,null,null,null,null,8]
Output: 15

Example 2:

Input: root = [6,7,8,2,7,1,3,9,null,1,4,null,null,null,5]
Output: 19

Constraints:

The number of nodes in the tree is in the range [1, 104].
1 <= Node.val <= 100

1304. Find N Unique Integers Sum up to Zero

Given an integer n, return any array containing n unique integers such that they add up to 0.

Example 1:

Input: n = 5
Output: [-7,-1,1,3,4]
Explanation: These arrays also are accepted [-5,-1,1,2,3] , [-3,-1,2,-2,4].

Example 2:

Input: n = 3
Output: [-1,0,1]

Example 3:

Input: n = 1
Output: [0]

Constraints:

1 <= n <= 1000

1305. All Elements in Two Binary Search Trees

Given two binary search trees root1 and root2.
Return a list containing all the integers from both trees sorted in ascending order.

Example 1:

Input: root1 = [2,1,4], root2 = [1,0,3]
Output: [0,1,1,2,3,4]

Example 2:

Input: root1 = [0,-10,10], root2 = [5,1,7,0,2]
Output: [-10,0,0,1,2,5,7,10]

Example 3:

Input: root1 = [], root2 = [5,1,7,0,2]
Output: [0,1,2,5,7]

Example 4:

Input: root1 = [0,-10,10], root2 = []
Output: [-10,0,10]

Example 5:

Input: root1 = [1,null,8], root2 = [8,1]
Output: [1,1,8,8]

Constraints:

Each tree has at most 5000 nodes.
Each node's value is between [-10^5, 10^5].

1306. Jump Game III

Given an array of non-negative integers arr, you are initially positioned at start index of the array. When you are at i ndex i, you can jump to i + arr[i] or i - arr[i], check if you can reach to any index with value 0.
Notice that you can not jump outside of the array at any time.

Example 1:

Input: arr = [4,2,3,0,3,1,2], start = 5
Output: true
Explanation:
All possible ways to reach at index 3 with value 0 are:
index 5 -> index 4 -> index 1 -> index 3
index 5 -> index 6 -> index 4 -> index 1 -> index 3

Example 2:

Input: arr = [4,2,3,0,3,1,2], start = 0
Output: true
Explanation:
One possible way to reach at index 3 with value 0 is:
index 0 -> index 4 -> index 1 -> index 3

Example 3:

Input: arr = [3,0,2,1,2], start = 2
Output: false
Explanation: There is no way to reach at index 1 with value 0.

Constraints:

1 <= arr.length <= 5 * 104
0 <= arr[i] < arr.length
0 <= start < arr.length

1307. Verbal Arithmetic Puzzle

Given an equation, represented by words on left side and the result on right side.
You need to check if the equation is solvable under the following rules:

Each character is decoded as one digit (0 - 9).
Every pair of different characters they must map to different digits.
Each words[i] and result are decoded as one number without leading zeros.
Sum of numbers on left side (words) will equal to the number on right side (result).

Return True if the equation is solvable otherwise return False.

Example 1:

Input: words = ["SEND","MORE"], result = "MONEY"
Output: true
Explanation: Map 'S'-> 9, 'E'->5, 'N'->6, 'D'->7, 'M'->1, 'O'->0, 'R'->8, 'Y'->'2'
Such that: "SEND" + "MORE" = "MONEY" , 9567 + 1085 = 10652
Example 2:

Input: words = ["SIX","SEVEN","SEVEN"], result = "TWENTY"
Output: true
Explanation: Map 'S'-> 6, 'I'->5, 'X'->0, 'E'->8, 'V'->7, 'N'->2, 'T'->1, 'W'->'3', 'Y'->4
Such that: "SIX" + "SEVEN" + "SEVEN" = "TWENTY" , 650 + 68782 + 68782 = 138214
Example 3:

Input: words = ["THIS","IS","TOO"], result = "FUNNY"
Output: true

Example 4:

Input: words = ["LEET","CODE"], result = "POINT"
Output: false

Constraints:

2 <= words.length <= 5
1 <= words[i].length, result.length <= 7
words[i], result contain only uppercase English letters.
The number of different characters used in the expression is at most 10.

1309. Decrypt String from Alphabet to Integer Mapping

Given a string s formed by digits ('0' - '9') and '#' . We want to map s to English lowercase characters as follows:

Characters ('a' to 'i') are represented by ('1' to '9') respectively.
Characters ('j' to 'z') are represented by ('10#' to '26#') respectively.

Return the string formed after mapping.
It's guaranteed that a unique mapping will always exist.

Example 1:

Input: s = "10#11#12"
Output: "jkab"
Explanation: "j" -> "10#" , "k" -> "11#" , "a" -> "1" , "b" -> "2".

Example 2:

Input: s = "1326#"
Output: "acz"

Example 3:

Input: s = "25#"
Output: "y"

Example 4:

Input: s = "12345678910#11#12#13#14#15#16#17#18#19#20#21#22#23#24#25#26#"
Output: "abcdefghijklmnopqrstuvwxyz"

Constraints:

1 <= s.length <= 1000
s[i] only contains digits letters ('0'-'9') and '#' letter.
s will be valid string such that mapping is always possible.

1310. XOR Queries of a Subarray

You are given an array arr of positive integers. You are also given the array queries where queries[i] = [lefti, righti].

For each query i compute the XOR of elements from lefti to righti (that is, arr[lefti] XOR arr[lefti + 1] XOR ... XOR arr[righti]).
Return an array answer where answer[i] is the answer to the ith query.

Example 1:

Input: arr = [1,3,4,8], queries = [[0,1],[1,2],[0,3],[3,3]]
Output: [2,7,14,8]
Explanation:
The binary representation of the elements in the array are:
1 = 0001
3 = 0011
4 = 0100
8 = 1000
The XOR values for queries are:
[0,1] = 1 xor 3 = 2
[1,2] = 3 xor 4 = 7
[0,3] = 1 xor 3 xor 4 xor 8 = 14
[3,3] = 8

Example 2:

Input: arr = [4,8,2,10], queries = [[2,3],[1,3],[0,0],[0,3]]
Output: [8,0,4,4]

Constraints:

1 <= arr.length, queries.length <= 3 * 104
1 <= arr[i] <= 109
queries[i].length == 2
0 <= lefti <= righti < arr.length

1311. Get Watched Videos by Your Friends

There are n people, each person has a unique id between 0 and n-1. Given the arrays watchedVideos and friends, wh ere watchedVideos[i] and friends[i] contain the list of watched videos and the list of friends respectively for the pers on with id = i.
Level 1 of videos are all watched videos by your friends, level 2 of videos are all watched videos by the friends of y our friends and so on. In general, the level k of videos are all watched videos by people with the shortest path exactly equal to k with you. Given your id and the level of videos, return the list of videos ordered by their frequencies (incr easing). For videos with the same frequency order them alphabetically from least to greatest.

Example 1:

Input: watchedVideos = [["A","B"],["C"],["B","C"],["D"]], friends = [[1,2],[0,3],[0,3],[1,2]], id = 0, level = 1
Output: ["B","C"]
Explanation:
You have id = 0 (green color in the figure) and your friends are (yellow color in the figure):
Person with id = 1 -> watchedVideos = ["C"]
Person with id = 2 -> watchedVideos = ["B","C"]
The frequencies of watchedVideos by your friends are:
B -> 1
C -> 2

Example 2:

Input: watchedVideos = [["A","B"],["C"],["B","C"],["D"]], friends = [[1,2],[0,3],[0,3],[1,2]], id = 0, level = 2
Output: ["D"]
Explanation:
You have id = 0 (green color in the figure) and the only friend of your friends is the person with id = 3 (yellow color in the figure).

Constraints:

n == watchedVideos.length == friends.length
2 <= n <= 100
1 <= watchedVideos[i].length <= 100
1 <= watchedVideos[i][j].length <= 8
0 <= friends[i].length < n
0 <= friends[i][j] < n
0 <= id < n
1 <= level < n
if friends[i] contains j, then friends[j] contains i

1312. Minimum Insertion Steps to Make a String Palindrome

Given a string s. In one step you can insert any character at any index of the string.
Return the minimum number of steps to make s palindrome.
A Palindrome String is one that reads the same backward as well as forward.

Example 1:

Input: s = "zzazz"
Output: 0
Explanation: The string "zzazz" is already palindrome we don't need any insertions.

Example 2:

Input: s = "mbadm"
Output: 2
Explanation: String can be "mbdadbm" or "mdbabdm".

Example 3:

Input: s = "leetcode"
Output: 5
Explanation: Inserting 5 characters the string becomes "leetcodocteel".

Example 4:

Input: s = "g"
Output: 0

Example 5:

Input: s = "no"
Output: 1

Constraints:

1 <= s.length <= 500
All characters of s are lower case English letters.

1313. Decompress Run-Length Encoded List

We are given a list nums of integers representing a list compressed with run-length encoding.
Consider each adjacent pair of elements [freq, val] = [nums[2*i], nums[2*i+1]] (with i >= 0). For each such pair, th ere are freq elements with value val concatenated in a sublist. Concatenate all the sublists from left to right to genera te the decompressed list.
Return the decompressed list.

Example 1:

Input: nums = [1,2,3,4]
Output: [2,4,4,4]
Explanation: The first pair [1,2] means we have freq = 1 and val = 2 so we generate the array [2].
The second pair [3,4] means we have freq = 3 and val = 4 so we generate [4,4,4].
At the end the concatenation [2] + [4,4,4] is [2,4,4,4].

Example 2:

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

Constraints:

2 <= nums.length <= 100
nums.length % 2 == 0
1 <= nums[i] <= 100

1314. Matrix Block Sum

Given a m x n matrix mat and an integer k, return a matrix answer where each answer[i][j] is the sum of all elements mat[r][c] for:

i - k <= r <= i + k,
j - k <= c <= j + k, and
(r, c) is a valid position in the matrix.

Example 1:

Input: mat = [[1,2,3],[4,5,6],[7,8,9]], k = 1
Output: [[12,21,16],[27,45,33],[24,39,28]]

Example 2:

Input: mat = [[1,2,3],[4,5,6],[7,8,9]], k = 2
Output: [[45,45,45],[45,45,45],[45,45,45]]

Constraints:

m == mat.length
n == mat[i].length
1 <= m, n, k <= 100
1 <= mat[i][j] <= 100

1315. Sum of Nodes with Even-Valued Grandparent

Given the root of a binary tree, return the sum of values of nodes with an even-valued grandparent. If there are no no des with an even-valued grandparent, return 0.
A grandparent of a node is the parent of its parent if it exists.

Example 1:

Input: root = [6,7,8,2,7,1,3,9,null,1,4,null,null,null,5]
Output: 18
Explanation: The red nodes are the nodes with even-value grandparent while the blue nodes are the even-value grand parents.

Example 2:

Input: root = [1]
Output: 0

Constraints:

The number of nodes in the tree is in the range [1, 104].
1 <= Node.val <= 100

1316. Distinct Echo Substrings

Return the number of distinct non-empty substrings of text that can be written as the concatenation of some string wi th itself (i.e. it can be written as a + a where a is some string).

Example 1:

Input: text = "abcabcabc"
Output: 3
Explanation: The 3 substrings are "abcabc", "bcabca" and "cabcab".

Example 2:

Input: text = "leetcodeleetcode"
Output: 2
Explanation: The 2 substrings are "ee" and "leetcodeleetcode".

Constraints:

1 <= text.length <= 2000
text has only lowercase English letters.

1317. Convert Integer to the Sum of Two No-Zero Integers

Given an integer n. No-Zero integer is a positive integer which doesn't contain any 0 in its decimal representation. Return a list of two integers [A, B] where:

A and B are No-Zero integers.
A + B = n

It's guarateed that there is at least one valid solution. If there are many valid solutions you can return any of them.

Example 1:

Input: n = 2
Output: [1,1]
Explanation: A = 1, B = 1. A + B = n and both A and B don't contain any 0 in their decimal representation.

Example 2:

Input: n = 11
Output: [2,9]

Example 3:

Input: n = 10000
Output: [1,9999]

Example 4:

Input: n = 69
Output: [1,68]

Example 5:

Input: n = 1010
Output: [11,999]

Constraints:

2 <= n <= 10^4

1318. Minimum Flips to Make a OR b Equal to c

Given 3 positives numbers a, b and c. Return the minimum flips required in some bits of a and b to make (a OR b = = c). (bitwise OR operation).
Flip operation consists of change any single bit 1 to 0 or change the bit 0 to 1 in their binary representation.

Example 1:

Input: a = 2, b = 6, c = 5
Output: 3
Explanation: After flips a = 1 , b = 4 , c = 5 such that (a OR b == c)
Example 2:

Input: a = 4, b = 2, c = 7
Output: 1

Example 3:

Input: a = 1, b = 2, c = 3
Output: 0

Constraints:

1 <= a <= 10^9
1 <= b <= 10^9
1 <= c <= 10^9

1319. Number of Operations to Make Network Connected

There are n computers numbered from 0 to n-1 connected by ethernet cables connections forming a network where c onnections[i] = [a, b] represents a connection between computers a and b. Any computer can reach any other comput er directly or indirectly through the network.
Given an initial computer network connections. You can extract certain cables between two directly connected comp uters, and place them between any pair of disconnected computers to make them directly connected. Return the mini mum number of times you need to do this in order to make all the computers connected. If it's not possible, return -1.

Example 1:

Input: n = 4, connections = [[0,1],[0,2],[1,2]]
Output: 1

Explanation: Remove cable between computer 1 and 2 and place between computers 1 and 3.

Example 2:

Input: n = 6, connections = [[0,1],[0,2],[0,3],[1,2],[1,3]]
Output: 2

Example 3:

Input: n = 6, connections = [[0,1],[0,2],[0,3],[1,2]]
Output: -1
Explanation: There are not enough cables.

Example 4:

Input: n = 5, connections = [[0,1],[0,2],[3,4],[2,3]]
Output: 0

Constraints:

1 <= n <= 10^5
1 <= connections.length <= min(n*(n-1)/2, 10^5)
connections[i].length == 2
0 <= connections[i][0], connections[i][1] < n
connections[i][0] != connections[i][1]
There are no repeated connections.
No two computers are connected by more than one cable.

1320. Minimum Distance to Type a Word Using Two Fingers

You have a keyboard layout as shown above in the X-Y plane, where each English uppercase letter is located at som e coordinate.

For example, the letter 'A' is located at coordinate (0, 0), the letter 'B' is located at coordinate (0, 1), the letter 'P' is lo cated at coordinate (2, 3) and the letter 'Z' is located at coordinate (4, 1).

Given the string word, return the minimum total distance to type such string using only two fingers.
The distance between coordinates (x1, y1) and (x2, y2) is |x1 - x2| + |y1 - y2|.
Note that the initial positions of your two fingers are considered free so do not count towards your total distance, als o your two fingers do not have to start at the first letter or the first two letters.

Example 1:

Input: word = "CAKE"
Output: 3
Explanation:
Using two fingers, one optimal way to type "CAKE" is:
Finger 1 on letter 'C' -> cost = 0
Finger 1 on letter 'A' -> cost = Distance from letter 'C' to letter 'A' = 2
Finger 2 on letter 'K' -> cost = 0
Finger 2 on letter 'E' -> cost = Distance from letter 'K' to letter 'E' = 1
Total distance = 3

Example 2:

Input: word = "HAPPY"
Output: 6
Explanation:
Using two fingers, one optimal way to type "HAPPY" is:
Finger 1 on letter 'H' -> cost = 0
Finger 1 on letter 'A' -> cost = Distance from letter 'H' to letter 'A' = 2
Finger 2 on letter 'P' -> cost = 0
Finger 2 on letter 'P' -> cost = Distance from letter 'P' to letter 'P' = 0
Finger 1 on letter 'Y' -> cost = Distance from letter 'A' to letter 'Y' = 4
Total distance = 6

Example 3:

Input: word = "NEW"
Output: 3

Example 4:

Input: word = "YEAR"
Output: 7

Constraints:

2 <= word.length <= 300
word consists of uppercase English letters.

1323. Maximum 69 Number

Given a positive integer num consisting only of digits 6 and 9.
Return the maximum number you can get by changing at most one digit (6 becomes 9, and 9 becomes 6).

Example 1:

Input: num = 9669
Output: 9969
Explanation:
Changing the first digit results in 6669.
Changing the second digit results in 9969.
Changing the third digit results in 9699.
Changing the fourth digit results in 9666.
The maximum number is 9969.

Example 2:

Input: num = 9996
Output: 9999
Explanation: Changing the last digit 6 to 9 results in the maximum number.
Example 3:

Input: num = 9999
Output: 9999
Explanation: It is better not to apply any change.

Constraints:

1 <= num <= 10^4
num's digits are 6 or 9.

1324. Print Words Vertically

Given a string s. Return all the words vertically in the same order in which they appear in s.
Words are returned as a list of strings, complete with spaces when is necessary. (Trailing spaces are not allowed).
Each word would be put on only one column and that in one column there will be only one word.

Example 1:

Input: s = "HOW ARE YOU"
Output: ["HAY","ORO","WEU"]
Explanation: Each word is printed vertically.
 "HAY"
 "ORO"
 "WEU"

Example 2:

Input: s = "TO BE OR NOT TO BE"
Output: ["TBONTB","OEROOE","   T"]
Explanation: Trailing spaces is not allowed.
"TBONTB"
"OEROOE"

"   T"

Example 3:

Input: s = "CONTEST IS COMING"
Output: ["CIC","OSO","N M","T I","E N","S G","T"]

Constraints:

1 <= s.length <= 200
s contains only upper case English letters.
It's guaranteed that there is only one space between 2 words.

1325. Delete Leaves With a Given Value

Given a binary tree root and an integer target, delete all the leaf nodes with value target.
Note that once you delete a leaf node with value target, if it's parent node becomes a leaf node and has the value targ et, it should also be deleted (you need to continue doing that until you can't).

Example 1:

Input: root = [1,2,3,2,null,2,4], target = 2
Output: [1,null,3,null,4]
Explanation: Leaf nodes in green with value (target = 2) are removed (Picture in left).
After removing, new nodes become leaf nodes with value (target = 2) (Picture in center).

Example 2:

Input: root = [1,3,3,3,2], target = 3
Output: [1,3,null,null,2]

Example 3:

Input: root = [1,2,null,2,null,2], target = 2
Output: [1]
Explanation: Leaf nodes in green with value (target = 2) are removed at each step.

Example 4:

Input: root = [1,1,1], target = 1
Output: []

Example 5:

Input: root = [1,2,3], target = 1
Output: [1,2,3]

Constraints:

1 <= target <= 1000
The given binary tree will have between 1 and 3000 nodes.
Each node's value is between [1, 1000].

1326. Minimum Number of Taps to Open to Water a Garden

There is a one-dimensional garden on the x-axis. The garden starts at the point 0 and ends at the point n. (i.e The len gth of the garden is n).
There are n + 1 taps located at points [0, 1, ..., n] in the garden.
Given an integer n and an integer array ranges of length n + 1 where ranges[i] (0-indexed) means the i-th tap can wat er the area [i - ranges[i], i + ranges[i]] if it was open.
Return the minimum number of taps that should be open to water the whole garden, If the garden cannot be watered return -1.

Example 1:

Input: n = 5, ranges = [3,4,1,1,0,0]
Output: 1
Explanation: The tap at point 0 can cover the interval [-3,3]
The tap at point 1 can cover the interval [-3,5]
The tap at point 2 can cover the interval [1,3]
The tap at point 3 can cover the interval [2,4]
The tap at point 4 can cover the interval [4,4]
The tap at point 5 can cover the interval [5,5]
Opening Only the second tap will water the whole garden [0,5]

Example 2:

Input: n = 3, ranges = [0,0,0,0]
Output: -1
Explanation: Even if you activate all the four taps you cannot water the whole garden.

Example 3:

Input: n = 7, ranges = [1,2,1,0,2,1,0,1]
Output: 3

Example 4:

Input: n = 8, ranges = [4,0,0,0,0,0,0,0,4]
Output: 2

Example 5:

Input: n = 8, ranges = [4,0,0,0,4,0,0,0,4]
Output: 1

Constraints:

1 <= n <= 104
ranges.length == n + 1
0 <= ranges[i] <= 100

1328. Break a Palindrome

Given a palindromic string of lowercase English letters palindrome, replace exactly one character with any lowercas e English letter so that the resulting string is not a palindrome and that it is the lexicographically smallest one possibl e.
Return the resulting string. If there is no way to replace a character to make it not a palindrome, return an empty strin g.
A string a is lexicographically smaller than a string b (of the same length) if in the first position where a and b differ, a has a character strictly smaller than the corresponding character in b. For example, "abcc" is lexicographically sma ller than "abcd" because the first position they differ is at the fourth character, and 'c' is smaller than 'd'.

Example 1:

Input: palindrome = "abccba"
Output: "aaccba"
Explanation: There are many ways to make "abccba" not a palindrome, such as "zbccba", "aaccba", and "abacba".
Of all the ways, "aaccba" is the lexicographically smallest.

Example 2:

Input: palindrome = "a"
Output: ""
Explanation: There is no way to replace a single character to make "a" not a palindrome, so return an empty string.

Example 3:

Input: palindrome = "aa"
Output: "ab"
Example 4:

Input: palindrome = "aba"
Output: "abb"

Constraints:

1 <= palindrome.length <= 1000
palindrome consists of only lowercase English letters.

1329. Sort the Matrix Diagonally

A matrix diagonal is a diagonal line of cells starting from some cell in either the topmost row or leftmost column and going in the bottom-right direction until reaching the matrix's end. For example, the matrix diagonal starting from mat[2][0], where mat is a 6 x 3 matrix, includes cells mat[2][0], mat[3][1], and mat[4][2].
Given an m x n matrix mat of integers, sort each matrix diagonal in ascending order and return the resulting matrix.

Example 1:

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

Example 2:

Input: mat = [[11,25,66,1,69,7],[23,55,17,45,15,52],[75,31,36,44,58,8],[22,27,33,25,68,4],[84,28,14,11,5,50]]
Output: [[5,17,4,1,52,7],[11,11,25,45,8,69],[14,23,25,44,58,15],[22,27,31,36,50,66],[84,28,75,33,55,68]]

Constraints:

m == mat.length
n == mat[i].length
1 <= m, n <= 100
1 <= mat[i][j] <= 100

1330. Reverse Subarray To Maximize Array Value

You are given an integer array nums. The value of this array is defined as the sum of |nums[i] - nums[i + 1]| for all 0

<= i < nums.length - 1.
You are allowed to select any subarray of the given array and reverse it. You can perform this operation only once.
Find maximum possible value of the final array.

Example 1:

Input: nums = [2,3,1,5,4]
Output: 10
Explanation: By reversing the subarray [3,1,5] the array becomes [2,5,1,3,4] whose value is 10.

Example 2:

Input: nums = [2,4,9,24,2,1,10]
Output: 68

Constraints:

1 <= nums.length <= 3 * 104
-105 <= nums[i] <= 105

1331. Rank Transform of an Array

Given an array of integers arr, replace each element with its rank.
The rank represents how large the element is. The rank has the following rules:

Rank is an integer starting from 1.
The larger the element, the larger the rank. If two elements are equal, their rank must be the same.
Rank should be as small as possible.

Example 1:

Input: arr = [40,10,20,30]
Output: [4,1,2,3]
Explanation: 40 is the largest element. 10 is the smallest. 20 is the second smallest. 30 is the third smallest.
Example 2:

Input: arr = [100,100,100]
Output: [1,1,1]
Explanation: Same elements share the same rank.

Example 3:

Input: arr = [37,12,28,9,100,56,80,5,12]
Output: [5,3,4,2,8,6,7,1,3]

Constraints:

0 <= arr.length <= 105
-109 <= arr[i] <= 109

1332. Remove Palindromic Subsequences

You are given a string s consisting only of letters 'a' and 'b'. In a single step you can remove one palindromic subseq uence from s.
Return the minimum number of steps to make the given string empty.
A string is a subsequence of a given string if it is generated by deleting some characters of a given string without cha nging its order. Note that a subsequence does not necessarily need to be contiguous.
A string is called palindrome if is one that reads the same backward as well as forward.

Example 1:

Input: s = "ababa"
Output: 1
Explanation: s is already a palindrome, so its entirety can be removed in a single step.

Example 2:

Input: s = "abb"
Output: 2
Explanation: "abb" -> "bb" -> "".
Remove palindromic subsequence "a" then "bb".

Example 3:

Input: s = "baabb"
Output: 2
Explanation: "baabb" -> "b" -> "".
Remove palindromic subsequence "baab" then "b".

Constraints:

1 <= s.length <= 1000
s[i] is either 'a' or 'b'.

1333. Filter Restaurants by Vegan-Friendly, Price and Distance

Given the array restaurants where restaurants[i] = [idi, ratingi, veganFriendlyi, pricei, distancei]. You have to filter t he restaurants using three filters.
The veganFriendly filter will be either true (meaning you should only include restaurants with veganFriendlyi set to t rue) or false (meaning you can include any restaurant). In addition, you have the filters maxPrice and maxDistance w hich are the maximum value for price and distance of restaurants you should consider respectively.
Return the array of restaurant IDs after filtering, ordered by rating from highest to lowest. For restaurants with the sa me rating, order them by id from highest to lowest. For simplicity veganFriendlyi and veganFriendly take value 1 w hen it is true, and 0 when it is false.

Example 1:

Input: restaurants = [[1,4,1,40,10],[2,8,0,50,5],[3,8,1,30,4],[4,10,0,10,3],[5,1,1,15,1]], veganFriendly = 1, maxPrice = 50, maxDistance = 10
Output: [3,1,5]
Explanation:
The restaurants are:
Restaurant 1 [id=1, rating=4, veganFriendly=1, price=40, distance=10]
Restaurant 2 [id=2, rating=8, veganFriendly=0, price=50, distance=5]
Restaurant 3 [id=3, rating=8, veganFriendly=1, price=30, distance=4]
Restaurant 4 [id=4, rating=10, veganFriendly=0, price=10, distance=3]
Restaurant 5 [id=5, rating=1, veganFriendly=1, price=15, distance=1]
After filter restaurants with veganFriendly = 1, maxPrice = 50 and maxDistance = 10 we have restaurant 3, restauran t 1 and restaurant 5 (ordered by rating from highest to lowest).

Example 2:

Input: restaurants = [[1,4,1,40,10],[2,8,0,50,5],[3,8,1,30,4],[4,10,0,10,3],[5,1,1,15,1]], veganFriendly = 0, maxPrice = 50, maxDistance = 10
Output: [4,3,2,1,5]
Explanation: The restaurants are the same as in example 1, but in this case the filter veganFriendly = 0, therefore all restaurants are considered.

Example 3:

Input: restaurants = [[1,4,1,40,10],[2,8,0,50,5],[3,8,1,30,4],[4,10,0,10,3],[5,1,1,15,1]], veganFriendly = 0, maxPrice = 30, maxDistance = 3
Output: [4,5]

Constraints:

1 <= restaurants.length <= 10^4
restaurants[i].length == 5
1 <= idi, ratingi, pricei, distancei <= 10^5
1 <= maxPrice, maxDistance <= 10^5
veganFriendlyi and veganFriendly are 0 or 1.
All idi are distinct.

1334. Find the City With the Smallest Number of Neighbors at a Threshold Distance

There are n cities numbered from 0 to n-1. Given the array edges where edges[i] = [fromi, toi, weighti] represents a bidirectional and weighted edge between cities fromi and toi, and given the integer distanceThreshold.
Return the city with the smallest number of cities that are reachable through some path and whose distance is at most distanceThreshold, If there are multiple such cities, return the city with the greatest number.
Notice that the distance of a path connecting cities i and j is equal to the sum of the edges' weights along that path.

Example 1:

Input: n = 4, edges = [[0,1,3],[1,2,1],[1,3,4],[2,3,1]], distanceThreshold = 4
Output: 3
Explanation: The figure above describes the graph.
The neighboring cities at a distanceThreshold = 4 for each city are:
City 0 -> [City 1, City 2]
City 1 -> [City 0, City 2, City 3]
City 2 -> [City 0, City 1, City 3]
City 3 -> [City 1, City 2]
Cities 0 and 3 have 2 neighboring cities at a distanceThreshold = 4, but we have to return city 3 since it has the great est number.

Example 2:

Input: n = 5, edges = [[0,1,2],[0,4,8],[1,2,3],[1,4,2],[2,3,1],[3,4,1]], distanceThreshold = 2
Output: 0
Explanation: The figure above describes the graph.
The neighboring cities at a distanceThreshold = 2 for each city are:
City 0 -> [City 1]
City 1 -> [City 0, City 4]
City 2 -> [City 3, City 4]
City 3 -> [City 2, City 4]
City 4 -> [City 1, City 2, City 3]
The city 0 has 1 neighboring city at a distanceThreshold = 2.

Constraints:

2 <= n <= 100
1 <= edges.length <= n * (n - 1) / 2
edges[i].length == 3
0 <= fromi < toi < n
1 <= weighti, distanceThreshold <= 10^4
All pairs (fromi, toi) are distinct.

1335. Minimum Difficulty of a Job Schedule

You want to schedule a list of jobs in d days. Jobs are dependent (i.e To work on the i-th job, you have to finish all t he jobs j where 0 <= j < i).
You have to finish at least one task every day. The difficulty of a job schedule is the sum of difficulties of each day o f the d days. The difficulty of a day is the maximum difficulty of a job done in that day.
Given an array of integers jobDifficulty and an integer d. The difficulty of the i-th job is jobDifficulty[i].
Return the minimum difficulty of a job schedule. If you cannot find a schedule for the jobs return -1.

Example 1:

Input: jobDifficulty = [6,5,4,3,2,1], d = 2
Output: 7
Explanation: First day you can finish the first 5 jobs, total difficulty = 6.
Second day you can finish the last job, total difficulty = 1.
The difficulty of the schedule = 6 + 1 = 7

Example 2:

Input: jobDifficulty = [9,9,9], d = 4
Output: -1
Explanation: If you finish a job per day you will still have a free day. you cannot find a schedule for the given jobs.

Example 3:

Input: jobDifficulty = [1,1,1], d = 3
Output: 3
Explanation: The schedule is one job per day. total difficulty will be 3.

Example 4:

Input: jobDifficulty = [7,1,7,1,7,1], d = 3
Output: 15

Example 5:

Input: jobDifficulty = [11,111,22,222,33,333,44,444], d = 6
Output: 843

Constraints:

1 <= jobDifficulty.length <= 300
0 <= jobDifficulty[i] <= 1000
1 <= d <= 10

1337. The K Weakest Rows in a Matrix

You are given an m x n binary matrix mat of 1's (representing soldiers) and 0's (representing civilians). The soldiers are positioned in front of the civilians. That is, all the 1's will appear to the left of all the 0's in each row.
A row i is weaker than a row j if one of the following is true:

The number of soldiers in row i is less than the number of soldiers in row j.
Both rows have the same number of soldiers and i < j.

Return the indices of the k weakest rows in the matrix ordered from weakest to strongest.

Example 1:

Input: mat =
[[1,1,0,0,0],
 [1,1,1,1,0],
 [1,0,0,0,0],
 [1,1,0,0,0],
 [1,1,1,1,1]],
k = 3
Output: [2,0,3]
Explanation:
The number of soldiers in each row is:
- Row 0: 2
- Row 1: 4
- Row 2: 1
- Row 3: 2
- Row 4: 5
The rows ordered from weakest to strongest are [2,0,3,1,4].

Example 2:

Input: mat =
[[1,0,0,0],
 [1,1,1,1],
 [1,0,0,0],
 [1,0,0,0]],
k = 2
Output: [0,2]
Explanation:
The number of soldiers in each row is:
- Row 0: 1
- Row 1: 4
- Row 2: 1
- Row 3: 1

The rows ordered from weakest to strongest are [0,2,3,1].

Constraints:

m == mat.length
n == mat[i].length
2 <= n, m <= 100
1 <= k <= m
matrix[i][j] is either 0 or 1.

1338. Reduce Array Size to The Half

You are given an integer array arr. You can choose a set of integers and remove all the occurrences of these integers in the array.
Return the minimum size of the set so that at least half of the integers of the array are removed.

Example 1:

Input: arr = [3,3,3,3,5,5,5,2,2,7]
Output: 2
Explanation: Choosing {3,7} will make the new array [5,5,5,2,2] which has size 5 (i.e equal to half of the size of the old array).
Possible sets of size 2 are {3,5},{3,2},{5,2}.
Choosing set {2,7} is not possible as it will make the new array [3,3,3,3,5,5,5] which has size greater than half of the size of the old array.

Example 2:

Input: arr = [7,7,7,7,7,7]
Output: 1
Explanation: The only possible set you can choose is {7}. This will make the new array empty.

Example 3:

Input: arr = [1,9]
Output: 1

Example 4:

Input: arr = [1000,1000,3,7]
Output: 1

Example 5:

Input: arr = [1,2,3,4,5,6,7,8,9,10]

Output: 5

Constraints:

1 <= arr.length <= 105
arr.length is even.
1 <= arr[i] <= 105

1339. Maximum Product of Splitted Binary Tree

Given the root of a binary tree, split the binary tree into two subtrees by removing one edge such that the product of t he sums of the subtrees is maximized.
Return the maximum product of the sums of the two subtrees. Since the answer may be too large, return it modulo 1 09 + 7.
Note that you need to maximize the answer before taking the mod and not after taking it.

Example 1:

Input: root = [1,2,3,4,5,6]
Output: 110
Explanation: Remove the red edge and get 2 binary trees with sum 11 and 10. Their product is 110 (11*10)

Example 2:

Input: root = [1,null,2,3,4,null,null,5,6]
Output: 90
Explanation: Remove the red edge and get 2 binary trees with sum 15 and 6.Their product is 90 (15*6)

Example 3:

Input: root = [2,3,9,10,7,8,6,5,4,11,1]
Output: 1025

Example 4:

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

Constraints:

The number of nodes in the tree is in the range [2, 5 * 104].
1 <= Node.val <= 104

1340. Jump Game V

Given an array of integers arr and an integer d. In one step you can jump from index i to index:

i + x where: i + x < arr.length and 0 < x <= d.
i - x where: i - x >= 0 and 0 < x <= d.

In addition, you can only jump from index i to index j if arr[i] > arr[j] and arr[i] > arr[k] for all indices k between i a nd j (More formally min(i, j) < k < max(i, j)).
You can choose any index of the array and start jumping. Return the maximum number of indices you can visit.
Notice that you can not jump outside of the array at any time.

Example 1:

Input: arr = [6,4,14,6,8,13,9,7,10,6,12], d = 2
Output: 4
Explanation: You can start at index 10. You can jump 10 --> 8 --> 6 --> 7 as shown.
Note that if you start at index 6 you can only jump to index 7. You cannot jump to index 5 because 13 > 9. You cann ot jump to index 4 because index 5 is between index 4 and 6 and 13 > 9.
Similarly You cannot jump from index 3 to index 2 or index 1.

Example 2:

Input: arr = [3,3,3,3,3], d = 3
Output: 1
Explanation: You can start at any index. You always cannot jump to any index.

Example 3:

Input: arr = [7,6,5,4,3,2,1], d = 1
Output: 7
Explanation: Start at index 0. You can visit all the indicies.

Example 4:

Input: arr = [7,1,7,1,7,1], d = 2
Output: 2

Example 5:

Input: arr = [66], d = 1
Output: 1

Constraints:

1 <= arr.length <= 1000
1 <= arr[i] <= 10^5
1 <= d <= arr.length

1342. Number of Steps to Reduce a Number to Zero

Given an integer num, return the number of steps to reduce it to zero.
In one step, if the current number is even, you have to divide it by 2, otherwise, you have to subtract 1 from it.

Example 1:

Input: num = 14
Output: 6
Explanation:
Step 1) 14 is even; divide by 2 and obtain 7.
Step 2) 7 is odd; subtract 1 and obtain 6.
Step 3) 6 is even; divide by 2 and obtain 3.
Step 4) 3 is odd; subtract 1 and obtain 2.
Step 5) 2 is even; divide by 2 and obtain 1.
Step 6) 1 is odd; subtract 1 and obtain 0.

Example 2:

Input: num = 8
Output: 4
Explanation:
Step 1) 8 is even; divide by 2 and obtain 4.
Step 2) 4 is even; divide by 2 and obtain 2.
Step 3) 2 is even; divide by 2 and obtain 1.
Step 4) 1 is odd; subtract 1 and obtain 0.

Example 3:

Input: num = 123
Output: 12

Constraints:

0 <= num <= 106

1343. Number of Sub-arrays of Size K and Average Greater than or Equal to Threshold

Given an array of integers arr and two integers k and threshold.
Return the number of sub-arrays of size k and average greater than or equal to threshold.

Example 1:

Input: arr = [2,2,2,2,5,5,5,8], k = 3, threshold = 4
Output: 3
Explanation: Sub-arrays [2,5,5],[5,5,5] and [5,5,8] have averages 4, 5 and 6 respectively. All other sub-arrays of size 3 have averages less than 4 (the threshold).

Example 2:

Input: arr = [1,1,1,1,1], k = 1, threshold = 0
Output: 5

Example 3:

Input: arr = [11,13,17,23,29,31,7,5,2,3], k = 3, threshold = 5
Output: 6
Explanation: The first 6 sub-arrays of size 3 have averages greater than 5. Note that averages are not integers.

Example 4:

Input: arr = [7,7,7,7,7,7,7], k = 7, threshold = 7
Output: 1

Example 5:

Input: arr = [4,4,4,4], k = 4, threshold = 1
Output: 1

Constraints:

1 <= arr.length <= 10^5
1 <= arr[i] <= 10^4
1 <= k <= arr.length
0 <= threshold <= 10^4

1344. Angle Between Hands of a Clock

Given two numbers, hour and minutes. Return the smaller angle (in degrees) formed between the hour and the minut e hand.

Example 1:

Input: hour = 12, minutes = 30
Output: 165

Example 2:

Input: hour = 3, minutes = 30
Output: 75

Example 3:

Input: hour = 3, minutes = 15
Output: 7.5

Example 4:

Input: hour = 4, minutes = 50
Output: 155

Example 5:

Input: hour = 12, minutes = 0
Output: 0

Constraints:

1 <= hour <= 12
0 <= minutes <= 59
Answers within 10^-5 of the actual value will be accepted as correct.

1345. Jump Game IV

Given an array of integers arr, you are initially positioned at the first index of the array.
In one step you can jump from index i to index:

i + 1 where: i + 1 < arr.length.
i - 1 where: i - 1 >= 0.

j where: arr[i] == arr[j] and i != j.

Return the minimum number of steps to reach the last index of the array.
Notice that you can not jump outside of the array at any time.

Example 1:

Input: arr = [100,-23,-23,404,100,23,23,23,3,404]
Output: 3
Explanation: You need three jumps from index 0 --> 4 --> 3 --> 9. Note that index 9 is the last index of the array.

Example 2:

Input: arr = [7]
Output: 0
Explanation: Start index is the last index. You don't need to jump.

Example 3:

Input: arr = [7,6,9,6,9,6,9,7]
Output: 1
Explanation: You can jump directly from index 0 to index 7 which is last index of the array.

Example 4:

Input: arr = [6,1,9]
Output: 2

Example 5:

Input: arr = [11,22,7,7,7,7,7,7,7,22,13]
Output: 3

Constraints:

1 <= arr.length <= 5 * 104
-108 <= arr[i] <= 108

1346. Check If N and Its Double Exist

Given an array arr of integers, check if there exists two integers N and M such that N is the double of M (i.e. N = 2 * M).
More formally check if there exists two indices i and j such that :

i != j

0 <= i, j < arr.length
arr[i] == 2 * arr[j]

Example 1:

Input: arr = [10,2,5,3]
Output: true
Explanation: N = 10 is the double of M = 5,that is, 10 = 2 * 5.

Example 2:

Input: arr = [7,1,14,11]
Output: true
Explanation: N = 14 is the double of M = 7,that is, 14 = 2 * 7.

Example 3:

Input: arr = [3,1,7,11]
Output: false
Explanation: In this case does not exist N and M, such that N = 2 * M.

Constraints:

2 <= arr.length <= 500
-10^3 <= arr[i] <= 10^3

1347. Minimum Number of Steps to Make Two Strings Anagram

Given two equal-size strings s and t. In one step you can choose any character of t and replace it with another charact
er.
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.

Example 4:

Input: s = "xxyyzz", t = "xxyyzz"
Output: 0

Example 5:

Input: s = "friend", t = "family"
Output: 4

Constraints:

1 <= s.length <= 50000
s.length == t.length
s and t contain lower-case English letters only.

1348. Tweet Counts Per Frequency

A social media company is trying to monitor activity on their site by analyzing the number of tweets that occur in sel ect periods of time. These periods can be partitioned into smaller time chunks based on a certain frequency (every mi nute, hour, or day).
For example, the period [10, 10000] (in seconds) would be partitioned into the following time chunks with these freq uencies:

Every minute (60-second chunks): [10,69], [70,129], [130,189], ..., [9970,10000]
Every hour (3600-second chunks): [10,3609], [3610,7209], [7210,10000]
Every day (86400-second chunks): [10,10000]

Notice that the last chunk may be shorter than the specified frequency's chunk size and will always end with the end time of the period (10000 in the above example).
Design and implement an API to help the company with their analysis.
Implement the TweetCounts class:

TweetCounts() Initializes the TweetCounts object.
void recordTweet(String tweetName, int time) Stores the tweetName at the recorded time (in seconds).
List<Integer> getTweetCountsPerFrequency(String freq, String tweetName, int startTime, int endTime) Returns a lis t of integers representing the number of tweets with tweetName in each time chunk for the given period of time [start Time, endTime] (in seconds) and frequency freq.

freq is one of "minute", "hour", or "day" representing a frequency of every minute, hour, or day respectively.

Example:

Input
["TweetCounts","recordTweet","recordTweet","recordTweet","getTweetCountsPerFrequency","getTweetCountsPerFrequency","recordTweet","getTweetCountsPerFrequency"]
[[],["tweet3",0],["tweet3",60],["tweet3",10],["minute","tweet3",0,59],["minute","tweet3",0,60],["tweet3",120],["hour","tweet3",0,210]]

Output
[null,null,null,null,[2],[2,1],null,[4]]

Explanation
TweetCounts tweetCounts = new TweetCounts();
tweetCounts.recordTweet("tweet3", 0);   // New tweet "tweet3" at time 0
tweetCounts.recordTweet("tweet3", 60);   // New tweet "tweet3" at time 60
tweetCounts.recordTweet("tweet3", 10);   // New tweet "tweet3" at time 10
tweetCounts.getTweetCountsPerFrequency("minute", "tweet3", 0, 59); // return [2]; chunk [0,59] had 2 tweets
tweetCounts.getTweetCountsPerFrequency("minute", "tweet3", 0, 60); // return [2,1]; chunk [0,59] had 2 tweets, chunk [60,60] had 1 tweet
tweetCounts.recordTweet("tweet3", 120);   // New tweet "tweet3" at time 120
tweetCounts.getTweetCountsPerFrequency("hour", "tweet3", 0, 210); // return [4]; chunk [0,210] had 4 tweets

Constraints:

0 <= time, startTime, endTime <= 109
0 <= endTime - startTime <= 104
There will be at most 104 calls in total to recordTweet and getTweetCountsPerFrequency.

1349. Maximum Students Taking Exam

Given a m * n matrix seats that represent seats distributions in a classroom. If a seat is broken, it is denoted by '#' character otherwise it is denoted by a '.' character.
Students can see the answers of those sitting next to the left, right, upper left and upper right, but he cannot see the answers of the student sitting directly in front or behind him. Return the maximum number of students that can take the exam together without any cheating being possible..
Students must be placed in seats in good condition.

Example 1:

Input: seats = [["#",".","#","#",".","#"],
[".","#","#","#","#","."],
["#",".","#","#",".","#"]]
Output: 4
Explanation: Teacher can place 4 students in available seats so they don't cheat on the exam.

Example 2:

Input: seats = [[".","#"],
["#","#"],
["#","."],
["#","#"],
[".","#"]]
Output: 3
Explanation: Place all students in available seats.

Example 3:

Input: seats = [["#",".",".",".",".","#"],
[".","#",".",".","#","."],
[".",".","#",".",".","."],
[".","#",".",".","#","."],
["#",".",".",".",".","#"]]
Output: 10
Explanation: Place students in available seats in column 1, 3 and 5.

Constraints:

seats contains only characters '.' and'#'.
m == seats.length
n == seats[i].length
1 <= m <= 8
1 <= n <= 8

1351. Count Negative Numbers in a Sorted Matrix

Given a m x n matrix grid which is sorted in non-increasing order both row-wise and column-wise, return the numbe r of negative numbers in grid.

Example 1:

Input: grid = [[4,3,2,-1],[3,2,1,-1],[1,1,-1,-2],[-1,-1,-2,-3]]
Output: 8

Explanation: There are 8 negatives number in the matrix.

Example 2:

Input: grid = [[3,2],[1,0]]
Output: 0

Example 3:

Input: grid = [[1,-1],[-1,-1]]
Output: 3

Example 4:

Input: grid = [[-1]]
Output: 1

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 100
-100 <= grid[i][j] <= 100

Follow up: Could you find an O(n + m) solution?

1352. Product of the Last K Numbers

Design an algorithm that accepts a stream of integers and retrieves the product of the last k integers of the stream.
Implement the ProductOfNumbers class:

ProductOfNumbers() Initializes the object with an empty stream.
void add(int num) Appends the integer num to the stream.
int getProduct(int k) Returns the product of the last k numbers in the current list. You can assume that always the cur rent list has at least k numbers.

The test cases are generated so that, at any time, the product of any contiguous sequence of numbers will fit into a si ngle 32-bit integer without overflowing.

Example:

Input
["ProductOfNumbers","add","add","add","add","add","getProduct","getProduct","getProduct","add","getProduct"]
[[],[3],[0],[2],[5],[4],[2],[3],[4],[8],[2]]

Output

[null,null,null,null,null,null,20,40,0,null,32]

Explanation
ProductOfNumbers productOfNumbers = new ProductOfNumbers();
productOfNumbers.add(3);   // [3]
productOfNumbers.add(0);   // [3,0]
productOfNumbers.add(2);   // [3,0,2]
productOfNumbers.add(5);   // [3,0,2,5]
productOfNumbers.add(4);   // [3,0,2,5,4]
productOfNumbers.getProduct(2); // return 20. The product of the last 2 numbers is 5 * 4 = 20
productOfNumbers.getProduct(3); // return 40. The product of the last 3 numbers is 2 * 5 * 4 = 40
productOfNumbers.getProduct(4); // return 0. The product of the last 4 numbers is 0 * 2 * 5 * 4 = 0
productOfNumbers.add(8);   // [3,0,2,5,4,8]
productOfNumbers.getProduct(2); // return 32. The product of the last 2 numbers is 4 * 8 = 32

Constraints:

0 <= num <= 100
1 <= k <= 4 * 104
At most 4 * 104 calls will be made to add and getProduct.
The product of the stream at any point in time will fit in a 32-bit integer.

1353. Maximum Number of Events That Can Be Attended

Given an array of events where events[i] = [startDayi, endDayi]. Every event i starts at startDayi and ends at endDay
i.
You can attend an event i at any day d where startTimei <= d <= endTimei. Notice that you can only attend one even
t at any time d.
Return the maximum number of events you can attend.

Example 1:

Input: events = [[1,2],[2,3],[3,4]]
Output: 3
Explanation: You can attend all the three events.
One way to attend them all is as shown.
Attend the first event on day 1.
Attend the second event on day 2.
Attend the third event on day 3.

Example 2:

Input: events= [[1,2],[2,3],[3,4],[1,2]]
Output: 4

Example 3:

Input: events = [[1,4],[4,4],[2,2],[3,4],[1,1]]
Output: 4

Example 4:

Input: events = [[1,100000]]
Output: 1

Example 5:

Input: events = [[1,1],[1,2],[1,3],[1,4],[1,5],[1,6],[1,7]]
Output: 7

Constraints:

1 <= events.length <= 105
events[i].length == 2
1 <= startDayi <= endDayi <= 105

1354. Construct Target Array With Multiple Sums

You are given an array target of n integers. From a starting array arr consisting of n 1's, you may perform the followi
ng procedure :

let x be the sum of all elements currently in your array.
choose index i, such that 0 <= i < n and set the value of arr at index i to x.
You may repeat this procedure as many times as needed.

Return true if it is possible to construct the target array from arr, otherwise, return false.

Example 1:

Input: target = [9,3,5]
Output: true
Explanation: Start with arr = [1, 1, 1]
[1, 1, 1], sum = 3 choose index 1
[1, 3, 1], sum = 5 choose index 2
[1, 3, 5], sum = 9 choose index 0
[9, 3, 5] Done

Example 2:

Input: target = [1,1,1,2]
Output: false
Explanation: Impossible to create target array from [1,1,1,1].

Example 3:

Input: target = [8,5]
Output: true

Constraints:

n == target.length
1 <= n <= 5 * 104
1 <= target[i] <= 109

1356. Sort Integers by The Number of 1 Bits

Given an integer array arr. You have to sort the integers in the array in ascending order by the number of 1's in their binary representation and in case of two or more integers have the same number of 1's you have to sort them in ascen ding order.
Return the sorted array.

Example 1:

Input: arr = [0,1,2,3,4,5,6,7,8]
Output: [0,1,2,4,8,3,5,6,7]
Explantion: [0] is the only integer with 0 bits.
[1,2,4,8] all have 1 bit.
[3,5,6] have 2 bits.
[7] has 3 bits.
The sorted array by bits is [0,1,2,4,8,3,5,6,7]

Example 2:

Input: arr = [1024,512,256,128,64,32,16,8,4,2,1]
Output: [1,2,4,8,16,32,64,128,256,512,1024]
Explantion: All integers have 1 bit in the binary representation, you should just sort them in ascending order.

Example 3:

Input: arr = [10000,10000]
Output: [10000,10000]

Example 4:

Input: arr = [2,3,5,7,11,13,17,19]
Output: [2,3,5,17,7,11,13,19]

Example 5:

Input: arr = [10,100,1000,10000]
Output: [10,100,10000,1000]

Constraints:

1 <= arr.length <= 500
0 <= arr[i] <= 10^4

1357. Apply Discount Every n Orders

There is a supermarket that is frequented by many customers. The products sold at the supermarket are represented a s two parallel integer arrays products and prices, where the ith product has an ID of products[i] and a price of prices[i].
When a customer is paying, their bill is represented as two parallel integer arrays product and amount, where the jth product they purchased has an ID of product[j], and amount[j] is how much of the product they bought. Their subtot al is calculated as the sum of each amount[j] * (price of the jth product).
The supermarket decided to have a sale. Every nth customer paying for their groceries will be given a percentage dis count. The discount amount is given by discount, where they will be given discount percent off their subtotal. More f ormally, if their subtotal is bill, then they would actually pay bill * ((100 - discount) / 100).
Implement the Cashier class:

Cashier(int n, int discount, int[] products, int[] prices) Initializes the object with n, the discount, and the products and their prices.
double getBill(int[] product, int[] amount) Returns the final total of the bill with the discount applied (if any). Answe rs within 10-5 of the actual value will be accepted.

Example 1:

Input
["Cashier","getBill","getBill","getBill","getBill","getBill","getBill","getBill"]
[[3,50,[1,2,3,4,5,6,7],[100,200,300,400,300,200,100]],[[1,2],[1,2]],[[3,7],[10,10]],[[1,2,3,4,5,6,7],[1,1,1,1,1,1,1]],[[4] ,[10]],[[7,3],[10,10]],[[7,5,3,1,6,4,2],[10,10,10,9,9,9,7]],[[2,3,5],[5,3,2]]]
Output
[null,500.0,4000.0,800.0,4000.0,4000.0,7350.0,2500.0]
Explanation
Cashier cashier = new Cashier(3,50,[1,2,3,4,5,6,7],[100,200,300,400,300,200,100]);
cashier.getBill([1,2],[1,2]); // return 500.0. 1st customer, no discount.
// bill = 1 * 100 + 2 * 200 = 500.
cashier.getBill([3,7],[10,10]); // return 4000.0. 2nd customer, no discount.

```
                                        // bill = 10 * 300 + 10 * 100 = 4000.
cashier.getBill([1,2,3,4,5,6,7],[1,1,1,1,1,1,1]);    // return 800.0. 3rd customer, 50% discount.
                                        // Original bill = 1600
                                        // Actual bill = 1600 * ((100 - 50) / 100) = 800.
cashier.getBill([4],[10]);                          // return 4000.0. 4th customer, no discount.
cashier.getBill([7,3],[10,10]);                       // return 4000.0. 5th customer, no discount.
cashier.getBill([7,5,3,1,6,4,2],[10,10,10,9,9,9,7]); // return 7350.0. 6th customer, 50% discount.
                                        // Original bill = 14700, but with
                                        // Actual bill = 14700 * ((100 - 50) / 100) = 7350.
cashier.getBill([2,3,5],[5,3,2]);                   // return 2500.0.  6th customer, no discount.
```

Constraints:

1 <= n <= 104
0 <= discount <= 100
1 <= products.length <= 200
prices.length == products.length
1 <= products[i] <= 200
1 <= prices[i] <= 1000
The elements in products are unique.
1 <= product.length <= products.length
amount.length == product.length
product[j] exists in products.
1 <= amount[j] <= 1000
The elements of product are unique.
At most 1000 calls will be made to getBill.
Answers within 10-5 of the actual value will be accepted.

1358. Number of Substrings Containing All Three Characters

Given a string s consisting only of characters a, b and c.
Return the number of substrings containing at least one occurrence of all these characters a, b and c.

Example 1:

Input: s = "abcabc"
Output: 10
Explanation: The substrings containing at least one occurrence of the characters a, b and c are "abc", "abca", "abcab", "abcabc", "bca", "bcab", "bcabc", "cab", "cabc" and "abc" (again).

Example 2:

Input: s = "aaacb"
Output: 3
Explanation: The substrings containing at least one occurrence of the characters a, b and c are "aaacb", "aacb" and "a

cb".

Example 3:

Input: s = "abc"
Output: 1

Constraints:

3 <= s.length <= 5 x 10^4
s only consists of a, b or c characters.

1359. Count All Valid Pickup and Delivery Options

Given n orders, each order consist in pickup and delivery services.
Count all valid pickup/delivery possible sequences such that delivery(i) is always after of pickup(i).
Since the answer may be too large, return it modulo 10^9 + 7.

Example 1:

Input: n = 1
Output: 1
Explanation: Unique order (P1, D1), Delivery 1 always is after of Pickup 1.

Example 2:

Input: n = 2
Output: 6
Explanation: All possible orders:
(P1,P2,D1,D2), (P1,P2,D2,D1), (P1,D1,P2,D2), (P2,P1,D1,D2), (P2,P1,D2,D1) and (P2,D2,P1,D1).
This is an invalid order (P1,D2,P2,D1) because Pickup 2 is after of Delivery 2.

Example 3:

Input: n = 3
Output: 90

Constraints:

1 <= n <= 500

1360. Number of Days Between Two Dates

Write a program to count the number of days between two dates.
The two dates are given as strings, their format is YYYY-MM-DD as shown in the examples.

Example 1:
Input: date1 = "2019-06-29", date2 = "2019-06-30"
Output: 1
Example 2:
Input: date1 = "2020-01-15", date2 = "2019-12-31"
Output: 15

Constraints:

The given dates are valid dates between the years 1971 and 2100.

1361. Validate Binary Tree Nodes

You have n binary tree nodes numbered from 0 to n - 1 where node i has two children leftChild[i] and rightChild[i], return true if and only if all the given nodes form exactly one valid binary tree.
If node i has no left child then leftChild[i] will equal -1, similarly for the right child.
Note that the nodes have no values and that we only use the node numbers in this problem.

Example 1:

Input: n = 4, leftChild = [1,-1,3,-1], rightChild = [2,-1,-1,-1]
Output: true

Example 2:

Input: n = 4, leftChild = [1,-1,3,-1], rightChild = [2,3,-1,-1]
Output: false

Example 3:

Input: n = 2, leftChild = [1,0], rightChild = [-1,-1]
Output: false

Example 4:

Input: n = 6, leftChild = [1,-1,-1,4,-1,-1], rightChild = [2,-1,-1,5,-1,-1]
Output: false

Constraints:

1 <= n <= 104
leftChild.length == rightChild.length == n
-1 <= leftChild[i], rightChild[i] <= n - 1

1362. Closest Divisors

Given an integer num, find the closest two integers in absolute difference whose product equals num + 1 or num + 2. Return the two integers in any order.

Example 1:

Input: num = 8
Output: [3,3]
Explanation: For num + 1 = 9, the closest divisors are 3 & 3, for num + 2 = 10, the closest divisors are 2 & 5, hence 3 & 3 is chosen.

Example 2:

Input: num = 123
Output: [5,25]

Example 3:

Input: num = 999
Output: [40,25]

Constraints:

1 <= num <= 10^9

1363. Largest Multiple of Three

Given an array of digits digits, return the largest multiple of three that can be formed by concatenating some of the gi ven digits in any order. If there is no answer return an empty string.
Since the answer may not fit in an integer data type, return the answer as a string. Note that the returning answer mus t not contain unnecessary leading zeros.

Example 1:

Input: digits = [8,1,9]
Output: "981"

Example 2:

Input: digits = [8,6,7,1,0]
Output: "8760"

Example 3:

Input: digits = [1]
Output: ""

Example 4:

Input: digits = [0,0,0,0,0,0]
Output: "0"

Constraints:

1 <= digits.length <= 104
0 <= digits[i] <= 9

1365. How Many Numbers Are Smaller Than the Current Number

Given the array nums, for each nums[i] find out how many numbers in the array are smaller than it. That is, for each nums[i] you have to count the number of valid j's such that j != i and nums[j] < nums[i].
Return the answer in an array.

Example 1:

Input: nums = [8,1,2,2,3]
Output: [4,0,1,1,3]

Explanation:
For nums[0]=8 there exist four smaller numbers than it (1, 2, 2 and 3).
For nums[1]=1 does not exist any smaller number than it.
For nums[2]=2 there exist one smaller number than it (1).
For nums[3]=2 there exist one smaller number than it (1).
For nums[4]=3 there exist three smaller numbers than it (1, 2 and 2).

Example 2:

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

Example 3:

Input: nums = [7,7,7,7]
Output: [0,0,0,0]

Constraints:

2 <= nums.length <= 500
0 <= nums[i] <= 100

1366. Rank Teams by Votes

In a special ranking system, each voter gives a rank from highest to lowest to all teams participated in the competitio n.
The ordering of teams is decided by who received the most position-one votes. If two or more teams tie in the first p osition, we consider the second position to resolve the conflict, if they tie again, we continue this process until the tie s are resolved. If two or more teams are still tied after considering all positions, we rank them alphabetically based o n their team letter.
Given an array of strings votes which is the votes of all voters in the ranking systems. Sort all teams according to the ranking system described above.
Return a string of all teams sorted by the ranking system.

Example 1:

Input: votes = ["ABC","ACB","ABC","ACB","ACB"]
Output: "ACB"
Explanation: Team A was ranked first place by 5 voters. No other team was voted as first place so team A is the first team.
Team B was ranked second by 2 voters and was ranked third by 3 voters.
Team C was ranked second by 3 voters and was ranked third by 2 voters.
As most of the voters ranked C second, team C is the second team and team B is the third.

Example 2:

Input: votes = ["WXYZ","XYZW"]
Output: "XWYZ"
Explanation: X is the winner due to tie-breaking rule. X has same votes as W for the first position but X has one vote as second position while W doesn't have any votes as second position.

Example 3:

Input: votes = ["ZMNAGUEDSJYLBOPHRQICWFXTVK"]
Output: "ZMNAGUEDSJYLBOPHRQICWFXTVK"
Explanation: Only one voter so his votes are used for the ranking.

Example 4:

Input: votes = ["BCA","CAB","CBA","ABC","ACB","BAC"]
Output: "ABC"
Explanation:
Team A was ranked first by 2 voters, second by 2 voters and third by 2 voters.
Team B was ranked first by 2 voters, second by 2 voters and third by 2 voters.
Team C was ranked first by 2 voters, second by 2 voters and third by 2 voters.
There is a tie and we rank teams ascending by their IDs.

Example 5:

Input: votes = ["M","M","M","M"]
Output: "M"
Explanation: Only team M in the competition so it has the first rank.

Constraints:

1 <= votes.length <= 1000
1 <= votes[i].length <= 26
votes[i].length == votes[j].length for 0 <= i, j < votes.length.
votes[i][j] is an English upper-case letter.
All characters of votes[i] are unique.
All the characters that occur in votes[0] also occur in votes[j] where 1 <= j < votes.length.

1367. Linked List in Binary Tree

Given a binary tree root and a linked list with head as the first node.
Return True if all the elements in the linked list starting from the head correspond to some downward path connected in the binary tree otherwise return False.
In this context downward path means a path that starts at some node and goes downwards.

Example 1:

Input: head = [4,2,8], root = [1,4,4,null,2,2,null,1,null,6,8,null,null,null,null,1,3]
Output: true
Explanation: Nodes in blue form a subpath in the binary Tree.

Example 2:

Input: head = [1,4,2,6], root = [1,4,4,null,2,2,null,1,null,6,8,null,null,null,null,1,3]
Output: true

Example 3:

Input: head = [1,4,2,6,8], root = [1,4,4,null,2,2,null,1,null,6,8,null,null,null,null,1,3]
Output: false
Explanation: There is no path in the binary tree that contains all the elements of the linked list from head.

Constraints:

The number of nodes in the tree will be in the range [1, 2500].
The number of nodes in the list will be in the range [1, 100].
1 <= Node.val <= 100 for each node in the linked list and binary tree.

1368. Minimum Cost to Make at Least One Valid Path in a Grid

Given a m x n grid. Each cell of the grid has a sign pointing to the next cell you should visit if you are currently in th is cell. The sign of grid[i][j] can be:

1 which means go to the cell to the right. (i.e go from grid[i][j] to grid[i][j + 1])
2 which means go to the cell to the left. (i.e go from grid[i][j] to grid[i][j - 1])
3 which means go to the lower cell. (i.e go from grid[i][j] to grid[i + 1][j])
4 which means go to the upper cell. (i.e go from grid[i][j] to grid[i - 1][j])

Notice that there could be some invalid signs on the cells of the grid which points outside the grid.
You will initially start at the upper left cell (0,0). A valid path in the grid is a path which starts from the upper left ce ll (0,0) and ends at the bottom-right cell (m - 1, n - 1) following the signs on the grid. The valid path doesn't have to be the shortest.
You can modify the sign on a cell with cost = 1. You can modify the sign on a cell one time only.
Return the minimum cost to make the grid have at least one valid path.

Example 1:

Input: grid = [[1,1,1,1],[2,2,2,2],[1,1,1,1],[2,2,2,2]]

Output: 3
Explanation: You will start at point (0, 0).
The path to (3, 3) is as follows. (0, 0) --> (0, 1) --> (0, 2) --> (0, 3) change the arrow to down with cost = 1 --> (1, 3) --> (1, 2) --> (1, 1) --> (1, 0) change the arrow to down with cost = 1 --> (2, 0) --> (2, 1) --> (2, 2) --> (2, 3) change t he arrow to down with cost = 1 --> (3, 3)
The total cost = 3.

Example 2:

Input: grid = [[1,1,3],[3,2,2],[1,1,4]]
Output: 0
Explanation: You can follow the path from (0, 0) to (2, 2).

Example 3:

Input: grid = [[1,2],[4,3]]
Output: 1

Example 4:

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

Example 5:

Input: grid = [[4]]
Output: 0

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 100

1370. Increasing Decreasing String

Given a string s. You should re-order the string using the following algorithm:

Pick the smallest character from s and append it to the result.
Pick the smallest character from s which is greater than the last appended character to the result and append it.
Repeat step 2 until you cannot pick more characters.
Pick the largest character from s and append it to the result.
Pick the largest character from s which is smaller than the last appended character to the result and append it.

Repeat step 5 until you cannot pick more characters.
Repeat the steps from 1 to 6 until you pick all characters from s.

In each step, If the smallest or the largest character appears more than once you can choose any occurrence and appe nd it to the result.
Return the result string after sorting s with this algorithm.

Example 1:

Input: s = "aaaabbbbcccc"
Output: "abccbaabccba"
Explanation: After steps 1, 2 and 3 of the first iteration, result = "abc"
After steps 4, 5 and 6 of the first iteration, result = "abccba"
First iteration is done. Now s = "aabbcc" and we go back to step 1
After steps 1, 2 and 3 of the second iteration, result = "abccbaabc"
After steps 4, 5 and 6 of the second iteration, result = "abccbaabccba"

Example 2:

Input: s = "rat"
Output: "art"
Explanation: The word "rat" becomes "art" after re-ordering it with the mentioned algorithm.

Example 3:

Input: s = "leetcode"
Output: "cdelotee"

Example 4:

Input: s = "ggggggg"
Output: "ggggggg"

Example 5:

Input: s = "spo"
Output: "ops"

Constraints:

1 <= s.length <= 500
s contains only lower-case English letters.

1371. Find the Longest Substring Containing Vowels in Even Counts

Given the string s, return the size of the longest substring containing each vowel an even number of times. That is, 'a', 'e', 'i', 'o', and 'u' must appear an even number of times.

Example 1:

Input: s = "eleetminicoworoep"
Output: 13
Explanation: The longest substring is "leetminicowor" which contains two each of the vowels: e, i and o and zero of the vowels: a and u.

Example 2:

Input: s = "leetcodeisgreat"
Output: 5
Explanation: The longest substring is "leetc" which contains two e's.

Example 3:

Input: s = "bcbcbc"
Output: 6
Explanation: In this case, the given string "bcbcbc" is the longest because all vowels: a, e, i, o and u appear zero times.

Constraints:

1 <= s.length <= 5 x 10^5
s contains only lowercase English letters.

1372. Longest ZigZag Path in a Binary Tree

You are given the root of a binary tree.
A ZigZag path for a binary tree is defined as follow:

Choose any node in the binary tree and a direction (right or left).
If the current direction is right, move to the right child of the current node; otherwise, move to the left child.
Change the direction from right to left or from left to right.
Repeat the second and third steps until you can't move in the tree.

Zigzag length is defined as the number of nodes visited - 1. (A single node has a length of 0).
Return the longest ZigZag path contained in that tree.

Example 1:

Input: root = [1,null,1,1,1,null,null,1,1,null,1,null,null,null,1,null,1]

Output: 3
Explanation: Longest ZigZag path in blue nodes (right -> left -> right).

Example 2:

Input: root = [1,1,1,null,1,null,null,1,1,null,1]
Output: 4
Explanation: Longest ZigZag path in blue nodes (left -> right -> left -> right).

Example 3:

Input: root = [1]
Output: 0

Constraints:

The number of nodes in the tree is in the range [1, 5 * 104].
1 <= Node.val <= 100

1373. Maximum Sum BST in Binary Tree

Given a binary tree root, return the maximum sum of all keys of any sub-tree which is also a Binary Search Tree (BST).
Assume a BST is defined as follows:

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

Example 1:

Input: root = [1,4,3,2,4,2,5,null,null,null,null,null,null,4,6]
Output: 20
Explanation: Maximum sum in a valid Binary search tree is obtained in root node with key equal to 3.

Example 2:

Input: root = [4,3,null,1,2]
Output: 2
Explanation: Maximum sum in a valid Binary search tree is obtained in a single root node with key equal to 2.

Example 3:

Input: root = [-4,-2,-5]
Output: 0
Explanation: All values are negatives. Return an empty BST.

Example 4:

Input: root = [2,1,3]
Output: 6

Example 5:

Input: root = [5,4,8,3,null,6,3]
Output: 7

Constraints:

The number of nodes in the tree is in the range [1, 4 * 104].
-4 * 104 <= Node.val <= 4 * 104

1374. Generate a String With Characters That Have Odd Counts

Given an integer n, return a string with n characters such that each character in such string occurs an odd number of t imes.
The returned string must contain only lowercase English letters. If there are multiples valid strings, return any of the m.

Example 1:

Input: n = 4
Output: "pppz"
Explanation: "pppz" is a valid string since the character 'p' occurs three times and the character 'z' occurs once. Note that there are many other valid strings such as "ohhh" and "love".

Example 2:

Input: n = 2
Output: "xy"
Explanation: "xy" is a valid string since the characters 'x' and 'y' occur once. Note that there are many other valid stri ngs such as "ag" and "ur".

Example 3:

Input: n = 7

Output: "holasss"

Constraints:

1 <= n <= 500

1375. Bulb Switcher III

There is a room with n bulbs, numbered from 1 to n, arranged in a row from left to right. Initially, all the bulbs are tu rned off.
At moment k (for k from 0 to n - 1), we turn on the light[k] bulb. A bulb changes color to blue only if it is on and all the previous bulbs (to the left) are turned on too.
Return the number of moments in which all turned-on bulbs are blue.

Example 1:

Input: light = [2,1,3,5,4]
Output: 3
Explanation: All bulbs turned on, are blue at the moment 1, 2 and 4.

Example 2:

Input: light = [3,2,4,1,5]
Output: 2
Explanation: All bulbs turned on, are blue at the moment 3, and 4 (index-0).

Example 3:

Input: light = [4,1,2,3]
Output: 1
Explanation: All bulbs turned on, are blue at the moment 3 (index-0).
Bulb 4th changes to blue at the moment 3.

Example 4:

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

Example 5:

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

Constraints:

n == light.length
1 <= n <= 5 * 104
light is a permutation of the numbers in the range [1, n]

1376. Time Needed to Inform All Employees

A company has n employees with a unique ID for each employee from 0 to n - 1. The head of the company is the on e with headID.
Each employee has one direct manager given in the manager array where manager[i] is the direct manager of the i-th employee, manager[headID] = -1. Also, it is guaranteed that the subordination relationships have a tree structure.
The head of the company wants to inform all the company employees of an urgent piece of news. He will inform his direct subordinates, and they will inform their subordinates, and so on until all employees know about the urgent ne ws.
The i-th employee needs informTime[i] minutes to inform all of his direct subordinates (i.e., After informTime[i] mi nutes, all his direct subordinates can start spreading the news).
Return the number of minutes needed to inform all the employees about the urgent news.

Example 1:

Input: n = 1, headID = 0, manager = [-1], informTime = [0]
Output: 0
Explanation: The head of the company is the only employee in the company.

Example 2:

Input: n = 6, headID = 2, manager = [2,2,-1,2,2,2], informTime = [0,0,1,0,0,0]
Output: 1
Explanation: The head of the company with id = 2 is the direct manager of all the employees in the company and nee ds 1 minute to inform them all.
The tree structure of the employees in the company is shown.

Example 3:

Input: n = 7, headID = 6, manager = [1,2,3,4,5,6,-1], informTime = [0,6,5,4,3,2,1]
Output: 21
Explanation: The head has id = 6. He will inform employee with id = 5 in 1 minute.
The employee with id = 5 will inform the employee with id = 4 in 2 minutes.
The employee with id = 4 will inform the employee with id = 3 in 3 minutes.
The employee with id = 3 will inform the employee with id = 2 in 4 minutes.
The employee with id = 2 will inform the employee with id = 1 in 5 minutes.
The employee with id = 1 will inform the employee with id = 0 in 6 minutes.
Needed time = 1 + 2 + 3 + 4 + 5 + 6 = 21.

Example 4:

Input: n = 15, headID = 0, manager = [-1,0,0,1,1,2,2,3,3,4,4,5,5,6,6], informTime = [1,1,1,1,1,1,1,0,0,0,0,0,0,0,0]
Output: 3
Explanation: The first minute the head will inform employees 1 and 2.
The second minute they will inform employees 3, 4, 5 and 6.
The third minute they will inform the rest of employees.

Example 5:

Input: n = 4, headID = 2, manager = [3,3,-1,2], informTime = [0,0,162,914]
Output: 1076

Constraints:

1 <= n <= 105
0 <= headID < n
manager.length == n
0 <= manager[i] < n
manager[headID] == -1
informTime.length == n
0 <= informTime[i] <= 1000
informTime[i] == 0 if employee i has no subordinates.
It is guaranteed that all the employees can be informed.

1377. Frog Position After T Seconds

Given an undirected tree consisting of n vertices numbered from 1 to n. A frog starts jumping from vertex 1. In one s econd, the frog jumps from its current vertex to another unvisited vertex if they are directly connected. The frog can not jump back to a visited vertex. In case the frog can jump to several vertices, it jumps randomly to one of them wit h the same probability. Otherwise, when the frog can not jump to any unvisited vertex, it jumps forever on the same vertex.
The edges of the undirected tree are given in the array edges, where edges[i] = [ai, bi] means that exists an edge conn ecting the vertices ai and bi.
Return the probability that after t seconds the frog is on the vertex target.

Example 1:

Input: n = 7, edges = [[1,2],[1,3],[1,7],[2,4],[2,6],[3,5]], t = 2, target = 4
Output: 0.16666666666666666
Explanation: The figure above shows the given graph. The frog starts at vertex 1, jumping with 1/3 probability to the vertex 2 after second 1 and then jumping with 1/2 probability to vertex 4 after second 2. Thus the probability for the frog is on the vertex 4 after 2 seconds is 1/3 * 1/2 = 1/6 = 0.16666666666666666.

Example 2:

Input: n = 7, edges = [[1,2],[1,3],[1,7],[2,4],[2,6],[3,5]], t = 1, target = 7
Output: 0.3333333333333333
Explanation: The figure above shows the given graph. The frog starts at vertex 1, jumping with 1/3 = 0.3333333333333333 probability to the vertex 7 after second 1.

Example 3:

Input: n = 7, edges = [[1,2],[1,3],[1,7],[2,4],[2,6],[3,5]], t = 20, target = 6
Output: 0.16666666666666666

Constraints:

1 <= n <= 100
edges.length == n - 1
edges[i].length == 2
1 <= ai, bi <= n
1 <= t <= 50
1 <= target <= n
Answers within 10-5 of the actual value will be accepted as correct.

1379. Find a Corresponding Node of a Binary Tree in a Clone of That Tree

Given two binary trees original and cloned and given a reference to a node target in the original tree.
The cloned tree is a copy of the original tree.
Return a reference to the same node in the cloned tree.
Note that you are not allowed to change any of the two trees or the target node and the answer must be a reference to a node in the cloned tree.

Example 1:

Input: tree = [7,4,3,null,null,6,19], target = 3
Output: 3
Explanation: In all examples the original and cloned trees are shown. The target node is a green node from the original tree. The answer is the yellow node from the cloned tree.

Example 2:

Input: tree = [7], target = 7
Output: 7

Example 3:

Input: tree = [8,null,6,null,5,null,4,null,3,null,2,null,1], target = 4
Output: 4

Example 4:

Input: tree = [1,2,3,4,5,6,7,8,9,10], target = 5
Output: 5

Example 5:

Input: tree = [1,2,null,3], target = 2
Output: 2

Constraints:

The number of nodes in the tree is in the range [1, 104].
The values of the nodes of the tree are unique.
target node is a node from the original tree and is not null.

Follow up: Could you solve the problem if repeated values on the tree are allowed?

1380. Lucky Numbers in a Matrix

Given an m x n matrix of distinct numbers, return all lucky numbers in the matrix in any order.
A lucky number is an element of the matrix such that it is the minimum element in its row and maximum in its colu mn.

Example 1:

Input: matrix = [[3,7,8],[9,11,13],[15,16,17]]
Output: [15]
Explanation: 15 is the only lucky number since it is the minimum in its row and the maximum in its column

Example 2:

Input: matrix = [[1,10,4,2],[9,3,8,7],[15,16,17,12]]
Output: [12]
Explanation: 12 is the only lucky number since it is the minimum in its row and the maximum in its column.

Example 3:

Input: matrix = [[7,8],[1,2]]
Output: [7]
Explanation: 7 is the only lucky number since it is the minimum in its row and the maximum in its column.

Example 4:

Input: matrix = [[3,6],[7,1],[5,2],[4,8]]
Output: []
Explanation: There is no lucky number.

Constraints:

m == mat.length
n == mat[i].length
1 <= n, m <= 50
1 <= matrix[i][j] <= 105.
All elements in the matrix are distinct.

1381. Design a Stack With Increment Operation

Design a stack which supports the following operations.
Implement the CustomStack class:

CustomStack(int maxSize) Initializes the object with maxSize which is the maximum number of elements in the stac k or do nothing if the stack reached the maxSize.
void push(int x) Adds x to the top of the stack if the stack hasn't reached the maxSize.
int pop() Pops and returns the top of stack or -1 if the stack is empty.
void inc(int k, int val) Increments the bottom k elements of the stack by val. If there are less than k elements in the st ack, just increment all the elements in the stack.

Example 1:

Input
["CustomStack","push","push","pop","push","push","push","increment","increment","pop","pop","pop","pop"]
[[3],[1],[2],[],[2],[3],[4],[5,100],[2,100],[],[],[],[]]
Output
[null,null,null,2,null,null,null,null,null,103,202,201,-1]
Explanation
CustomStack customStack = new CustomStack(3); // Stack is Empty []
customStack.push(1); // stack becomes [1]
customStack.push(2); // stack becomes [1, 2]
customStack.pop(); // return 2 --> Return top of the stack 2, stack becomes [1]

```
customStack.push(2);                  // stack becomes [1, 2]
customStack.push(3);                  // stack becomes [1, 2, 3]
customStack.push(4);                  // stack still [1, 2, 3], Don't add another elements as size is 4
customStack.increment(5, 100);            // stack becomes [101, 102, 103]
customStack.increment(2, 100);            // stack becomes [201, 202, 103]
customStack.pop();                   // return 103 --> Return top of the stack 103, stack becomes [201, 202]
customStack.pop();                   // return 202 --> Return top of the stack 102, stack becomes [201]
customStack.pop();                   // return 201 --> Return top of the stack 101, stack becomes []
customStack.pop();                   // return -1 --> Stack is empty return -1.
```

Constraints:

1 <= maxSize <= 1000
1 <= x <= 1000
1 <= k <= 1000
0 <= val <= 100
At most 1000 calls will be made to each method of increment, push and pop each separately.

1382. Balance a Binary Search Tree

Given the root of a binary search tree, return a balanced binary search tree with the same node values. If there is mor e than one answer, return any of them.
A binary search tree is balanced if the depth of the two subtrees of every node never differs by more than 1.

Example 1:

Input: root = [1,null,2,null,3,null,4,null,null]
Output: [2,1,3,null,null,null,4]
Explanation: This is not the only correct answer, [3,1,4,null,2] is also correct.

Example 2:

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

Constraints:

The number of nodes in the tree is in the range [1, 104].
1 <= Node.val <= 105

1383. Maximum Performance of a Team

You are given two integers n and k and two integer arrays speed and efficiency both of length n. There are n enginee rs numbered from 1 to n. speed[i] and efficiency[i] represent the speed and efficiency of the ith engineer respectively .
Choose at most k different engineers out of the n engineers to form a team with the maximum performance.
The performance of a team is the sum of their engineers' speeds multiplied by the minimum efficiency among their e ngineers.
Return the maximum performance of this team. Since the answer can be a huge number, return it modulo 109 + 7.

Example 1:

Input: n = 6, speed = [2,10,3,1,5,8], efficiency = [5,4,3,9,7,2], k = 2
Output: 60
Explanation:
We have the maximum performance of the team by selecting engineer 2 (with speed=10 and efficiency=4) and engin eer 5 (with speed=5 and efficiency=7). That is, performance = (10 + 5) * min(4, 7) = 60.

Example 2:

Input: n = 6, speed = [2,10,3,1,5,8], efficiency = [5,4,3,9,7,2], k = 3
Output: 68
Explanation:
This is the same example as the first but k = 3. We can select engineer 1, engineer 2 and engineer 5 to get the maxim um performance of the team. That is, performance = (2 + 10 + 5) * min(5, 4, 7) = 68.

Example 3:

Input: n = 6, speed = [2,10,3,1,5,8], efficiency = [5,4,3,9,7,2], k = 4
Output: 72

Constraints:

1 <= k <= n <= 105
speed.length == n
efficiency.length == n
1 <= speed[i] <= 105
1 <= efficiency[i] <= 108

1385. Find the Distance Value Between Two Arrays

Given two integer arrays arr1 and arr2, and the integer d, return the distance value between the two arrays.
The distance value is defined as the number of elements arr1[i] such that there is not any element arr2[j] where |arr1[i]-arr2[j]| <= d.

Example 1:

Input: arr1 = [4,5,8], arr2 = [10,9,1,8], d = 2
Output: 2
Explanation:
For arr1[0]=4 we have:
|4-10|=6 > d=2
|4-9|=5 > d=2
|4-1|=3 > d=2
|4-8|=4 > d=2
For arr1[1]=5 we have:
|5-10|=5 > d=2
|5-9|=4 > d=2
|5-1|=4 > d=2
|5-8|=3 > d=2
For arr1[2]=8 we have:
|8-10|=2 <= d=2
|8-9|=1 <= d=2
|8-1|=7 > d=2
|8-8|=0 <= d=2

Example 2:

Input: arr1 = [1,4,2,3], arr2 = [-4,-3,6,10,20,30], d = 3
Output: 2

Example 3:

Input: arr1 = [2,1,100,3], arr2 = [-5,-2,10,-3,7], d = 6
Output: 1

Constraints:

1 <= arr1.length, arr2.length <= 500
-1000 <= arr1[i], arr2[j] <= 1000
0 <= d <= 100

1386. Cinema Seat Allocation

A cinema has n rows of seats, numbered from 1 to n and there are ten seats in each row, labelled from 1 to 10 as sho
wn in the figure above.
Given the array reservedSeats containing the numbers of seats already reserved, for example, reservedSeats[i] = [3,8
] means the seat located in row 3 and labelled with 8 is already reserved.
Return the maximum number of four-person groups you can assign on the cinema seats. A four-person group occupi
es four adjacent seats in one single row. Seats across an aisle (such as [3,3] and [3,4]) are not considered to be adjace
nt, but there is an exceptional case on which an aisle split a four-person group, in that case, the aisle split a four-pers
on group in the middle, which means to have two people on each side.

Example 1:

Input: n = 3, reservedSeats = [[1,2],[1,3],[1,8],[2,6],[3,1],[3,10]]
Output: 4
Explanation: The figure above shows the optimal allocation for four groups, where seats mark with blue are already reserved and contiguous seats mark with orange are for one group.

Example 2:

Input: n = 2, reservedSeats = [[2,1],[1,8],[2,6]]
Output: 2

Example 3:

Input: n = 4, reservedSeats = [[4,3],[1,4],[4,6],[1,7]]
Output: 4

Constraints:

1 <= n <= 10^9
1 <= reservedSeats.length <= min(10*n, 10^4)
reservedSeats[i].length == 2
1 <= reservedSeats[i][0] <= n
1 <= reservedSeats[i][1] <= 10
All reservedSeats[i] are distinct.

1387. Sort Integers by The Power Value

The power of an integer x is defined as the number of steps needed to transform x into 1 using the following steps:

if x is even then x = x / 2
if x is odd then x = 3 * x + 1

For example, the power of x = 3 is 7 because 3 needs 7 steps to become 1 (3 --> 10 --> 5 --> 16 --> 8 --> 4 --> 2 -->

1).
Given three integers lo, hi and k. The task is to sort all integers in the interval [lo, hi] by the power value in ascendin g order, if two or more integers have the same power value sort them by ascending order.
Return the k-th integer in the range [lo, hi] sorted by the power value.
Notice that for any integer x (lo <= x <= hi) it is guaranteed that x will transform into 1 using these steps and that the power of x is will fit in 32 bit signed integer.

Example 1:

Input: lo = 12, hi = 15, k = 2
Output: 13
Explanation: The power of 12 is 9 (12 --> 6 --> 3 --> 10 --> 5 --> 16 --> 8 --> 4 --> 2 --> 1)
The power of 13 is 9
The power of 14 is 17
The power of 15 is 17
The interval sorted by the power value [12,13,14,15]. For k = 2 answer is the second element which is 13.
Notice that 12 and 13 have the same power value and we sorted them in ascending order. Same for 14 and 15.

Example 2:

Input: lo = 1, hi = 1, k = 1
Output: 1

Example 3:

Input: lo = 7, hi = 11, k = 4
Output: 7
Explanation: The power array corresponding to the interval [7, 8, 9, 10, 11] is [16, 3, 19, 6, 14].
The interval sorted by power is [8, 10, 11, 7, 9].
The fourth number in the sorted array is 7.

Example 4:

Input: lo = 10, hi = 20, k = 5
Output: 13

Example 5:

Input: lo = 1, hi = 1000, k = 777
Output: 570

Constraints:

1 <= lo <= hi <= 1000
1 <= k <= hi - lo + 1

1388. Pizza With 3n Slices

There is a pizza with 3n slices of varying size, you and your friends will take slices of pizza as follows:

You will pick any pizza slice.
Your friend Alice will pick next slice in anti clockwise direction of your pick.
Your friend Bob will pick next slice in clockwise direction of your pick.
Repeat until there are no more slices of pizzas.

Sizes of Pizza slices is represented by circular array slices in clockwise direction.
Return the maximum possible sum of slice sizes which you can have.

Example 1:

Input: slices = [1,2,3,4,5,6]
Output: 10
Explanation: Pick pizza slice of size 4, Alice and Bob will pick slices with size 3 and 5 respectively. Then Pick slice s with size 6, finally Alice and Bob will pick slice of size 2 and 1 respectively. Total = 4 + 6.

Example 2:

Input: slices = [8,9,8,6,1,1]
Output: 16
Output: Pick pizza slice of size 8 in each turn. If you pick slice with size 9 your partners will pick slices of size 8.

Example 3:

Input: slices = [4,1,2,5,8,3,1,9,7]
Output: 21

Example 4:

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

Constraints:

1 <= slices.length <= 500
slices.length % 3 == 0
1 <= slices[i] <= 1000

1389. Create Target Array in the Given Order

Given two arrays of integers nums and index. Your task is to create target array under the following rules:

Initially target array is empty.
From left to right read nums[i] and index[i], insert at index index[i] the value nums[i] in target array.
Repeat the previous step until there are no elements to read in nums and index.

Return the target array.
It is guaranteed that the insertion operations will be valid.

Example 1:

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

nums	index	target
0	0	[0]
1	1	[0,1]
2	2	[0,1,2]
3	2	[0,1,3,2]
4	1	[0,4,1,3,2]

Example 2:

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

nums	index	target
1	0	[1]
2	1	[1,2]
3	2	[1,2,3]
4	3	[1,2,3,4]
0	0	[0,1,2,3,4]

Example 3:

Input: nums = [1], index = [0]
Output: [1]

Constraints:

1 <= nums.length, index.length <= 100
nums.length == index.length
0 <= nums[i] <= 100
0 <= index[i] <= i

1390. Four Divisors

Given an integer array nums, return the sum of divisors of the integers in that array that have exactly four divisors. If there is no such integer in the array, return 0.

Example 1:

Input: nums = [21,4,7]
Output: 32
Explanation:
21 has 4 divisors: 1, 3, 7, 21
4 has 3 divisors: 1, 2, 4
7 has 2 divisors: 1, 7
The answer is the sum of divisors of 21 only.

Example 2:

Input: nums = [21,21]
Output: 64

Example 3:

Input: nums = [1,2,3,4,5]
Output: 0

Constraints:

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

1391. Check if There is a Valid Path in a Grid

Given a m x n grid. Each cell of the grid represents a street. The street of grid[i][j] can be:

1 which means a street connecting the left cell and the right cell.
2 which means a street connecting the upper cell and the lower cell.
3 which means a street connecting the left cell and the lower cell.
4 which means a street connecting the right cell and the lower cell.
5 which means a street connecting the left cell and the upper cell.
6 which means a street connecting the right cell and the upper cell.

You will initially start at the street of the upper-left cell (0,0). A valid path in the grid is a path which starts from the upper left cell (0,0) and ends at the bottom-right cell (m - 1, n - 1). The path should only follow the streets.
Notice that you are not allowed to change any street.

Return true if there is a valid path in the grid or false otherwise.

Example 1:

Input: grid = [[2,4,3],[6,5,2]]
Output: true
Explanation: As shown you can start at cell (0, 0) and visit all the cells of the grid to reach (m - 1, n - 1).

Example 2:

Input: grid = [[1,2,1],[1,2,1]]
Output: false
Explanation: As shown you the street at cell (0, 0) is not connected with any street of any other cell and you will get stuck at cell (0, 0)

Example 3:

Input: grid = [[1,1,2]]
Output: false
Explanation: You will get stuck at cell (0, 1) and you cannot reach cell (0, 2).

Example 4:

Input: grid = [[1,1,1,1,1,1,3]]
Output: true

Example 5:

Input: grid = [[2],[2],[2],[2],[2],[2],[6]]
Output: true

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 300
1 <= grid[i][j] <= 6

1392. Longest Happy Prefix

A string is called a happy prefix if is a non-empty prefix which is also a suffix (excluding itself).
Given a string s, return the longest happy prefix of s. Return an empty string "" if no such prefix exists.

Example 1:

Input: s = "level"
Output: "l"
Explanation: s contains 4 prefix excluding itself ("l", "le", "lev", "leve"), and suffix ("l", "el", "vel", "evel"). The larg est prefix which is also suffix is given by "l".

Example 2:

Input: s = "abababab"
Output: "abab"
Explanation: "abab" is the largest prefix which is also suffix. They can overlap in the original string.

Example 3:

Input: s = "leetcodeleet"
Output: "leet"

Example 4:

Input: s = "a"
Output: ""

Constraints:

1 <= s.length <= 105
s contains only lowercase English letters.

1394. Find Lucky Integer in an Array

Given an array of integers arr, a lucky integer is an integer which has a frequency in the array equal to its value. Return a lucky integer in the array. If there are multiple lucky integers return the largest of them. If there is no lucky integer return -1.

Example 1:

Input: arr = [2,2,3,4]
Output: 2
Explanation: The only lucky number in the array is 2 because frequency[2] == 2.

Example 2:

Input: arr = [1,2,2,3,3,3]
Output: 3
Explanation: 1, 2 and 3 are all lucky numbers, return the largest of them.

Example 3:

Input: arr = [2,2,2,3,3]
Output: -1
Explanation: There are no lucky numbers in the array.

Example 4:

Input: arr = [5]
Output: -1

Example 5:

Input: arr = [7,7,7,7,7,7,7]
Output: 7

Constraints:

1 <= arr.length <= 500
1 <= arr[i] <= 500

1395. Count Number of Teams

There are n soldiers standing in a line. Each soldier is assigned a unique rating value.
You have to form a team of 3 soldiers amongst them under the following rules:

Choose 3 soldiers with index (i, j, k) with rating (rating[i], rating[j], rating[k]).
A team is valid if: (rating[i] < rating[j] < rating[k]) or (rating[i] > rating[j] > rating[k]) where (0 <= i < j < k < n).

Return the number of teams you can form given the conditions. (soldiers can be part of multiple teams).

Example 1:

Input: rating = [2,5,3,4,1]
Output: 3
Explanation: We can form three teams given the conditions. (2,3,4), (5,4,1), (5,3,1).

Example 2:

Input: rating = [2,1,3]
Output: 0
Explanation: We can't form any team given the conditions.

Example 3:

Input: rating = [1,2,3,4]
Output: 4

Constraints:

n == rating.length
3 <= n <= 1000
1 <= rating[i] <= 105
All the integers in rating are unique.

1396. Design Underground System

An underground railway system is keeping track of customer travel times between different stations. They are using this data to calculate the average time it takes to travel from one station to another.
Implement the UndergroundSystem class:

void checkIn(int id, string stationName, int t)

A customer with a card ID equal to id, checks in at the station stationName at time t.
A customer can only be checked into one place at a time.

void checkOut(int id, string stationName, int t)

A customer with a card ID equal to id, checks out from the station stationName at time t.

double getAverageTime(string startStation, string endStation)

Returns the average time it takes to travel from startStation to endStation.
The average time is computed from all the previous traveling times from startStation to endStation that happened dir ectly, meaning a check in at startStation followed by a check out from endStation.
The time it takes to travel from startStation to endStation may be different from the time it takes to travel from endSt ation to startStation.
There will be at least one customer that has traveled from startStation to endStation before getAverageTime is called .

You may assume all calls to the checkIn and checkOut methods are consistent. If a customer checks in at time t1 the n checks out at time t2, then t1 < t2. All events happen in chronological order.

Example 1:

Input

["UndergroundSystem","checkIn","checkIn","checkIn","checkOut","checkOut","checkOut","getAverageTime","get
AverageTime","checkIn","getAverageTime","checkOut","getAverageTime"]
[[],[45,"Leyton",3],[32,"Paradise",8],[27,"Leyton",10],[45,"Waterloo",15],[27,"Waterloo",20],[32,"Cambridge",22],
["Paradise","Cambridge"],["Leyton","Waterloo"],[10,"Leyton",24],["Leyton","Waterloo"],[10,"Waterloo",38],["Ley
ton","Waterloo"]]

Output
[null,null,null,null,null,null,null,14.00000,11.00000,null,11.00000,null,12.00000]

Explanation
UndergroundSystem undergroundSystem = new UndergroundSystem();
undergroundSystem.checkIn(45, "Leyton", 3);
undergroundSystem.checkIn(32, "Paradise", 8);
undergroundSystem.checkIn(27, "Leyton", 10);
undergroundSystem.checkOut(45, "Waterloo", 15); // Customer 45 "Leyton" -> "Waterloo" in 15-3 = 12
undergroundSystem.checkOut(27, "Waterloo", 20); // Customer 27 "Leyton" -> "Waterloo" in 20-10 = 10
undergroundSystem.checkOut(32, "Cambridge", 22); // Customer 32 "Paradise" -> "Cambridge" in 22-8 = 14
undergroundSystem.getAverageTime("Paradise", "Cambridge"); // return 14.00000. One trip "Paradise" -> "Cambri
dge", (14) / 1 = 14
undergroundSystem.getAverageTime("Leyton", "Waterloo");   // return 11.00000. Two trips "Leyton" -> "Waterloo
", (10 + 12) / 2 = 11
undergroundSystem.checkIn(10, "Leyton", 24);
undergroundSystem.getAverageTime("Leyton", "Waterloo");   // return 11.00000
undergroundSystem.checkOut(10, "Waterloo", 38); // Customer 10 "Leyton" -> "Waterloo" in 38-24 = 14
undergroundSystem.getAverageTime("Leyton", "Waterloo");   // return 12.00000. Three trips "Leyton" -> "Waterlo
o", (10 + 12 + 14) / 3 = 12

Example 2:

Input
["UndergroundSystem","checkIn","checkOut","getAverageTime","checkIn","checkOut","getAverageTime","checkI
n","checkOut","getAverageTime"]
[[],[10,"Leyton",3],[10,"Paradise",8],["Leyton","Paradise"],[5,"Leyton",10],[5,"Paradise",16],["Leyton","Paradise"],
[2,"Leyton",21],[2,"Paradise",30],["Leyton","Paradise"]]

Output
[null,null,null,5.00000,null,null,5.50000,null,null,6.66667]

Explanation
UndergroundSystem undergroundSystem = new UndergroundSystem();
undergroundSystem.checkIn(10, "Leyton", 3);
undergroundSystem.checkOut(10, "Paradise", 8); // Customer 10 "Leyton" -> "Paradise" in 8-3 = 5
undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 5.00000, (5) / 1 = 5
undergroundSystem.checkIn(5, "Leyton", 10);
undergroundSystem.checkOut(5, "Paradise", 16); // Customer 5 "Leyton" -> "Paradise" in 16-10 = 6
undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 5.50000, (5 + 6) / 2 = 5.5
undergroundSystem.checkIn(2, "Leyton", 21);
undergroundSystem.checkOut(2, "Paradise", 30); // Customer 2 "Leyton" -> "Paradise" in 30-21 = 9
undergroundSystem.getAverageTime("Leyton", "Paradise"); // return 6.66667, (5 + 6 + 9) / 3 = 6.66667

Constraints:

1 <= id, t <= 106

1 <= stationName.length, startStation.length, endStation.length <= 10
All strings consist of uppercase and lowercase English letters and digits.
There will be at most 2 * 104 calls in total to checkIn, checkOut, and getAverageTime.
Answers within 10-5 of the actual value will be accepted.

1397. Find All Good Strings

Given the strings s1 and s2 of size n and the string evil, return the number of good strings.
A good string has size n, it is alphabetically greater than or equal to s1, it is alphabetically smaller than or equal to s2 , and it does not contain the string evil as a substring. Since the answer can be a huge number, return this modulo 10 9 + 7.

Example 1:

Input: n = 2, s1 = "aa", s2 = "da", evil = "b"
Output: 51
Explanation: There are 25 good strings starting with 'a': "aa","ac","ad",...,"az". Then there are 25 good strings startin g with 'c': "ca","cc","cd",...,"cz" and finally there is one good string starting with 'd': "da".

Example 2:

Input: n = 8, s1 = "leetcode", s2 = "leetgoes", evil = "leet"
Output: 0
Explanation: All strings greater than or equal to s1 and smaller than or equal to s2 start with the prefix "leet", therefo re, there is not any good string.

Example 3:

Input: n = 2, s1 = "gx", s2 = "gz", evil = "x"
Output: 2

Constraints:

s1.length == n
s2.length == n
s1 <= s2
1 <= n <= 500
1 <= evil.length <= 50
All strings consist of lowercase English letters.

1399. Count Largest Group

Given an integer n. Each number from 1 to n is grouped according to the sum of its digits.
Return how many groups have the largest size.

Example 1:

Input: n = 13
Output: 4
Explanation: There are 9 groups in total, they are grouped according sum of its digits of numbers from 1 to 13: [1,10], [2,11], [3,12], [4,13], [5], [6], [7], [8], [9]. There are 4 groups with largest size.

Example 2:

Input: n = 2
Output: 2
Explanation: There are 2 groups [1], [2] of size 1.

Example 3:

Input: n = 15
Output: 6

Example 4:

Input: n = 24
Output: 5

Constraints:

1 <= n <= 10^4

1400. Construct K Palindrome Strings

Given a string s and an integer k. You should construct k non-empty palindrome strings using all the characters in s.
Return True if you can use all the characters in s to construct k palindrome strings or False otherwise.

Example 1:

Input: s = "annabelle", k = 2
Output: true
Explanation: You can construct two palindromes using all characters in s.

Some possible constructions "anna" + "elble", "anbna" + "elle", "anellena" + "b"

Example 2:

Input: s = "leetcode", k = 3
Output: false
Explanation: It is impossible to construct 3 palindromes using all the characters of s.

Example 3:

Input: s = "true", k = 4
Output: true
Explanation: The only possible solution is to put each character in a separate string.

Example 4:

Input: s = "yzyzyzyzyzyzyzy", k = 2
Output: true
Explanation: Simply you can put all z's in one string and all y's in the other string. Both strings will be palindrome.

Example 5:

Input: s = "cr", k = 7
Output: false
Explanation: We don't have enough characters in s to construct 7 palindromes.

Constraints:

1 <= s.length <= 10^5
All characters in s are lower-case English letters.
1 <= k <= 10^5

1401. Circle and Rectangle Overlapping

Given a circle represented as (radius, x_center, y_center) and an axis-aligned rectangle represented as (x1, y1, x2, y2), where (x1, y1) are the coordinates of the bottom-left corner, and (x2, y2) are the coordinates of the top-right corner of the rectangle.
Return True if the circle and rectangle are overlapped otherwise return False.
In other words, check if there are any point (xi, yi) such that belongs to the circle and the rectangle at the same time.

Example 1:

Input: radius = 1, x_center = 0, y_center = 0, x1 = 1, y1 = -1, x2 = 3, y2 = 1
Output: true
Explanation: Circle and rectangle share the point (1,0)

Example 2:

Input: radius = 1, x_center = 0, y_center = 0, x1 = -1, y1 = 0, x2 = 0, y2 = 1
Output: true

Example 3:

Input: radius = 1, x_center = 1, y_center = 1, x1 = -3, y1 = -3, x2 = 3, y2 = 3
Output: true

Example 4:

Input: radius = 1, x_center = 1, y_center = 1, x1 = 1, y1 = -3, x2 = 2, y2 = -1
Output: false

Constraints:

1 <= radius <= 2000
-10^4 <= x_center, y_center, x1, y1, x2, y2 <= 10^4
x1 < x2
y1 < y2

1402. Reducing Dishes

A chef has collected data on the satisfaction level of his n dishes. Chef can cook any dish in 1 unit of time.
Like-time coefficient of a dish is defined as the time taken to cook that dish including previous dishes multiplied by i ts satisfaction level i.e. time[i] * satisfaction[i].
Return the maximum sum of like-time coefficient that the chef can obtain after dishes preparation.
Dishes can be prepared in any order and the chef can discard some dishes to get this maximum value.

Example 1:

Input: satisfaction = [-1,-8,0,5,-9]
Output: 14
Explanation: After Removing the second and last dish, the maximum total like-time coefficient will be equal to (-1* 1 + 0*2 + 5*3 = 14).
Each dish is prepared in one unit of time.
Example 2:

Input: satisfaction = [4,3,2]
Output: 20
Explanation: Dishes can be prepared in any order, (2*1 + 3*2 + 4*3 = 20)

Example 3:

Input: satisfaction = [-1,-4,-5]
Output: 0
Explanation: People don't like the dishes. No dish is prepared.

Example 4:

Input: satisfaction = [-2,5,-1,0,3,-3]
Output: 35

Constraints:

n == satisfaction.length
1 <= n <= 500
-1000 <= satisfaction[i] <= 1000

1403. Minimum Subsequence in Non-Increasing Order

Given the array nums, obtain a subsequence of the array whose sum of elements is strictly greater than the sum of th e non included elements in such subsequence.
If there are multiple solutions, return the subsequence with minimum size and if there still exist multiple solutions, re turn the subsequence with the maximum total sum of all its elements. A subsequence of an array can be obtained by erasing some (possibly zero) elements from the array.
Note that the solution with the given constraints is guaranteed to be unique. Also return the answer sorted in non-inc reasing order.

Example 1:

Input: nums = [4,3,10,9,8]
Output: [10,9]
Explanation: The subsequences [10,9] and [10,8] are minimal such that the sum of their elements is strictly greater th an the sum of elements not included, however, the subsequence [10,9] has the maximum total sum of its elements.

Example 2:

Input: nums = [4,4,7,6,7]
Output: [7,7,6]
Explanation: The subsequence [7,7] has the sum of its elements equal to 14 which is not strictly greater than the sum of elements not included (14 = 4 + 4 + 6). Therefore, the subsequence [7,6,7] is the minimal satisfying the condition s. Note the subsequence has to returned in non-decreasing order.

Example 3:

Input: nums = [6]
Output: [6]

Constraints:

1 <= nums.length <= 500
1 <= nums[i] <= 100

1404. Number of Steps to Reduce a Number in Binary Representation to One

Given the binary representation of an integer as a string s, return the number of steps to reduce it to 1 under the follo wing rules:

If the current number is even, you have to divide it by 2.

If the current number is odd, you have to add 1 to it.

It is guaranteed that you can always reach one for all test cases.

Example 1:

Input: s = "1101"
Output: 6
Explanation: "1101" corressponds to number 13 in their decimal representation.
Step 1) 13 is odd, add 1 and obtain 14.
Step 2) 14 is even, divide by 2 and obtain 7.
Step 3) 7 is odd, add 1 and obtain 8.
Step 4) 8 is even, divide by 2 and obtain 4.
Step 5) 4 is even, divide by 2 and obtain 2.
Step 6) 2 is even, divide by 2 and obtain 1.

Example 2:

Input: s = "10"
Output: 1
Explanation: "10" corressponds to number 2 in their decimal representation.
Step 1) 2 is even, divide by 2 and obtain 1.

Example 3:

Input: s = "1"
Output: 0

Constraints:

1 <= s.length <= 500
s consists of characters '0' or '1'
s[0] == '1'

1405. Longest Happy String

A string is called happy if it does not have any of the strings 'aaa', 'bbb' or 'ccc' as a substring.
Given three integers a, b and c, return any string s, which satisfies following conditions:

s is happy and longest possible.
s contains at most a occurrences of the letter 'a', at most b occurrences of the letter 'b' and at most c occurrences of th
e letter 'c'.
s will only contain 'a', 'b' and 'c' letters.

If there is no such string s return the empty string "".

Example 1:

Input: a = 1, b = 1, c = 7
Output: "ccaccbcc"
Explanation: "ccbccacc" would also be a correct answer.

Example 2:

Input: a = 2, b = 2, c = 1
Output: "aabbc"

Example 3:

Input: a = 7, b = 1, c = 0
Output: "aabaa"
Explanation: It's the only correct answer in this case.

Constraints:

0 <= a, b, c <= 100
a + b + c > 0

1406. Stone Game III

Alice and Bob continue their games with piles of stones. There are several stones arranged in a row, and each stone has an associated value which is an integer given in the array stoneValue.
Alice and Bob take turns, with Alice starting first. On each player's turn, that player can take 1, 2 or 3 stones from th e first remaining stones in the row.
The score of each player is the sum of values of the stones taken. The score of each player is 0 initially.
The objective of the game is to end with the highest score, and the winner is the player with the highest score and the re could be a tie. The game continues until all the stones have been taken.
Assume Alice and Bob play optimally.
Return "Alice" if Alice will win, "Bob" if Bob will win or "Tie" if they end the game with the same score.

Example 1:

Input: values = [1,2,3,7]
Output: "Bob"
Explanation: Alice will always lose. Her best move will be to take three piles and the score become 6. Now the score of Bob is 7 and Bob wins.

Example 2:

Input: values = [1,2,3,-9]
Output: "Alice"
Explanation: Alice must choose all the three piles at the first move to win and leave Bob with negative score.
If Alice chooses one pile her score will be 1 and the next move Bob's score becomes 5. The next move Alice will tak e the pile with value = -9 and lose.
If Alice chooses two piles her score will be 3 and the next move Bob's score becomes 3. The next move Alice will ta ke the pile with value = -9 and also lose.
Remember that both play optimally so here Alice will choose the scenario that makes her win.

Example 3:

Input: values = [1,2,3,6]
Output: "Tie"
Explanation: Alice cannot win this game. She can end the game in a draw if she decided to choose all the first three piles, otherwise she will lose.

Example 4:

Input: values = [1,2,3,-1,-2,-3,7]
Output: "Alice"

Example 5:

Input: values = [-1,-2,-3]
Output: "Tie"

Constraints:

1 <= values.length <= 50000
-1000 <= values[i] <= 1000

1408. String Matching in an Array

Given an array of string words. Return all strings in words which is substring of another word in any order.
String words[i] is substring of words[j], if can be obtained removing some characters to left and/or right side of word s[j].

Example 1:

Input: words = ["mass","as","hero","superhero"]
Output: ["as","hero"]
Explanation: "as" is substring of "mass" and "hero" is substring of "superhero".
["hero","as"] is also a valid answer.

Example 2:

Input: words = ["leetcode","et","code"]
Output: ["et","code"]
Explanation: "et", "code" are substring of "leetcode".

Example 3:

Input: words = ["blue","green","bu"]
Output: []

Constraints:

1 <= words.length <= 100
1 <= words[i].length <= 30
words[i] contains only lowercase English letters.
It's guaranteed that words[i] will be unique.

1409. Queries on a Permutation With Key

Given the array queries of positive integers between 1 and m, you have to process all queries[i] (from i=0 to i=queries.length-1) according to the following rules:

In the beginning, you have the permutation P=[1,2,3,...,m].
For the current i, find the position of queries[i] in the permutation P (indexing from 0) and then move this at the begi nning of the permutation P. Notice that the position of queries[i] in P is the result for queries[i].

Return an array containing the result for the given queries.

Example 1:

Input: queries = [3,1,2,1], m = 5
Output: [2,1,2,1]
Explanation: The queries are processed as follow:
For i=0: queries[i]=3, P=[1,2,3,4,5], position of 3 in P is 2, then we move 3 to the beginning of P resulting in P=[3,1, 2,4,5].
For i=1: queries[i]=1, P=[3,1,2,4,5], position of 1 in P is 1, then we move 1 to the beginning of P resulting in P=[1,3, 2,4,5].
For i=2: queries[i]=2, P=[1,3,2,4,5], position of 2 in P is 2, then we move 2 to the beginning of P resulting in P=[2,1, 3,4,5].
For i=3: queries[i]=1, P=[2,1,3,4,5], position of 1 in P is 1, then we move 1 to the beginning of P resulting in P=[1,2, 3,4,5].
Therefore, the array containing the result is [2,1,2,1].

Example 2:

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

Example 3:

Input: queries = [7,5,5,8,3], m = 8
Output: [6,5,0,7,5]

Constraints:

1 <= m <= 10^3
1 <= queries.length <= m
1 <= queries[i] <= m

1410. HTML Entity Parser

HTML entity parser is the parser that takes HTML code as input and replace all the entities of the special characters by the characters itself.
The special characters and their entities for HTML are:

Quotation Mark: the entity is " and symbol character is ".
Single Quote Mark: the entity is ' and symbol character is '.

Ampersand: the entity is & and symbol character is &.
Greater Than Sign: the entity is > and symbol character is >.
Less Than Sign: the entity is < and symbol character is <.
Slash: the entity is ⁄ and symbol character is /.

Given the input text string to the HTML parser, you have to implement the entity parser.
Return the text after replacing the entities by the special characters.

Example 1:

Input: text = "& is an HTML entity but &ambassador; is not."
Output: "& is an HTML entity but &ambassador; is not."
Explanation: The parser will replace the & entity by &

Example 2:

Input: text = "and I quote: "...""
Output: "and I quote: \"...\""

Example 3:

Input: text = "Stay home! Practice on Leetcode :)"
Output: "Stay home! Practice on Leetcode :)"

Example 4:

Input: text = "x > y && x < y is always false"
Output: "x > y && x < y is always false"

Example 5:

Input: text = "leetcode.com⁄problemset⁄all"
Output: "leetcode.com/problemset/all"

Constraints:

1 <= text.length <= 10^5
The string may contain any possible characters out of all the 256 ASCII characters.

1411. Number of Ways to Paint N × 3 Grid

You have a grid of size n x 3 and you want to paint each cell of the grid with exactly one of the three colors: Red, Ye llow, or Green while making sure that no two adjacent cells have the same color (i.e., no two cells that share vertical or horizontal sides have the same color).
Given n the number of rows of the grid, return the number of ways you can paint this grid. As the answer may grow

large, the answer must be computed modulo 109 + 7.

Example 1:

Input: n = 1
Output: 12
Explanation: There are 12 possible way to paint the grid as shown.

Example 2:

Input: n = 2
Output: 54

Example 3:

Input: n = 3
Output: 246

Example 4:

Input: n = 7
Output: 106494

Example 5:

Input: n = 5000
Output: 30228214

Constraints:

n == grid.length
grid[i].length == 3
1 <= n <= 5000

1413. Minimum Value to Get Positive Step by Step Sum

Given an array of integers nums, you start with an initial positive value startValue.
In each iteration, you calculate the step by step sum of startValue plus elements in nums (from left to right).
Return the minimum positive value of startValue such that the step by step sum is never less than 1.

Example 1:

Input: nums = [-3,2,-3,4,2]
Output: 5

Explanation: If you choose startValue = 4, in the third iteration your step by step sum is less than 1.
step by step sum
startValue = 4 | startValue = 5 | nums
(4 -3) = 1 | (5 -3) = 2 | -3
(1 +2) = 3 | (2 +2) = 4 | 2
(3 -3) = 0 | (4 -3) = 1 | -3
(0 +4) = 4 | (1 +4) = 5 | 4
(4 +2) = 6 | (5 +2) = 7 | 2

Example 2:

Input: nums = [1,2]
Output: 1
Explanation: Minimum start value should be positive.

Example 3:

Input: nums = [1,-2,-3]
Output: 5

Constraints:

1 <= nums.length <= 100
-100 <= nums[i] <= 100

1414. Find the Minimum Number of Fibonacci Numbers Whose Sum Is K

Given an integer k, return the minimum number of Fibonacci numbers whose sum is equal to k. The same Fibonacci number can be used multiple times.
The Fibonacci numbers are defined as:

F1 = 1
F2 = 1
Fn = Fn-1 + Fn-2 for n > 2.

It is guaranteed that for the given constraints we can always find such Fibonacci numbers that sum up to k.

Example 1:

Input: k = 7
Output: 2
Explanation: The Fibonacci numbers are: 1, 1, 2, 3, 5, 8, 13, ...
For k = 7 we can use 2 + 5 = 7.
Example 2:

Input: k = 10
Output: 2
Explanation: For k = 10 we can use 2 + 8 = 10.

Example 3:

Input: k = 19
Output: 3
Explanation: For k = 19 we can use 1 + 5 + 13 = 19.

Constraints:

1 <= k <= 109

1415. The k-th Lexicographical String of All Happy Strings of Length n

A happy string is a string that:

consists only of letters of the set ['a', 'b', 'c'].
s[i] != s[i + 1] for all values of i from 1 to s.length - 1 (string is 1-indexed).

For example, strings "abc", "ac", "b" and "abcbabcbcb" are all happy strings and strings "aa", "baa" and "ababbc" are not happy strings.
Given two integers n and k, consider a list of all happy strings of length n sorted in lexicographical order.
Return the kth string of this list or return an empty string if there are less than k happy strings of length n.

Example 1:

Input: n = 1, k = 3
Output: "c"
Explanation: The list ["a", "b", "c"] contains all happy strings of length 1. The third string is "c".

Example 2:

Input: n = 1, k = 4
Output: ""
Explanation: There are only 3 happy strings of length 1.

Example 3:

Input: n = 3, k = 9
Output: "cab"
Explanation: There are 12 different happy string of length 3 ["aba", "abc", "aca", "acb", "bab", "bac", "bca", "bcb", "cab", "cac", "cba", "cbc"]. You will find the 9th string = "cab"

Example 4:

Input: n = 2, k = 7
Output: ""

Example 5:

Input: n = 10, k = 100
Output: "abacbabacb"

Constraints:

1 <= n <= 10
1 <= k <= 100

1416. Restore The Array

A program was supposed to print an array of integers. The program forgot to print whitespaces and the array is print ed as a string of digits s and all we know is that all integers in the array were in the range [1, k] and there are no leadi ng zeros in the array.
Given the string s and the integer k, return the number of the possible arrays that can be printed as s using the mentio ned program. Since the answer may be very large, return it modulo 109 + 7.

Example 1:

Input: s = "1000", k = 10000
Output: 1
Explanation: The only possible array is [1000]

Example 2:

Input: s = "1000", k = 10
Output: 0
Explanation: There cannot be an array that was printed this way and has all integer >= 1 and <= 10.

Example 3:

Input: s = "1317", k = 2000
Output: 8
Explanation: Possible arrays are [1317],[131,7],[13,17],[1,317],[13,1,7],[1,31,7],[1,3,17],[1,3,1,7]

Example 4:

Input: s = "2020", k = 30
Output: 1

Explanation: The only possible array is [20,20]. [2020] is invalid because 2020 > 30. [2,020] is ivalid because 020 c ontains leading zeros.

Example 5:

Input: s = "1234567890", k = 90
Output: 34

Constraints:

1 <= s.length <= 105
s consists of only digits and does not contain leading zeros.
1 <= k <= 109

1417. Reformat The String

Given alphanumeric string s. (Alphanumeric string is a string consisting of lowercase English letters and digits).
You have to find a permutation of the string where no letter is followed by another letter and no digit is followed by another digit. That is, no two adjacent characters have the same type.
Return the reformatted string or return an empty string if it is impossible to reformat the string.

Example 1:

Input: s = "a0b1c2"
Output: "0a1b2c"
Explanation: No two adjacent characters have the same type in "0a1b2c". "a0b1c2", "0a1b2c", "0c2a1b" are also vali d permutations.

Example 2:

Input: s = "leetcode"
Output: ""
Explanation: "leetcode" has only characters so we cannot separate them by digits.

Example 3:

Input: s = "1229857369"
Output: ""
Explanation: "1229857369" has only digits so we cannot separate them by characters.

Example 4:

Input: s = "covid2019"
Output: "c2o0v1i9d"

Example 5:

Input: s = "ab123"
Output: "1a2b3"

Constraints:

1 <= s.length <= 500
s consists of only lowercase English letters and/or digits.

1418. Display Table of Food Orders in a Restaurant

Given the array orders, which represents the orders that customers have done in a restaurant. More specifically order s[i]=[customerNamei,tableNumberi,foodItemi] where customerNamei is the name of the customer, tableNumberi is t he table customer sit at, and foodItemi is the item customer orders.
Return the restaurant's “display table”. The “display table” is a table whose row entries denote how many of each fo od item each table ordered. The first column is the table number and the remaining columns correspond to each food item in alphabetical order. The first row should be a header whose first column is “Table”, followed by the names of the food items. Note that the customer names are not part of the table. Additionally, the rows should be sorted in nu merically increasing order.

Example 1:

Input: orders = [["David","3","Ceviche"],["Corina","10","Beef Burrito"],["David","3","Fried Chicken"],["Carla","5" ,"Water"],["Carla","5","Ceviche"],["Rous","3","Ceviche"]]
Output: [["Table","Beef Burrito","Ceviche","Fried Chicken","Water"],["3","0","2","1","0"],["5","0","1","0","1"],["1 0","1","0","0","0"]]
Explanation:
The displaying table looks like:
Table,Beef Burrito,Ceviche,Fried Chicken,Water
3 ,0 ,2 ,1 ,0
5 ,0 ,1 ,0 ,1
10 ,1 ,0 ,0 ,0
For the table 3: David orders "Ceviche" and "Fried Chicken", and Rous orders "Ceviche".
For the table 5: Carla orders "Water" and "Ceviche".
For the table 10: Corina orders "Beef Burrito".

Example 2:

Input: orders = [["James","12","Fried Chicken"],["Ratesh","12","Fried Chicken"],["Amadeus","12","Fried Chicken"],["Adam","1","Canadian Waffles"],["Brianna","1","Canadian Waffles"]]
Output: [["Table","Canadian Waffles","Fried Chicken"],["1","2","0"],["12","0","3"]]
Explanation:
For the table 1: Adam and Brianna order "Canadian Waffles".
For the table 12: James, Ratesh and Amadeus order "Fried Chicken".

Example 3:

Input: orders = [["Laura","2","Bean Burrito"],["Jhon","2","Beef Burrito"],["Melissa","2","Soda"]]
Output: [["Table","Bean Burrito","Beef Burrito","Soda"],["2","1","1","1"]]

Constraints:

1 <= orders.length <= 5 * 10^4
orders[i].length == 3
1 <= customerNamei.length, foodItemi.length <= 20
customerNamei and foodItemi consist of lowercase and uppercase English letters and the space character.
tableNumberi is a valid integer between 1 and 500.

1419. Minimum Number of Frogs Croaking

You are given the string croakOfFrogs, which represents a combination of the string "croak" from different frogs, th at is, multiple frogs can croak at the same time, so multiple "croak" are mixed.
Return the minimum number of different frogs to finish all the croaks in the given string.
A valid "croak" means a frog is printing five letters 'c', 'r', 'o', 'a', and 'k' sequentially. The frogs have to print all five l etters to finish a croak. If the given string is not a combination of a valid "croak" return -1.

Example 1:

Input: croakOfFrogs = "croakcroak"
Output: 1
Explanation: One frog yelling "croak" twice.

Example 2:

Input: croakOfFrogs = "crcoakroak"
Output: 2
Explanation: The minimum number of frogs is two.
The first frog could yell "crcoakroak".
The second frog could yell later "crcoakroak".

Example 3:

Input: croakOfFrogs = "croakcrook"
Output: -1
Explanation: The given string is an invalid combination of "croak" from different frogs.

Constraints:

1 <= croakOfFrogs.length <= 105

croakOfFrogs is either 'c', 'r', 'o', 'a', or 'k'.

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

Given three integers n, m and k. Consider the following algorithm to find the maximum element of an array of positi ve 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 10^9 + 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 satisify 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]

Example 4:

Input: n = 50, m = 100, k = 25
Output: 34549172
Explanation: Don't forget to compute the answer modulo 1000000007

Example 5:

Input: n = 37, m = 17, k = 7
Output: 418930126

Constraints:

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

1422. Maximum Score After Splitting a String

Given a string s of zeros and ones, return the maximum score after splitting the string into two non-empty substrings (i.e. left substring and right substring).
The score after splitting a string is the number of zeros in the left substring plus the number of ones in the right subst ring.

Example 1:

Input: s = "011101"
Output: 5
Explanation:
All possible ways of splitting s into two non-empty substrings are:
left = "0" and right = "11101", score = 1 + 4 = 5
left = "01" and right = "1101", score = 1 + 3 = 4
left = "011" and right = "101", score = 1 + 2 = 3
left = "0111" and right = "01", score = 1 + 1 = 2
left = "01110" and right = "1", score = 2 + 1 = 3

Example 2:

Input: s = "00111"
Output: 5
Explanation: When left = "00" and right = "111", we get the maximum score = 2 + 3 = 5

Example 3:

Input: s = "1111"
Output: 3

Constraints:

2 <= s.length <= 500
The string s consists of characters '0' and '1' only.

1423. Maximum Points You Can Obtain from Cards

There are several cards arranged in a row, and each card has an associated number of points. The points are given in the integer array cardPoints.
In one step, you can take one card from the beginning or from the end of the row. You have to take exactly k cards.
Your score is the sum of the points of the cards you have taken.
Given the integer array cardPoints and the integer k, return the maximum score you can obtain.

Example 1:

Input: cardPoints = [1,2,3,4,5,6,1], k = 3
Output: 12
Explanation: After the first step, your score will always be 1. However, choosing the rightmost card first will maxim ize your total score. The optimal strategy is to take the three cards on the right, giving a final score of 1 + 6 + 5 = 12.

Example 2:

Input: cardPoints = [2,2,2], k = 2
Output: 4
Explanation: Regardless of which two cards you take, your score will always be 4.

Example 3:

Input: cardPoints = [9,7,7,9,7,7,9], k = 7
Output: 55
Explanation: You have to take all the cards. Your score is the sum of points of all cards.

Example 4:

Input: cardPoints = [1,1000,1], k = 1
Output: 1
Explanation: You cannot take the card in the middle. Your best score is 1.

Example 5:

Input: cardPoints = [1,79,80,1,1,1,200,1], k = 3
Output: 202

Constraints:

1 <= cardPoints.length <= 105
1 <= cardPoints[i] <= 104
1 <= k <= cardPoints.length

1424. Diagonal Traverse II

Given a list of lists of integers, nums, return all elements of nums in diagonal order as shown in the below images.

Example 1:

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

Example 2:

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

Example 3:

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

Example 4:

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

Constraints:

1 <= nums.length <= 10^5
1 <= nums[i].length <= 10^5
1 <= nums[i][j] <= 10^9
There at most 10^5 elements in nums.

1425. Constrained Subsequence Sum

Given an integer array nums and an integer k, return the maximum sum of a non-empty subsequence of that array su ch that for every two consecutive integers in the subsequence, nums[i] and nums[j], where i < j, the condition j - i <= k is satisfied.
A subsequence of an array is obtained by deleting some number of elements (can be zero) from the array, leaving the remaining elements in their original order.

Example 1:

Input: nums = [10,2,-10,5,20], k = 2
Output: 37
Explanation: The subsequence is [10, 2, 5, 20].

Example 2:

Input: nums = [-1,-2,-3], k = 1
Output: -1
Explanation: The subsequence must be non-empty, so we choose the largest number.

Example 3:

Input: nums = [10,-2,-10,-5,20], k = 2
Output: 23
Explanation: The subsequence is [10, -2, -5, 20].

Constraints:

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

1431. Kids With the Greatest Number of Candies

There are n kids with candies. You are given an integer array candies, where each candies[i] represents the number o f candies the ith kid has, and an integer extraCandies, denoting the number of extra candies that you have.
Return a boolean array result of length n, where result[i] is true if, after giving the ith kid all the extraCandies, they will have the greatest number of candies among all the kids, or false otherwise.
Note that multiple kids can have the greatest number of candies.

Example 1:

Input: candies = [2,3,5,1,3], extraCandies = 3
Output: [true,true,true,false,true]
Explanation: If you give all extraCandies to:
- Kid 1, they will have 2 + 3 = 5 candies, which is the greatest among the kids.
- Kid 2, they will have 3 + 3 = 6 candies, which is the greatest among the kids.
- Kid 3, they will have 5 + 3 = 8 candies, which is the greatest among the kids.
- Kid 4, they will have 1 + 3 = 4 candies, which is not the greatest among the kids.
- Kid 5, they will have 3 + 3 = 6 candies, which is the greatest among the kids.

Example 2:

Input: candies = [4,2,1,1,2], extraCandies = 1
Output: [true,false,false,false,false]
Explanation: There is only 1 extra candy.

Kid 1 will always have the greatest number of candies, even if a different kid is given the extra candy.

Example 3:

Input: candies = [12,1,12], extraCandies = 10
Output: [true,false,true]

Constraints:

n == candies.length
2 <= n <= 100
1 <= candies[i] <= 100
1 <= extraCandies <= 50

1432. Max Difference You Can Get From Changing an Integer

You are given an integer num. You will apply the following steps exactly two times:

Pick a digit x (0 <= x <= 9).
Pick another digit y (0 <= y <= 9). The digit y can be equal to x.
Replace all the occurrences of x in the decimal representation of num by y.
The new integer cannot have any leading zeros, also the new integer cannot be 0.

Let a and b be the results of applying the operations to num the first and second times, respectively.
Return the max difference between a and b.

Example 1:

Input: num = 555
Output: 888
Explanation: The first time pick x = 5 and y = 9 and store the new integer in a.
The second time pick x = 5 and y = 1 and store the new integer in b.
We have now a = 999 and b = 111 and max difference = 888

Example 2:

Input: num = 9
Output: 8
Explanation: The first time pick x = 9 and y = 9 and store the new integer in a.
The second time pick x = 9 and y = 1 and store the new integer in b.
We have now a = 9 and b = 1 and max difference = 8

Example 3:

Input: num = 123456

Output: 820000

Example 4:

Input: num = 10000
Output: 80000

Example 5:

Input: num = 9288
Output: 8700

Constraints:

1 <= num <= 10^8

1433. Check If a String Can Break Another String

Given two strings: s1 and s2 with the same size, check if some permutation of string s1 can break some permutation of string s2 or vice-versa. In other words s2 can break s1 or vice-versa.
A string x can break string y (both of size n) if x[i] >= y[i] (in alphabetical order) for all i between 0 and n-1.

Example 1:

Input: s1 = "abc", s2 = "xya"
Output: true
Explanation: "ayx" is a permutation of s2="xya" which can break to string "abc" which is a permutation of s1="abc".

Example 2:

Input: s1 = "abe", s2 = "acd"
Output: false
Explanation: All permutations for s1="abe" are: "abe", "aeb", "bae", "bea", "eab" and "eba" and all permutation for s2="acd" are: "acd", "adc", "cad", "cda", "dac" and "dca". However, there is not any permutation from s1 which can break some permutation from s2 and vice-versa.

Example 3:

Input: s1 = "leetcodee", s2 = "interview"
Output: true

Constraints:

s1.length == n

s2.length == n
1 <= n <= 10^5
All strings consist of lowercase English letters.

1434. Number of Ways to Wear Different Hats to Each Other

There are n people and 40 types of hats labeled from 1 to 40.
Given a list of list of integers hats, where hats[i] is a list of all hats preferred by the i-th person.
Return the number of ways that the n people wear different hats to each other.
Since the answer may be too large, return it modulo 10^9 + 7.

Example 1:

Input: hats = [[3,4],[4,5],[5]]
Output: 1
Explanation: There is only one way to choose hats given the conditions.
First person choose hat 3, Second person choose hat 4 and last one hat 5.
Example 2:

Input: hats = [[3,5,1],[3,5]]
Output: 4
Explanation: There are 4 ways to choose hats
(3,5), (5,3), (1,3) and (1,5)

Example 3:

Input: hats = [[1,2,3,4],[1,2,3,4],[1,2,3,4],[1,2,3,4]]
Output: 24
Explanation: Each person can choose hats labeled from 1 to 4.
Number of Permutations of (1,2,3,4) = 24.

Example 4:

Input: hats = [[1,2,3],[2,3,5,6],[1,3,7,9],[1,8,9],[2,5,7]]
Output: 111

Constraints:

n == hats.length
1 <= n <= 10
1 <= hats[i].length <= 40
1 <= hats[i][j] <= 40
hats[i] contains a list of unique integers.

1436. Destination City

You are given the array paths, where paths[i] = [cityAi, cityBi] means there exists a direct path going from cityAi to cityBi. Return the destination city, that is, the city without any path outgoing to another city.
It is guaranteed that the graph of paths forms a line without any loop, therefore, there will be exactly one destination city.

Example 1:

Input: paths = [["London","New York"],["New York","Lima"],["Lima","Sao Paulo"]]
Output: "Sao Paulo"
Explanation: Starting at "London" city you will reach "Sao Paulo" city which is the destination city. Your trip consis t of: "London" -> "New York" -> "Lima" -> "Sao Paulo".

Example 2:

Input: paths = [["B","C"],["D","B"],["C","A"]]
Output: "A"
Explanation: All possible trips are:
"D" -> "B" -> "C" -> "A".
"B" -> "C" -> "A".
"C" -> "A".
"A".
Clearly the destination city is "A".

Example 3:

Input: paths = [["A","Z"]]
Output: "Z"

Constraints:

1 <= paths.length <= 100
paths[i].length == 2
1 <= cityAi.length, cityBi.length <= 10
cityAi != cityBi
All strings consist of lowercase and uppercase English letters and the space character.

1437. Check If All 1's Are at Least Length K Places Away

Given an array nums of 0s and 1s and an integer k, return True if all 1's are at least k places away from each other, ot herwise return False.

Example 1:

Input: nums = [1,0,0,0,1,0,0,1], k = 2
Output: true
Explanation: Each of the 1s are at least 2 places away from each other.

Example 2:

Input: nums = [1,0,0,1,0,1], k = 2
Output: false
Explanation: The second 1 and third 1 are only one apart from each other.
Example 3:

Input: nums = [1,1,1,1,1], k = 0
Output: true

Example 4:

Input: nums = [0,1,0,1], k = 1
Output: true

Constraints:

1 <= nums.length <= 105
0 <= k <= nums.length
nums[i] is 0 or 1

1438. Longest Continuous Subarray With Absolute Diff Less Than or Equal to Limit

Given an array of integers nums and an integer limit, return the size of the longest non-empty subarray such that the absolute difference between any two elements of this subarray is less than or equal to limit.

Example 1:

Input: nums = [8,2,4,7], limit = 4
Output: 2
Explanation: All subarrays are:
[8] with maximum absolute diff |8-8| = 0 <= 4.
[8,2] with maximum absolute diff |8-2| = 6 > 4.

[8,2,4] with maximum absolute diff |8-2| = 6 > 4.
[8,2,4,7] with maximum absolute diff |8-2| = 6 > 4.
[2] with maximum absolute diff |2-2| = 0 <= 4.
[2,4] with maximum absolute diff |2-4| = 2 <= 4.
[2,4,7] with maximum absolute diff |2-7| = 5 > 4.
[4] with maximum absolute diff |4-4| = 0 <= 4.
[4,7] with maximum absolute diff |4-7| = 3 <= 4.
[7] with maximum absolute diff |7-7| = 0 <= 4.
Therefore, the size of the longest subarray is 2.

Example 2:

Input: nums = [10,1,2,4,7,2], limit = 5
Output: 4
Explanation: The subarray [2,4,7,2] is the longest since the maximum absolute diff is |2-7| = 5 <= 5.

Example 3:

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

Constraints:

1 <= nums.length <= 105
1 <= nums[i] <= 109
0 <= limit <= 109

1439. Find the Kth Smallest Sum of a Matrix With Sorted Rows

You are given an m * n matrix, mat, and an integer k, which has its rows sorted in non-decreasing order.
You are allowed to choose exactly 1 element from each row to form an array. Return the Kth smallest array sum am ong all possible arrays.

Example 1:

Input: mat = [[1,3,11],[2,4,6]], k = 5
Output: 7
Explanation: Choosing one element from each row, the first k smallest sum are:
[1,2], [1,4], [3,2], [3,4], [1,6]. Where the 5th sum is 7.
Example 2:

Input: mat = [[1,3,11],[2,4,6]], k = 9
Output: 17

Example 3:

Input: mat = [[1,10,10],[1,4,5],[2,3,6]], k = 7
Output: 9
Explanation: Choosing one element from each row, the first k smallest sum are:
[1,1,2], [1,1,3], [1,4,2], [1,4,3], [1,1,6], [1,5,2], [1,5,3]. Where the 7th sum is 9.

Example 4:

Input: mat = [[1,1,10],[2,2,9]], k = 7
Output: 12

Constraints:

m == mat.length
n == mat.length[i]
1 <= m, n <= 40
1 <= k <= min(200, n ^ m)
1 <= mat[i][j] <= 5000
mat[i] is a non decreasing array.

1441. Build an Array With Stack Operations

Given an array target and an integer n. In each iteration, you will read a number from list = {1,2,3..., n}.
Build the target array using the following operations:

Push: Read a new element from the beginning list, and push it in the array.
Pop: delete the last element of the array.
If the target array is already built, stop reading more elements.

Return the operations to build the target array. You are guaranteed that the answer is unique.

Example 1:

Input: target = [1,3], n = 3
Output: ["Push","Push","Pop","Push"]
Explanation:
Read number 1 and automatically push in the array -> [1]
Read number 2 and automatically push in the array then Pop it -> [1]
Read number 3 and automatically push in the array -> [1,3]

Example 2:

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

Example 3:

Input: target = [1,2], n = 4
Output: ["Push","Push"]
Explanation: You only need to read the first 2 numbers and stop.

Example 4:

Input: target = [2,3,4], n = 4
Output: ["Push","Pop","Push","Push","Push"]

Constraints:

1 <= target.length <= 100
1 <= target[i] <= n
1 <= n <= 100
target is strictly increasing.

1442. Count Triplets That Can Form Two Arrays of Equal XOR

Given an array of integers arr.
We want to select three indices i, j and k where (0 <= i < j <= k < arr.length).
Let's define a and b as follows:

a = arr[i] ^ arr[i + 1] ^ ... ^ arr[j - 1]
b = arr[j] ^ arr[j + 1] ^ ... ^ arr[k]

Note that ^ denotes the bitwise-xor operation.
Return the number of triplets (i, j and k) Where a == b.

Example 1:

Input: arr = [2,3,1,6,7]
Output: 4
Explanation: The triplets are (0,1,2), (0,2,2), (2,3,4) and (2,4,4)

Example 2:

Input: arr = [1,1,1,1,1]
Output: 10

Example 3:

Input: arr = [2,3]
Output: 0

Example 4:

Input: arr = [1,3,5,7,9]
Output: 3

Example 5:

Input: arr = [7,11,12,9,5,2,7,17,22]
Output: 8

Constraints:

1 <= arr.length <= 300
1 <= arr[i] <= 10^8

1443. Minimum Time to Collect All Apples in a Tree

Given an undirected tree consisting of n vertices numbered from 0 to n-1, which has some apples in their vertices. You spend 1 second to walk over one edge of the tree. Return the minimum time in seconds you have to spend to collect all apples in the tree, starting at vertex 0 and coming back to this vertex.
The edges of the undirected tree are given in the array edges, where edges[i] = [ai, bi] means that exists an edge connecting the vertices ai and bi. Additionally, there is a boolean array hasApple, where hasApple[i] = true means that vertex i has an apple; otherwise, it does not have any apple.

Example 1:

Input: n = 7, edges = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], hasApple = [false,false,true,false,true,true,false]
Output: 8
Explanation: The figure above represents the given tree where red vertices have an apple. One optimal path to collect all apples is shown by the green arrows.

Example 2:

Input: n = 7, edges = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], hasApple = [false,false,true,false,false,true,false]
Output: 6
Explanation: The figure above represents the given tree where red vertices have an apple. One optimal path to collect all apples is shown by the green arrows.

Example 3:

Input: n = 7, edges = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], hasApple = [false,false,false,false,false,false,false]
Output: 0

Constraints:

1 <= n <= 105
edges.length == n - 1
edges[i].length == 2
0 <= ai < bi <= n - 1
fromi < toi
hasApple.length == n

1444. Number of Ways of Cutting a Pizza

Given a rectangular pizza represented as a rows x cols matrix containing the following characters: 'A' (an apple) and '.' (empty cell) and given the integer k. You have to cut the pizza into k pieces using k-1 cuts.
For each cut you choose the direction: vertical or horizontal, then you choose a cut position at the cell boundary and cut the pizza into two pieces. If you cut the pizza vertically, give the left part of the pizza to a person. If you cut the p izza horizontally, give the upper part of the pizza to a person. Give the last piece of pizza to the last person.
Return the number of ways of cutting the pizza such that each piece contains at least one apple. Since the answer can be a huge number, return this modulo 10^9 + 7.

Example 1:

Input: pizza = ["A..","AAA","..."], k = 3
Output: 3
Explanation: The figure above shows the three ways to cut the pizza. Note that pieces must contain at least one apple .

Example 2:

Input: pizza = ["A..","AA.","..."], k = 3
Output: 1

Example 3:

Input: pizza = ["A..","A..","..."], k = 1
Output: 1

Constraints:

1 <= rows, cols <= 50
rows == pizza.length
cols == pizza[i].length
1 <= k <= 10
pizza consists of characters 'A' and '.' only.

1446. Consecutive Characters

The power of the string is the maximum length of a non-empty substring that contains only one unique character. Given a string s, return the power of s.

Example 1:

Input: s = "leetcode"
Output: 2
Explanation: The substring "ee" is of length 2 with the character 'e' only.

Example 2:

Input: s = "abbcccddddeeeeedcba"
Output: 5
Explanation: The substring "eeeee" is of length 5 with the character 'e' only.

Example 3:

Input: s = "triplepilloooooow"
Output: 5

Example 4:

Input: s = "hooraaaaaaaaaaay"
Output: 11

Example 5:

Input: s = "tourist"
Output: 1

Constraints:

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

1447. Simplified Fractions

Given an integer n, return a list of all simplified fractions between 0 and 1 (exclusive) such that the denominator is le ss-than-or-equal-to n. The fractions can be in any order.

Example 1:

Input: n = 2
Output: ["1/2"]
Explanation: "1/2" is the only unique fraction with a denominator less-than-or-equal-to 2.
Example 2:

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

Example 3:

Input: n = 4
Output: ["1/2","1/3","1/4","2/3","3/4"]
Explanation: "2/4" is not a simplified fraction because it can be simplified to "1/2".
Example 4:

Input: n = 1
Output: []

Constraints:

1 <= n <= 100

1448. Count Good Nodes in Binary Tree

Given a binary tree root, a node X in the tree is named good if in the path from root to X there are no nodes with a va lue greater than X.
Return the number of good nodes in the binary tree.

Example 1:

Input: root = [3,1,4,3,null,1,5]
Output: 4
Explanation: Nodes in blue are good.
Root Node (3) is always a good node.
Node 4 -> (3,4) is the maximum value in the path starting from the root.
Node 5 -> (3,4,5) is the maximum value in the path
Node 3 -> (3,1,3) is the maximum value in the path.

Example 2:

Input: root = [3,3,null,4,2]
Output: 3
Explanation: Node 2 -> (3, 3, 2) is not good, because "3" is higher than it.
Example 3:

Input: root = [1]
Output: 1
Explanation: Root is considered as good.

Constraints:

The number of nodes in the binary tree is in the range [1, 10^5].
Each node's value is between [-10^4, 10^4].

1449. Form Largest Integer With Digits That Add up to Target

Given an array of integers cost and an integer target. Return the maximum integer you can paint under the following rules:

The cost of painting a digit (i+1) is given by cost[i] (0 indexed).
The total cost used must be equal to target.
Integer does not have digits 0.

Since the answer may be too large, return it as string.
If there is no way to paint any integer given the condition, return "0".

Example 1:

Input: cost = [4,3,2,5,6,7,2,5,5], target = 9
Output: "7772"
Explanation: The cost to paint the digit '7' is 2, and the digit '2' is 3. Then cost("7772") = 2*3+ 3*1 = 9. You could a lso paint "977", but "7772" is the largest number.
Digit cost
 1 -> 4
 2 -> 3
 3 -> 2
 4 -> 5
 5 -> 6
 6 -> 7
 7 -> 2
 8 -> 5
 9 -> 5

Example 2:

Input: cost = [7,6,5,5,5,6,8,7,8], target = 12
Output: "85"
Explanation: The cost to paint the digit '8' is 7, and the digit '5' is 5. Then cost("85") = 7 + 5 = 12.

Example 3:

Input: cost = [2,4,6,2,4,6,4,4,4], target = 5
Output: "0"
Explanation: It's not possible to paint any integer with total cost equal to target.

Example 4:

Input: cost = [6,10,15,40,40,40,40,40,40], target = 47
Output: "32211"

Constraints:

cost.length == 9
1 <= cost[i] <= 5000
1 <= target <= 5000

1450. Number of Students Doing Homework at a Given Time

Given two integer arrays startTime and endTime and given an integer queryTime.
The ith student started doing their homework at the time startTime[i] and finished it at time endTime[i].
Return the number of students doing their homework at time queryTime. More formally, return the number of studen ts where queryTime lays in the interval [startTime[i], endTime[i]] inclusive.

Example 1:

Input: startTime = [1,2,3], endTime = [3,2,7], queryTime = 4
Output: 1
Explanation: We have 3 students where:
The first student started doing homework at time 1 and finished at time 3 and wasn't doing anything at time 4.
The second student started doing homework at time 2 and finished at time 2 and also wasn't doing anything at time 4 .
The third student started doing homework at time 3 and finished at time 7 and was the only student doing homework at time 4.

Example 2:

Input: startTime = [4], endTime = [4], queryTime = 4
Output: 1
Explanation: The only student was doing their homework at the queryTime.

Example 3:

Input: startTime = [4], endTime = [4], queryTime = 5
Output: 0

Example 4:

Input: startTime = [1,1,1,1], endTime = [1,3,2,4], queryTime = 7
Output: 0

Example 5:

Input: startTime = [9,8,7,6,5,4,3,2,1], endTime = [10,10,10,10,10,10,10,10,10], queryTime = 5
Output: 5

Constraints:

startTime.length == endTime.length
1 <= startTime.length <= 100
1 <= startTime[i] <= endTime[i] <= 1000
1 <= queryTime <= 1000

1451. Rearrange Words in a Sentence

Given a sentence text (A sentence is a string of space-separated words) in the following format:

First letter is in upper case.
Each word in text are separated by a single space.

Your task is to rearrange the words in text such that all words are rearranged in an increasing order of their lengths. I f two words have the same length, arrange them in their original order.
Return the new text following the format shown above.

Example 1:

Input: text = "Leetcode is cool"
Output: "Is cool leetcode"
Explanation: There are 3 words, "Leetcode" of length 8, "is" of length 2 and "cool" of length 4.
Output is ordered by length and the new first word starts with capital letter.

Example 2:

Input: text = "Keep calm and code on"
Output: "On and keep calm code"

Explanation: Output is ordered as follows:
"On" 2 letters.
"and" 3 letters.
"keep" 4 letters in case of tie order by position in original text.
"calm" 4 letters.
"code" 4 letters.

Example 3:

Input: text = "To be or not to be"
Output: "To be or to be not"

Constraints:

text begins with a capital letter and then contains lowercase letters and single space between words.
1 <= text.length <= 10^5

1452. People Whose List of Favorite Companies Is Not a Subset of Another List

Given the array favoriteCompanies where favoriteCompanies[i] is the list of favorites companies for the ith person (i ndexed from 0).
Return the indices of people whose list of favorite companies is not a subset of any other list of favorites companies. You must return the indices in increasing order.

Example 1:

Input: favoriteCompanies = [["leetcode","google","facebook"],["google","microsoft"],["google","facebook"],["googl e"],["amazon"]]
Output: [0,1,4]
Explanation:
Person with index=2 has favoriteCompanies[2]=["google","facebook"] which is a subset of favoriteCompanies[0]=[" leetcode","google","facebook"] corresponding to the person with index 0.
Person with index=3 has favoriteCompanies[3]=["google"] which is a subset of favoriteCompanies[0]=["leetcode"," google","facebook"] and favoriteCompanies[1]=["google","microsoft"].
Other lists of favorite companies are not a subset of another list, therefore, the answer is [0,1,4].

Example 2:

Input: favoriteCompanies = [["leetcode","google","facebook"],["leetcode","amazon"],["facebook","google"]]
Output: [0,1]
Explanation: In this case favoriteCompanies[2]=["facebook","google"] is a subset of favoriteCompanies[0]=["leetco de","google","facebook"], therefore, the answer is [0,1].

Example 3:

Input: favoriteCompanies = [["leetcode"],["google"],["facebook"],["amazon"]]
Output: [0,1,2,3]

Constraints:

1 <= favoriteCompanies.length <= 100
1 <= favoriteCompanies[i].length <= 500
1 <= favoriteCompanies[i][j].length <= 20
All strings in favoriteCompanies[i] are distinct.
All lists of favorite companies are distinct, that is, If we sort alphabetically each list then favoriteCompanies[i] != fav oriteCompanies[j].
All strings consist of lowercase English letters only.

1453. Maximum Number of Darts Inside of a Circular Dartboard

You have a very large square wall and a circular dartboard placed on the wall. You have been challenged to throw da rts into the board blindfolded. Darts thrown at the wall are represented as an array of points on a 2D plane.
Return the maximum number of points that are within or lie on any circular dartboard of radius r.

Example 1:

Input: points = [[-2,0],[2,0],[0,2],[0,-2]], r = 2
Output: 4
Explanation: Circle dartboard with center in (0,0) and radius = 2 contain all points.

Example 2:

Input: points = [[-3,0],[3,0],[2,6],[5,4],[0,9],[7,8]], r = 5
Output: 5
Explanation: Circle dartboard with center in (0,4) and radius = 5 contain all points except the point (7,8).

Example 3:

Input: points = [[-2,0],[2,0],[0,2],[0,-2]], r = 1
Output: 1

Example 4:

Input: points = [[1,2],[3,5],[1,-1],[2,3],[4,1],[1,3]], r = 2
Output: 4

Constraints:

1 <= points.length <= 100
points[i].length == 2
-10^4 <= points[i][0], points[i][1] <= 10^4
1 <= r <= 5000

1455. Check If a Word Occurs As a Prefix of Any Word in a Sentence

Given a sentence that consists of some words separated by a single space, and a searchWord, check if searchWord is a prefix of any word in sentence.
Return the index of the word in sentence (1-indexed) where searchWord is a prefix of this word. If searchWord is a prefix of more than one word, return the index of the first word (minimum index). If there is no such word return -1.
A prefix of a string s is any leading contiguous substring of s.

Example 1:

Input: sentence = "i love eating burger", searchWord = "burg"
Output: 4
Explanation: "burg" is prefix of "burger" which is the 4th word in the sentence.

Example 2:

Input: sentence = "this problem is an easy problem", searchWord = "pro"
Output: 2
Explanation: "pro" is prefix of "problem" which is the 2nd and the 6th word in the sentence, but we return 2 as it's th e minimal index.

Example 3:

Input: sentence = "i am tired", searchWord = "you"
Output: -1
Explanation: "you" is not a prefix of any word in the sentence.

Example 4:

Input: sentence = "i use triple pillow", searchWord = "pill"
Output: 4

Example 5:

Input: sentence = "hello from the other side", searchWord = "they"
Output: -1

Constraints:

1 <= sentence.length <= 100

1 <= searchWord.length <= 10
sentence consists of lowercase English letters and spaces.
searchWord consists of lowercase English letters.

1456. Maximum Number of Vowels in a Substring of Given Length

Given a string s and an integer k.
Return the maximum number of vowel letters in any substring of s with length k.
Vowel letters in English are (a, e, i, o, u).

Example 1:

Input: s = "abciiidef", k = 3
Output: 3
Explanation: The substring "iii" contains 3 vowel letters.

Example 2:

Input: s = "aeiou", k = 2
Output: 2
Explanation: Any substring of length 2 contains 2 vowels.

Example 3:

Input: s = "leetcode", k = 3
Output: 2
Explanation: "lee", "eet" and "ode" contain 2 vowels.

Example 4:

Input: s = "rhythms", k = 4
Output: 0
Explanation: We can see that s doesn't have any vowel letters.

Example 5:

Input: s = "tryhard", k = 4
Output: 1

Constraints:

1 <= s.length <= 10^5
s consists of lowercase English letters.
1 <= k <= s.length

1457. Pseudo-Palindromic Paths in a Binary Tree

Given a binary tree where node values are digits from 1 to 9. A path in the binary tree is said to be pseudo-palindrom ic if at least one permutation of the node values in the path is a palindrome.
Return the number of pseudo-palindromic paths going from the root node to leaf nodes.

Example 1:

Input: root = [2,3,1,3,1,null,1]
Output: 2
Explanation: The figure above represents the given binary tree. There are three paths going from the root node to lea f nodes: the red path [2,3,3], the green path [2,1,1], and the path [2,3,1]. Among these paths only red path and green path are pseudo-palindromic paths since the red path [2,3,3] can be rearranged in [3,2,3] (palindrome) and the green path [2,1,1] can be rearranged in [1,2,1] (palindrome).

Example 2:

Input: root = [2,1,1,1,3,null,null,null,null,null,1]
Output: 1
Explanation: The figure above represents the given binary tree. There are three paths going from the root node to lea f nodes: the green path [2,1,1], the path [2,1,3,1], and the path [2,1]. Among these paths only the green path is pseud o-palindromic since [2,1,1] can be rearranged in [1,2,1] (palindrome).

Example 3:

Input: root = [9]
Output: 1

Constraints:

The number of nodes in the tree is in the range [1, 105].
1 <= Node.val <= 9

1458. Max Dot Product of Two Subsequences

Given two arrays nums1 and nums2.

Return the maximum dot product between non-empty subsequences of nums1 and nums2 with the same length.
A subsequence of a array is a new array which is formed from the original array by deleting some (can be none) of t he characters without disturbing the relative positions of the remaining characters. (ie, [2,3,5] is a subsequence of [1, 2,3,4,5] while [1,5,3] is not).

Example 1:

Input: nums1 = [2,1,-2,5], nums2 = [3,0,-6]
Output: 18
Explanation: Take subsequence [2,-2] from nums1 and subsequence [3,-6] from nums2.
Their dot product is (2*3 + (-2)*(-6)) = 18.
Example 2:

Input: nums1 = [3,-2], nums2 = [2,-6,7]
Output: 21
Explanation: Take subsequence [3] from nums1 and subsequence [7] from nums2.
Their dot product is (3*7) = 21.
Example 3:

Input: nums1 = [-1,-1], nums2 = [1,1]
Output: -1
Explanation: Take subsequence [-1] from nums1 and subsequence [1] from nums2.
Their dot product is -1.

Constraints:

1 <= nums1.length, nums2.length <= 500
-1000 <= nums1[i], nums2[i] <= 1000

1460. Make Two Arrays Equal by Reversing Sub-arrays

Given two integer arrays of equal length target and arr.
In one step, you can select any non-empty sub-array of arr and reverse it. You are allowed to make any number of st eps.
Return True if you can make arr equal to target, or False otherwise.

Example 1:

Input: target = [1,2,3,4], arr = [2,4,1,3]
Output: true
Explanation: You can follow the next steps to convert arr to target:
1- Reverse sub-array [2,4,1], arr becomes [1,4,2,3]
2- Reverse sub-array [4,2], arr becomes [1,2,4,3]
3- Reverse sub-array [4,3], arr becomes [1,2,3,4]
There are multiple ways to convert arr to target, this is not the only way to do so.

Example 2:

Input: target = [7], arr = [7]
Output: true
Explanation: arr is equal to target without any reverses.

Example 3:

Input: target = [1,12], arr = [12,1]
Output: true

Example 4:

Input: target = [3,7,9], arr = [3,7,11]
Output: false
Explanation: arr doesn't have value 9 and it can never be converted to target.

Example 5:

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

Constraints:

target.length == arr.length
1 <= target.length <= 1000
1 <= target[i] <= 1000
1 <= arr[i] <= 1000

1461. Check If a String Contains All Binary Codes of Size K

Given a binary string s and an integer k.
Return true if every binary code of length k is a substring of s. Otherwise, return false.

Example 1:

Input: s = "00110110", k = 2
Output: true
Explanation: The binary codes of length 2 are "00", "01", "10" and "11". They can be all found as substrings at indici es 0, 1, 3 and 2 respectively.

Example 2:

Input: s = "00110", k = 2
Output: true

Example 3:

Input: s = "0110", k = 1
Output: true
Explanation: The binary codes of length 1 are "0" and "1", it is clear that both exist as a substring.

Example 4:

Input: s = "0110", k = 2
Output: false
Explanation: The binary code "00" is of length 2 and doesn't exist in the array.

Example 5:

Input: s = "0000000001011100", k = 4
Output: false

Constraints:

1 <= s.length <= 5 * 105
s[i] is either '0' or '1'.
1 <= k <= 20

1462. Course Schedule IV

There are a total of numCourses courses you have to take, labeled from 0 to numCourses - 1. You are given an array prerequisites where prerequisites[i] = [ai, bi] indicates that you must take course ai first if you want to take course bi.

For example, the pair [0, 1] indicates that you have to take course 0 before you can take course 1.

Prerequisites can also be indirect. If course a is a prerequisite of course b, and course b is a prerequisite of course c, t hen course a is a prerequisite of course c.
You are also given an array queries where queries[j] = [uj, vj]. For the jth query, you should answer whether course uj is a prerequisite of course vj or not.
Return a boolean array answer, where answer[j] is the answer to the jth query.

Example 1:

Input: numCourses = 2, prerequisites = [[1,0]], queries = [[0,1],[1,0]]
Output: [false,true]
Explanation: The pair [1, 0] indicates that you have to take course 1 before you can take course 0.
Course 0 is not a prerequisite of course 1, but the opposite is true.

Example 2:

Input: numCourses = 2, prerequisites = [], queries = [[1,0],[0,1]]
Output: [false,false]
Explanation: There are no prerequisites, and each course is independent.

Example 3:

Input: numCourses = 3, prerequisites = [[1,2],[1,0],[2,0]], queries = [[1,0],[1,2]]
Output: [true,true]

Constraints:

2 <= numCourses <= 100
0 <= prerequisites.length <= (numCourses * (numCourses - 1) / 2)
prerequisites[i].length == 2
0 <= ai, bi <= n - 1
ai != bi
All the pairs [ai, bi] are unique.
The prerequisites graph has no cycles.
1 <= queries.length <= 104
0 <= ui, vi <= n - 1
ui != vi

1463. Cherry Pickup II

Given a rows x cols matrix grid representing a field of cherries. Each cell in grid represents the number of cherries th at you can collect.
You have two robots that can collect cherries for you, Robot #1 is located at the top-left corner (0,0) , and Robot #2 i s located at the top-right corner (0, cols-1) of the grid.
Return the maximum number of cherries collection using both robots by following the rules below:

From a cell (i,j), robots can move to cell (i+1, j-1) , (i+1, j) or (i+1, j+1).
When any robot is passing through a cell, It picks it up all cherries, and the cell becomes an empty cell (0).
When both robots stay on the same cell, only one of them takes the cherries.
Both robots cannot move outside of the grid at any moment.
Both robots should reach the bottom row in the grid.

Example 1:

Input: grid = [[3,1,1],[2,5,1],[1,5,5],[2,1,1]]
Output: 24

Explanation: Path of robot #1 and #2 are described in color green and blue respectively.
Cherries taken by Robot #1, (3 + 2 + 5 + 2) = 12.
Cherries taken by Robot #2, (1 + 5 + 5 + 1) = 12.
Total of cherries: 12 + 12 = 24.

Example 2:

Input: grid = [[1,0,0,0,0,0,1],[2,0,0,0,0,3,0],[2,0,9,0,0,0,0],[0,3,0,5,4,0,0],[1,0,2,3,0,0,6]]
Output: 28
Explanation: Path of robot #1 and #2 are described in color green and blue respectively.
Cherries taken by Robot #1, (1 + 9 + 5 + 2) = 17.
Cherries taken by Robot #2, (1 + 3 + 4 + 3) = 11.
Total of cherries: 17 + 11 = 28.

Example 3:

Input: grid = [[1,0,0,3],[0,0,0,3],[0,0,3,3],[9,0,3,3]]
Output: 22

Example 4:

Input: grid = [[1,1],[1,1]]
Output: 4

Constraints:

rows == grid.length
cols == grid[i].length
2 <= rows, cols <= 70
0 <= grid[i][j] <= 100

1464. Maximum Product of Two Elements in an Array

Given the array of integers nums, you will choose two different indices i and j of that array. Return the maximum val ue of (nums[i]-1)*(nums[j]-1).

Example 1:

Input: nums = [3,4,5,2]
Output: 12
Explanation: If you choose the indices i=1 and j=2 (indexed from 0), you will get the maximum value, that is, (nums [1]-1)*(nums[2]-1) = (4-1)*(5-1) = 3*4 = 12.

Example 2:

Input: nums = [1,5,4,5]
Output: 16
Explanation: Choosing the indices i=1 and j=3 (indexed from 0), you will get the maximum value of (5-1)*(5-1) = 16.

Example 3:

Input: nums = [3,7]
Output: 12

Constraints:

2 <= nums.length <= 500
1 <= nums[i] <= 10^3

1465. Maximum Area of a Piece of Cake After Horizontal and Vertical Cuts

You are given a rectangular cake of size h x w and two arrays of integers horizontalCuts and verticalCuts where:

horizontalCuts[i] is the distance from the top of the rectangular cake to the ith horizontal cut and similarly, and verticalCuts[j] is the distance from the left of the rectangular cake to the jth vertical cut.

Return the maximum area of a piece of cake after you cut at each horizontal and vertical position provided in the arrays horizontalCuts and verticalCuts. Since the answer can be a large number, return this modulo 109 + 7.

Example 1:

Input: h = 5, w = 4, horizontalCuts = [1,2,4], verticalCuts = [1,3]
Output: 4
Explanation: The figure above represents the given rectangular cake. Red lines are the horizontal and vertical cuts. After you cut the cake, the green piece of cake has the maximum area.

Example 2:

Input: h = 5, w = 4, horizontalCuts = [3,1], verticalCuts = [1]
Output: 6
Explanation: The figure above represents the given rectangular cake. Red lines are the horizontal and vertical cuts. After you cut the cake, the green and yellow pieces of cake have the maximum area.

Example 3:

Input: h = 5, w = 4, horizontalCuts = [3], verticalCuts = [3]

Output: 9

Constraints:

2 <= h, w <= 109
1 <= horizontalCuts.length <= min(h - 1, 105)
1 <= verticalCuts.length <= min(w - 1, 105)
1 <= horizontalCuts[i] < h
1 <= verticalCuts[i] < w
All the elements in horizontalCuts are distinct.
All the elements in verticalCuts are distinct.

1466. Reorder Routes to Make All Paths Lead to the City Zero

There are n cities numbered from 0 to n - 1 and n - 1 roads such that there is only one way to travel between two diff erent cities (this network form a tree). Last year, The ministry of transport decided to orient the roads in one directio n because they are too narrow.
Roads are represented by connections where connections[i] = [ai, bi] represents a road from city ai to city bi.
This year, there will be a big event in the capital (city 0), and many people want to travel to this city.
Your task consists of reorienting some roads such that each city can visit the city 0. Return the minimum number of edges changed.
It's guaranteed that each city can reach city 0 after reorder.

Example 1:

Input: n = 6, connections = [[0,1],[1,3],[2,3],[4,0],[4,5]]
Output: 3
Explanation: Change the direction of edges show in red such that each node can reach the node 0 (capital).

Example 2:

Input: n = 5, connections = [[1,0],[1,2],[3,2],[3,4]]
Output: 2
Explanation: Change the direction of edges show in red such that each node can reach the node 0 (capital).

Example 3:

Input: n = 3, connections = [[1,0],[2,0]]
Output: 0

Constraints:

2 <= n <= 5 * 104
connections.length == n - 1
connections[i].length == 2
0 <= ai, bi <= n - 1
ai != bi

1467. Probability of a Two Boxes Having The Same Number of Distinct Balls

Given 2n balls of k distinct colors. You will be given an integer array balls of size k where balls[i] is the number of b alls of color i.
All the balls will be shuffled uniformly at random, then we will distribute the first n balls to the first box and the rem aining n balls to the other box (Please read the explanation of the second example carefully).
Please note that the two boxes are considered different. For example, if we have two balls of colors a and b, and two boxes [] and (), then the distribution [a] (b) is considered different than the distribution [b] (a) (Please read the expla nation of the first example carefully).
Return the probability that the two boxes have the same number of distinct balls. Answers within 10-5 of the actual v alue will be accepted as correct.

Example 1:

Input: balls = [1,1]
Output: 1.00000
Explanation: Only 2 ways to divide the balls equally:
- A ball of color 1 to box 1 and a ball of color 2 to box 2
- A ball of color 2 to box 1 and a ball of color 1 to box 2
In both ways, the number of distinct colors in each box is equal. The probability is 2/2 = 1

Example 2:

Input: balls = [2,1,1]
Output: 0.66667
Explanation: We have the set of balls [1, 1, 2, 3]
This set of balls will be shuffled randomly and we may have one of the 12 distinct shuffles with equal probability (i. e. 1/12):
[1,1 / 2,3], [1,1 / 3,2], [1,2 / 1,3], [1,2 / 3,1], [1,3 / 1,2], [1,3 / 2,1], [2,1 / 1,3], [2,1 / 3,1], [2,3 / 1,1], [3,1 / 1,2], [3,1 / 2,1], [3,2 / 1,1]
After that, we add the first two balls to the first box and the second two balls to the second box.
We can see that 8 of these 12 possible random distributions have the same number of distinct colors of balls in each box.
Probability is 8/12 = 0.66667

Example 3:

Input: balls = [1,2,1,2]
Output: 0.60000
Explanation: The set of balls is [1, 2, 2, 3, 4, 4]. It is hard to display all the 180 possible random shuffles of this set b

ut it is easy to check that 108 of them will have the same number of distinct colors in each box.
Probability = 108 / 180 = 0.6

Example 4:

Input: balls = [3,2,1]
Output: 0.30000
Explanation: The set of balls is [1, 1, 1, 2, 2, 3]. It is hard to display all the 60 possible random shuffles of this set bu t it is easy to check that 18 of them will have the same number of distinct colors in each box.
Probability = 18 / 60 = 0.3

Example 5:

Input: balls = [6,6,6,6,6,6]
Output: 0.90327

Constraints:

1 <= balls.length <= 8
1 <= balls[i] <= 6
sum(balls) is even.

1470. Shuffle the Array

Given the array nums consisting of 2n elements in the form [x1,x2,...,xn,y1,y2,...,yn].
Return the array in the form [x1,y1,x2,y2,...,xn,yn].

Example 1:

Input: nums = [2,5,1,3,4,7], n = 3
Output: [2,3,5,4,1,7]
Explanation: Since x1=2, x2=5, x3=1, y1=3, y2=4, y3=7 then the answer is [2,3,5,4,1,7].

Example 2:

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

Example 3:

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

Constraints:

1 <= n <= 500
nums.length == 2n
1 <= nums[i] <= 10^3

1471. The k Strongest Values in an Array

Given an array of integers arr and an integer k.
A value arr[i] is said to be stronger than a value arr[j] if |arr[i] - m| > |arr[j] - m| where m is the median of the array.
If |arr[i] - m| == |arr[j] - m|, then arr[i] is said to be stronger than arr[j] if arr[i] > arr[j].
Return a list of the strongest k values in the array. return the answer in any arbitrary order.
Median is the middle value in an ordered integer list. More formally, if the length of the list is n, the median is the el ement in position ((n - 1) / 2) in the sorted list (0-indexed).

For arr = [6, -3, 7, 2, 11], n = 5 and the median is obtained by sorting the array arr = [-3, 2, 6, 7, 11] and the median i s arr[m] where m = ((5 - 1) / 2) = 2. The median is 6.
For arr = [-7, 22, 17, 3], n = 4 and the median is obtained by sorting the array arr = [-7, 3, 17, 22] and the median is a rr[m] where m = ((4 - 1) / 2) = 1. The median is 3.

Example 1:

Input: arr = [1,2,3,4,5], k = 2
Output: [5,1]
Explanation: Median is 3, the elements of the array sorted by the strongest are [5,1,4,2,3]. The strongest 2 elements a re [5, 1]. [1, 5] is also accepted answer.
Please note that although |5 - 3| == |1 - 3| but 5 is stronger than 1 because 5 > 1.

Example 2:

Input: arr = [1,1,3,5,5], k = 2
Output: [5,5]
Explanation: Median is 3, the elements of the array sorted by the strongest are [5,5,1,1,3]. The strongest 2 elements a re [5, 5].

Example 3:

Input: arr = [6,7,11,7,6,8], k = 5
Output: [11,8,6,6,7]
Explanation: Median is 7, the elements of the array sorted by the strongest are [11,8,6,6,7,7].
Any permutation of [11,8,6,6,7] is accepted.

Example 4:

Input: arr = [6,-3,7,2,11], k = 3
Output: [-3,11,2]

Example 5:

Input: arr = [-7,22,17,3], k = 2
Output: [22,17]

Constraints:

1 <= arr.length <= 10^5
-10^5 <= arr[i] <= 10^5
1 <= k <= arr.length

1472. Design Browser History

You have a browser of one tab where you start on the homepage and you can visit another url, get back in the history number of steps or move forward in the history number of steps.
Implement the BrowserHistory class:

BrowserHistory(string homepage) Initializes the object with the homepage of the browser.
void visit(string url) Visits url from the current page. It clears up all the forward history.
string back(int steps) Move steps back in history. If you can only return x steps in the history and steps > x, you will return only x steps. Return the current url after moving back in history at most steps.
string forward(int steps) Move steps forward in history. If you can only forward x steps in the history and steps > x, you will forward only x steps. Return the current url after forwarding in history at most steps.

Example:

Input:
["BrowserHistory","visit","visit","visit","back","back","forward","visit","forward","back","back"]
[["leetcode.com"],["google.com"],["facebook.com"],["youtube.com"],[1],[1],[1],["linkedin.com"],[2],[2],[7]]
Output:
[null,null,null,null,"facebook.com","google.com","facebook.com",null,"linkedin.com","google.com","leetcode.com"]

Explanation:
BrowserHistory browserHistory = new BrowserHistory("leetcode.com");
browserHistory.visit("google.com"); // You are in "leetcode.com". Visit "google.com"
browserHistory.visit("facebook.com"); // You are in "google.com". Visit "facebook.com"
browserHistory.visit("youtube.com"); // You are in "facebook.com". Visit "youtube.com"
browserHistory.back(1); // You are in "youtube.com", move back to "facebook.com" return "facebook.com"
browserHistory.back(1); // You are in "facebook.com", move back to "google.com" return "google.com"
browserHistory.forward(1); // You are in "google.com", move forward to "facebook.com" return "facebook.com"
browserHistory.visit("linkedin.com"); // You are in "facebook.com". Visit "linkedin.com"
browserHistory.forward(2); // You are in "linkedin.com", you cannot move forward any steps.

browserHistory.back(2); // You are in "linkedin.com", move back two steps to "facebook.com" then to "g
oogle.com". return "google.com"
browserHistory.back(7); // You are in "google.com", you can move back only one step to "leetcode.com".
 return "leetcode.com"

Constraints:

1 <= homepage.length <= 20
1 <= url.length <= 20
1 <= steps <= 100
homepage and url consist of '.' or lower case English letters.
At most 5000 calls will be made to visit, back, and forward.

1473. Paint House III

There is a row of m houses in a small city, each house must be painted with one of the n colors (labeled from 1 to n),
 some houses that have been painted last summer should not be painted again.
A neighborhood is a maximal group of continuous houses that are painted with the same color.

For example: houses = [1,2,2,3,3,2,1,1] contains 5 neighborhoods [{1}, {2,2}, {3,3}, {2}, {1,1}].

Given an array houses, an m x n matrix cost and an integer target where:

houses[i]: is the color of the house i, and 0 if the house is not painted yet.
cost[i][j]: is the cost of paint the house i with the color j + 1.

Return the minimum cost of painting all the remaining houses in such a way that there are exactly target neighborho
ods. If it is not possible, return -1.

Example 1:

Input: houses = [0,0,0,0,0], cost = [[1,10],[10,1],[10,1],[1,10],[5,1]], m = 5, n = 2, target = 3
Output: 9
Explanation: Paint houses of this way [1,2,2,1,1]
This array contains target = 3 neighborhoods, [{1}, {2,2}, {1,1}].
Cost of paint all houses (1 + 1 + 1 + 1 + 5) = 9.

Example 2:

Input: houses = [0,2,1,2,0], cost = [[1,10],[10,1],[10,1],[1,10],[5,1]], m = 5, n = 2, target = 3
Output: 11
Explanation: Some houses are already painted, Paint the houses of this way [2,2,1,2,2]
This array contains target = 3 neighborhoods, [{2,2}, {1}, {2,2}].
Cost of paint the first and last house (10 + 1) = 11.

Example 3:

Input: houses = [0,0,0,0,0], cost = [[1,10],[10,1],[1,10],[10,1],[1,10]], m = 5, n = 2, target = 5
Output: 5

Example 4:

Input: houses = [3,1,2,3], cost = [[1,1,1],[1,1,1],[1,1,1],[1,1,1]], m = 4, n = 3, target = 3
Output: -1
Explanation: Houses are already painted with a total of 4 neighborhoods [{3},{1},{2},{3}] different of target = 3.

Constraints:

m == houses.length == cost.length
n == cost[i].length
1 <= m <= 100
1 <= n <= 20
1 <= target <= m
0 <= houses[i] <= n
1 <= cost[i][j] <= 10^4

1475. Final Prices With a Special Discount in a Shop

Given the array prices where prices[i] is the price of the ith item in a shop. There is a special discount for items in th e shop, if you buy the ith item, then you will receive a discount equivalent to prices[j] where j is the minimum index such that j > i and prices[j] <= prices[i], otherwise, you will not receive any discount at all.
Return an array where the ith element is the final price you will pay for the ith item of the shop considering the speci al discount.

Example 1:

Input: prices = [8,4,6,2,3]
Output: [4,2,4,2,3]
Explanation:
For item 0 with price[0]=8 you will receive a discount equivalent to prices[1]=4, therefore, the final price you will p ay is 8 - 4 = 4.
For item 1 with price[1]=4 you will receive a discount equivalent to prices[3]=2, therefore, the final price you will p ay is 4 - 2 = 2.
For item 2 with price[2]=6 you will receive a discount equivalent to prices[3]=2, therefore, the final price you will p ay is 6 - 2 = 4.
For items 3 and 4 you will not receive any discount at all.

Example 2:

Input: prices = [1,2,3,4,5]

Output: [1,2,3,4,5]
Explanation: In this case, for all items, you will not receive any discount at all.

Example 3:

Input: prices = [10,1,1,6]
Output: [9,0,1,6]

Constraints:

1 <= prices.length <= 500
1 <= prices[i] <= 10^3

1476. Subrectangle Queries

Implement the class SubrectangleQueries which receives a rows x cols rectangle as a matrix of integers in the constr uctor and supports two methods:
1. updateSubrectangle(int row1, int col1, int row2, int col2, int newValue)

Updates all values with newValue in the subrectangle whose upper left coordinate is (row1,col1) and bottom right co ordinate is (row2,col2).

2. getValue(int row, int col)

Returns the current value of the coordinate (row,col) from the rectangle.

Example 1:

Input
["SubrectangleQueries","getValue","updateSubrectangle","getValue","getValue","updateSubrectangle","getValue","getValue"]
[[[[1,2,1],[4,3,4],[3,2,1],[1,1,1]]],[0,2],[0,0,3,2,5],[0,2],[3,1],[3,0,3,2,10],[3,1],[0,2]]
Output
[null,1,null,5,5,null,10,5]
Explanation
SubrectangleQueries subrectangleQueries = new SubrectangleQueries([[1,2,1],[4,3,4],[3,2,1],[1,1,1]]);
// The initial rectangle (4x3) looks like:
// 1 2 1
// 4 3 4
// 3 2 1
// 1 1 1
subrectangleQueries.getValue(0, 2); // return 1
subrectangleQueries.updateSubrectangle(0, 0, 3, 2, 5);
// After this update the rectangle looks like:

// 5 5 5
// 5 5 5
// 5 5 5
// 5 5 5
subrectangleQueries.getValue(0, 2); // return 5
subrectangleQueries.getValue(3, 1); // return 5
subrectangleQueries.updateSubrectangle(3, 0, 3, 2, 10);
// After this update the rectangle looks like:
// 5   5   5
// 5   5   5
// 5   5   5
// 10 10 10
subrectangleQueries.getValue(3, 1); // return 10
subrectangleQueries.getValue(0, 2); // return 5

Example 2:

Input
["SubrectangleQueries","getValue","updateSubrectangle","getValue","getValue","updateSubrectangle","getValue"]
[[[[1,1,1],[2,2,2],[3,3,3]]],[0,0],[0,0,2,2,100],[0,0],[2,2],[1,1,2,2,20],[2,2]]
Output
[null,1,null,100,100,null,20]
Explanation
SubrectangleQueries subrectangleQueries = new SubrectangleQueries([[1,1,1],[2,2,2],[3,3,3]]);
subrectangleQueries.getValue(0, 0); // return 1
subrectangleQueries.updateSubrectangle(0, 0, 2, 2, 100);
subrectangleQueries.getValue(0, 0); // return 100
subrectangleQueries.getValue(2, 2); // return 100
subrectangleQueries.updateSubrectangle(1, 1, 2, 2, 20);
subrectangleQueries.getValue(2, 2); // return 20

Constraints:

There will be at most 500 operations considering both methods: updateSubrectangle and getValue.
1 <= rows, cols <= 100
rows == rectangle.length
cols == rectangle[i].length
0 <= row1 <= row2 < rows
0 <= col1 <= col2 < cols
1 <= newValue, rectangle[i][j] <= 10^9
0 <= row < rows
0 <= col < cols

1477. Find Two Non-overlapping Sub-arrays Each With Target Sum

Given an array of integers arr and an integer target.
You have to find two non-overlapping sub-arrays of arr each with a sum equal target. There can be multiple answers so you have to find an answer where the sum of the lengths of the two sub-arrays is minimum.
Return the minimum sum of the lengths of the two required sub-arrays, or return -1 if you cannot find such two sub-arrays.

Example 1:

Input: arr = [3,2,2,4,3], target = 3
Output: 2
Explanation: Only two sub-arrays have sum = 3 ([3] and [3]). The sum of their lengths is 2.

Example 2:

Input: arr = [7,3,4,7], target = 7
Output: 2
Explanation: Although we have three non-overlapping sub-arrays of sum = 7 ([7], [3,4] and [7]), but we will choose the first and third sub-arrays as the sum of their lengths is 2.

Example 3:

Input: arr = [4,3,2,6,2,3,4], target = 6
Output: -1
Explanation: We have only one sub-array of sum = 6.

Example 4:

Input: arr = [5,5,4,4,5], target = 3
Output: -1
Explanation: We cannot find a sub-array of sum = 3.

Example 5:

Input: arr = [3,1,1,1,5,1,2,1], target = 3
Output: 3
Explanation: Note that sub-arrays [1,2] and [2,1] cannot be an answer because they overlap.

Constraints:

1 <= arr.length <= 105
1 <= arr[i] <= 1000
1 <= target <= 108

1478. Allocate Mailboxes

Given the array houses and an integer k. where houses[i] is the location of the ith house along a street, your task is to allocate k mailboxes in the street.
Return the minimum total distance between each house and its nearest mailbox.
The answer is guaranteed to fit in a 32-bit signed integer.

Example 1:

Input: houses = [1,4,8,10,20], k = 3
Output: 5
Explanation: Allocate mailboxes in position 3, 9 and 20.
Minimum total distance from each houses to nearest mailboxes is |3-1| + |4-3| + |9-8| + |10-9| + |20-20| = 5

Example 2:

Input: houses = [2,3,5,12,18], k = 2
Output: 9
Explanation: Allocate mailboxes in position 3 and 14.
Minimum total distance from each houses to nearest mailboxes is |2-3| + |3-3| + |5-3| + |12-14| + |18-14| = 9.

Example 3:

Input: houses = [7,4,6,1], k = 1
Output: 8

Example 4:

Input: houses = [3,6,14,10], k = 4
Output: 0

Constraints:

n == houses.length
1 <= n <= 100
1 <= houses[i] <= 10^4
1 <= k <= n
Array houses contain unique integers.

1480. Running Sum of 1d Array

Given an array nums. We define a running sum of an array as runningSum[i] = sum(nums[0]…nums[i]).
Return the running sum of nums.

Example 1:

Input: nums = [1,2,3,4]
Output: [1,3,6,10]
Explanation: Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].
Example 2:

Input: nums = [1,1,1,1,1]
Output: [1,2,3,4,5]
Explanation: Running sum is obtained as follows: [1, 1+1, 1+1+1, 1+1+1+1, 1+1+1+1+1].
Example 3:

Input: nums = [3,1,2,10,1]
Output: [3,4,6,16,17]

Constraints:

1 <= nums.length <= 1000
-10^6 <= nums[i] <= 10^6

1481. Least Number of Unique Integers after K Removals

Given an array of integers arr and an integer k. Find the least number of unique integers after removing exactly k ele ments.

Example 1:

Input: arr = [5,5,4], k = 1
Output: 1
Explanation: Remove the single 4, only 5 is left.

Example 2:

Input: arr = [4,3,1,1,3,3,2], k = 3
Output: 2
Explanation: Remove 4, 2 and either one of the two 1s or three 3s. 1 and 3 will be left.

Constraints:

1 <= arr.length <= 10^5
1 <= arr[i] <= 10^9
0 <= k <= arr.length

1482. Minimum Number of Days to Make m Bouquets

Given an integer array bloomDay, an integer m and an integer k.
We need to make m bouquets. To make a bouquet, you need to use k adjacent flowers from the garden.
The garden consists of n flowers, the ith flower will bloom in the bloomDay[i] and then can be used in exactly one b ouquet.
Return the minimum number of days you need to wait to be able to make m bouquets from the garden. If it is imposs ible to make m bouquets return -1.

Example 1:

Input: bloomDay = [1,10,3,10,2], m = 3, k = 1
Output: 3
Explanation: Let's see what happened in the first three days. x means flower bloomed and _ means flower didn't bloo m in the garden.
We need 3 bouquets each should contain 1 flower.
After day 1: [x, _, _, _, _] // we can only make one bouquet.
After day 2: [x, _, _, _, x] // we can only make two bouquets.
After day 3: [x, _, x, _, x] // we can make 3 bouquets. The answer is 3.

Example 2:

Input: bloomDay = [1,10,3,10,2], m = 3, k = 2
Output: -1
Explanation: We need 3 bouquets each has 2 flowers, that means we need 6 flowers. We only have 5 flowers so it is impossible to get the needed bouquets and we return -1.

Example 3:

Input: bloomDay = [7,7,7,7,12,7,7], m = 2, k = 3
Output: 12
Explanation: We need 2 bouquets each should have 3 flowers.
Here's the garden after the 7 and 12 days:
After day 7: [x, x, x, x, _, x, x]
We can make one bouquet of the first three flowers that bloomed. We cannot make another bouquet from the last thr ee flowers that bloomed because they are not adjacent.
After day 12: [x, x, x, x, x, x, x]
It is obvious that we can make two bouquets in different ways.

Example 4:

Input: bloomDay = [1000000000,1000000000], m = 1, k = 1
Output: 1000000000
Explanation: You need to wait 1000000000 days to have a flower ready for a bouquet.

Example 5:

Input: bloomDay = [1,10,2,9,3,8,4,7,5,6], m = 4, k = 2
Output: 9

Constraints:

bloomDay.length == n
1 <= n <= 10^5
1 <= bloomDay[i] <= 10^9
1 <= m <= 10^6
1 <= k <= n

1483. Kth Ancestor of a Tree Node

You are given a tree with n nodes numbered from 0 to n - 1 in the form of a parent array parent where parent[i] is the parent of ith node. The root of the tree is node 0. Find the kth ancestor of a given node.
The kth ancestor of a tree node is the kth node in the path from that node to the root node.
Implement the TreeAncestor class:

TreeAncestor(int n, int[] parent) Initializes the object with the number of nodes in the tree and the parent array.
int getKthAncestor(int node, int k) return the kth ancestor of the given node node. If there is no such ancestor, return -1.

Example 1:

Input
["TreeAncestor", "getKthAncestor", "getKthAncestor", "getKthAncestor"]
[[7, [-1, 0, 0, 1, 1, 2, 2]], [3, 1], [5, 2], [6, 3]]
Output
[null, 1, 0, -1]

Explanation
TreeAncestor treeAncestor = new TreeAncestor(7, [-1, 0, 0, 1, 1, 2, 2]);
treeAncestor.getKthAncestor(3, 1); // returns 1 which is the parent of 3
treeAncestor.getKthAncestor(5, 2); // returns 0 which is the grandparent of 5
treeAncestor.getKthAncestor(6, 3); // returns -1 because there is no such ancestor

Constraints:

1 <= k <= n <= 5 * 104
parent.length == n
parent[0] == -1
0 <= parent[i] < n for all 0 < i < n
0 <= node < n
There will be at most 5 * 104 queries.

1486. XOR Operation in an Array

Given an integer n and an integer start.
Define an array nums where nums[i] = start + 2*i (0-indexed) and n == nums.length.
Return the bitwise XOR of all elements of nums.

Example 1:

Input: n = 5, start = 0
Output: 8
Explanation: Array nums is equal to [0, 2, 4, 6, 8] where (0 ^ 2 ^ 4 ^ 6 ^ 8) = 8.
Where "^" corresponds to bitwise XOR operator.

Example 2:

Input: n = 4, start = 3
Output: 8
Explanation: Array nums is equal to [3, 5, 7, 9] where (3 ^ 5 ^ 7 ^ 9) = 8.
Example 3:

Input: n = 1, start = 7
Output: 7

Example 4:

Input: n = 10, start = 5
Output: 2

Constraints:

1 <= n <= 1000
0 <= start <= 1000
n == nums.length

1487. Making File Names Unique

Given an array of strings names of size n. You will create n folders in your file system such that, at the ith minute, y ou will create a folder with the name names[i].
Since two files cannot have the same name, if you enter a folder name which is previously used, the system will hav e a suffix addition to its name in the form of (k), where, k is the smallest positive integer such that the obtained name

remains unique.
Return an array of strings of length n where ans[i] is the actual name the system will assign to the ith folder when yo
u create it.

Example 1:

Input: names = ["pes","fifa","gta","pes(2019)"]
Output: ["pes","fifa","gta","pes(2019)"]
Explanation: Let's see how the file system creates folder names:
"pes" --> not assigned before, remains "pes"
"fifa" --> not assigned before, remains "fifa"
"gta" --> not assigned before, remains "gta"
"pes(2019)" --> not assigned before, remains "pes(2019)"

Example 2:

Input: names = ["gta","gta(1)","gta","avalon"]
Output: ["gta","gta(1)","gta(2)","avalon"]
Explanation: Let's see how the file system creates folder names:
"gta" --> not assigned before, remains "gta"
"gta(1)" --> not assigned before, remains "gta(1)"
"gta" --> the name is reserved, system adds (k), since "gta(1)" is also reserved, systems put k = 2. it becomes "gta(2)
"
"avalon" --> not assigned before, remains "avalon"

Example 3:

Input: names = ["onepiece","onepiece(1)","onepiece(2)","onepiece(3)","onepiece"]
Output: ["onepiece","onepiece(1)","onepiece(2)","onepiece(3)","onepiece(4)"]
Explanation: When the last folder is created, the smallest positive valid k is 4, and it becomes "onepiece(4)".

Example 4:

Input: names = ["wano","wano","wano","wano"]
Output: ["wano","wano(1)","wano(2)","wano(3)"]
Explanation: Just increase the value of k each time you create folder "wano".

Example 5:

Input: names = ["kaido","kaido(1)","kaido","kaido(1)"]
Output: ["kaido","kaido(1)","kaido(2)","kaido(1)(1)"]
Explanation: Please note that system adds the suffix (k) to current name even it contained the same suffix before.

Constraints:

1 <= names.length <= 5 * 10^4
1 <= names[i].length <= 20
names[i] consists of lower case English letters, digits and/or round brackets.

1488. Avoid Flood in The City

Your country has an infinite number of lakes. Initially, all the lakes are empty, but when it rains over the nth lake, th e nth lake becomes full of water. If it rains over a lake which is full of water, there will be a flood. Your goal is to av oid the flood in any lake.
Given an integer array rains where:

rains[i] > 0 means there will be rains over the rains[i] lake.
rains[i] == 0 means there are no rains this day and you can choose one lake this day and dry it.

Return an array ans where:

ans.length == rains.length
ans[i] == -1 if rains[i] > 0.
ans[i] is the lake you choose to dry in the ith day if rains[i] == 0.

If there are multiple valid answers return any of them. If it is impossible to avoid flood return an empty array.
Notice that if you chose to dry a full lake, it becomes empty, but if you chose to dry an empty lake, nothing changes. (see example 4)

Example 1:

Input: rains = [1,2,3,4]
Output: [-1,-1,-1,-1]
Explanation: After the first day full lakes are [1]
After the second day full lakes are [1,2]
After the third day full lakes are [1,2,3]
After the fourth day full lakes are [1,2,3,4]
There's no day to dry any lake and there is no flood in any lake.

Example 2:

Input: rains = [1,2,0,0,2,1]
Output: [-1,-1,2,1,-1,-1]
Explanation: After the first day full lakes are [1]
After the second day full lakes are [1,2]
After the third day, we dry lake 2. Full lakes are [1]
After the fourth day, we dry lake 1. There is no full lakes.
After the fifth day, full lakes are [2].
After the sixth day, full lakes are [1,2].
It is easy that this scenario is flood-free. [-1,-1,1,2,-1,-1] is another acceptable scenario.

Example 3:

Input: rains = [1,2,0,1,2]
Output: []
Explanation: After the second day, full lakes are [1,2]. We have to dry one lake in the third day.
After that, it will rain over lakes [1,2]. It's easy to prove that no matter which lake you choose to dry in the 3rd day, t he other one will flood.

Example 4:

Input: rains = [69,0,0,0,69]
Output: [-1,69,1,1,-1]
Explanation: Any solution on one of the forms [-1,69,x,y,-1], [-1,x,69,y,-1] or [-1,x,y,69,-1] is acceptable where 1 <= x,y <= 10^9

Example 5:

Input: rains = [10,20,20]
Output: []
Explanation: It will rain over lake 20 two consecutive days. There is no chance to dry any lake.

Constraints:

1 <= rains.length <= 105
0 <= rains[i] <= 109

1489. Find Critical and Pseudo-Critical Edges in Minimum Spanning Tree

Given a weighted undirected connected graph with n vertices numbered from 0 to n - 1, and an array edges where edges[i] = [ai, bi, weighti] represents a bidirectional and weighted edge between nodes ai and bi. A minimum spanning tree (MST) is a subset of the graph's edges that connects all vertices without cycles and with the minimum possible total edge weight.
Find all the critical and pseudo-critical edges in the given graph's minimum spanning tree (MST). An MST edge whose deletion from the graph would cause the MST weight to increase is called a critical edge. On the other hand, a pseudo-critical edge is that which can appear in some MSTs but not all.
Note that you can return the indices of the edges in any order.

Example 1:

Input: n = 5, edges = [[0,1,1],[1,2,1],[2,3,2],[0,3,2],[0,4,3],[3,4,3],[1,4,6]]
Output: [[0,1],[2,3,4,5]]
Explanation: The figure above describes the graph.
The following figure shows all the possible MSTs:

Notice that the two edges 0 and 1 appear in all MSTs, therefore they are critical edges, so we return them in the first list of the output.
The edges 2, 3, 4, and 5 are only part of some MSTs, therefore they are considered pseudo-critical edges. We add them to the second list of the output.

Example 2:

Input: n = 4, edges = [[0,1,1],[1,2,1],[2,3,1],[0,3,1]]

Output: [[],[0,1,2,3]]
Explanation: We can observe that since all 4 edges have equal weight, choosing any 3 edges from the given 4 will yi eld an MST. Therefore all 4 edges are pseudo-critical.

Constraints:

2 <= n <= 100
1 <= edges.length <= min(200, n * (n - 1) / 2)
edges[i].length == 3
0 <= ai < bi < n
1 <= weighti <= 1000
All pairs (ai, bi) are distinct.

1491. Average Salary Excluding the Minimum and Maximum Salary

Given an array of unique integers salary where salary[i] is the salary of the employee i.
Return the average salary of employees excluding the minimum and maximum salary.

Example 1:

Input: salary = [4000,3000,1000,2000]
Output: 2500.00000
Explanation: Minimum salary and maximum salary are 1000 and 4000 respectively.
Average salary excluding minimum and maximum salary is (2000+3000)/2= 2500

Example 2:

Input: salary = [1000,2000,3000]
Output: 2000.00000
Explanation: Minimum salary and maximum salary are 1000 and 3000 respectively.
Average salary excluding minimum and maximum salary is (2000)/1= 2000

Example 3:

Input: salary = [6000,5000,4000,3000,2000,1000]
Output: 3500.00000

Example 4:

Input: salary = [8000,9000,2000,3000,6000,1000]
Output: 4750.00000

Constraints:

3 <= salary.length <= 100
10^3 <= salary[i] <= 10^6
salary[i] is unique.
Answers within 10^-5 of the actual value will be accepted as correct.

1492. The kth Factor of n

Given two positive integers n and k.
A factor of an integer n is defined as an integer i where n % i == 0.
Consider a list of all factors of n sorted in ascending order, return the kth factor in this list or return -1 if n has less th an k factors.

Example 1:

Input: n = 12, k = 3
Output: 3
Explanation: Factors list is [1, 2, 3, 4, 6, 12], the 3rd factor is 3.

Example 2:

Input: n = 7, k = 2
Output: 7
Explanation: Factors list is [1, 7], the 2nd factor is 7.

Example 3:

Input: n = 4, k = 4
Output: -1
Explanation: Factors list is [1, 2, 4], there is only 3 factors. We should return -1.

Example 4:

Input: n = 1, k = 1
Output: 1
Explanation: Factors list is [1], the 1st factor is 1.

Example 5:

Input: n = 1000, k = 3
Output: 4
Explanation: Factors list is [1, 2, 4, 5, 8, 10, 20, 25, 40, 50, 100, 125, 200, 250, 500, 1000].

Constraints:

1 <= k <= n <= 1000

1493. Longest Subarray of 1's After Deleting One Element

Given a binary array nums, you should delete one element from it.
Return the size of the longest non-empty subarray containing only 1's in the resulting array. Return 0 if there is no su
ch subarray.

Example 1:

Input: nums = [1,1,0,1]
Output: 3
Explanation: After deleting the number in position 2, [1,1,1] contains 3 numbers with value of 1's.
Example 2:

Input: nums = [0,1,1,1,0,1,1,0,1]
Output: 5
Explanation: After deleting the number in position 4, [0,1,1,1,1,1,0,1] longest subarray with value of 1's is [1,1,1,1,1
].
Example 3:

Input: nums = [1,1,1]
Output: 2
Explanation: You must delete one element.
Example 4:

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

Example 5:

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

Constraints:

1 <= nums.length <= 105
nums[i] is either 0 or 1.

1494. Parallel Courses II

You are given an integer n, which indicates that there are n courses labeled from 1 to n. You are also given an array r elations where relations[i] = [prevCoursei, nextCoursei], representing a prerequisite relationship between course pre vCoursei and course nextCoursei: course prevCoursei has to be taken before course nextCoursei. Also, you are given the integer k.
In one semester, you can take at most k courses as long as you have taken all the prerequisites in the previous semest er for the courses you are taking.
Return the minimum number of semesters needed to take all courses. The testcases will be generated such that it is p ossible to take every course.

Example 1:

Input: n = 4, dependencies = [[2,1],[3,1],[1,4]], k = 2
Output: 3
Explanation: The figure above represents the given graph.
In the first semester, you can take courses 2 and 3.
In the second semester, you can take course 1.
In the third semester, you can take course 4.

Example 2:

Input: n = 5, dependencies = [[2,1],[3,1],[4,1],[1,5]], k = 2
Output: 4
Explanation: The figure above represents the given graph.
In the first semester, you can take courses 2 and 3 only since you cannot take more than two per semester.
In the second semester, you can take course 4.
In the third semester, you can take course 1.
In the fourth semester, you can take course 5.

Example 3:

Input: n = 11, dependencies = [], k = 2
Output: 6

Constraints:

1 <= n <= 15
1 <= k <= n
0 <= relations.length <= n * (n-1) / 2
relations[i].length == 2
1 <= prevCoursei, nextCoursei <= n
prevCoursei != nextCoursei
All the pairs [prevCoursei, nextCoursei] are unique.
The given graph is a directed acyclic graph.

1496. Path Crossing

Given a string path, where path[i] = 'N', 'S', 'E' or 'W', each representing moving one unit north, south, east, or west, r
espectively. 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 visit
ed. 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'.

1497. Check If Array Pairs Are Divisible by k

Given an array of integers arr of even length n and an integer k.
We want to divide the array into exactly n / 2 pairs such that the sum of each pair is divisible by k.
Return True If you can find a way to do that or False otherwise.

Example 1:

Input: arr = [1,2,3,4,5,10,6,7,8,9], k = 5
Output: true
Explanation: Pairs are (1,9),(2,8),(3,7),(4,6) and (5,10).

Example 2:

Input: arr = [1,2,3,4,5,6], k = 7
Output: true
Explanation: Pairs are (1,6),(2,5) and(3,4).

Example 3:

Input: arr = [1,2,3,4,5,6], k = 10
Output: false
Explanation: You can try all possible pairs to see that there is no way to divide arr into 3 pairs each with sum divisibl
e by 10.

Example 4:

Input: arr = [-10,10], k = 2
Output: true

Example 5:

Input: arr = [-1,1,-2,2,-3,3,-4,4], k = 3
Output: true

Constraints:

arr.length == n
1 <= n <= 105
n is even.
-109 <= arr[i] <= 109
1 <= k <= 105

1498. Number of Subsequences That Satisfy the Given Sum Condition

Given an array of integers nums and an integer target.
Return the number of non-empty subsequences of nums such that the sum of the minimum and maximum element o
n it is less or equal to target. Since the answer may be too large, return it modulo 109 + 7.

Example 1:

Input: nums = [3,5,6,7], target = 9
Output: 4
Explanation: There are 4 subsequences that satisfy the condition.
[3] -> Min value + max value <= target (3 + 3 <= 9)
[3,5] -> (3 + 5 <= 9)
[3,5,6] -> (3 + 6 <= 9)
[3,6] -> (3 + 6 <= 9)

Example 2:

Input: nums = [3,3,6,8], target = 10

Output: 6
Explanation: There are 6 subsequences that satisfy the condition. (nums can have repeated numbers).
[3] , [3] , [3,3], [3,6] , [3,6] , [3,3,6]
Example 3:

Input: nums = [2,3,3,4,6,7], target = 12
Output: 61
Explanation: There are 63 non-empty subsequences, two of them don't satisfy the condition ([6,7], [7]).
Number of valid subsequences (63 - 2 = 61).

Example 4:

Input: nums = [5,2,4,1,7,6,8], target = 16
Output: 127
Explanation: All non-empty subset satisfy the condition (2^7 - 1) = 127

Constraints:

1 <= nums.length <= 105
1 <= nums[i] <= 106
1 <= target <= 106

1499. Max Value of Equation

You are given an array points containing the coordinates of points on a 2D plane, sorted by the x-values, where point s[i] = [xi, yi] such that xi < xj for all 1 <= i < j <= points.length. You are also given an integer k.
Return the maximum value of the equation yi + yj + |xi - xj| where |xi - xj| <= k and 1 <= i < j <= points.length.
It is guaranteed that there exists at least one pair of points that satisfy the constraint |xi - xj| <= k.

Example 1:

Input: points = [[1,3],[2,0],[5,10],[6,-10]], k = 1
Output: 4
Explanation: The first two points satisfy the condition |xi - xj| <= 1 and if we calculate the equation we get 3 + 0 + |1 - 2| = 4. Third and fourth points also satisfy the condition and give a value of 10 + -10 + |5 - 6| = 1.
No other pairs satisfy the condition, so we return the max of 4 and 1.

Example 2:

Input: points = [[0,0],[3,0],[9,2]], k = 3
Output: 3
Explanation: Only the first two points have an absolute difference of 3 or less in the x-values, and give the value of 0 + 0 + |0 - 3| = 3.

Constraints:

2 <= points.length <= 105
points[i].length == 2
-108 <= xi, yi <= 108
0 <= k <= 2 * 108
xi < xj for all 1 <= i < j <= points.length
xi form a strictly increasing sequence.

1502. Can Make Arithmetic Progression From Sequence

A sequence of numbers is called an arithmetic progression if the difference between any two consecutive elements is the same.
Given an array of numbers arr, return true if the array can be rearranged to form an arithmetic progression. Otherwise, return false.

Example 1:

Input: arr = [3,5,1]
Output: true
Explanation: We can reorder the elements as [1,3,5] or [5,3,1] with differences 2 and -2 respectively, between each consecutive elements.

Example 2:

Input: arr = [1,2,4]
Output: false
Explanation: There is no way to reorder the elements to obtain an arithmetic progression.

Constraints:

2 <= arr.length <= 1000
-106 <= arr[i] <= 106

1503. Last Moment Before All Ants Fall Out of a Plank

We have a wooden plank of the length n units. Some ants are walking on the plank, each ant moves with speed 1 unit per second. Some of the ants move to the left, the other move to the right.

When two ants moving in two different directions meet at some point, they change their directions and continue mov ing again. Assume changing directions doesn't take any additional time.
When an ant reaches one end of the plank at a time t, it falls out of the plank imediately.
Given an integer n and two integer arrays left and right, the positions of the ants moving to the left and the right. Ret urn the moment when the last ant(s) fall out of the plank.

Example 1:

Input: n = 4, left = [4,3], right = [0,1]
Output: 4
Explanation: In the image above:
-The ant at index 0 is named A and going to the right.
-The ant at index 1 is named B and going to the right.
-The ant at index 3 is named C and going to the left.
-The ant at index 4 is named D and going to the left.
Note that the last moment when an ant was on the plank is t = 4 second, after that it falls imediately out of the plank. (i.e. We can say that at t = 4.0000000001, there is no ants on the plank).

Example 2:

Input: n = 7, left = [], right = [0,1,2,3,4,5,6,7]
Output: 7
Explanation: All ants are going to the right, the ant at index 0 needs 7 seconds to fall.

Example 3:

Input: n = 7, left = [0,1,2,3,4,5,6,7], right = []
Output: 7
Explanation: All ants are going to the left, the ant at index 7 needs 7 seconds to fall.

Example 4:

Input: n = 9, left = [5], right = [4]
Output: 5
Explanation: At t = 1 second, both ants will be at the same intial position but with different direction.

Example 5:

Input: n = 6, left = [6], right = [0]
Output: 6

Constraints:

1 <= n <= 10^4
0 <= left.length <= n + 1
0 <= left[i] <= n
0 <= right.length <= n + 1
0 <= right[i] <= n
1 <= left.length + right.length <= n + 1
All values of left and right are unique, and each value can appear only in one of the two arrays.

1504. Count Submatrices With All Ones

Given an m x n binary matrix mat, return the number of submatrices that have all ones.

Example 1:

Input: mat = [[1,0,1],[1,1,0],[1,1,0]]
Output: 13
Explanation:
There are 6 rectangles of side 1x1.
There are 2 rectangles of side 1x2.
There are 3 rectangles of side 2x1.
There is 1 rectangle of side 2x2.
There is 1 rectangle of side 3x1.
Total number of rectangles = 6 + 2 + 3 + 1 + 1 = 13.

Example 2:

Input: mat = [[0,1,1,0],[0,1,1,1],[1,1,1,0]]
Output: 24
Explanation:
There are 8 rectangles of side 1x1.
There are 5 rectangles of side 1x2.
There are 2 rectangles of side 1x3.
There are 4 rectangles of side 2x1.
There are 2 rectangles of side 2x2.
There are 2 rectangles of side 3x1.
There is 1 rectangle of side 3x2.
Total number of rectangles = 8 + 5 + 2 + 4 + 2 + 2 + 1 = 24.

Constraints:

1 <= m, n <= 150
mat[i][j] is either 0 or 1.

1505. Minimum Possible Integer After at Most K Adjacent Swaps On Digits

Given a string num representing the digits of a very large integer and an integer k.
You are allowed to swap any two adjacent digits of the integer at most k times.
Return the minimum integer you can obtain also as a string.

Example 1:

Input: num = "4321", k = 4
Output: "1342"
Explanation: The steps to obtain the minimum integer from 4321 with 4 adjacent swaps are shown.

Example 2:

Input: num = "100", k = 1
Output: "010"
Explanation: It's ok for the output to have leading zeros, but the input is guaranteed not to have any leading zeros.

Example 3:

Input: num = "36789", k = 1000
Output: "36789"
Explanation: We can keep the number without any swaps.

Example 4:

Input: num = "22", k = 22
Output: "22"

Example 5:

Input: num = "9438957234785635408", k = 23
Output: "0345989723478563548"

Constraints:

1 <= num.length <= 30000
num contains digits only and doesn't have leading zeros.
1 <= k <= 10^9

1507. Reformat Date

Given a date string in the form Day Month Year, where:

Day is in the set {"1st", "2nd", "3rd", "4th", ..., "30th", "31st"}.
Month is in the set {"Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"}.
Year is in the range [1900, 2100].

Convert the date string to the format YYYY-MM-DD, where:

YYYY denotes the 4 digit year.
MM denotes the 2 digit month.
DD denotes the 2 digit day.

Example 1:

Input: date = "20th Oct 2052"
Output: "2052-10-20"

Example 2:

Input: date = "6th Jun 1933"
Output: "1933-06-06"

Example 3:

Input: date = "26th May 1960"
Output: "1960-05-26"

Constraints:

The given dates are guaranteed to be valid, so no error handling is necessary.

1508. Range Sum of Sorted Subarray Sums

You are given the array nums consisting of n positive integers. You computed the sum of all non-empty continuous s ubarrays from the array and then sorted them in non-decreasing order, creating a new array of n * (n + 1) / 2 number s.
Return the sum of the numbers from index left to index right (indexed from 1), inclusive, in the new array. Since the answer can be a huge number return it modulo 109 + 7.

Example 1:

Input: nums = [1,2,3,4], n = 4, left = 1, right = 5
Output: 13
Explanation: All subarray sums are 1, 3, 6, 10, 2, 5, 9, 3, 7, 4. After sorting them in non-decreasing order we have th e new array [1, 2, 3, 3, 4, 5, 6, 7, 9, 10]. The sum of the numbers from index le = 1 to ri = 5 is 1 + 2 + 3 + 3 + 4 = 13.

Example 2:

Input: nums = [1,2,3,4], n = 4, left = 3, right = 4
Output: 6
Explanation: The given array is the same as example 1. We have the new array [1, 2, 3, 3, 4, 5, 6, 7, 9, 10]. The sum of the numbers from index le = 3 to ri = 4 is 3 + 3 = 6.

Example 3:

Input: nums = [1,2,3,4], n = 4, left = 1, right = 10
Output: 50

Constraints:

n == nums.length
1 <= nums.length <= 1000
1 <= nums[i] <= 100
1 <= left <= right <= n * (n + 1) / 2

1509. Minimum Difference Between Largest and Smallest Value in Three Moves

Given an array nums, you are allowed to choose one element of nums and change it by any value in one move.
Return the minimum difference between the largest and smallest value of nums after perfoming at most 3 moves.

Example 1:

Input: nums = [5,3,2,4]
Output: 0
Explanation: Change the array [5,3,2,4] to [2,2,2,2].
The difference between the maximum and minimum is 2-2 = 0.
Example 2:

Input: nums = [1,5,0,10,14]
Output: 1
Explanation: Change the array [1,5,0,10,14] to [1,1,0,1,1].
The difference between the maximum and minimum is 1-0 = 1.

Example 3:

Input: nums = [6,6,0,1,1,4,6]
Output: 2

Example 4:

Input: nums = [1,5,6,14,15]
Output: 1

Constraints:

1 <= nums.length <= 10^5
-10^9 <= nums[i] <= 10^9

1510. Stone Game IV

Alice and Bob take turns playing a game, with Alice starting first.
Initially, there are n stones in a pile. On each player's turn, that player makes a move consisting of removing any no n-zero square number of stones in the pile.
Also, if a player cannot make a move, he/she loses the game.
Given a positive integer n. Return True if and only if Alice wins the game otherwise return False, assuming both pla yers play optimally.

Example 1:

Input: n = 1
Output: true
Explanation: Alice can remove 1 stone winning the game because Bob doesn't have any moves.
Example 2:

Input: n = 2
Output: false
Explanation: Alice can only remove 1 stone, after that Bob removes the last one winning the game (2 -> 1 -> 0).
Example 3:

Input: n = 4
Output: true
Explanation: n is already a perfect square, Alice can win with one move, removing 4 stones (4 -> 0).

Example 4:

Input: n = 7
Output: false
Explanation: Alice can't win the game if Bob plays optimally.
If Alice starts removing 4 stones, Bob will remove 1 stone then Alice should remove only 1 stone and finally Bob re moves the last one (7 -> 3 -> 2 -> 1 -> 0).
If Alice starts removing 1 stone, Bob will remove 4 stones then Alice only can remove 1 stone and finally Bob remo ves the last one (7 -> 6 -> 2 -> 1 -> 0).
Example 5:

Input: n = 17
Output: false

Explanation: Alice can't win the game if Bob plays optimally.

Constraints:

1 <= n <= 10^5

1512. Number of Good Pairs

Given an array of integers nums, return the number of good pairs.
A pair (i, j) is called good if nums[i] == nums[j] and i < j.

Example 1:

Input: nums = [1,2,3,1,1,3]
Output: 4
Explanation: There are 4 good pairs (0,3), (0,4), (3,4), (2,5) 0-indexed.

Example 2:

Input: nums = [1,1,1,1]
Output: 6
Explanation: Each pair in the array are good.

Example 3:

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

Constraints:

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

1513. Number of Substrings With Only 1s

Given a binary string s, return the number of substrings with all characters 1's. Since the answer may be too large, ret

urn it modulo 109 + 7.

Example 1:

Input: s = "0110111"
Output: 9
Explanation: There are 9 substring in total with only 1's characters.
"1" -> 5 times.
"11" -> 3 times.
"111" -> 1 time.
Example 2:

Input: s = "101"
Output: 2
Explanation: Substring "1" is shown 2 times in s.

Example 3:

Input: s = "111111"
Output: 21
Explanation: Each substring contains only 1's characters.

Example 4:

Input: s = "000"
Output: 0

Constraints:

1 <= s.length <= 105
s[i] is either '0' or '1'.

1514. Path with Maximum Probability

You are given an undirected weighted graph of n nodes (0-indexed), represented by an edge list where edges[i] = [a, b] is an undirected edge connecting the nodes a and b with a probability of success of traversing that edge succProb[i].
Given two nodes start and end, find the path with the maximum probability of success to go from start to end and ret urn its success probability.
If there is no path from start to end, return 0. Your answer will be accepted if it differs from the correct answer by at most 1e-5.

Example 1:

Input: n = 3, edges = [[0,1],[1,2],[0,2]], succProb = [0.5,0.5,0.2], start = 0, end = 2
Output: 0.25000
Explanation: There are two paths from start to end, one having a probability of success = 0.2 and the other has 0.5 * 0.5 = 0.25.

Example 2:

Input: n = 3, edges = [[0,1],[1,2],[0,2]], succProb = [0.5,0.5,0.3], start = 0, end = 2
Output: 0.30000

Example 3:

Input: n = 3, edges = [[0,1]], succProb = [0.5], start = 0, end = 2
Output: 0.00000
Explanation: There is no path between 0 and 2.

Constraints:

2 <= n <= 10^4
0 <= start, end < n
start != end
0 <= a, b < n
a != b
0 <= succProb.length == edges.length <= 2*10^4
0 <= succProb[i] <= 1
There is at most one edge between every two nodes.

1515. Best Position for a Service Centre

A delivery company wants to build a new service centre in a new city. The company knows the positions of all the c ustomers in this city on a 2D-Map and wants to build the new centre in a position such that the sum of the euclidean distances to all customers is minimum.
Given an array positions where positions[i] = [xi, yi] is the position of the ith customer on the map, return the minim um sum of the euclidean distances to all customers.
In other words, you need to choose the position of the service centre [xcentre, ycentre] such that the following formu la is minimized:

Answers within 10^-5 of the actual value will be accepted.

Example 1:

Input: positions = [[0,1],[1,0],[1,2],[2,1]]
Output: 4.00000

Explanation: As shown, you can see that choosing [xcentre, ycentre] = [1, 1] will make the distance to each custome r = 1, the sum of all distances is 4 which is the minimum possible we can achieve.

Example 2:

Input: positions = [[1,1],[3,3]]
Output: 2.82843
Explanation: The minimum possible sum of distances = sqrt(2) + sqrt(2) = 2.82843

Example 3:

Input: positions = [[1,1]]
Output: 0.00000

Example 4:

Input: positions = [[1,1],[0,0],[2,0]]
Output: 2.73205
Explanation: At the first glance, you may think that locating the centre at [1, 0] will achieve the minimum sum, but l ocating it at [1, 0] will make the sum of distances = 3.
Try to locate the centre at [1.0, 0.5773502711] you will see that the sum of distances is 2.73205.
Be careful with the precision!

Example 5:

Input: positions = [[0,1],[3,2],[4,5],[7,6],[8,9],[11,1],[2,12]]
Output: 32.94036
Explanation: You can use [4.3460852395, 4.9813795505] as the position of the centre.

Constraints:

1 <= positions.length <= 50
positions[i].length == 2
0 <= positions[i][0], positions[i][1] <= 100

1518. Water Bottles

Given numBottles full water bottles, you can exchange numExchange empty water bottles for one full water bottle.
The operation of drinking a full water bottle turns it into an empty bottle.
Return the maximum number of water bottles you can drink.

Example 1:

Input: numBottles = 9, numExchange = 3
Output: 13
Explanation: You can exchange 3 empty bottles to get 1 full water bottle.
Number of water bottles you can drink: 9 + 3 + 1 = 13.

Example 2:

Input: numBottles = 15, numExchange = 4
Output: 19
Explanation: You can exchange 4 empty bottles to get 1 full water bottle.
Number of water bottles you can drink: 15 + 3 + 1 = 19.

Example 3:

Input: numBottles = 5, numExchange = 5
Output: 6

Example 4:

Input: numBottles = 2, numExchange = 3
Output: 2

Constraints:

1 <= numBottles <= 100
2 <= numExchange <= 100

1519. Number of Nodes in the Sub-Tree With the Same Label

Given a tree (i.e. a connected, undirected graph that has no cycles) consisting of n nodes numbered from 0 to n - 1 an d exactly n - 1 edges. The root of the tree is the node 0, and each node of the tree has a label which is a lower-case ch aracter given in the string labels (i.e. The node with the number i has the label labels[i]).
The edges array is given on the form edges[i] = [ai, bi], which means there is an edge between nodes ai and bi in the tree.
Return an array of size n where ans[i] is the number of nodes in the subtree of the ith node which have the same labe l as node i.
A subtree of a tree T is the tree consisting of a node in T and all of its descendant nodes.

Example 1:

Input: n = 7, edges = [[0,1],[0,2],[1,4],[1,5],[2,3],[2,6]], labels = "abaedcd"
Output: [2,1,1,1,1,1,1]
Explanation: Node 0 has label 'a' and its sub-tree has node 2 with label 'a' as well, thus the answer is 2. Notice that an y node is part of its sub-tree.

Node 1 has a label 'b'. The sub-tree of node 1 contains nodes 1,4 and 5, as nodes 4 and 5 have different labels than n ode 1, the answer is just 1 (the node itself).

Example 2:

Input: n = 4, edges = [[0,1],[1,2],[0,3]], labels = "bbbb"
Output: [4,2,1,1]
Explanation: The sub-tree of node 2 contains only node 2, so the answer is 1.
The sub-tree of node 3 contains only node 3, so the answer is 1.
The sub-tree of node 1 contains nodes 1 and 2, both have label 'b', thus the answer is 2.
The sub-tree of node 0 contains nodes 0, 1, 2 and 3, all with label 'b', thus the answer is 4.

Example 3:

Input: n = 5, edges = [[0,1],[0,2],[1,3],[0,4]], labels = "aabab"
Output: [3,2,1,1,1]

Example 4:

Input: n = 6, edges = [[0,1],[0,2],[1,3],[3,4],[4,5]], labels = "cbabaa"
Output: [1,2,1,1,2,1]

Example 5:

Input: n = 7, edges = [[0,1],[1,2],[2,3],[3,4],[4,5],[5,6]], labels = "aaabaaa"
Output: [6,5,4,1,3,2,1]

Constraints:

1 <= n <= 10^5
edges.length == n - 1
edges[i].length == 2
0 <= ai, bi < n
ai != bi
labels.length == n
labels is consisting of only of lower-case English letters.

1520. Maximum Number of Non-Overlapping Substrings

Given a string s of lowercase letters, you need to find the maximum number of non-empty substrings of s that meet t he following conditions:

The substrings do not overlap, that is for any two substrings s[i..j] and s[k..l], either j < k or i > l is true.
A substring that contains a certain character c must also contain all occurrences of c.

Find the maximum number of substrings that meet the above conditions. If there are multiple solutions with the sam e number of substrings, return the one with minimum total length. It can be shown that there exists a unique solution of minimum total length.
Notice that you can return the substrings in any order.

Example 1:

Input: s = "adefaddaccc"
Output: ["e","f","ccc"]
Explanation: The following are all the possible substrings that meet the conditions:
[
  "adefaddaccc"
  "adefadda",
  "ef",
  "e",
  "f",
  "ccc",
]
If we choose the first string, we cannot choose anything else and we'd get only 1. If we choose "adefadda", we are le ft with "ccc" which is the only one that doesn't overlap, thus obtaining 2 substrings. Notice also, that it's not optimal to choose "ef" since it can be split into two. Therefore, the optimal way is to choose ["e","f","ccc"] which gives us 3 substrings. No other solution of the same number of substrings exist.

Example 2:

Input: s = "abbaccd"
Output: ["d","bb","cc"]
Explanation: Notice that while the set of substrings ["d","abba","cc"] also has length 3, it's considered incorrect since it has larger total length.

Constraints:

1 <= s.length <= 10^5
s contains only lowercase English letters.

1521. Find a Value of a Mysterious Function Closest to Target

Winston was given the above mysterious function func. He has an integer array arr and an integer target and he want s to find the values l and r that make the value |func(arr, l, r) - target| minimum possible.
Return the minimum possible value of |func(arr, l, r) - target|.
Notice that func should be called with the values l and r where 0 <= l, r < arr.length.

Example 1:

Input: arr = [9,12,3,7,15], target = 5
Output: 2
Explanation: Calling func with all the pairs of [l,r] = [[0,0],[1,1],[2,2],[3,3],[4,4],[0,1],[1,2],[2,3],[3,4],[0,2],[1,3],[2,4],[0,3],[1,4],[0,4]], Winston got the following results [9,12,3,7,15,8,0,3,7,0,0,3,0,0,0]. The value closest to 5 is 7 and 3, thus the minimum difference is 2.

Example 2:

Input: arr = [1000000,1000000,1000000], target = 1
Output: 999999
Explanation: Winston called the func with all possible values of [l,r] and he always got 1000000, thus the min difference is 999999.

Example 3:

Input: arr = [1,2,4,8,16], target = 0
Output: 0

Constraints:

1 <= arr.length <= 105
1 <= arr[i] <= 106
0 <= target <= 107

1523. Count Odd Numbers in an Interval Range

Given two non-negative integers low and high. Return the count of odd numbers between low and high (inclusive).

Example 1:

Input: low = 3, high = 7
Output: 3
Explanation: The odd numbers between 3 and 7 are [3,5,7].
Example 2:

Input: low = 8, high = 10
Output: 1
Explanation: The odd numbers between 8 and 10 are [9].

Constraints:

0 <= low <= high <= 10^9

1524. Number of Sub-arrays With Odd Sum

Given an array of integers arr, return the number of subarrays with an odd sum.
Since the answer can be very large, return it modulo 109 + 7.

Example 1:

Input: arr = [1,3,5]
Output: 4
Explanation: All subarrays are [[1],[1,3],[1,3,5],[3],[3,5],[5]]
All sub-arrays sum are [1,4,9,3,8,5].
Odd sums are [1,9,3,5] so the answer is 4.

Example 2:

Input: arr = [2,4,6]
Output: 0
Explanation: All subarrays are [[2],[2,4],[2,4,6],[4],[4,6],[6]]
All sub-arrays sum are [2,6,12,4,10,6].
All sub-arrays have even sum and the answer is 0.

Example 3:

Input: arr = [1,2,3,4,5,6,7]
Output: 16

Constraints:

1 <= arr.length <= 105
1 <= arr[i] <= 100

1525. Number of Good Ways to Split a String

You are given a string s, a split is called good if you can split s into 2 non-empty strings p and q where its concatenat ion is equal to s and the number of distinct letters in p and q are the same.
Return the number of good splits you can make in s.

Example 1:

Input: s = "aacaba"
Output: 2
Explanation: There are 5 ways to split "aacaba" and 2 of them are good.
("a", "acaba") Left string and right string contains 1 and 3 different letters respectively.
("aa", "caba") Left string and right string contains 1 and 3 different letters respectively.
("aac", "aba") Left string and right string contains 2 and 2 different letters respectively (good split).
("aaca", "ba") Left string and right string contains 2 and 2 different letters respectively (good split).
("aacab", "a") Left string and right string contains 3 and 1 different letters respectively.

Example 2:

Input: s = "abcd"
Output: 1
Explanation: Split the string as follows ("ab", "cd").

Example 3:

Input: s = "aaaaa"
Output: 4
Explanation: All possible splits are good.
Example 4:

Input: s = "acbadbaada"
Output: 2

Constraints:

s contains only lowercase English letters.
1 <= s.length <= 10^5

1526. Minimum Number of Increments on Subarrays to Form a Target Array

Given an array of positive integers target and an array initial of same size with all zeros.
Return the minimum number of operations to form a target array from initial if you are allowed to do the following o peration:

Choose any subarray from initial and increment each value by one.

The answer is guaranteed to fit within the range of a 32-bit signed integer.

Example 1:

Input: target = [1,2,3,2,1]
Output: 3
Explanation: We need at least 3 operations to form the target array from the initial array.
[0,0,0,0,0] increment 1 from index 0 to 4 (inclusive).

[1,1,1,1,1] increment 1 from index 1 to 3 (inclusive).
[1,2,2,2,1] increment 1 at index 2.
[1,2,3,2,1] target array is formed.

Example 2:

Input: target = [3,1,1,2]
Output: 4
Explanation: (initial)[0,0,0,0] -> [1,1,1,1] -> [1,1,1,2] -> [2,1,1,2] -> [3,1,1,2] (target).

Example 3:

Input: target = [3,1,5,4,2]
Output: 7
Explanation: (initial)[0,0,0,0,0] -> [1,1,1,1,1] -> [2,1,1,1,1] -> [3,1,1,1,1]
-> [3,1,2,2,2] -> [3,1,3,3,2] -> [3,1,4,4,2] -> [3,1,5,4,2] (target).

Example 4:

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

Constraints:

1 <= target.length <= 10^5
1 <= target[i] <= 10^5

1528. Shuffle String

Given a string s and an integer array indices of the same length.
The string s will be shuffled such that the character at the ith position moves to indices[i] in the shuffled string.
Return the shuffled string.

Example 1:

Input: s = "codeleet", indices = [4,5,6,7,0,2,1,3]
Output: "leetcode"
Explanation: As shown, "codeleet" becomes "leetcode" after shuffling.

Example 2:

Input: s = "abc", indices = [0,1,2]
Output: "abc"
Explanation: After shuffling, each character remains in its position.

Example 3:

Input: s = "aiohn", indices = [3,1,4,2,0]
Output: "nihao"

Example 4:

Input: s = "aaiougrt", indices = [4,0,2,6,7,3,1,5]
Output: "arigatou"

Example 5:

Input: s = "art", indices = [1,0,2]
Output: "rat"

Constraints:

s.length == indices.length == n
1 <= n <= 100
s contains only lower-case English letters.
0 <= indices[i] < n
All values of indices are unique (i.e. indices is a permutation of the integers from 0 to n - 1).

1529. Bulb Switcher IV

There is a room with n bulbs, numbered from 0 to n - 1, arranged in a row from left to right. Initially, all the bulbs ar e turned off.
Your task is to obtain the configuration represented by target where target[i] is '1' if the ith bulb is turned on and is '0 ' if it is turned off.
You have a switch to flip the state of the bulb, a flip operation is defined as follows:

Choose any bulb (index i) of your current configuration.
Flip each bulb from index i to index n - 1.

When any bulb is flipped it means that if it is '0' it changes to '1' and if it is '1' it changes to '0'.
Return the minimum number of flips required to form target.

Example 1:

Input: target = "10111"
Output: 3
Explanation: Initial configuration "00000".
flip from the third bulb: "00000" -> "00111"
flip from the first bulb: "00111" -> "11000"
flip from the second bulb: "11000" -> "10111"

We need at least 3 flip operations to form target.
Example 2:

Input: target = "101"
Output: 3
Explanation: "000" -> "111" -> "100" -> "101".

Example 3:

Input: target = "00000"
Output: 0

Example 4:

Input: target = "001011101"
Output: 5

Constraints:

1 <= target.length <= 105
target[i] is either '0' or '1'.

1530. Number of Good Leaf Nodes Pairs

Given the root of a binary tree and an integer distance. A pair of two different leaf nodes of a binary tree is said to be good if the length of the shortest path between them is less than or equal to distance.
Return the number of good leaf node pairs in the tree.

Example 1:

Input: root = [1,2,3,null,4], distance = 3
Output: 1
Explanation: The leaf nodes of the tree are 3 and 4 and the length of the shortest path between them is 3. This is the only good pair.

Example 2:

Input: root = [1,2,3,4,5,6,7], distance = 3
Output: 2
Explanation: The good pairs are [4,5] and [6,7] with shortest path = 2. The pair [4,6] is not good because the length of ther shortest path between them is 4.

Example 3:

Input: root = [7,1,4,6,null,5,3,null,null,null,null,null,2], distance = 3
Output: 1
Explanation: The only good pair is [2,5].

Example 4:

Input: root = [100], distance = 1
Output: 0

Example 5:

Input: root = [1,1,1], distance = 2
Output: 1

Constraints:

The number of nodes in the tree is in the range [1, 2^10].
Each node's value is between [1, 100].
1 <= distance <= 10

1531. String Compression II

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.

1534. Count Good Triplets

Given an array of integers arr, and three integers a, b and c. You need to find the number of good triplets.
A triplet (arr[i], arr[j], arr[k]) is good if the following conditions are true:

0 <= i < j < k < arr.length
|arr[i] - arr[j]| <= a
|arr[j] - arr[k]| <= b
|arr[i] - arr[k]| <= c

Where |x| denotes the absolute value of x.
Return the number of good triplets.

Example 1:

Input: arr = [3,0,1,1,9,7], a = 7, b = 2, c = 3
Output: 4
Explanation: There are 4 good triplets: [(3,0,1), (3,0,1), (3,1,1), (0,1,1)].

Example 2:

Input: arr = [1,1,2,2,3], a = 0, b = 0, c = 1
Output: 0
Explanation: No triplet satisfies all conditions.

Constraints:

3 <= arr.length <= 100
0 <= arr[i] <= 1000
0 <= a, b, c <= 1000

1535. Find the Winner of an Array Game

Given an integer array arr of distinct integers and an integer k.
A game will be played between the first two elements of the array (i.e. arr[0] and arr[1]). In each round of the game, we compare arr[0] with arr[1], the larger integer wins and remains at position 0 and the smaller integer moves to the end of the array. The game ends when an integer wins k consecutive rounds.
Return the integer which will win the game.
It is guaranteed that there will be a winner of the game.

Example 1:

Input: arr = [2,1,3,5,4,6,7], k = 2
Output: 5
Explanation: Let's see the rounds of the game:
Round | arr | winner | win_count
1 | [2,1,3,5,4,6,7] | 2 | 1
2 | [2,3,5,4,6,7,1] | 3 | 1
3 | [3,5,4,6,7,1,2] | 5 | 1
4 | [5,4,6,7,1,2,3] | 5 | 2
So we can see that 4 rounds will be played and 5 is the winner because it wins 2 consecutive games.

Example 2:

Input: arr = [3,2,1], k = 10
Output: 3
Explanation: 3 will win the first 10 rounds consecutively.

Example 3:

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

Example 4:

Input: arr = [1,11,22,33,44,55,66,77,88,99], k = 1000000000
Output: 99

Constraints:

2 <= arr.length <= 105
1 <= arr[i] <= 106
arr contains distinct integers.
1 <= k <= 109

1536. Minimum Swaps to Arrange a Binary Grid

Given an n x n binary grid, in one step you can choose two adjacent rows of the grid and swap them.
A grid is said to be valid if all the cells above the main diagonal are zeros.
Return the minimum number of steps needed to make the grid valid, or -1 if the grid cannot be valid.
The main diagonal of a grid is the diagonal that starts at cell (1, 1) and ends at cell (n, n).

Example 1:

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

Example 2:

Input: grid = [[0,1,1,0],[0,1,1,0],[0,1,1,0],[0,1,1,0]]
Output: -1
Explanation: All rows are similar, swaps have no effect on the grid.

Example 3:

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

Constraints:

n == grid.length == grid[i].length
1 <= n <= 200
grid[i][j] is either 0 or 1

1537. Get the Maximum Score

You are given two sorted arrays of distinct integers nums1 and nums2.
A valid path is defined as follows:

Choose array nums1 or nums2 to traverse (from index-0).
Traverse the current array from left to right.
If you are reading any value that is present in nums1 and nums2 you are allowed to change your path to the other arr

ay. (Only one repeated value is considered in the valid path).

The score is defined as the sum of uniques values in a valid path.
Return the maximum score you can obtain of all possible valid paths. Since the answer may be too large, return it m odulo 109 + 7.

Example 1:

Input: nums1 = [2,4,5,8,10], nums2 = [4,6,8,9]
Output: 30
Explanation: Valid paths:
[2,4,5,8,10], [2,4,5,8,9], [2,4,6,8,9], [2,4,6,8,10], (starting from nums1)
[4,6,8,9], [4,5,8,10], [4,5,8,9], [4,6,8,10] (starting from nums2)
The maximum is obtained with the path in green [2,4,6,8,10].

Example 2:

Input: nums1 = [1,3,5,7,9], nums2 = [3,5,100]
Output: 109
Explanation: Maximum sum is obtained with the path [1,3,5,100].

Example 3:

Input: nums1 = [1,2,3,4,5], nums2 = [6,7,8,9,10]
Output: 40
Explanation: There are no common elements between nums1 and nums2.
Maximum sum is obtained with the path [6,7,8,9,10].

Example 4:

Input: nums1 = [1,4,5,8,9,11,19], nums2 = [2,3,4,11,12]
Output: 61

Constraints:

1 <= nums1.length, nums2.length <= 105
1 <= nums1[i], nums2[i] <= 107
nums1 and nums2 are strictly increasing.

1539. Kth Missing Positive Number

Given an array arr of positive integers sorted in a strictly increasing order, and an integer k.
Find the kth positive integer that is missing from this array.

Example 1:

Input: arr = [2,3,4,7,11], k = 5
Output: 9
Explanation: The missing positive integers are [1,5,6,8,9,10,12,13,...]. The 5th missing positive integer is 9.

Example 2:

Input: arr = [1,2,3,4], k = 2
Output: 6
Explanation: The missing positive integers are [5,6,7,...]. The 2nd missing positive integer is 6.

Constraints:

1 <= arr.length <= 1000
1 <= arr[i] <= 1000
1 <= k <= 1000
arr[i] < arr[j] for 1 <= i < j <= arr.length

1540. Can Convert String in K Moves

Given two strings s and t, your goal is to convert s into t in k moves or less.
During the ith (1 <= i <= k) move you can:

Choose any index j (1-indexed) from s, such that 1 <= j <= s.length and j has not been chosen in any previous move, and shift the character at that index i times.
Do nothing.

Shifting a character means replacing it by the next letter in the alphabet (wrapping around so that 'z' becomes 'a'). Shi fting a character by i means applying the shift operations i times.
Remember that any index j can be picked at most once.
Return true if it's possible to convert s into t in no more than k moves, otherwise return false.

Example 1:

Input: s = "input", t = "ouput", k = 9
Output: true
Explanation: In the 6th move, we shift 'i' 6 times to get 'o'. And in the 7th move we shift 'n' to get 'u'.

Example 2:

Input: s = "abc", t = "bcd", k = 10
Output: false
Explanation: We need to shift each character in s one time to convert it into t. We can shift 'a' to 'b' during the 1st mo ve. However, there is no way to shift the other characters in the remaining moves to obtain t from s.

Example 3:

Input: s = "aab", t = "bbb", k = 27
Output: true
Explanation: In the 1st move, we shift the first 'a' 1 time to get 'b'. In the 27th move, we shift the second 'a' 27 times t o get 'b'.

Constraints:

1 <= s.length, t.length <= 10^5
0 <= k <= 10^9
s, t contain only lowercase English letters.

1541. Minimum Insertions to Balance a Parentheses String

Given a parentheses string s containing only the characters '(' and ')'. A parentheses string is balanced if:

Any left parenthesis '(' must have a corresponding two consecutive right parenthesis '))'.
Left parenthesis '(' must go before the corresponding two consecutive right parenthesis '))'.

In other words, we treat '(' as openning parenthesis and '))' as closing parenthesis.
For example, "())", "())(())))" and "(())())))" are balanced, ")()", "))))" and "(()))" are not balanced.
You can insert the characters '(' and ')' at any position of the string to balance it if needed.
Return the minimum number of insertions needed to make s balanced.

Example 1:

Input: s = "(()))"
Output: 1
Explanation: The second '(' has two matching '))', but the first '(' has only ')' matching. We need to to add one more ')' at the end of the string to be "(())))" which is balanced.

Example 2:

Input: s = "())"
Output: 0
Explanation: The string is already balanced.

Example 3:

Input: s = "))())("
Output: 3
Explanation: Add '(' to match the first '))', Add '))' to match the last '('.

Example 4:

Input: s = "((((((("
Output: 12
Explanation: Add 12 ')' to balance the string.

Example 5:

Input: s = "))))))))"
Output: 5
Explanation: Add 4 '(' at the beginning of the string and one ')' at the end. The string becomes "(((()))))))))".

Constraints:

1 <= s.length <= 10^5
s consists of '(' and ')' only.

1542. Find Longest Awesome Substring

Given a string s. An awesome substring is a non-empty substring of s such that we can make any number of swaps in order to make it palindrome.
Return the length of the maximum length awesome substring of s.

Example 1:

Input: s = "3242415"
Output: 5
Explanation: "24241" is the longest awesome substring, we can form the palindrome "24142" with some swaps.

Example 2:

Input: s = "12345678"
Output: 1

Example 3:

Input: s = "213123"
Output: 6
Explanation: "213123" is the longest awesome substring, we can form the palindrome "231132" with some swaps.

Example 4:

Input: s = "00"
Output: 2

Constraints:

1 <= s.length <= 105
s consists only of digits.

1544. Make The String Great

Given a string s of lower and upper case English letters.
A good string is a string which doesn't have two adjacent characters s[i] and s[i + 1] where:

0 <= i <= s.length - 2
s[i] is a lower-case letter and s[i + 1] is the same letter but in upper-case or vice-versa.

To make the string good, you can choose two adjacent characters that make the string bad and remove them. You ca
n keep doing this until the string becomes good.
Return the string after making it good. The answer is guaranteed to be unique under the given constraints.
Notice that an empty string is also good.

Example 1:

Input: s = "leEeetcode"
Output: "leetcode"
Explanation: In the first step, either you choose i = 1 or i = 2, both will result "leEeetcode" to be reduced to "leetcode
".

Example 2:

Input: s = "abBAcC"
Output: ""
Explanation: We have many possible scenarios, and all lead to the same answer. For example:
"abBAcC" --> "aAcC" --> "cC" --> ""
"abBAcC" --> "abBA" --> "aA" --> ""

Example 3:

Input: s = "s"
Output: "s"

Constraints:

1 <= s.length <= 100
s contains only lower and upper case English letters.

1545. Find Kth Bit in Nth Binary String

Given two positive integers n and k, the binary string Sn is formed as follows:

S1 = "0"
Si = Si-1 + "1" + reverse(invert(Si-1)) for i > 1

Where + denotes the concatenation operation, reverse(x) returns the reversed string x, and invert(x) inverts all the bit s in x (0 changes to 1 and 1 changes to 0).
For example, the first 4 strings in the above sequence are:

S1 = "0"
S2 = "011"
S3 = "0111001"
S4 = "011100110110001"

Return the kth bit in Sn. It is guaranteed that k is valid for the given n.

Example 1:

Input: n = 3, k = 1
Output: "0"
Explanation: S3 is "0111001". The first bit is "0".

Example 2:

Input: n = 4, k = 11
Output: "1"
Explanation: S4 is "011100110110001". The 11th bit is "1".

Example 3:

Input: n = 1, k = 1
Output: "0"

Example 4:

Input: n = 2, k = 3
Output: "1"

Constraints:

1 <= n <= 20
1 <= k <= 2n - 1

1546. Maximum Number of Non-Overlapping Subarrays With Sum Equals Target

Given an array nums and an integer target.
Return the maximum number of non-empty non-overlapping subarrays such that the sum of values in each subarray i s equal to target.

Example 1:

Input: nums = [1,1,1,1,1], target = 2
Output: 2
Explanation: There are 2 non-overlapping subarrays [1,1,1,1,1] with sum equals to target(2).

Example 2:

Input: nums = [-1,3,5,1,4,2,-9], target = 6
Output: 2
Explanation: There are 3 subarrays with sum equal to 6.
([5,1], [4,2], [3,5,1,4,2,-9]) but only the first 2 are non-overlapping.
Example 3:

Input: nums = [-2,6,6,3,5,4,1,2,8], target = 10
Output: 3

Example 4:

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

Constraints:

1 <= nums.length <= 10^5
-10^4 <= nums[i] <= 10^4
0 <= target <= 10^6

1547. Minimum Cost to Cut a Stick

Given a wooden stick of length n units. The stick is labelled from 0 to n. For example, a stick of length 6 is labelled as follows:

Given an integer array cuts where cuts[i] denotes a position you should perform a cut at.
You should perform the cuts in order, you can change the order of the cuts as you wish.
The cost of one cut is the length of the stick to be cut, the total cost is the sum of costs of all cuts. When you cut a sti ck, it will be split into two smaller sticks (i.e. the sum of their lengths is the length of the stick before the cut). Please refer to the first example for a better explanation.
Return the minimum total cost of the cuts.

Example 1:

Input: n = 7, cuts = [1,3,4,5]
Output: 16
Explanation: Using cuts order = [1, 3, 4, 5] as in the input leads to the following scenario:

The first cut is done to a rod of length 7 so the cost is 7. The second cut is done to a rod of length 6 (i.e. the second p art of the first cut), the third is done to a rod of length 4 and the last cut is to a rod of length 3. The total cost is 7 + 6 + 4 + 3 = 20.
Rearranging the cuts to be [3, 5, 1, 4] for example will lead to a scenario with total cost = 16 (as shown in the examp le photo 7 + 4 + 3 + 2 = 16).
Example 2:

Input: n = 9, cuts = [5,6,1,4,2]
Output: 22
Explanation: If you try the given cuts ordering the cost will be 25.
There are much ordering with total cost <= 25, for example, the order [4, 6, 5, 2, 1] has total cost = 22 which is the minimum possible.

Constraints:

2 <= n <= 106
1 <= cuts.length <= min(n - 1, 100)
1 <= cuts[i] <= n - 1
All the integers in cuts array are distinct.

1550. Three Consecutive Odds

Given an integer array arr, return true if there are three consecutive odd numbers in the array. Otherwise, return false .

Example 1:

Input: arr = [2,6,4,1]
Output: false
Explanation: There are no three consecutive odds.

Example 2:

Input: arr = [1,2,34,3,4,5,7,23,12]
Output: true
Explanation: [5,7,23] are three consecutive odds.

Constraints:

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

1551. Minimum Operations to Make Array Equal

You have an array arr of length n where arr[i] = (2 * i) + 1 for all valid values of i (i.e., 0 <= i < n).
In one operation, you can select two indices x and y where 0 <= x, y < n and subtract 1 from arr[x] and add 1 to arr[y] (i.e., perform arr[x] -=1 and arr[y] += 1). The goal is to make all the elements of the array equal. It is guaranteed th at all the elements of the array can be made equal using some operations.
Given an integer n, the length of the array, return the minimum number of operations needed to make all the element s of arr equal.

Example 1:

Input: n = 3
Output: 2
Explanation: arr = [1, 3, 5]
First operation choose x = 2 and y = 0, this leads arr to be [2, 3, 4]
In the second operation choose x = 2 and y = 0 again, thus arr = [3, 3, 3].

Example 2:

Input: n = 6
Output: 9

Constraints:

1 <= n <= 104

1552. Magnetic Force Between Two Balls

In the universe Earth C-137, Rick discovered a special form of magnetic force between two balls if they are put in hi s new invented basket. Rick has n empty baskets, the ith basket is at position[i], Morty has m balls and needs to distr ibute the balls into the baskets such that the minimum magnetic force between any two balls is maximum.
Rick stated that magnetic force between two different balls at positions x and y is |x - y|.
Given the integer array position and the integer m. Return the required force.

Example 1:

Input: position = [1,2,3,4,7], m = 3
Output: 3
Explanation: Distributing the 3 balls into baskets 1, 4 and 7 will make the magnetic force between ball pairs [3, 3, 6]. The minimum magnetic force is 3. We cannot achieve a larger minimum magnetic force than 3.

Example 2:

Input: position = [5,4,3,2,1,1000000000], m = 2
Output: 999999999
Explanation: We can use baskets 1 and 1000000000.

Constraints:

n == position.length
2 <= n <= 105
1 <= position[i] <= 109
All integers in position are distinct.
2 <= m <= position.length

1553. Minimum Number of Days to Eat N Oranges

There are n oranges in the kitchen and you decided to eat some of these oranges every day as follows:

Eat one orange.
If the number of remaining oranges n is divisible by 2 then you can eat n / 2 oranges.
If the number of remaining oranges n is divisible by 3 then you can eat 2 * (n / 3) oranges.

You can only choose one of the actions per day.
Return the minimum number of days to eat n oranges.

Example 1:

Input: n = 10
Output: 4
Explanation: You have 10 oranges.
Day 1: Eat 1 orange, 10 - 1 = 9.
Day 2: Eat 6 oranges, 9 - 2*(9/3) = 9 - 6 = 3. (Since 9 is divisible by 3)
Day 3: Eat 2 oranges, 3 - 2*(3/3) = 3 - 2 = 1.
Day 4: Eat the last orange 1 - 1 = 0.
You need at least 4 days to eat the 10 oranges.

Example 2:

Input: n = 6
Output: 3
Explanation: You have 6 oranges.
Day 1: Eat 3 oranges, 6 - 6/2 = 6 - 3 = 3. (Since 6 is divisible by 2).
Day 2: Eat 2 oranges, 3 - 2*(3/3) = 3 - 2 = 1. (Since 3 is divisible by 3)
Day 3: Eat the last orange 1 - 1 = 0.
You need at least 3 days to eat the 6 oranges.

Example 3:

Input: n = 1
Output: 1

Example 4:

Input: n = 56
Output: 6

Constraints:

1 <= n <= 2 * 109

1556. Thousand Separator

Given an integer n, add a dot (".") as the thousands separator and return it in string format.

Example 1:

Input: n = 987
Output: "987"

Example 2:

Input: n = 1234

Output: "1.234"

Example 3:

Input: n = 123456789
Output: "123.456.789"

Example 4:

Input: n = 0
Output: "0"

Constraints:

0 <= n < 231

1557. Minimum Number of Vertices to Reach All Nodes

Given a directed acyclic graph, with n vertices numbered from 0 to n-1, and an array edges where edges[i] = [fromi, toi] represents a directed edge from node fromi to node toi.
Find the smallest set of vertices from which all nodes in the graph are reachable. It's guaranteed that a unique solutio n exists.
Notice that you can return the vertices in any order.

Example 1:

Input: n = 6, edges = [[0,1],[0,2],[2,5],[3,4],[4,2]]
Output: [0,3]
Explanation: It's not possible to reach all the nodes from a single vertex. From 0 we can reach [0,1,2,5]. From 3 we c an reach [3,4,2,5]. So we output [0,3].
Example 2:

Input: n = 5, edges = [[0,1],[2,1],[3,1],[1,4],[2,4]]
Output: [0,2,3]
Explanation: Notice that vertices 0, 3 and 2 are not reachable from any other node, so we must include them. Also an y of these vertices can reach nodes 1 and 4.

Constraints:

2 <= n <= 10^5
1 <= edges.length <= min(10^5, n * (n - 1) / 2)
edges[i].length == 2

0 <= fromi, toi < n
All pairs (fromi, toi) are distinct.

1558. Minimum Numbers of Function Calls to Make Target Array

Your task is to form an integer array nums from an initial array of zeros arr that is the same size as nums.
Return the minimum number of function calls to make nums from arr.
The answer is guaranteed to fit in a 32-bit signed integer.

Example 1:

Input: nums = [1,5]
Output: 5
Explanation: Increment by 1 (second element): [0, 0] to get [0, 1] (1 operation).
Double all the elements: [0, 1] -> [0, 2] -> [0, 4] (2 operations).
Increment by 1 (both elements) [0, 4] -> [1, 4] -> [1, 5] (2 operations).
Total of operations: 1 + 2 + 2 = 5.

Example 2:

Input: nums = [2,2]
Output: 3
Explanation: Increment by 1 (both elements) [0, 0] -> [0, 1] -> [1, 1] (2 operations).
Double all the elements: [1, 1] -> [2, 2] (1 operation).
Total of operations: 2 + 1 = 3.

Example 3:

Input: nums = [4,2,5]
Output: 6
Explanation: (initial)[0,0,0] -> [1,0,0] -> [1,0,1] -> [2,0,2] -> [2,1,2] -> [4,2,4] -> [4,2,5](nums).

Example 4:

Input: nums = [3,2,2,4]
Output: 7

Example 5:

Input: nums = [2,4,8,16]
Output: 8

Constraints:

1 <= nums.length <= 10^5

0 <= nums[i] <= 10^9

1559. Detect Cycles in 2D Grid

Given a 2D array of characters grid of size m x n, you need to find if there exists any cycle consisting of the same va lue in grid.
A cycle is a path of length 4 or more in the grid that starts and ends at the same cell. From a given cell, you can mov e to one of the cells adjacent to it - in one of the four directions (up, down, left, or right), if it has the same value of t he current cell.
Also, you cannot move to the cell that you visited in your last move. For example, the cycle (1, 1) -> (1, 2) -> (1, 1) i s invalid because from (1, 2) we visited (1, 1) which was the last visited cell.
Return true if any cycle of the same value exists in grid, otherwise, return false.

Example 1:

Input: grid = [["a","a","a","a"],["a","b","b","a"],["a","b","b","a"],["a","a","a","a"]]
Output: true
Explanation: There are two valid cycles shown in different colors in the image below:

Example 2:

Input: grid = [["c","c","c","a"],["c","d","c","c"],["c","c","e","c"],["f","c","c","c"]]
Output: true
Explanation: There is only one valid cycle highlighted in the image below:

Example 3:

Input: grid = [["a","b","b"],["b","z","b"],["b","b","a"]]
Output: false

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 500
grid consists only of lowercase English letters.

1560. Most Visited Sector in a Circular Track

Given an integer n and an integer array rounds. We have a circular track which consists of n sectors labeled from 1 t
o n. A marathon will be held on this track, the marathon consists of m rounds. The ith round starts at sector rounds[i
- 1] and ends at sector rounds[i]. For example, round 1 starts at sector rounds[0] and ends at sector rounds[1]
Return an array of the most visited sectors sorted in ascending order.
Notice that you circulate the track in ascending order of sector numbers in the counter-clockwise direction (See the fi
rst example).

Example 1:

Input: n = 4, rounds = [1,3,1,2]
Output: [1,2]
Explanation: The marathon starts at sector 1. The order of the visited sectors is as follows:
1 --> 2 --> 3 (end of round 1) --> 4 --> 1 (end of round 2) --> 2 (end of round 3 and the marathon)
We can see that both sectors 1 and 2 are visited twice and they are the most visited sectors. Sectors 3 and 4 are visite
d only once.
Example 2:

Input: n = 2, rounds = [2,1,2,1,2,1,2,1,2]
Output: [2]

Example 3:

Input: n = 7, rounds = [1,3,5,7]
Output: [1,2,3,4,5,6,7]

Constraints:

2 <= n <= 100
1 <= m <= 100
rounds.length == m + 1
1 <= rounds[i] <= n
rounds[i] != rounds[i + 1] for 0 <= i < m

1561. Maximum Number of Coins You Can Get

There are 3n piles of coins of varying size, you and your friends will take piles of coins as follows:

In each step, you will choose any 3 piles of coins (not necessarily consecutive).
Of your choice, Alice will pick the pile with the maximum number of coins.
You will pick the next pile with the maximum number of coins.
Your friend Bob will pick the last pile.
Repeat until there are no more piles of coins.

Given an array of integers piles where piles[i] is the number of coins in the ith pile.
Return the maximum number of coins that you can have.

Example 1:

Input: piles = [2,4,1,2,7,8]
Output: 9
Explanation: Choose the triplet (2, 7, 8), Alice Pick the pile with 8 coins, you the pile with 7 coins and Bob the last o ne.
Choose the triplet (1, 2, 4), Alice Pick the pile with 4 coins, you the pile with 2 coins and Bob the last one.
The maximum number of coins which you can have are: 7 + 2 = 9.
On the other hand if we choose this arrangement (1, 2, 8), (2, 4, 7) you only get 2 + 4 = 6 coins which is not optimal.

Example 2:

Input: piles = [2,4,5]
Output: 4

Example 3:

Input: piles = [9,8,7,6,5,1,2,3,4]
Output: 18

Constraints:

3 <= piles.length <= 105
piles.length % 3 == 0
1 <= piles[i] <= 104

1562. Find Latest Group of Size M

Given an array arr that represents a permutation of numbers from 1 to n.
You have a binary string of size n that initially has all its bits set to zero. At each step i (assuming both the binary stri ng and arr are 1-indexed) from 1 to n, the bit at position arr[i] is set to 1.
You are also given an integer m. Find the latest step at which there exists a group of ones of length m. A group of on es is a contiguous substring of 1's such that it cannot be extended in either direction.
Return the latest step at which there exists a group of ones of length exactly m. If no such group exists, return -1.

Example 1:

Input: arr = [3,5,1,2,4], m = 1
Output: 4
Explanation:
Step 1: "00100", groups: ["1"]
Step 2: "00101", groups: ["1", "1"]
Step 3: "10101", groups: ["1", "1", "1"]
Step 4: "11101", groups: ["111", "1"]
Step 5: "11111", groups: ["11111"]
The latest step at which there exists a group of size 1 is step 4.
Example 2:

Input: arr = [3,1,5,4,2], m = 2
Output: -1
Explanation:
Step 1: "00100", groups: ["1"]
Step 2: "10100", groups: ["1", "1"]
Step 3: "10101", groups: ["1", "1", "1"]
Step 4: "10111", groups: ["1", "111"]
Step 5: "11111", groups: ["11111"]
No group of size 2 exists during any step.

Example 3:

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

Example 4:

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

Constraints:

n == arr.length
1 <= m <= n <= 105
1 <= arr[i] <= n
All integers in arr are distinct.

1563. Stone Game V

There are several stones arranged in a row, and each stone has an associated value which is an integer given in the ar ray stoneValue.
In each round of the game, Alice divides the row into two non-empty rows (i.e. left row and right row), then Bob cal culates the value of each row which is the sum of the values of all the stones in this row. Bob throws away the row w

hich has the maximum value, and Alice's score increases by the value of the remaining row. If the value of the two r ows are equal, Bob lets Alice decide which row will be thrown away. The next round starts with the remaining row. The game ends when there is only one stone remaining. Alice's is initially zero.
Return the maximum score that Alice can obtain.

Example 1:

Input: stoneValue = [6,2,3,4,5,5]
Output: 18
Explanation: In the first round, Alice divides the row to [6,2,3], [4,5,5]. The left row has the value 11 and the right ro w has value 14. Bob throws away the right row and Alice's score is now 11.
In the second round Alice divides the row to [6], [2,3]. This time Bob throws away the left row and Alice's score bec omes 16 (11 + 5).
The last round Alice has only one choice to divide the row which is [2], [3]. Bob throws away the right row and Alic e's score is now 18 (16 + 2). The game ends because only one stone is remaining in the row.

Example 2:

Input: stoneValue = [7,7,7,7,7,7,7]
Output: 28

Example 3:

Input: stoneValue = [4]
Output: 0

Constraints:

1 <= stoneValue.length <= 500
1 <= stoneValue[i] <= 106

1566. Detect Pattern of Length M Repeated K or More Times

Given an array of positive integers arr, find a pattern of length m that is repeated k or more times.
A pattern is a subarray (consecutive sub-sequence) that consists of one or more values, repeated multiple times conse cutively without overlapping. A pattern is defined by its length and the number of repetitions.
Return true if there exists a pattern of length m that is repeated k or more times, otherwise return false.

Example 1:

Input: arr = [1,2,4,4,4,4], m = 1, k = 3
Output: true
Explanation: The pattern (4) of length 1 is repeated 4 consecutive times. Notice that pattern can be repeated k or mor e times but not less.

Example 2:

Input: arr = [1,2,1,2,1,1,1,3], m = 2, k = 2
Output: true
Explanation: The pattern (1,2) of length 2 is repeated 2 consecutive times. Another valid pattern (2,1) is also repeate d 2 times.

Example 3:

Input: arr = [1,2,1,2,1,3], m = 2, k = 3
Output: false
Explanation: The pattern (1,2) is of length 2 but is repeated only 2 times. There is no pattern of length 2 that is repeat ed 3 or more times.

Example 4:

Input: arr = [1,2,3,1,2], m = 2, k = 2
Output: false
Explanation: Notice that the pattern (1,2) exists twice but not consecutively, so it doesn't count.

Example 5:

Input: arr = [2,2,2,2], m = 2, k = 3
Output: false
Explanation: The only pattern of length 2 is (2,2) however it's repeated only twice. Notice that we do not count overl apping repetitions.

Constraints:

2 <= arr.length <= 100
1 <= arr[i] <= 100
1 <= m <= 100
2 <= k <= 100

1567. Maximum Length of Subarray With Positive Product

Given an array of integers nums, find the maximum length of a subarray where the product of all its elements is posi tive.
A subarray of an array is a consecutive sequence of zero or more values taken out of that array.
Return the maximum length of a subarray with positive product.

Example 1:

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

Explanation: The array nums already has a positive product of 24.

Example 2:

Input: nums = [0,1,-2,-3,-4]
Output: 3
Explanation: The longest subarray with positive product is [1,-2,-3] which has a product of 6.
Notice that we cannot include 0 in the subarray since that'll make the product 0 which is not positive.
Example 3:

Input: nums = [-1,-2,-3,0,1]
Output: 2
Explanation: The longest subarray with positive product is [-1,-2] or [-2,-3].

Example 4:

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

Example 5:

Input: nums = [1,2,3,5,-6,4,0,10]
Output: 4

Constraints:

1 <= nums.length <= 10^5
-10^9 <= nums[i] <= 10^9

1568. Minimum Number of Days to Disconnect Island

Given a 2D grid consisting of 1s (land) and 0s (water). An island is a maximal 4-directionally (horizontal or vertical) connected group of 1s.
The grid is said to be connected if we have exactly one island, otherwise is said disconnected.
In one day, we are allowed to change any single land cell (1) into a water cell (0).
Return the minimum number of days to disconnect the grid.

Example 1:

Input: grid = [[0,1,1,0],[0,1,1,0],[0,0,0,0]]
Output: 2
Explanation: We need at least 2 days to get a disconnected grid.
Change land grid[1][1] and grid[0][2] to water and get 2 disconnected island.

Example 2:

Input: grid = [[1,1]]
Output: 2
Explanation: Grid of full water is also disconnected ([[1,1]] -> [[0,0]]), 0 islands.

Example 3:

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

Example 4:

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

Example 5:

Input: grid = [[1,1,0,1,1],
[1,1,1,1,1],
[1,1,0,1,1],
[1,1,1,1,1]]
Output: 2

Constraints:

1 <= grid.length, grid[i].length <= 30
grid[i][j] is 0 or 1.

1569. Number of Ways to Reorder Array to Get Same BST

Given an array nums that represents a permutation of integers from 1 to n. We are going to construct a binary search tree (BST) by inserting the elements of nums in order into an initially empty BST. Find the number of different ways to reorder nums so that the constructed BST is identical to that formed from the original array nums.
For example, given nums = [2,1,3], we will have 2 as the root, 1 as a left child, and 3 as a right child. The array [2,3,1] also yields the same BST but [3,2,1] yields a different BST.
Return the number of ways to reorder nums such that the BST formed is identical to the original BST formed from nums.
Since the answer may be very large, return it modulo 10^9 + 7.

Example 1:

Input: nums = [2,1,3]
Output: 1
Explanation: We can reorder nums to be [2,3,1] which will yield the same BST. There are no other ways to reorder n ums which will yield the same BST.

Example 2:

Input: nums = [3,4,5,1,2]
Output: 5
Explanation: The following 5 arrays will yield the same BST:
[3,1,2,4,5]
[3,1,4,2,5]
[3,1,4,5,2]
[3,4,1,2,5]
[3,4,1,5,2]

Example 3:

Input: nums = [1,2,3]
Output: 0
Explanation: There are no other orderings of nums that will yield the same BST.

Example 4:

Input: nums = [3,1,2,5,4,6]
Output: 19

Example 5:

Input: nums = [9,4,2,1,3,6,5,7,8,14,11,10,12,13,16,15,17,18]
Output: 216212978
Explanation: The number of ways to reorder nums to get the same BST is 3216212999. Taking this number modulo 10^9 + 7 gives 216212978.

Constraints:

1 <= nums.length <= 1000
1 <= nums[i] <= nums.length
All integers in nums are distinct.

1572. Matrix Diagonal Sum

Given a square matrix mat, return the sum of the matrix diagonals.
Only include the sum of all the elements on the primary diagonal and all the elements on the secondary diagonal that are not part of the primary diagonal.

Example 1:

Input: mat = [[1,2,3],
[4,5,6],
[7,8,9]]
Output: 25
Explanation: Diagonals sum: 1 + 5 + 9 + 3 + 7 = 25
Notice that element mat[1][1] = 5 is counted only once.

Example 2:

Input: mat = [[1,1,1,1],
[1,1,1,1],
[1,1,1,1],
[1,1,1,1]]
Output: 8

Example 3:

Input: mat = [[5]]
Output: 5

Constraints:

n == mat.length == mat[i].length
1 <= n <= 100
1 <= mat[i][j] <= 100

1573. Number of Ways to Split a String

Given a binary string s (a string consisting only of '0's and '1's), we can split s into 3 non-empty strings s1, s2, s3 (s1 + s2+ s3 = s).
Return the number of ways s can be split such that the number of characters '1' is the same in s1, s2, and s3.
Since the answer may be too large, return it modulo 10^9 + 7.

Example 1:

Input: s = "10101"
Output: 4

Explanation: There are four ways to split s in 3 parts where each part contain the same number of letters '1'.
"1|010|1"
"1|01|01"
"10|10|1"
"10|1|01"

Example 2:

Input: s = "1001"
Output: 0

Example 3:

Input: s = "0000"
Output: 3
Explanation: There are three ways to split s in 3 parts.
"0|0|00"
"0|00|0"
"00|0|0"

Example 4:

Input: s = "100100010100110"
Output: 12

Constraints:

3 <= s.length <= 10^5
s[i] is '0' or '1'.

1574. Shortest Subarray to be Removed to Make Array Sorted

Given an integer array arr, remove a subarray (can be empty) from arr such that the remaining elements in arr are no n-decreasing.
Return the length of the shortest subarray to remove.
A subarray is a contiguous subsequence of the array.

Example 1:

Input: arr = [1,2,3,10,4,2,3,5]
Output: 3
Explanation: The shortest subarray we can remove is [10,4,2] of length 3. The remaining elements after that will be [1,2,3,3,5] which are sorted.
Another correct solution is to remove the subarray [3,10,4].
Example 2:

Input: arr = [5,4,3,2,1]
Output: 4
Explanation: Since the array is strictly decreasing, we can only keep a single element. Therefore we need to remove a subarray of length 4, either [5,4,3,2] or [4,3,2,1].

Example 3:

Input: arr = [1,2,3]
Output: 0
Explanation: The array is already non-decreasing. We do not need to remove any elements.

Example 4:

Input: arr = [1]
Output: 0

Constraints:

1 <= arr.length <= 105
0 <= arr[i] <= 109

1575. Count All Possible Routes

You are given an array of distinct positive integers locations where locations[i] represents the position of city i. You are also given integers start, finish and fuel representing the starting city, ending city, and the initial amount of fuel y ou have, respectively.
At each step, if you are at city i, you can pick any city j such that j != i and 0 <= j < locations.length and move to city j. Moving from city i to city j reduces the amount of fuel you have by |locations[i] - locations[j]|. Please notice that | x| denotes the absolute value of x.
Notice that fuel cannot become negative at any point in time, and that you are allowed to visit any city more than on ce (including start and finish).
Return the count of all possible routes from start to finish. Since the answer may be too large, return it modulo 109 + 7.

Example 1:

Input: locations = [2,3,6,8,4], start = 1, finish = 3, fuel = 5
Output: 4
Explanation: The following are all possible routes, each uses 5 units of fuel:
1 -> 3
1 -> 2 -> 3
1 -> 4 -> 3
1 -> 4 -> 2 -> 3

Example 2:

Input: locations = [4,3,1], start = 1, finish = 0, fuel = 6
Output: 5
Explanation: The following are all possible routes:
1 -> 0, used fuel = 1
1 -> 2 -> 0, used fuel = 5
1 -> 2 -> 1 -> 0, used fuel = 5
1 -> 0 -> 1 -> 0, used fuel = 3
1 -> 0 -> 1 -> 0 -> 1 -> 0, used fuel = 5

Example 3:

Input: locations = [5,2,1], start = 0, finish = 2, fuel = 3
Output: 0
Explanation: It's impossible to get from 0 to 2 using only 3 units of fuel since the shortest route needs 4 units of fuel.
Example 4:

Input: locations = [2,1,5], start = 0, finish = 0, fuel = 3
Output: 2
Explanation: There are two possible routes, 0 and 0 -> 1 -> 0.
Example 5:

Input: locations = [1,2,3], start = 0, finish = 2, fuel = 40
Output: 615088286
Explanation: The total number of possible routes is 2615088300. Taking this number modulo 10^9 + 7 gives us 615 088286.

Constraints:

2 <= locations.length <= 100
1 <= locations[i] <= 109
All integers in locations are distinct.
0 <= start, finish < locations.length
1 <= fuel <= 200

1576. Replace All ?'s to Avoid Consecutive Repeating Characters

Given a string s containing only lowercase English letters and the '?' character, convert all the '?' characters into lowe rcase letters such that the final string does not contain any consecutive repeating characters. You cannot modify the non '?' characters.
It is guaranteed that there are no consecutive repeating characters in the given string except for '?'.
Return the final string after all the conversions (possibly zero) have been made. If there is more than one solution, ret urn any of them. It can be shown that an answer is always possible with the given constraints.

Example 1:

Input: s = "?zs"
Output: "azs"
Explanation: There are 25 solutions for this problem. From "azs" to "yzs", all are valid. Only "z" is an invalid modifi cation as the string will consist of consecutive repeating characters in "zzs".
Example 2:

Input: s = "ubv?w"
Output: "ubvaw"
Explanation: There are 24 solutions for this problem. Only "v" and "w" are invalid modifications as the strings will c onsist of consecutive repeating characters in "ubvvw" and "ubvww".

Example 3:

Input: s = "j?qg??b"
Output: "jaqgacb"

Example 4:

Input: s = "??yw?ipkj?"
Output: "acywaipkja"

Constraints:

1 <= s.length <= 100
s consist of lowercase English letters and '?'.

1577. Number of Ways Where Square of Number Is Equal to Product of Two Numbers

Given two arrays of integers nums1 and nums2, return the number of triplets formed (type 1 and type 2) under the fo llowing rules:

Type 1: Triplet (i, j, k) if nums1[i]2 == nums2[j] * nums2[k] where 0 <= i < nums1.length and 0 <= j < k < nums2.le ngth.
Type 2: Triplet (i, j, k) if nums2[i]2 == nums1[j] * nums1[k] where 0 <= i < nums2.length and 0 <= j < k < nums1.le ngth.

Example 1:

Input: nums1 = [7,4], nums2 = [5,2,8,9]
Output: 1
Explanation: Type 1: (1,1,2), nums1[1]^2 = nums2[1] * nums2[2]. (4^2 = 2 * 8).

Example 2:

Input: nums1 = [1,1], nums2 = [1,1,1]
Output: 9
Explanation: All Triplets are valid, because 1^2 = 1 * 1.
Type 1: (0,0,1), (0,0,2), (0,1,2), (1,0,1), (1,0,2), (1,1,2). nums1[i]^2 = nums2[j] * nums2[k].
Type 2: (0,0,1), (1,0,1), (2,0,1). nums2[i]^2 = nums1[j] * nums1[k].

Example 3:

Input: nums1 = [7,7,8,3], nums2 = [1,2,9,7]
Output: 2
Explanation: There are 2 valid triplets.
Type 1: (3,0,2). nums1[3]^2 = nums2[0] * nums2[2].
Type 2: (3,0,1). nums2[3]^2 = nums1[0] * nums1[1].

Example 4:

Input: nums1 = [4,7,9,11,23], nums2 = [3,5,1024,12,18]
Output: 0
Explanation: There are no valid triplets.

Constraints:

1 <= nums1.length, nums2.length <= 1000
1 <= nums1[i], nums2[i] <= 10^5

1578. Minimum Deletion Cost to Avoid Repeating Letters

Given a string s and an array of integers cost where cost[i] is the cost of deleting the ith character in s.
Return the minimum cost of deletions such that there are no two identical letters next to each other.
Notice that you will delete the chosen characters at the same time, in other words, after deleting a character, the costs of deleting other characters will not change.

Example 1:

Input: s = "abaac", cost = [1,2,3,4,5]
Output: 3
Explanation: Delete the letter "a" with cost 3 to get "abac" (String without two identical letters next to each other).

Example 2:

Input: s = "abc", cost = [1,2,3]
Output: 0
Explanation: You don't need to delete any character because there are no identical letters next to each other.

Example 3:

Input: s = "aabaa", cost = [1,2,3,4,1]
Output: 2
Explanation: Delete the first and the last character, getting the string ("aba").

Constraints:

s.length == cost.length
1 <= s.length, cost.length <= 10^5
1 <= cost[i] <= 10^4
s contains only lowercase English letters.

1579. Remove Max Number of Edges to Keep Graph Fully Traversable

Alice and Bob have an undirected graph of n nodes and 3 types of edges:

Type 1: Can be traversed by Alice only.
Type 2: Can be traversed by Bob only.
Type 3: Can by traversed by both Alice and Bob.

Given an array edges where edges[i] = [typei, ui, vi] represents a bidirectional edge of type typei between nodes ui a nd vi, find the maximum number of edges you can remove so that after removing the edges, the graph can still be ful ly traversed by both Alice and Bob. The graph is fully traversed by Alice and Bob if starting from any node, they ca n reach all other nodes.
Return the maximum number of edges you can remove, or return -1 if it's impossible for the graph to be fully travers ed by Alice and Bob.

Example 1:

Input: n = 4, edges = [[3,1,2],[3,2,3],[1,1,3],[1,2,4],[1,1,2],[2,3,4]]
Output: 2
Explanation: If we remove the 2 edges [1,1,2] and [1,1,3]. The graph will still be fully traversable by Alice and Bob. Removing any additional edge will not make it so. So the maximum number of edges we can remove is 2.

Example 2:

Input: n = 4, edges = [[3,1,2],[3,2,3],[1,1,4],[2,1,4]]
Output: 0
Explanation: Notice that removing any edge will not make the graph fully traversable by Alice and Bob.

Example 3:

Input: n = 4, edges = [[3,2,3],[1,1,2],[2,3,4]]
Output: -1
Explanation: In the current graph, Alice cannot reach node 4 from the other nodes. Likewise, Bob cannot reach 1. Th
erefore it's impossible to make the graph fully traversable.

Constraints:

1 <= n <= 10^5
1 <= edges.length <= min(10^5, 3 * n * (n-1) / 2)
edges[i].length == 3
1 <= edges[i][0] <= 3
1 <= edges[i][1] < edges[i][2] <= n
All tuples (typei, ui, vi) are distinct.

1582. Special Positions in a Binary Matrix

Given a rows x cols matrix mat, where mat[i][j] is either 0 or 1, return the number of special positions in mat.
A position (i,j) is called special if mat[i][j] == 1 and all other elements in row i and column j are 0 (rows and column
s are 0-indexed).

Example 1:

Input: mat = [[1,0,0],
[0,0,1],
[1,0,0]]
Output: 1
Explanation: (1,2) is a special position because mat[1][2] == 1 and all other elements in row 1 and column 2 are 0.

Example 2:

Input: mat = [[1,0,0],
[0,1,0],
[0,0,1]]
Output: 3
Explanation: (0,0), (1,1) and (2,2) are special positions.

Example 3:

Input: mat = [[0,0,0,1],
[1,0,0,0],
[0,1,1,0],
[0,0,0,0]]
Output: 2

Example 4:

Input: mat = [[0,0,0,0,0],
[1,0,0,0,0],
[0,1,0,0,0],
[0,0,1,0,0],
[0,0,0,1,1]]
Output: 3

Constraints:

rows == mat.length
cols == mat[i].length
1 <= rows, cols <= 100
mat[i][j] is 0 or 1.

1583. Count Unhappy Friends

You are given a list of preferences for n friends, where n is always even.
For each person i, preferences[i] contains a list of friends sorted in the order of preference. In other words, a friend e arlier in the list is more preferred than a friend later in the list. Friends in each list are denoted by integers from 0 to n -1.
All the friends are divided into pairs. The pairings are given in a list pairs, where pairs[i] = [xi, yi] denotes xi is paire d with yi and yi is paired with xi.
However, this pairing may cause some of the friends to be unhappy. A friend x is unhappy if x is paired with y and t here exists a friend u who is paired with v but:

x prefers u over y, and
u prefers x over v.

Return the number of unhappy friends.

Example 1:

Input: n = 4, preferences = [[1, 2, 3], [3, 2, 0], [3, 1, 0], [1, 2, 0]], pairs = [[0, 1], [2, 3]]
Output: 2
Explanation:
Friend 1 is unhappy because:
- 1 is paired with 0 but prefers 3 over 0, and
- 3 prefers 1 over 2.
Friend 3 is unhappy because:
- 3 is paired with 2 but prefers 1 over 2, and
- 1 prefers 3 over 0.
Friends 0 and 2 are happy.

Example 2:

Input: n = 2, preferences = [[1], [0]], pairs = [[1, 0]]
Output: 0
Explanation: Both friends 0 and 1 are happy.

Example 3:

Input: n = 4, preferences = [[1, 3, 2], [2, 3, 0], [1, 3, 0], [0, 2, 1]], pairs = [[1, 3], [0, 2]]
Output: 4

Constraints:

2 <= n <= 500
n is even.
preferences.length == n
preferences[i].length == n - 1
0 <= preferences[i][j] <= n - 1
preferences[i] does not contain i.
All values in preferences[i] are unique.
pairs.length == n/2
pairs[i].length == 2
xi != yi
0 <= xi, yi <= n - 1
Each person is contained in exactly one pair.

1584. Min Cost to Connect All Points

You are given an array points representing integer coordinates of some points on a 2D-plane, where points[i] = [xi, y i].
The cost of connecting two points [xi, yi] and [xj, yj] is the manhattan distance between them: |xi - xj| + |yi - yj|, whe re |val| denotes the absolute value of val.
Return the minimum cost to make all points connected. All points are connected if there is exactly one simple path b etween any two points.

Example 1:

Input: points = [[0,0],[2,2],[3,10],[5,2],[7,0]]
Output: 20
Explanation:

We can connect the points as shown above to get the minimum cost of 20.
Notice that there is a unique path between every pair of points.

Example 2:

Input: points = [[3,12],[-2,5],[-4,1]]
Output: 18

Example 3:

Input: points = [[0,0],[1,1],[1,0],[-1,1]]
Output: 4

Example 4:

Input: points = [[-1000000,-1000000],[1000000,1000000]]
Output: 4000000

Example 5:

Input: points = [[0,0]]
Output: 0

Constraints:

1 <= points.length <= 1000
-106 <= xi, yi <= 106
All pairs (xi, yi) are distinct.

1585. Check If String Is Transformable With Substring Sort Operations

Given two strings s and t, you want to transform string s into string t using the following operation any number of ti mes:

Choose a non-empty substring in s and sort it in-place so the characters are in ascending order.

For example, applying the operation on the underlined substring in "14234" results in "12344".
Return true if it is possible to transform string s into string t. Otherwise, return false.
A substring is a contiguous sequence of characters within a string.

Example 1:

Input: s = "84532", t = "34852"
Output: true
Explanation: You can transform s into t using the following sort operations:
"84532" (from index 2 to 3) -> "84352"
"84352" (from index 0 to 2) -> "34852"

Example 2:

Input: s = "34521", t = "23415"
Output: true
Explanation: You can transform s into t using the following sort operations:
"34521" -> "23451"
"23451" -> "23415"

Example 3:

Input: s = "12345", t = "12435"
Output: false

Example 4:

Input: s = "1", t = "2"
Output: false

Constraints:

s.length == t.length
1 <= s.length <= 105
s and t only contain digits from '0' to '9'.

1588. Sum of All Odd Length Subarrays

Given an array of positive integers arr, calculate the sum of all possible odd-length subarrays.
A subarray is a contiguous subsequence of the array.
Return the sum of all odd-length subarrays of arr.

Example 1:

Input: arr = [1,4,2,5,3]
Output: 58
Explanation: The odd-length subarrays of arr and their sums are:
[1] = 1
[4] = 4
[2] = 2
[5] = 5
[3] = 3
[1,4,2] = 7
[4,2,5] = 11
[2,5,3] = 10
[1,4,2,5,3] = 15

If we add all these together we get 1 + 4 + 2 + 5 + 3 + 7 + 11 + 10 + 15 = 58
Example 2:

Input: arr = [1,2]
Output: 3
Explanation: There are only 2 subarrays of odd length, [1] and [2]. Their sum is 3.
Example 3:

Input: arr = [10,11,12]
Output: 66

Constraints:

1 <= arr.length <= 100
1 <= arr[i] <= 1000

1589. Maximum Sum Obtained of Any Permutation

We have an array of integers, nums, and an array of requests where requests[i] = [starti, endi]. The ith request asks f or the sum of nums[starti] + nums[starti + 1] + ... + nums[endi - 1] + nums[endi]. Both starti and endi are 0-indexed.
Return the maximum total sum of all requests among all permutations of nums.
Since the answer may be too large, return it modulo 109 + 7.

Example 1:

Input: nums = [1,2,3,4,5], requests = [[1,3],[0,1]]
Output: 19
Explanation: One permutation of nums is [2,1,3,4,5] with the following result:
requests[0] -> nums[1] + nums[2] + nums[3] = 1 + 3 + 4 = 8
requests[1] -> nums[0] + nums[1] = 2 + 1 = 3
Total sum: 8 + 3 = 11.
A permutation with a higher total sum is [3,5,4,2,1] with the following result:
requests[0] -> nums[1] + nums[2] + nums[3] = 5 + 4 + 2 = 11
requests[1] -> nums[0] + nums[1] = 3 + 5 = 8
Total sum: 11 + 8 = 19, which is the best that you can do.

Example 2:

Input: nums = [1,2,3,4,5,6], requests = [[0,1]]
Output: 11
Explanation: A permutation with the max total sum is [6,5,4,3,2,1] with request sums [11].
Example 3:

Input: nums = [1,2,3,4,5,10], requests = [[0,2],[1,3],[1,1]]
Output: 47

Explanation: A permutation with the max total sum is [4,10,5,3,2,1] with request sums [19,18,10].

Constraints:

n == nums.length
1 <= n <= 105
0 <= nums[i] <= 105
1 <= requests.length <= 105
requests[i].length == 2
0 <= starti <= endi < n

1590. Make Sum Divisible by P

Given an array of positive integers nums, remove the smallest subarray (possibly empty) such that the sum of the re maining elements is divisible by p. It is not allowed to remove the whole array.
Return the length of the smallest subarray that you need to remove, or -1 if it's impossible.
A subarray is defined as a contiguous block of elements in the array.

Example 1:

Input: nums = [3,1,4,2], p = 6
Output: 1
Explanation: The sum of the elements in nums is 10, which is not divisible by 6. We can remove the subarray [4], an d the sum of the remaining elements is 6, which is divisible by 6.

Example 2:

Input: nums = [6,3,5,2], p = 9
Output: 2
Explanation: We cannot remove a single element to get a sum divisible by 9. The best way is to remove the subarray [5,2], leaving us with [6,3] with sum 9.

Example 3:

Input: nums = [1,2,3], p = 3
Output: 0
Explanation: Here the sum is 6. which is already divisible by 3. Thus we do not need to remove anything.

Example 4:

Input: nums = [1,2,3], p = 7
Output: -1
Explanation: There is no way to remove a subarray in order to get a sum divisible by 7.

Example 5:

Input: nums = [1000000000,1000000000,1000000000], p = 3
Output: 0

Constraints:

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

1591. Strange Printer II

There is a strange printer with the following two special requirements:

On each turn, the printer will print a solid rectangular pattern of a single color on the grid. This will cover up the exis ting colors in the rectangle.
Once the printer has used a color for the above operation, the same color cannot be used again.

You are given a m x n matrix targetGrid, where targetGrid[row][col] is the color in the position (row, col) of the grid .
Return true if it is possible to print the matrix targetGrid, otherwise, return false.

Example 1:

Input: targetGrid = [[1,1,1,1],[1,2,2,1],[1,2,2,1],[1,1,1,1]]
Output: true

Example 2:

Input: targetGrid = [[1,1,1,1],[1,1,3,3],[1,1,3,4],[5,5,1,4]]
Output: true

Example 3:

Input: targetGrid = [[1,2,1],[2,1,2],[1,2,1]]
Output: false
Explanation: It is impossible to form targetGrid because it is not allowed to print the same color in different turns.
Example 4:

Input: targetGrid = [[1,1,1],[3,1,3]]
Output: false

Constraints:

m == targetGrid.length
n == targetGrid[i].length
1 <= m, n <= 60
1 <= targetGrid[row][col] <= 60

1592. Rearrange Spaces Between Words

You are given a string text of words that are placed among some number of spaces. Each word consists of one or mo re lowercase English letters and are separated by at least one space. It's guaranteed that text contains at least one wor d.
Rearrange the spaces so that there is an equal number of spaces between every pair of adjacent words and that numb er is maximized. If you cannot redistribute all the spaces equally, place the extra spaces at the end, meaning the retur ned string should be the same length as text.
Return the string after rearranging the spaces.

Example 1:

Input: text = " this   is a sentence "
Output: "this   is   a   sentence"
Explanation: There are a total of 9 spaces and 4 words. We can evenly divide the 9 spaces between the words: 9 / (4-1) = 3 spaces.

Example 2:

Input: text = " practice   makes   perfect"
Output: "practice   makes   perfect "
Explanation: There are a total of 7 spaces and 3 words. 7 / (3-1) = 3 spaces plus 1 extra space. We place this extra sp ace at the end of the string.

Example 3:

Input: text = "hello   world"
Output: "hello   world"

Example 4:

Input: text = " walks udp package   into bar a"
Output: "walks udp package into bar a "

Example 5:

Input: text = "a"
Output: "a"

Constraints:

1 <= text.length <= 100
text consists of lowercase English letters and ' '.
text contains at least one word.

1593. Split a String Into the Max Number of Unique Substrings

Given a string s, return the maximum number of unique substrings that the given string can be split into.
You can split string s into any list of non-empty substrings, where the concatenation of the substrings forms the origi nal string. However, you must split the substrings such that all of them are unique.
A substring is a contiguous sequence of characters within a string.

Example 1:

Input: s = "ababccc"
Output: 5
Explanation: One way to split maximally is ['a', 'b', 'ab', 'c', 'cc']. Splitting like ['a', 'b', 'a', 'b', 'c', 'cc'] is not valid as yo u have 'a' and 'b' multiple times.

Example 2:

Input: s = "aba"
Output: 2
Explanation: One way to split maximally is ['a', 'ba'].

Example 3:

Input: s = "aa"
Output: 1
Explanation: It is impossible to split the string any further.

Constraints:

1 <= s.length <= 16

s contains only lower case English letters.

1594. Maximum Non Negative Product in a Matrix

You are given a rows x cols matrix grid. Initially, you are located at the top-left corner (0, 0), and in each step, you c an only move right or down in the matrix.
Among all possible paths starting from the top-left corner (0, 0) and ending in the bottom-right corner (rows - 1, cols - 1), find the path with the maximum non-negative product. The product of a path is the product of all integers in the grid cells visited along the path.
Return the maximum non-negative product modulo 109 + 7. If the maximum product is negative return -1.
Notice that the modulo is performed after getting the maximum product.

Example 1:

Input: grid = [[-1,-2,-3],
[-2,-3,-3],
[-3,-3,-2]]
Output: -1
Explanation: It's not possible to get non-negative product in the path from (0, 0) to (2, 2), so return -1.

Example 2:

Input: grid = [[1,-2,1],
[1,-2,1],
[3,-4,1]]
Output: 8
Explanation: Maximum non-negative product is in bold (1 * 1 * -2 * -4 * 1 = 8).

Example 3:

Input: grid = [[1, 3],
[0,-4]]
Output: 0
Explanation: Maximum non-negative product is in bold (1 * 0 * -4 = 0).

Example 4:

Input: grid = [[1, 4,4,0],
[-2, 0,0,1],
[1,-1,1,1]]
Output: 2
Explanation: Maximum non-negative product is in bold (1 * -2 * 1 * -1 * 1 * 1 = 2).

Constraints:

1 <= rows, cols <= 15
-4 <= grid[i][j] <= 4

1595. Minimum Cost to Connect Two Groups of Points

You are given two groups of points where the first group has size1 points, the second group has size2 points, and siz
e1 >= size2.
The cost of the connection between any two points are given in an size1 x size2 matrix where cost[i][j] is the cost of connecting point i of the first group and point j of the second group. The groups are connected if each point in both g
roups is connected to one or more points in the opposite group. In other words, each point in the first group must be connected to at least one point in the second group, and each point in the second group must be connected to at least one point in the first group.
Return the minimum cost it takes to connect the two groups.

Example 1:

Input: cost = [[15, 96], [36, 2]]
Output: 17
Explanation: The optimal way of connecting the groups is:
1--A
2--B
This results in a total cost of 17.

Example 2:

Input: cost = [[1, 3, 5], [4, 1, 1], [1, 5, 3]]
Output: 4
Explanation: The optimal way of connecting the groups is:
1--A
2--B
2--C
3--A
This results in a total cost of 4.
Note that there are multiple points connected to point 2 in the first group and point A in the second group. This does not matter as there is no limit to the number of points that can be connected. We only care about the minimum total c
ost.

Example 3:

Input: cost = [[2, 5, 1], [3, 4, 7], [8, 1, 2], [6, 2, 4], [3, 8, 8]]
Output: 10

Constraints:

size1 == cost.length
size2 == cost[i].length
1 <= size1, size2 <= 12
size1 >= size2
0 <= cost[i][j] <= 100

1598. Crawler Log Folder

The Leetcode file system keeps a log each time some user performs a change folder operation.
The operations are described below:

"../" : Move to the parent folder of the current folder. (If you are already in the main folder, remain in the same folde
r).
"./" : Remain in the same folder.
"x/" : Move to the child folder named x (This folder is guaranteed to always exist).

You are given a list of strings logs where logs[i] is the operation performed by the user at the ith step.
The file system starts in the main folder, then the operations in logs are performed.
Return the minimum number of operations needed to go back to the main folder after the change folder operations.

Example 1:

Input: logs = ["d1/","d2/","../","d21/","./"]
Output: 2
Explanation: Use this change folder operation "../" 2 times and go back to the main folder.

Example 2:

Input: logs = ["d1/","d2/","./","d3/","../","d31/"]
Output: 3

Example 3:

Input: logs = ["d1/","../","../","../"]
Output: 0

Constraints:

1 <= logs.length <= 103
2 <= logs[i].length <= 10
logs[i] contains lowercase English letters, digits, '.', and '/'.
logs[i] follows the format described in the statement.
Folder names consist of lowercase English letters and digits.

1599. Maximum Profit of Operating a Centennial Wheel

You are the operator of a Centennial Wheel that has four gondolas, and each gondola has room for up to four people. You have the ability to rotate the gondolas counterclockwise, which costs you runningCost dollars.
You are given an array customers of length n where customers[i] is the number of new customers arriving just befor e the ith rotation (0-indexed). This means you must rotate the wheel i times before the customers[i] customers arrive. You cannot make customers wait if there is room in the gondola. Each customer pays boardingCost dollars when th ey board on the gondola closest to the ground and will exit once that gondola reaches the ground again.
You can stop the wheel at any time, including before serving all customers. If you decide to stop serving customers, all subsequent rotations are free in order to get all the customers down safely. Note that if there are currently more th an four customers waiting at the wheel, only four will board the gondola, and the rest will wait for the next rotation.
Return the minimum number of rotations you need to perform to maximize your profit. If there is no scenario where the profit is positive, return -1.

Example 1:

Input: customers = [8,3], boardingCost = 5, runningCost = 6
Output: 3
Explanation: The numbers written on the gondolas are the number of people currently there.
1. 8 customers arrive, 4 board and 4 wait for the next gondola, the wheel rotates. Current profit is 4 * $5 - 1 * $6 = $ 14.
2. 3 customers arrive, the 4 waiting board the wheel and the other 3 wait, the wheel rotates. Current profit is 8 * $5 - 2 * $6 = $28.
3. The final 3 customers board the gondola, the wheel rotates. Current profit is 11 * $5 - 3 * $6 = $37.
The highest profit was $37 after rotating the wheel 3 times.
Example 2:

Input: customers = [10,9,6], boardingCost = 6, runningCost = 4
Output: 7
Explanation:
1. 10 customers arrive, 4 board and 6 wait for the next gondola, the wheel rotates. Current profit is 4 * $6 - 1 * $4 = $20.
2. 9 customers arrive, 4 board and 11 wait (2 originally waiting, 9 newly waiting), the wheel rotates. Current profit is 8 * $6 - 2 * $4 = $40.
3. The final 6 customers arrive, 4 board and 13 wait, the wheel rotates. Current profit is 12 * $6 - 3 * $4 = $60.
4. 4 board and 9 wait, the wheel rotates. Current profit is 16 * $6 - 4 * $4 = $80.
5. 4 board and 5 wait, the wheel rotates. Current profit is 20 * $6 - 5 * $4 = $100.
6. 4 board and 1 waits, the wheel rotates. Current profit is 24 * $6 - 6 * $4 = $120.
7. 1 boards, the wheel rotates. Current profit is 25 * $6 - 7 * $4 = $122.
The highest profit was $122 after rotating the wheel 7 times.

Example 3:

Input: customers = [3,4,0,5,1], boardingCost = 1, runningCost = 92
Output: -1
Explanation:
1. 3 customers arrive, 3 board and 0 wait, the wheel rotates. Current profit is 3 * $1 - 1 * $92 = -$89.
2. 4 customers arrive, 4 board and 0 wait, the wheel rotates. Current profit is 7 * $1 - 2 * $92 = -$177.

3. 0 customers arrive, 0 board and 0 wait, the wheel rotates. Current profit is 7 * $1 - 3 * $92 = -$269.
4. 5 customers arrive, 4 board and 1 waits, the wheel rotates. Current profit is 11 * $1 - 4 * $92 = -$357.
5. 1 customer arrives, 2 board and 0 wait, the wheel rotates. Current profit is 13 * $1 - 5 * $92 = -$447.
The profit was never positive, so return -1.

Example 4:

Input: customers = [10,10,6,4,7], boardingCost = 3, runningCost = 8
Output: 9
Explanation:
1. 10 customers arrive, 4 board and 6 wait, the wheel rotates. Current profit is 4 * $3 - 1 * $8 = $4.
2. 10 customers arrive, 4 board and 12 wait, the wheel rotates. Current profit is 8 * $3 - 2 * $8 = $8.
3. 6 customers arrive, 4 board and 14 wait, the wheel rotates. Current profit is 12 * $3 - 3 * $8 = $12.
4. 4 customers arrive, 4 board and 14 wait, the wheel rotates. Current profit is 16 * $3 - 4 * $8 = $16.
5. 7 customers arrive, 4 board and 17 wait, the wheel rotates. Current profit is 20 * $3 - 5 * $8 = $20.
6. 4 board and 13 wait, the wheel rotates. Current profit is 24 * $3 - 6 * $8 = $24.
7. 4 board and 9 wait, the wheel rotates. Current profit is 28 * $3 - 7 * $8 = $28.
8. 4 board and 5 wait, the wheel rotates. Current profit is 32 * $3 - 8 * $8 = $32.
9. 4 board and 1 waits, the wheel rotates. Current profit is 36 * $3 - 9 * $8 = $36.
10. 1 board and 0 wait, the wheel rotates. Current profit is 37 * $3 - 10 * $8 = $31.
The highest profit was $36 after rotating the wheel 9 times.

Constraints:

n == customers.length
1 <= n <= 105
0 <= customers[i] <= 50
1 <= boardingCost, runningCost <= 100

1600. Throne Inheritance

A kingdom consists of a king, his children, his grandchildren, and so on. Every once in a while, someone in the fami ly dies or a child is born.
The kingdom has a well-defined order of inheritance that consists of the king as the first member. Let's define the rec ursive function Successor(x, curOrder), which given a person x and the inheritance order so far, returns who should be the next person after x in the order of inheritance.

```
Successor(x, curOrder):
    if x has no children or all of x's children are in curOrder:
        if x is the king return null
        else return Successor(x's parent, curOrder)
    else return x's oldest child who's not in curOrder
```

For example, assume we have a kingdom that consists of the king, his children Alice and Bob (Alice is older than Bo b), and finally Alice's son Jack.

In the beginning, curOrder will be ["king"].
Calling Successor(king, curOrder) will return Alice, so we append to curOrder to get ["king", "Alice"].
Calling Successor(Alice, curOrder) will return Jack, so we append to curOrder to get ["king", "Alice", "Jack"].
Calling Successor(Jack, curOrder) will return Bob, so we append to curOrder to get ["king", "Alice", "Jack", "Bob"].
Calling Successor(Bob, curOrder) will return null. Thus the order of inheritance will be ["king", "Alice", "Jack", "Bo
b"].

Using the above function, we can always obtain a unique order of inheritance.
Implement the ThroneInheritance class:

ThroneInheritance(string kingName) Initializes an object of the ThroneInheritance class. The name of the king is giv
en as part of the constructor.
void birth(string parentName, string childName) Indicates that parentName gave birth to childName.
void death(string name) Indicates the death of name. The death of the person doesn't affect the Successor function no
r the current inheritance order. You can treat it as just marking the person as dead.
string[] getInheritanceOrder() Returns a list representing the current order of inheritance excluding dead people.

Example 1:

Input
["ThroneInheritance", "birth", "birth", "birth", "birth", "birth", "birth", "getInheritanceOrder", "death", "getInheritanc
eOrder"]
[["king"], ["king", "andy"], ["king", "bob"], ["king", "catherine"], ["andy", "matthew"], ["bob", "alex"], ["bob", "ash
a"], [null], ["bob"], [null]]
Output
[null, null, null, null, null, null, null, ["king", "andy", "matthew", "bob", "alex", "asha", "catherine"], null, ["king", "a
ndy", "matthew", "alex", "asha", "catherine"]]

Explanation
ThroneInheritance t= new ThroneInheritance("king"); // order: king
t.birth("king", "andy"); // order: king > andy
t.birth("king", "bob"); // order: king > andy > bob
t.birth("king", "catherine"); // order: king > andy > bob > catherine
t.birth("andy", "matthew"); // order: king > andy > matthew > bob > catherine
t.birth("bob", "alex"); // order: king > andy > matthew > bob > alex > catherine
t.birth("bob", "asha"); // order: king > andy > matthew > bob > alex > asha > catherine
t.getInheritanceOrder(); // return ["king", "andy", "matthew", "bob", "alex", "asha", "catherine"]
t.death("bob"); // order: king > andy > matthew > bob > alex > asha > catherine
t.getInheritanceOrder(); // return ["king", "andy", "matthew", "alex", "asha", "catherine"]

Constraints:

1 <= kingName.length, parentName.length, childName.length, name.length <= 15
kingName, parentName, childName, and name consist of lowercase English letters only.
All arguments childName and kingName are distinct.
All name arguments of death will be passed to either the constructor or as childName to birth first.
For each call to birth(parentName, childName), it is guaranteed that parentName is alive.
At most 105 calls will be made to birth and death.
At most 10 calls will be made to getInheritanceOrder.

1601. Maximum Number of Achievable Transfer Requests

We have n buildings numbered from 0 to n - 1. Each building has a number of employees. It's transfer season, and so me employees want to change the building they reside in.
You are given an array requests where requests[i] = [fromi, toi] represents an employee's request to transfer from bui lding fromi to building toi.
All buildings are full, so a list of requests is achievable only if for each building, the net change in employee transfer s is zero. This means the number of employees leaving is equal to the number of employees moving in. For example if n = 3 and two employees are leaving building 0, one is leaving building 1, and one is leaving building 2, there sho uld be two employees moving to building 0, one employee moving to building 1, and one employee moving to buildi ng 2.
Return the maximum number of achievable requests.

Example 1:

Input: n = 5, requests = [[0,1],[1,0],[0,1],[1,2],[2,0],[3,4]]
Output: 5
Explantion: Let's see the requests:
From building 0 we have employees x and y and both want to move to building 1.
From building 1 we have employees a and b and they want to move to buildings 2 and 0 respectively.
From building 2 we have employee z and they want to move to building 0.
From building 3 we have employee c and they want to move to building 4.
From building 4 we don't have any requests.
We can achieve the requests of users x and b by swapping their places.
We can achieve the requests of users y, a and z by swapping the places in the 3 buildings.

Example 2:

Input: n = 3, requests = [[0,0],[1,2],[2,1]]
Output: 3
Explantion: Let's see the requests:
From building 0 we have employee x and they want to stay in the same building 0.
From building 1 we have employee y and they want to move to building 2.
From building 2 we have employee z and they want to move to building 1.
We can achieve all the requests.
Example 3:

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

Constraints:

1 <= n <= 20
1 <= requests.length <= 16

requests[i].length == 2
0 <= fromi, toi < n

1603. Design Parking System

Design a parking system for a parking lot. The parking lot has three kinds of parking spaces: big, medium, and small , with a fixed number of slots for each size.
Implement the ParkingSystem class:

ParkingSystem(int big, int medium, int small) Initializes object of the ParkingSystem class. The number of slots for each parking space are given as part of the constructor.
bool addCar(int carType) Checks whether there is a parking space of carType for the car that wants to get into the pa rking lot. carType can be of three kinds: big, medium, or small, which are represented by 1, 2, and 3 respectively. A car can only park in a parking space of its carType. If there is no space available, return false, else park the car in tha t size space and return true.

Example 1:

Input
["ParkingSystem", "addCar", "addCar", "addCar", "addCar"]
[[1, 1, 0], [1], [2], [3], [1]]
Output
[null, true, true, false, false]

Explanation
ParkingSystem parkingSystem = new ParkingSystem(1, 1, 0);
parkingSystem.addCar(1); // return true because there is 1 available slot for a big car
parkingSystem.addCar(2); // return true because there is 1 available slot for a medium car
parkingSystem.addCar(3); // return false because there is no available slot for a small car
parkingSystem.addCar(1); // return false because there is no available slot for a big car. It is already occupied.

Constraints:

0 <= big, medium, small <= 1000
carType is 1, 2, or 3
At most 1000 calls will be made to addCar

1604. Alert Using Same Key-Card Three or More Times in a One Hour Period

LeetCode company workers use key-cards to unlock office doors. Each time a worker uses their key-card, the securit y system saves the worker's name and the time when it was used. The system emits an alert if any worker uses the ke y-card three or more times in a one-hour period.
You are given a list of strings keyName and keyTime where [keyName[i], keyTime[i]] corresponds to a person's na me and the time when their key-card was used in a single day.
Access times are given in the 24-hour time format "HH:MM", such as "23:51" and "09:49".
Return a list of unique worker names who received an alert for frequent keycard use. Sort the names in ascending or der alphabetically.
Notice that "10:00" - "11:00" is considered to be within a one-hour period, while "22:51" - "23:52" is not considered to be within a one-hour period.

Example 1:

Input: keyName = ["daniel","daniel","daniel","luis","luis","luis","luis"], keyTime = ["10:00","10:40","11:00","09:00 ","11:00","13:00","15:00"]
Output: ["daniel"]
Explanation: "daniel" used the keycard 3 times in a one-hour period ("10:00","10:40", "11:00").

Example 2:

Input: keyName = ["alice","alice","alice","bob","bob","bob","bob"], keyTime = ["12:01","12:00","18:00","21:00","2 1:20","21:30","23:00"]
Output: ["bob"]
Explanation: "bob" used the keycard 3 times in a one-hour period ("21:00","21:20", "21:30").

Example 3:

Input: keyName = ["john","john","john"], keyTime = ["23:58","23:59","00:01"]
Output: []

Example 4:

Input: keyName = ["leslie","leslie","leslie","clare","clare","clare","clare"], keyTime = ["13:00","13:20","14:00","18: 00","18:51","19:30","19:49"]
Output: ["clare","leslie"]

Constraints:

1 <= keyName.length, keyTime.length <= 105
keyName.length == keyTime.length
keyTime[i] is in the format "HH:MM".
[keyName[i], keyTime[i]] is unique.
1 <= keyName[i].length <= 10
keyName[i] contains only lowercase English letters.

1605. Find Valid Matrix Given Row and Column Sums

You are given two arrays rowSum and colSum of non-negative integers where rowSum[i] is the sum of the elements in the ith row and colSum[j] is the sum of the elements of the jth column of a 2D matrix. In other words, you do not know the elements of the matrix, but you do know the sums of each row and column.
Find any matrix of non-negative integers of size rowSum.length x colSum.length that satisfies the rowSum and colSum requirements.
Return a 2D array representing any matrix that fulfills the requirements. It's guaranteed that at least one matrix that fulfills the requirements exists.

Example 1:

Input: rowSum = [3,8], colSum = [4,7]
Output: [[3,0],
[1,7]]
Explanation:
0th row: 3 + 0 = 3 == rowSum[0]
1st row: 1 + 7 = 8 == rowSum[1]
0th column: 3 + 1 = 4 == colSum[0]
1st column: 0 + 7 = 7 == colSum[1]
The row and column sums match, and all matrix elements are non-negative.
Another possible matrix is: [[1,2],
[3,5]]

Example 2:

Input: rowSum = [5,7,10], colSum = [8,6,8]
Output: [[0,5,0],
[6,1,0],
[2,0,8]]

Example 3:

Input: rowSum = [14,9], colSum = [6,9,8]
Output: [[0,9,5],
[6,0,3]]

Example 4:

Input: rowSum = [1,0], colSum = [1]
Output: [[1],
[0]]

Example 5:

Input: rowSum = [0], colSum = [0]
Output: [[0]]

Constraints:

1 <= rowSum.length, colSum.length <= 500
0 <= rowSum[i], colSum[i] <= 108
sum(rows) == sum(columns)

1606. Find Servers That Handled Most Number of Requests

You have k servers numbered from 0 to k-1 that are being used to handle multiple requests simultaneously. Each ser ver has infinite computational capacity but cannot handle more than one request at a time. The requests are assigned to servers according to a specific algorithm:

The ith (0-indexed) request arrives.
If all servers are busy, the request is dropped (not handled at all).
If the (i % k)th server is available, assign the request to that server.
Otherwise, assign the request to the next available server (wrapping around the list of servers and starting from 0 if n ecessary). For example, if the ith server is busy, try to assign the request to the (i+1)th server, then the (i+2)th server, and so on.

You are given a strictly increasing array arrival of positive integers, where arrival[i] represents the arrival time of the ith request, and another array load, where load[i] represents the load of the ith request (the time it takes to complete) . Your goal is to find the busiest server(s). A server is considered busiest if it handled the most number of requests su ccessfully among all the servers.
Return a list containing the IDs (0-indexed) of the busiest server(s). You may return the IDs in any order.

Example 1:

Input: k = 3, arrival = [1,2,3,4,5], load = [5,2,3,3,3]
Output: [1]
Explanation:
All of the servers start out available.
The first 3 requests are handled by the first 3 servers in order.
Request 3 comes in. Server 0 is busy, so it's assigned to the next available server, which is 1.
Request 4 comes in. It cannot be handled since all servers are busy, so it is dropped.
Servers 0 and 2 handled one request each, while server 1 handled two requests. Hence server 1 is the busiest server.

Example 2:

Input: k = 3, arrival = [1,2,3,4], load = [1,2,1,2]
Output: [0]
Explanation:
The first 3 requests are handled by first 3 servers.
Request 3 comes in. It is handled by server 0 since the server is available.
Server 0 handled two requests, while servers 1 and 2 handled one request each. Hence server 0 is the busiest server.

Example 3:

Input: k = 3, arrival = [1,2,3], load = [10,12,11]
Output: [0,1,2]
Explanation: Each server handles a single request, so they are all considered the busiest.

Example 4:

Input: k = 3, arrival = [1,2,3,4,8,9,10], load = [5,2,10,3,1,2,2]
Output: [1]

Example 5:

Input: k = 1, arrival = [1], load = [1]
Output: [0]

Constraints:

1 <= k <= 105
1 <= arrival.length, load.length <= 105
arrival.length == load.length
1 <= arrival[i], load[i] <= 109
arrival is strictly increasing.

1608. Special Array With X Elements Greater Than or Equal X

You are given an array nums of non-negative integers. nums is considered special if there exists a number x such tha t there are exactly x numbers in nums that are greater than or equal to x.
Notice that x does not have to be an element in nums.
Return x if the array is special, otherwise, return -1. It can be proven that if nums is special, the value for x is unique.

Example 1:

Input: nums = [3,5]
Output: 2
Explanation: There are 2 values (3 and 5) that are greater than or equal to 2.

Example 2:

Input: nums = [0,0]
Output: -1
Explanation: No numbers fit the criteria for x.
If x = 0, there should be 0 numbers >= x, but there are 2.
If x = 1, there should be 1 number >= x, but there are 0.
If x = 2, there should be 2 numbers >= x, but there are 0.
x cannot be greater since there are only 2 numbers in nums.

Example 3:

Input: nums = [0,4,3,0,4]
Output: 3
Explanation: There are 3 values that are greater than or equal to 3.

Example 4:

Input: nums = [3,6,7,7,0]
Output: -1

Constraints:

1 <= nums.length <= 100
0 <= nums[i] <= 1000

1609. Even Odd Tree

A binary tree is named Even-Odd if it meets the following conditions:

The root of the binary tree is at level index 0, its children are at level index 1, their children are at level index 2, etc.
For every even-indexed level, all nodes at the level have odd integer values in strictly increasing order (from left to r ight).
For every odd-indexed level, all nodes at the level have even integer values in strictly decreasing order (from left to r ight).

Given the root of a binary tree, return true if the binary tree is Even-Odd, otherwise return false.

Example 1:

Input: root = [1,10,4,3,null,7,9,12,8,6,null,null,2]
Output: true
Explanation: The node values on each level are:
Level 0: [1]
Level 1: [10,4]
Level 2: [3,7,9]
Level 3: [12,8,6,2]
Since levels 0 and 2 are all odd and increasing, and levels 1 and 3 are all even and decreasing, the tree is Even-Odd.

Example 2:

Input: root = [5,4,2,3,3,7]
Output: false

Explanation: The node values on each level are:
Level 0: [5]
Level 1: [4,2]
Level 2: [3,3,7]
Node values in the level 2 must be in strictly increasing order, so the tree is not Even-Odd.

Example 3:

Input: root = [5,9,1,3,5,7]
Output: false
Explanation: Node values in the level 1 should be even integers.

Example 4:

Input: root = [1]
Output: true

Example 5:

Input: root = [11,8,6,1,3,9,11,30,20,18,16,12,10,4,2,17]
Output: true

Constraints:

The number of nodes in the tree is in the range [1, 105].
1 <= Node.val <= 106

1610. Maximum Number of Visible Points

You are given an array points, an integer angle, and your location, where location = [posx, posy] and points[i] = [xi, yi] both denote integral coordinates on the X-Y plane.
Initially, you are facing directly east from your position. You cannot move from your position, but you can rotate. In other words, posx and posy cannot be changed. Your field of view in degrees is represented by angle, determining how wide you can see from any given view direction. Let d be the amount in degrees that you rotate counterclockwise. Then, your field of view is the inclusive range of angles [d - angle/2, d + angle/2].

Your browser does not support the video tag or this video format.

You can see some set of points if, for each point, the angle formed by the point, your position, and the immediate east direction from your position is in your field of view.
There can be multiple points at one coordinate. There may be points at your location, and you can always see these points regardless of your rotation. Points do not obstruct your vision to other points.
Return the maximum number of points you can see.

Example 1:

Input: points = [[2,1],[2,2],[3,3]], angle = 90, location = [1,1]
Output: 3
Explanation: The shaded region represents your field of view. All points can be made visible in your field of view, i ncluding [3,3] even though [2,2] is in front and in the same line of sight.

Example 2:

Input: points = [[2,1],[2,2],[3,4],[1,1]], angle = 90, location = [1,1]
Output: 4
Explanation: All points can be made visible in your field of view, including the one at your location.

Example 3:

Input: points = [[1,0],[2,1]], angle = 13, location = [1,1]
Output: 1
Explanation: You can only see one of the two points, as shown above.

Constraints:

1 <= points.length <= 105
points[i].length == 2
location.length == 2
0 <= angle < 360
0 <= posx, posy, xi, yi <= 100

1611. Minimum One Bit Operations to Make Integers Zero

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 t o 0.

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

Example 1:

Input: n = 0
Output: 0

Example 2:

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 3:

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.

Example 4:

Input: n = 9
Output: 14

Example 5:

Input: n = 333
Output: 393

Constraints:

0 <= n <= 109

1614. Maximum Nesting Depth of the Parentheses

A string is a valid parentheses string (denoted VPS) if it meets one of the following:

It is an empty string "", or a single character not equal to "(" or ")",
It can be written as AB (A concatenated with B), where A and B are VPS's, or
It can be written as (A), where A is a VPS.

We can similarly define the nesting depth depth(S) of any VPS S as follows:

depth("") = 0
depth(C) = 0, where C is a string with a single character not equal to "(" or ")".
depth(A + B) = max(depth(A), depth(B)), where A and B are VPS's.
depth("(" + A + ")") = 1 + depth(A), where A is a VPS.

For example, "", "()()", and "()(()())" are VPS's (with nesting depths 0, 1, and 2), and ")(" and "(()" are not VPS's.
Given a VPS represented as string s, return the nesting depth of s.

Example 1:

Input: s = "(1+(2*3)+((8)/4))+1"
Output: 3
Explanation: Digit 8 is inside of 3 nested parentheses in the string.

Example 2:

Input: s = "(1)+((2))+(((3)))"
Output: 3

Example 3:

Input: s = "1+(2*3)/(2-1)"
Output: 1

Example 4:

Input: s = "1"
Output: 0

Constraints:

1 <= s.length <= 100
s consists of digits 0-9 and characters '+', '-', '*', '/', '(', and ')'.
It is guaranteed that parentheses expression s is a VPS.

1615. Maximal Network Rank

There is an infrastructure of n cities with some number of roads connecting these cities. Each roads[i] = [ai, bi] indic ates that there is a bidirectional road between cities ai and bi.
The network rank of two different cities is defined as the total number of directly connected roads to either city. If a road is directly connected to both cities, it is only counted once.
The maximal network rank of the infrastructure is the maximum network rank of all pairs of different cities.
Given the integer n and the array roads, return the maximal network rank of the entire infrastructure.

Example 1:

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

Explanation: The network rank of cities 0 and 1 is 4 as there are 4 roads that are connected to either 0 or 1. The road between 0 and 1 is only counted once.

Example 2:

Input: n = 5, roads = [[0,1],[0,3],[1,2],[1,3],[2,3],[2,4]]
Output: 5
Explanation: There are 5 roads that are connected to cities 1 or 2.

Example 3:

Input: n = 8, roads = [[0,1],[1,2],[2,3],[2,4],[5,6],[5,7]]
Output: 5
Explanation: The network rank of 2 and 5 is 5. Notice that all the cities do not have to be connected.

Constraints:

2 <= n <= 100
0 <= roads.length <= n * (n - 1) / 2
roads[i].length == 2
0 <= ai, bi <= n-1
ai != bi
Each pair of cities has at most one road connecting them.

1616. Split Two Strings to Make Palindrome

You are given two strings a and b of the same length. Choose an index and split both strings at the same index, splitti ng a into two strings: aprefix and asuffix where a = aprefix + asuffix, and splitting b into two strings: bprefix and bsu ffix where b = bprefix + bsuffix. Check if aprefix + bsuffix or bprefix + asuffix forms a palindrome.
When you split a string s into sprefix and ssuffix, either ssuffix or sprefix is allowed to be empty. For example, if s = "abc", then "" + "abc", "a" + "bc", "ab" + "c" , and "abc" + "" are valid splits.
Return true if it is possible to form a palindrome string, otherwise return false.
Notice that x + y denotes the concatenation of strings x and y.

Example 1:

Input: a = "x", b = "y"
Output: true
Explaination: If either a or b are palindromes the answer is true since you can split in the following way:
aprefix = "", asuffix = "x"
bprefix = "", bsuffix = "y"
Then, aprefix + bsuffix = "" + "y" = "y", which is a palindrome.

Example 2:

Input: a = "abdef", b = "fecab"
Output: true

Example 3:

Input: a = "ulacfd", b = "jizalu"
Output: true
Explaination: Split them at index 3:
aprefix = "ula", asuffix = "cfd"
bprefix = "jiz", bsuffix = "alu"
Then, aprefix + bsuffix = "ula" + "alu" = "ulaalu", which is a palindrome.

Example 4:

Input: a = "xbdef", b = "xecab"
Output: false

Constraints:

1 <= a.length, b.length <= 105
a.length == b.length
a and b consist of lowercase English letters

1617. Count Subtrees With Max Distance Between Cities

There are n cities numbered from 1 to n. You are given an array edges of size n-1, where edges[i] = [ui, vi] represent s a bidirectional edge between cities ui and vi. There exists a unique path between each pair of cities. In other words, the cities form a tree.
A subtree is a subset of cities where every city is reachable from every other city in the subset, where the path betwe en each pair passes through only the cities from the subset. Two subtrees are different if there is a city in one subtree that is not present in the other.
For each d from 1 to n-1, find the number of subtrees in which the maximum distance between any two cities in the s ubtree is equal to d.
Return an array of size n-1 where the dth element (1-indexed) is the number of subtrees in which the maximum dista nce between any two cities is equal to d.
Notice that the distance between the two cities is the number of edges in the path between them.

Example 1:

Input: n = 4, edges = [[1,2],[2,3],[2,4]]
Output: [3,4,0]
Explanation:
The subtrees with subsets {1,2}, {2,3} and {2,4} have a max distance of 1.

The subtrees with subsets {1,2,3}, {1,2,4}, {2,3,4} and {1,2,3,4} have a max distance of 2.
No subtree has two nodes where the max distance between them is 3.

Example 2:

Input: n = 2, edges = [[1,2]]
Output: [1]

Example 3:

Input: n = 3, edges = [[1,2],[2,3]]
Output: [2,1]

Constraints:

2 <= n <= 15
edges.length == n-1
edges[i].length == 2
1 <= ui, vi <= n
All pairs (ui, vi) are distinct.

1619. Mean of Array After Removing Some Elements

Given an integer array arr, return the mean of the remaining integers after removing the smallest 5% and the largest 5% of the elements.
Answers within 10-5 of the actual answer will be considered accepted.

Example 1:

Input: arr = [1,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3]
Output: 2.00000
Explanation: After erasing the minimum and the maximum values of this array, all elements are equal to 2, so the me an is 2.

Example 2:

Input: arr = [6,2,7,5,1,2,0,3,10,2,5,0,5,5,0,8,7,6,8,0]
Output: 4.00000

Example 3:

Input: arr = [6,0,7,0,7,5,7,8,3,4,0,7,8,1,6,8,1,1,2,4,8,1,9,5,4,3,8,5,10,8,6,6,1,0,6,10,8,2,3,4]
Output: 4.77778

Example 4:

Input: arr = [9,7,8,7,7,8,4,4,6,8,8,7,6,8,8,9,2,6,0,0,1,10,8,6,3,3,5,1,10,9,0,7,10,0,10,4,1,10,6,9,3,6,0,0,2,7,0,6,7,2,9,7,7,3,0,1,6,1,10,3]
Output: 5.27778

Example 5:

Input: arr = [4,8,4,10,0,7,1,3,7,8,8,3,4,1,6,2,1,1,8,0,9,8,0,3,9,10,3,10,1,10,7,3,2,1,4,9,10,7,6,4,0,8,5,1,2,1,6,2,5,0,7,10,9,10,3,7,10,5,8,5,7,6,7,6,10,9,5,10,5,5,7,2,10,7,7,8,2,0,1,1]
Output: 5.29167

Constraints:

20 <= arr.length <= 1000
arr.length is a multiple of 20.
0 <= arr[i] <= 105

1620. Coordinate With Maximum Network Quality

You are given an array of network towers towers, where towers[i] = [xi, yi, qi] denotes the ith network tower with location (xi, yi) and quality factor qi. All the coordinates are integral coordinates on the X-Y plane, and the distance between the two coordinates is the Euclidean distance.
You are also given an integer radius where a tower is reachable if the distance is less than or equal to radius. Outside that distance, the signal becomes garbled, and the tower is not reachable.
The signal quality of the ith tower at a coordinate (x, y) is calculated with the formula □ qi / (1 + d)□ , where d is the distance between the tower and the coordinate. The network quality at a coordinate is the sum of the signal qualities from all the reachable towers.
Return the array [cx, cy] representing the integral coordinate (cx, cy) where the network quality is maximum. If there are multiple coordinates with the same network quality, return the lexicographically minimum non-negative coordinate.
Note:

A coordinate (x1, y1) is lexicographically smaller than (x2, y2) if either:

x1 < x2, or
x1 == x2 and y1 < y2.

□ val□ is the greatest integer less than or equal to val (the floor function).

Example 1:

Input: towers = [[1,2,5],[2,1,7],[3,1,9]], radius = 2

Output: [2,1]
Explanation: At coordinate (2, 1) the total quality is 13.
- Quality of 7 from (2, 1) results in □ 7 / (1 + sqrt(0)□ = □ 7□ = 7
- Quality of 5 from (1, 2) results in □ 5 / (1 + sqrt(2)□ = □ 2.07□ = 2
- Quality of 9 from (3, 1) results in □ 9 / (1 + sqrt(1)□ = □ 4.5□ = 4
No other coordinate has a higher network quality.
Example 2:

Input: towers = [[23,11,21]], radius = 9
Output: [23,11]
Explanation: Since there is only one tower, the network quality is highest right at the tower's location.

Example 3:

Input: towers = [[1,2,13],[2,1,7],[0,1,9]], radius = 2
Output: [1,2]
Explanation: Coordinate (1, 2) has the highest network quality.

Example 4:

Input: towers = [[2,1,9],[0,1,9]], radius = 2
Output: [0,1]
Explanation: Both (0, 1) and (2, 1) are optimal in terms of quality, but (0, 1) is lexicographically minimal.

Example 5:

Input: towers = [[42,0,0]], radius = 7
Output: [0,0]
Explanation: The network quality is 0 at every coordinate, even at the tower's location.
Thus, the lexicographically minimum non-negative coordinate is (0, 0).

Constraints:

1 <= towers.length <= 50
towers[i].length == 3
0 <= xi, yi, qi <= 50
1 <= radius <= 50

1621. Number of Sets of K Non-Overlapping Line Segments

Given n points on a 1-D plane, where the ith point (from 0 to n-1) is at x = i, find the number of ways we can draw e xactly k non-overlapping line segments such that each segment covers two or more points. The endpoints of each se gment must have integral coordinates. The k line segments do not have to cover all n points, and they are allowed to share endpoints.
Return the number of ways we can draw k non-overlapping line segments. Since this number can be huge, return it

modulo 109 + 7.

Example 1:

Input: n = 4, k = 2
Output: 5
Explanation:
The two line segments are shown in red and blue.
The image above shows the 5 different ways {(0,2),(2,3)}, {(0,1),(1,3)}, {(0,1),(2,3)}, {(1,2),(2,3)}, {(0,1),(1,2)}.
Example 2:

Input: n = 3, k = 1
Output: 3
Explanation: The 3 ways are {(0,1)}, {(0,2)}, {(1,2)}.

Example 3:

Input: n = 30, k = 7
Output: 796297179
Explanation: The total number of possible ways to draw 7 line segments is 3796297200. Taking this number modulo 109 + 7 gives us 796297179.

Example 4:

Input: n = 5, k = 3
Output: 7

Example 5:

Input: n = 3, k = 2
Output: 1

Constraints:

2 <= n <= 1000
1 <= k <= n-1

1622. Fancy Sequence

Write an API that generates fancy sequences using the append, addAll, and multAll operations.
Implement the Fancy class:

Fancy() Initializes the object with an empty sequence.
void append(val) Appends an integer val to the end of the sequence.
void addAll(inc) Increments all existing values in the sequence by an integer inc.

void multAll(m) Multiplies all existing values in the sequence by an integer m.
int getIndex(idx) Gets the current value at index idx (0-indexed) of the sequence modulo 109 + 7. If the index is grea ter or equal than the length of the sequence, return -1.

Example 1:

Input
["Fancy", "append", "addAll", "append", "multAll", "getIndex", "addAll", "append", "multAll", "getIndex", "getInde x", "getIndex"]
[[], [2], [3], [7], [2], [0], [3], [10], [2], [0], [1], [2]]
Output
[null, null, null, null, null, 10, null, null, null, 26, 34, 20]

Explanation
Fancy fancy = new Fancy();
fancy.append(2);   // fancy sequence: [2]
fancy.addAll(3);   // fancy sequence: [2+3] -> [5]
fancy.append(7);   // fancy sequence: [5, 7]
fancy.multAll(2); // fancy sequence: [5*2, 7*2] -> [10, 14]
fancy.getIndex(0); // return 10
fancy.addAll(3);   // fancy sequence: [10+3, 14+3] -> [13, 17]
fancy.append(10); // fancy sequence: [13, 17, 10]
fancy.multAll(2); // fancy sequence: [13*2, 17*2, 10*2] -> [26, 34, 20]
fancy.getIndex(0); // return 26
fancy.getIndex(1); // return 34
fancy.getIndex(2); // return 20

Constraints:

1 <= val, inc, m <= 100
0 <= idx <= 105
At most 105 calls total will be made to append, addAll, multAll, and getIndex.

1624. Largest Substring Between Two Equal Characters

Given a string s, return the length of the longest substring between two equal characters, excluding the two character s. If there is no such substring return -1.
A substring is a contiguous sequence of characters within a string.

Example 1:

Input: s = "aa"
Output: 0
Explanation: The optimal substring here is an empty substring between the two 'a's.

Example 2:

Input: s = "abca"
Output: 2
Explanation: The optimal substring here is "bc".

Example 3:

Input: s = "cbzxy"
Output: -1
Explanation: There are no characters that appear twice in s.

Example 4:

Input: s = "cabbac"
Output: 4
Explanation: The optimal substring here is "abba". Other non-optimal substrings include "bb" and "".

Constraints:

1 <= s.length <= 300
s contains only lowercase English letters.

1625. Lexicographically Smallest String After Applying Operations

You are given a string s of even length consisting of digits from 0 to 9, and two integers a and b.
You can apply either of the following two operations any number of times and in any order on s:

Add a to all odd indices of s (0-indexed). Digits post 9 are cycled back to 0. For example, if s = "3456" and a = 5, s b ecomes "3951".
Rotate s to the right by b positions. For example, if s = "3456" and b = 1, s becomes "6345".

Return the lexicographically smallest string you can obtain by applying the above operations any number of times on s.
A string a is lexicographically smaller than a string b (of the same length) 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. For example, "0158" is lexi cographically smaller than "0190" because the first position they differ is at the third letter, and '5' comes before '9'.

Example 1:

Input: s = "5525", a = 9, b = 2
Output: "2050"
Explanation: We can apply the following operations:
Start: "5525"
Rotate: "2555"

Add: "2454"
Add: "2353"
Rotate: "5323"
Add: "5222"
    Add: "5121"
    Rotate: "2151"
    Add: "2050"
There is no way to obtain a string that is lexicographically smaller then "2050".

Example 2:

Input: s = "74", a = 5, b = 1
Output: "24"
Explanation: We can apply the following operations:
Start: "74"
Rotate: "47"
    Add: "42"
    Rotate: "24"
There is no way to obtain a string that is lexicographically smaller then "24".

Example 3:

Input: s = "0011", a = 4, b = 2
Output: "0011"
Explanation: There are no sequence of operations that will give us a lexicographically smaller string than "0011".

Example 4:

Input: s = "43987654", a = 7, b = 3
Output: "00553311"

Constraints:

2 <= s.length <= 100
s.length is even.
s consists of digits from 0 to 9 only.
1 <= a <= 9
1 <= b <= s.length - 1

1626. Best Team With No Conflicts

You are the manager of a basketball team. For the upcoming tournament, you want to choose the team with the high est overall score. The score of the team is the sum of scores of all the players in the team.
However, the basketball team is not allowed to have conflicts. A conflict exists if a younger player has a strictly high er score than an older player. A conflict does not occur between players of the same age.
Given two lists, scores and ages, where each scores[i] and ages[i] represents the score and age of the ith player, respe

ctively, return the highest overall score of all possible basketball teams.

Example 1:

Input: scores = [1,3,5,10,15], ages = [1,2,3,4,5]
Output: 34
Explanation: You can choose all the players.

Example 2:

Input: scores = [4,5,6,5], ages = [2,1,2,1]
Output: 16
Explanation: It is best to choose the last 3 players. Notice that you are allowed to choose multiple people of the same age.

Example 3:

Input: scores = [1,2,3,5], ages = [8,9,10,1]
Output: 6
Explanation: It is best to choose the first 3 players.

Constraints:

1 <= scores.length, ages.length <= 1000
scores.length == ages.length
1 <= scores[i] <= 106
1 <= ages[i] <= 1000

1627. Graph Connectivity With Threshold

We have n cities labeled from 1 to n. Two different cities with labels x and y are directly connected by a bidirectiona l road if and only if x and y share a common divisor strictly greater than some threshold. More formally, cities with l abels x and y have a road between them if there exists an integer z such that all of the following are true:

x % z == 0,
y % z == 0, and
z > threshold.

Given the two integers, n and threshold, and an array of queries, you must determine for each queries[i] = [ai, bi] if c ities ai and bi are connected directly or indirectly. (i.e. there is some path between them).
Return an array answer, where answer.length == queries.length and answer[i] is true if for the ith query, there is a pa th between ai and bi, or answer[i] is false if there is no path.

Example 1:

Input: n = 6, threshold = 2, queries = [[1,4],[2,5],[3,6]]
Output: [false,false,true]
Explanation: The divisors for each number:
1: 1
2: 1, 2
3: 1, 3
4: 1, 2, 4
5: 1, 5
6: 1, 2, 3, 6
Using the underlined divisors above the threshold, only cities 3 and 6 share a common divisor, so they are the only ones directly connected. The result of each query:
[1,4] 1 is not connected to 4
[2,5] 2 is not connected to 5
[3,6] 3 is connected to 6 through path 3--6

Example 2:

Input: n = 6, threshold = 0, queries = [[4,5],[3,4],[3,2],[2,6],[1,3]]
Output: [true,true,true,true,true]
Explanation: The divisors for each number are the same as the previous example. However, since the threshold is 0, all divisors can be used. Since all numbers share 1 as a divisor, all cities are connected.

Example 3:

Input: n = 5, threshold = 1, queries = [[4,5],[4,5],[3,2],[2,3],[3,4]]
Output: [false,false,false,false,false]
Explanation: Only cities 2 and 4 share a common divisor 2 which is strictly greater than the threshold 1, so they are t he only ones directly connected.
Please notice that there can be multiple queries for the same pair of nodes [x, y], and that the query [x, y] is equivale nt to the query [y, x].

Constraints:

2 <= n <= 104
0 <= threshold <= n
1 <= queries.length <= 105
queries[i].length == 2
1 <= ai, bi <= cities
ai != bi

1629. Slowest Key

A newly designed keypad was tested, where a tester pressed a sequence of n keys, one at a time.
You are given a string keysPressed of length n, where keysPressed[i] was the ith key pressed in the testing sequence, and a sorted list releaseTimes, where releaseTimes[i] was the time the ith key was released. Both arrays are 0-indexed. The 0th key was pressed at the time 0, and every subsequent key was pressed at the exact time the previous key was released.
The tester wants to know the key of the keypress that had the longest duration. The ith keypress had a duration of releaseTimes[i] - releaseTimes[i - 1], and the 0th keypress had a duration of releaseTimes[0].
Note that the same key could have been pressed multiple times during the test, and these multiple presses of the same key may not have had the same duration.
Return the key of the keypress that had the longest duration. If there are multiple such keypresses, return the lexicographically largest key of the keypresses.

Example 1:

Input: releaseTimes = [9,29,49,50], keysPressed = "cbcd"
Output: "c"
Explanation: The keypresses were as follows:
Keypress for 'c' had a duration of 9 (pressed at time 0 and released at time 9).
Keypress for 'b' had a duration of 29 - 9 = 20 (pressed at time 9 right after the release of the previous character and released at time 29).
Keypress for 'c' had a duration of 49 - 29 = 20 (pressed at time 29 right after the release of the previous character and released at time 49).
Keypress for 'd' had a duration of 50 - 49 = 1 (pressed at time 49 right after the release of the previous character and released at time 50).
The longest of these was the keypress for 'b' and the second keypress for 'c', both with duration 20.
'c' is lexicographically larger than 'b', so the answer is 'c'.

Example 2:

Input: releaseTimes = [12,23,36,46,62], keysPressed = "spuda"
Output: "a"
Explanation: The keypresses were as follows:
Keypress for 's' had a duration of 12.
Keypress for 'p' had a duration of 23 - 12 = 11.
Keypress for 'u' had a duration of 36 - 23 = 13.
Keypress for 'd' had a duration of 46 - 36 = 10.
Keypress for 'a' had a duration of 62 - 46 = 16.
The longest of these was the keypress for 'a' with duration 16.

Constraints:

releaseTimes.length == n
keysPressed.length == n
2 <= n <= 1000
1 <= releaseTimes[i] <= 109
releaseTimes[i] < releaseTimes[i+1]
keysPressed contains only lowercase English letters.

1630. Arithmetic Subarrays

A sequence of numbers is called arithmetic if it consists of at least two elements, and the difference between every t
wo consecutive elements is the same. More formally, a sequence s is arithmetic if and only if s[i+1] - s[i] == s[1] - s[
0] for all valid i.
For example, these are arithmetic sequences:

1, 3, 5, 7, 9
7, 7, 7, 7
3, -1, -5, -9
The following sequence is not arithmetic:

1, 1, 2, 5, 7
You are given an array of n integers, nums, and two arrays of m integers each, l and r, representing the m range queri
es, where the ith query is the range [l[i], r[i]]. All the arrays are 0-indexed.
Return a list of boolean elements answer, where answer[i] is true if the subarray nums[l[i]], nums[l[i]+1], ... , nums[r
[i]] can be rearranged to form an arithmetic sequence, and false otherwise.

Example 1:

Input: nums = [4,6,5,9,3,7], l = [0,0,2], r = [2,3,5]
Output: [true,false,true]
Explanation:
In the 0th query, the subarray is [4,6,5]. This can be rearranged as [6,5,4], which is an arithmetic sequence.
In the 1st query, the subarray is [4,6,5,9]. This cannot be rearranged as an arithmetic sequence.
In the 2nd query, the subarray is [5,9,3,7]. This can be rearranged as [3,5,7,9], which is an arithmetic sequence.
Example 2:

Input: nums = [-12,-9,-3,-12,-6,15,20,-25,-20,-15,-10], l = [0,1,6,4,8,7], r = [4,4,9,7,9,10]
Output: [false,true,false,false,true,true]

Constraints:

n == nums.length
m == l.length
m == r.length
2 <= n <= 500
1 <= m <= 500
0 <= l[i] < r[i] < n
-105 <= nums[i] <= 105

1631. Path With Minimum Effort

You are a hiker preparing for an upcoming hike. You are given heights, a 2D array of size rows x columns, where he ights[row][col] represents the height of cell (row, col). You are situated in the top-left cell, (0, 0), and you hope to tra vel to the bottom-right cell, (rows-1, columns-1) (i.e., 0-indexed). You can move up, down, left, or right, and you wi sh to find a route that requires the minimum effort.
A route's effort is the maximum absolute difference in heights between two consecutive cells of the route.
Return the minimum effort required to travel from the top-left cell to the bottom-right cell.

Example 1:

Input: heights = [[1,2,2],[3,8,2],[5,3,5]]
Output: 2
Explanation: The route of [1,3,5,3,5] has a maximum absolute difference of 2 in consecutive cells.
This is better than the route of [1,2,2,2,5], where the maximum absolute difference is 3.

Example 2:

Input: heights = [[1,2,3],[3,8,4],[5,3,5]]
Output: 1
Explanation: The route of [1,2,3,4,5] has a maximum absolute difference of 1 in consecutive cells, which is better th an route [1,3,5,3,5].

Example 3:

Input: heights = [[1,2,1,1,1],[1,2,1,2,1],[1,2,1,2,1],[1,2,1,2,1],[1,1,1,2,1]]
Output: 0
Explanation: This route does not require any effort.

Constraints:

rows == heights.length
columns == heights[i].length
1 <= rows, columns <= 100
1 <= heights[i][j] <= 106

1632. Rank Transform of a Matrix

Given an m x n matrix, return a new matrix answer where answer[row][col] is the rank of matrix[row][col].
The rank is an integer that represents how large an element is compared to other elements. It is calculated using the f ollowing rules:

The rank is an integer starting from 1.
If two elements p and q are in the same row or column, then:

If p < q then rank(p) < rank(q)
If p == q then rank(p) == rank(q)
If p > q then rank(p) > rank(q)

The rank should be as small as possible.

It is guaranteed that answer is unique under the given rules.

Example 1:

Input: matrix = [[1,2],[3,4]]
Output: [[1,2],[2,3]]
Explanation:
The rank of matrix[0][0] is 1 because it is the smallest integer in its row and column.
The rank of matrix[0][1] is 2 because matrix[0][1] > matrix[0][0] and matrix[0][0] is rank 1.
The rank of matrix[1][0] is 2 because matrix[1][0] > matrix[0][0] and matrix[0][0] is rank 1.
The rank of matrix[1][1] is 3 because matrix[1][1] > matrix[0][1], matrix[1][1] > matrix[1][0], and both matrix[0][1] and matrix[1][0] are rank 2.

Example 2:

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

Example 3:

Input: matrix = [[20,-21,14],[-19,4,19],[22,-47,24],[-19,4,19]]
Output: [[4,2,3],[1,3,4],[5,1,6],[1,3,4]]

Example 4:

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

Constraints:

m == matrix.length
n == matrix[i].length
1 <= m, n <= 500
-109 <= matrix[row][col] <= 109

1636. Sort Array by Increasing Frequency

Given an array of integers nums, sort the array in increasing order based on the frequency of the values. If multiple v alues have the same frequency, sort them in decreasing order.
Return the sorted array.

Example 1:

Input: nums = [1,1,2,2,2,3]
Output: [3,1,1,2,2,2]
Explanation: '3' has a frequency of 1, '1' has a frequency of 2, and '2' has a frequency of 3.

Example 2:

Input: nums = [2,3,1,3,2]
Output: [1,3,3,2,2]
Explanation: '2' and '3' both have a frequency of 2, so they are sorted in decreasing order.

Example 3:

Input: nums = [-1,1,-6,4,5,-6,1,4,1]
Output: [5,-1,4,4,-6,-6,1,1,1]

Constraints:

1 <= nums.length <= 100
-100 <= nums[i] <= 100

1637. Widest Vertical Area Between Two Points Containing No Points

Given n points on a 2D plane where points[i] = [xi, yi], Return the widest vertical area between two points such that no points are inside the area.
A vertical area is an area of fixed-width extending infinitely along the y-axis (i.e., infinite height). The widest vertica l area is the one with the maximum width.
Note that points on the edge of a vertical area are not considered included in the area.

Example 1:

Input: points = [[8,7],[9,9],[7,4],[9,7]]
Output: 1
Explanation: Both the red and the blue area are optimal.

Example 2:

Input: points = [[3,1],[9,0],[1,0],[1,4],[5,3],[8,8]]
Output: 3

Constraints:

n == points.length
2 <= n <= 105
points[i].length == 2
0 <= xi, yi <= 109

1638. Count Substrings That Differ by One Character

Given two strings s and t, find the number of ways you can choose a non-empty substring of s and replace a single c haracter by a different character such that the resulting substring is a substring of t. In other words, find the number o f substrings in s that differ from some substring in t by exactly one character.
For example, the underlined substrings in "computer" and "computation" only differ by the 'e'/'a', so this is a valid w ay.
Return the number of substrings that satisfy the condition above.
A substring is a contiguous sequence of characters within a string.

Example 1:

Input: s = "aba", t = "baba"
Output: 6
Explanation: The following are the pairs of substrings from s and t that differ by exactly 1 character:
("aba", "baba")
("aba", "baba")
("aba", "baba")
("aba", "baba")
("aba", "baba")
("aba", "baba")
The underlined portions are the substrings that are chosen from s and t.

Example 2:

Input: s = "ab", t = "bb"
Output: 3
Explanation: The following are the pairs of substrings from s and t that differ by 1 character:
("ab", "bb")
("ab", "bb")
("ab", "bb")
The underlined portions are the substrings that are chosen from s and t.

Example 3:

Input: s = "a", t = "a"
Output: 0

Example 4:

Input: s = "abe", t = "bbc"
Output: 10

Constraints:

1 <= s.length, t.length <= 100
s and t consist of lowercase English letters only.

1639. Number of Ways to Form a Target String Given a Dictionary

You are given a list of strings of the same length words and a string target.
Your task is to form target using the given words under the following rules:

target should be formed from left to right.
To form the ith character (0-indexed) of target, you can choose the kth character of the jth string in words if target[i] = words[j][k].
Once you use the kth character of the jth string of words, you can no longer use the xth character of any string in words where x <= k. In other words, all characters to the left of or at index k become unusuable for every string.
Repeat the process until you form the string target.

Notice that you can use multiple characters from the same string in words provided the conditions above are met.
Return the number of ways to form target from words. Since the answer may be too large, return it modulo 109 + 7.

Example 1:

Input: words = ["acca","bbbb","caca"], target = "aba"
Output: 6
Explanation: There are 6 ways to form target.
"aba" -> index 0 ("acca"), index 1 ("bbbb"), index 3 ("caca")
"aba" -> index 0 ("acca"), index 2 ("bbbb"), index 3 ("caca")
"aba" -> index 0 ("acca"), index 1 ("bbbb"), index 3 ("acca")
"aba" -> index 0 ("acca"), index 2 ("bbbb"), index 3 ("acca")
"aba" -> index 1 ("caca"), index 2 ("bbbb"), index 3 ("acca")
"aba" -> index 1 ("caca"), index 2 ("bbbb"), index 3 ("caca")

Example 2:

Input: words = ["abba","baab"], target = "bab"
Output: 4
Explanation: There are 4 ways to form target.

"bab" -> index 0 ("baab"), index 1 ("baab"), index 2 ("abba")
"bab" -> index 0 ("baab"), index 1 ("baab"), index 3 ("baab")
"bab" -> index 0 ("baab"), index 2 ("baab"), index 3 ("baab")
"bab" -> index 1 ("abba"), index 2 ("baab"), index 3 ("baab")

Example 3:

Input: words = ["abcd"], target = "abcd"
Output: 1

Example 4:

Input: words = ["abab","baba","abba","baab"], target = "abba"
Output: 16

Constraints:

1 <= words.length <= 1000
1 <= words[i].length <= 1000
All strings in words have the same length.
1 <= target.length <= 1000
words[i] and target contain only lowercase English letters.

1640. Check Array Formation Through Concatenation

You are given an array of distinct integers arr and an array of integer arrays pieces, where the integers in pieces are d istinct. Your goal is to form arr by concatenating the arrays in pieces in any order. However, you are not allowed to r eorder the integers in each array pieces[i].
Return true if it is possible to form the array arr from pieces. Otherwise, return false.

Example 1:

Input: arr = [85], pieces = [[85]]
Output: true

Example 2:

Input: arr = [15,88], pieces = [[88],[15]]
Output: true
Explanation: Concatenate [15] then [88]

Example 3:

Input: arr = [49,18,16], pieces = [[16,18,49]]
Output: false

Explanation: Even though the numbers match, we cannot reorder pieces[0].

Example 4:

Input: arr = [91,4,64,78], pieces = [[78],[4,64],[91]]
Output: true
Explanation: Concatenate [91] then [4,64] then [78]
Example 5:

Input: arr = [1,3,5,7], pieces = [[2,4,6,8]]
Output: false

Constraints:

1 <= pieces.length <= arr.length <= 100
sum(pieces[i].length) == arr.length
1 <= pieces[i].length <= arr.length
1 <= arr[i], pieces[i][j] <= 100
The integers in arr are distinct.
The integers in pieces are distinct (i.e., If we flatten pieces in a 1D array, all the integers in this array are distinct).

1641. Count Sorted Vowel Strings

Given an integer n, return the number of strings of length n that consist only of vowels (a, e, i, o, u) and are lexicographically sorted.
A string s is lexicographically sorted if for all valid i, s[i] is the same as or comes before s[i+1] in the alphabet.

Example 1:

Input: n = 1
Output: 5
Explanation: The 5 sorted strings that consist of vowels only are ["a","e","i","o","u"].

Example 2:

Input: n = 2
Output: 15
Explanation: The 15 sorted strings that consist of vowels only are
["aa","ae","ai","ao","au","ee","ei","eo","eu","ii","io","iu","oo","ou","uu"].
Note that "ea" is not a valid string since 'e' comes after 'a' in the alphabet.

Example 3:

Input: n = 33
Output: 66045

Constraints:

1 <= n <= 50

1642. Furthest Building You Can Reach

You are given an integer array heights representing the heights of buildings, some bricks, and some ladders.
You start your journey from building 0 and move to the next building by possibly using bricks or ladders.
While moving from building i to building i+1 (0-indexed),

If the current building's height is greater than or equal to the next building's height, you do not need a ladder or brick
s.
If the current building's height is less than the next building's height, you can either use one ladder or (h[i+1] - h[i]) b
ricks.

Return the furthest building index (0-indexed) you can reach if you use the given ladders and bricks optimally.

Example 1:

Input: heights = [4,2,7,6,9,14,12], bricks = 5, ladders = 1
Output: 4
Explanation: Starting at building 0, you can follow these steps:
- Go to building 1 without using ladders nor bricks since 4 >= 2.
- Go to building 2 using 5 bricks. You must use either bricks or ladders because 2 < 7.
- Go to building 3 without using ladders nor bricks since 7 >= 6.
- Go to building 4 using your only ladder. You must use either bricks or ladders because 6 < 9.
It is impossible to go beyond building 4 because you do not have any more bricks or ladders.

Example 2:

Input: heights = [4,12,2,7,3,18,20,3,19], bricks = 10, ladders = 2
Output: 7

Example 3:

Input: heights = [14,3,19,3], bricks = 17, ladders = 0
Output: 3

Constraints:

1 <= heights.length <= 105
1 <= heights[i] <= 106

0 <= bricks <= 109
0 <= ladders <= heights.length

1643. Kth Smallest Instructions

Bob is standing at cell (0, 0), and he wants to reach destination: (row, column). He can only travel right and down. You are going to help Bob by providing instructions for him to reach destination.
The instructions are represented as a string, where each character is either:

'H', meaning move horizontally (go right), or
'V', meaning move vertically (go down).

Multiple instructions will lead Bob to destination. For example, if destination is (2, 3), both "HHHVV" and "HVHVH" are valid instructions.
However, Bob is very picky. Bob has a lucky number k, and he wants the kth lexicographically smallest instructions that will lead him to destination. k is 1-indexed.
Given an integer array destination and an integer k, return the kth lexicographically smallest instructions that will take Bob to destination.

Example 1:

Input: destination = [2,3], k = 1
Output: "HHHVV"
Explanation: All the instructions that reach (2, 3) in lexicographic order are as follows:
["HHHVV", "HHVHV", "HHVVH", "HVHHV", "HVHVH", "HVVHH", "VHHHV", "VHHVH", "VHVHH", "VVHHH"].

Example 2:

Input: destination = [2,3], k = 2
Output: "HHVHV"

Example 3:

Input: destination = [2,3], k = 3
Output: "HHVVH"

Constraints:

destination.length == 2
1 <= row, column <= 15
1 <= k <= nCr(row + column, row), where nCr(a, b) denotes a choose b .

1646. Get Maximum in Generated Array

You are given an integer n. A 0-indexed integer array nums of length n + 1 is generated in the following way:

nums[0] = 0
nums[1] = 1
nums[2 * i] = nums[i] when 2 <= 2 * i <= n
nums[2 * i + 1] = nums[i] + nums[i + 1] when 2 <= 2 * i + 1 <= n

Return the maximum integer in the array nums .

Example 1:

Input: n = 7
Output: 3
Explanation: According to the given rules:
nums[0] = 0
nums[1] = 1
nums[(1 * 2) = 2] = nums[1] = 1
nums[(1 * 2) + 1 = 3] = nums[1] + nums[2] = 1 + 1 = 2
nums[(2 * 2) = 4] = nums[2] = 1
nums[(2 * 2) + 1 = 5] = nums[2] + nums[3] = 1 + 2 = 3
nums[(3 * 2) = 6] = nums[3] = 2
nums[(3 * 2) + 1 = 7] = nums[3] + nums[4] = 2 + 1 = 3
Hence, nums = [0,1,1,2,1,3,2,3], and the maximum is max(0,1,1,2,1,3,2,3) = 3.

Example 2:

Input: n = 2
Output: 1
Explanation: According to the given rules, nums = [0,1,1]. The maximum is max(0,1,1) = 1.

Example 3:

Input: n = 3
Output: 2
Explanation: According to the given rules, nums = [0,1,1,2]. The maximum is max(0,1,1,2) = 2.

Constraints:

0 <= n <= 100

1647. Minimum Deletions to Make Character Frequencies Unique

A string s is called good if there are no two different characters in s that have the same frequency.
Given a string s, return the minimum number of characters you need to delete to make s good.
The frequency of a character in a string is the number of times it appears in the string. For example, in the string "aa b", the frequency of 'a' is 2, while the frequency of 'b' is 1.

Example 1:

Input: s = "aab"
Output: 0
Explanation: s is already good.

Example 2:

Input: s = "aaabbbcc"
Output: 2
Explanation: You can delete two 'b's resulting in the good string "aaabcc".
Another way it to delete one 'b' and one 'c' resulting in the good string "aaabbc".
Example 3:

Input: s = "ceabaacb"
Output: 2
Explanation: You can delete both 'c's resulting in the good string "eabaab".
Note that we only care about characters that are still in the string at the end (i.e. frequency of 0 is ignored).

Constraints:

1 <= s.length <= 105
s contains only lowercase English letters.

1648. Sell Diminishing-Valued Colored Balls

You have an inventory of different colored balls, and there is a customer that wants orders balls of any color.
The customer weirdly values the colored balls. Each colored ball's value is the number of balls of that color you curr ently have in your inventory. For example, if you own 6 yellow balls, the customer would pay 6 for the first yellow b all. After the transaction, there are only 5 yellow balls left, so the next yellow ball is then valued at 5 (i.e., the value of the balls decreases as you sell more to the customer).
You are given an integer array, inventory, where inventory[i] represents the number of balls of the ith color that you

initially own. You are also given an integer orders, which represents the total number of balls that the customer want s. You can sell the balls in any order.
Return the maximum total value that you can attain after selling orders colored balls. As the answer may be too large , return it modulo 109 + 7.

Example 1:

Input: inventory = [2,5], orders = 4
Output: 14
Explanation: Sell the 1st color 1 time (2) and the 2nd color 3 times (5 + 4 + 3).
The maximum total value is 2 + 5 + 4 + 3 = 14.

Example 2:

Input: inventory = [3,5], orders = 6
Output: 19
Explanation: Sell the 1st color 2 times (3 + 2) and the 2nd color 4 times (5 + 4 + 3 + 2).
The maximum total value is 3 + 2 + 5 + 4 + 3 + 2 = 19.

Example 3:

Input: inventory = [2,8,4,10,6], orders = 20
Output: 110

Example 4:

Input: inventory = [1000000000], orders = 1000000000
Output: 21
Explanation: Sell the 1st color 1000000000 times for a total value of 500000000500000000. 500000000500000000 modulo 109 + 7 = 21.

Constraints:

1 <= inventory.length <= 105
1 <= inventory[i] <= 109
1 <= orders <= min(sum(inventory[i]), 109)

1649. Create Sorted Array through Instructions

Given an integer array instructions, you are asked to create a sorted array from the elements in instructions. You start with an empty container nums. For each element from left to right in instructions, insert it into nums. The cost of ea ch insertion is the minimum of the following:

The number of elements currently in nums that are strictly less than instructions[i].

The number of elements currently in nums that are strictly greater than instructions[i].

For example, if inserting element 3 into nums = [1,2,3,5], the cost of insertion is min(2, 1) (elements 1 and 2 are less than 3, element 5 is greater than 3) and nums will become [1,2,3,3,5].
Return the total cost to insert all elements from instructions into nums. Since the answer may be large, return it modu lo 109 + 7

Example 1:

Input: instructions = [1,5,6,2]
Output: 1
Explanation: Begin with nums = [].
Insert 1 with cost min(0, 0) = 0, now nums = [1].
Insert 5 with cost min(1, 0) = 0, now nums = [1,5].
Insert 6 with cost min(2, 0) = 0, now nums = [1,5,6].
Insert 2 with cost min(1, 2) = 1, now nums = [1,2,5,6].
The total cost is 0 + 0 + 0 + 1 = 1.
Example 2:

Input: instructions = [1,2,3,6,5,4]
Output: 3
Explanation: Begin with nums = [].
Insert 1 with cost min(0, 0) = 0, now nums = [1].
Insert 2 with cost min(1, 0) = 0, now nums = [1,2].
Insert 3 with cost min(2, 0) = 0, now nums = [1,2,3].
Insert 6 with cost min(3, 0) = 0, now nums = [1,2,3,6].
Insert 5 with cost min(3, 1) = 1, now nums = [1,2,3,5,6].
Insert 4 with cost min(3, 2) = 2, now nums = [1,2,3,4,5,6].
The total cost is 0 + 0 + 0 + 0 + 1 + 2 = 3.

Example 3:

Input: instructions = [1,3,3,3,2,4,2,1,2]
Output: 4
Explanation: Begin with nums = [].
Insert 1 with cost min(0, 0) = 0, now nums = [1].
Insert 3 with cost min(1, 0) = 0, now nums = [1,3].
Insert 3 with cost min(1, 0) = 0, now nums = [1,3,3].
Insert 3 with cost min(1, 0) = 0, now nums = [1,3,3,3].
Insert 2 with cost min(1, 3) = 1, now nums = [1,2,3,3,3].
Insert 4 with cost min(5, 0) = 0, now nums = [1,2,3,3,3,4].
    Insert 2 with cost min(1, 4) = 1, now nums = [1,2,2,3,3,3,4].
    Insert 1 with cost min(0, 6) = 0, now nums = [1,1,2,2,3,3,3,4].
    Insert 2 with cost min(2, 4) = 2, now nums = [1,1,2,2,2,3,3,3,4].
The total cost is 0 + 0 + 0 + 0 + 1 + 0 + 1 + 0 + 2 = 4.

Constraints:

1 <= instructions.length <= 105
1 <= instructions[i] <= 105

1652. Defuse the Bomb

You have a bomb to defuse, and your time is running out! Your informer will provide you with a circular array code of length of n and a key k.
To decrypt the code, you must replace every number. All the numbers are replaced simultaneously.

If k > 0, replace the ith number with the sum of the next k numbers.
If k < 0, replace the ith number with the sum of the previous k numbers.
If k == 0, replace the ith number with 0.

As code is circular, the next element of code[n-1] is code[0], and the previous element of code[0] is code[n-1].
Given the circular array code and an integer key k, return the decrypted code to defuse the bomb!

Example 1:

Input: code = [5,7,1,4], k = 3
Output: [12,10,16,13]
Explanation: Each number is replaced by the sum of the next 3 numbers. The decrypted code is [7+1+4, 1+4+5, 4+5+7, 5+7+1]. Notice that the numbers wrap around.

Example 2:

Input: code = [1,2,3,4], k = 0
Output: [0,0,0,0]
Explanation: When k is zero, the numbers are replaced by 0.

Example 3:

Input: code = [2,4,9,3], k = -2
Output: [12,5,6,13]
Explanation: The decrypted code is [3+9, 2+3, 4+2, 9+4]. Notice that the numbers wrap around again. If k is negative, the sum is of the previous numbers.

Constraints:

n == code.length
1 <= n <= 100
1 <= code[i] <= 100
-(n - 1) <= k <= n - 1

1653. Minimum Deletions to Make String Balanced

You are given a string s consisting only of characters 'a' and 'b' .
You can delete any number of characters in s to make s balanced. s is balanced if there is no pair of indices (i,j) such that i < j and s[i] = 'b' and s[j]= 'a'.
Return the minimum number of deletions needed to make s balanced.

Example 1:

Input: s = "aababbab"
Output: 2
Explanation: You can either:
Delete the characters at 0-indexed positions 2 and 6 ("aababbab" -> "aaabbb"), or
Delete the characters at 0-indexed positions 3 and 6 ("aababbab" -> "aabbbb").

Example 2:

Input: s = "bbaaaaabb"
Output: 2
Explanation: The only solution is to delete the first two characters.

Constraints:

1 <= s.length <= 105
s[i] is 'a' or 'b' .

1654. Minimum Jumps to Reach Home

A certain bug's home is on the x-axis at position x. Help them get there from position 0.
The bug jumps according to the following rules:

It can jump exactly a positions forward (to the right).
It can jump exactly b positions backward (to the left).
It cannot jump backward twice in a row.
It cannot jump to any forbidden positions.

The bug may jump forward beyond its home, but it cannot jump to positions numbered with negative integers.
Given an array of integers forbidden, where forbidden[i] means that the bug cannot jump to the position forbidden[i] , and integers a, b, and x, return the minimum number of jumps needed for the bug to reach its home. If there is no p ossible sequence of jumps that lands the bug on position x, return -1.

Example 1:

Input: forbidden = [14,4,18,1,15], a = 3, b = 15, x = 9
Output: 3

Explanation: 3 jumps forward (0 -> 3 -> 6 -> 9) will get the bug home.

Example 2:

Input: forbidden = [8,3,16,6,12,20], a = 15, b = 13, x = 11
Output: -1

Example 3:

Input: forbidden = [1,6,2,14,5,17,4], a = 16, b = 9, x = 7
Output: 2
Explanation: One jump forward (0 -> 16) then one jump backward (16 -> 7) will get the bug home.

Constraints:

1 <= forbidden.length <= 1000
1 <= a, b, forbidden[i] <= 2000
0 <= x <= 2000
All the elements in forbidden are distinct.
Position x is not forbidden.

1655. Distribute Repeating Integers

You are given an array of n integers, nums, where there are at most 50 unique values in the array. You are also given an array of m customer order quantities, quantity, where quantity[i] is the amount of integers the ith customer ordered. Determine if it is possible to distribute nums such that:

The ith customer gets exactly quantity[i] integers,
The integers the ith customer gets are all equal, and
Every customer is satisfied.

Return true if it is possible to distribute nums according to the above conditions.

Example 1:

Input: nums = [1,2,3,4], quantity = [2]
Output: false
Explanation: The 0th customer cannot be given two different integers.

Example 2:

Input: nums = [1,2,3,3], quantity = [2]
Output: true
Explanation: The 0th customer is given [3,3]. The integers [1,2] are not used.

Example 3:

Input: nums = [1,1,2,2], quantity = [2,2]
Output: true
Explanation: The 0th customer is given [1,1], and the 1st customer is given [2,2].

Example 4:

Input: nums = [1,1,2,3], quantity = [2,2]
Output: false
Explanation: Although the 0th customer could be given [1,1], the 1st customer cannot be satisfied.
Example 5:

Input: nums = [1,1,1,1,1], quantity = [2,3]
Output: true
Explanation: The 0th customer is given [1,1], and the 1st customer is given [1,1,1].

Constraints:

n == nums.length
1 <= n <= 105
1 <= nums[i] <= 1000
m == quantity.length
1 <= m <= 10
1 <= quantity[i] <= 105
There are at most 50 unique values in nums.

1656. Design an Ordered Stream

There is a stream of n (idKey, value) pairs arriving in an arbitrary order, where idKey is an integer between 1 and n a nd value is a string. No two pairs have the same id.
Design a stream that returns the values in increasing order of their IDs by returning a chunk (list) of values after each insertion. The concatenation of all the chunks should result in a list of the sorted values.
Implement the OrderedStream class:

OrderedStream(int n) Constructs the stream to take n values.
String[] insert(int idKey, String value) Inserts the pair (idKey, value) into the stream, then returns the largest possibl e chunk of currently inserted values that appear next in the order.

Example:

Input
["OrderedStream", "insert", "insert", "insert", "insert", "insert"]

[[5], [3, "ccccc"], [1, "aaaaa"], [2, "bbbbb"], [5, "eeeee"], [4, "ddddd"]]
Output
[null, [], ["aaaaa"], ["bbbbb", "ccccc"], [], ["ddddd", "eeeee"]]

Explanation
// Note that the values ordered by ID is ["aaaaa", "bbbbb", "ccccc", "ddddd", "eeeee"].
OrderedStream os = new OrderedStream(5);
os.insert(3, "ccccc"); // Inserts (3, "ccccc"), returns [].
os.insert(1, "aaaaa"); // Inserts (1, "aaaaa"), returns ["aaaaa"].
os.insert(2, "bbbbb"); // Inserts (2, "bbbbb"), returns ["bbbbb", "ccccc"].
os.insert(5, "eeeee"); // Inserts (5, "eeeee"), returns [].
os.insert(4, "ddddd"); // Inserts (4, "ddddd"), returns ["ddddd", "eeeee"].
// Concatentating all the chunks returned:
// [] + ["aaaaa"] + ["bbbbb", "ccccc"] + [] + ["ddddd", "eeeee"] = ["aaaaa", "bbbbb", "ccccc", "ddddd", "eeeee"]
// The resulting order is the same as the order above.

Constraints:

1 <= n <= 1000
1 <= id <= n
value.length == 5
value consists only of lowercase letters.
Each call to insert will have a unique id.
Exactly n calls will be made to insert.

1657. Determine if Two Strings Are Close

Two strings are considered close if you can attain one from the other using the following operations:

Operation 1: Swap any two existing characters.

For example, abcde -> aecdb

Operation 2: Transform every occurrence of one existing character into another existing character, and do the same with the other character.

For example, aacabb -> bbcbaa (all a's turn into b's, and all b's turn into a's)

You can use the operations on either string as many times as necessary.
Given two strings, word1 and word2, return true if word1 and word2 are close, and false otherwise.

Example 1:

Input: word1 = "abc", word2 = "bca"
Output: true
Explanation: You can attain word2 from word1 in 2 operations.
Apply Operation 1: "abc" -> "acb"
Apply Operation 1: "acb" -> "bca"

Example 2:

Input: word1 = "a", word2 = "aa"
Output: false
Explanation: It is impossible to attain word2 from word1, or vice versa, in any number of operations.

Example 3:

Input: word1 = "cabbba", word2 = "abbccc"
Output: true
Explanation: You can attain word2 from word1 in 3 operations.
Apply Operation 1: "cabbba" -> "caabbb"
Apply Operation 2: "caabbb" -> "baaccc"
Apply Operation 2: "baaccc" -> "abbccc"

Example 4:

Input: word1 = "cabbba", word2 = "aabbss"
Output: false
Explanation: It is impossible to attain word2 from word1, or vice versa, in any amount of operations.

Constraints:

1 <= word1.length, word2.length <= 105
word1 and word2 contain only lowercase English letters.

1658. Minimum Operations to Reduce X to Zero

You are given an integer array nums and an integer x. In one operation, you can either remove the leftmost or the rig htmost element from the array nums and subtract its value from x. Note that this modifies the array for future operati ons.
Return the minimum number of operations to reduce x to exactly 0 if it is possible, otherwise, return -1.

Example 1:

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

Explanation: The optimal solution is to remove the last two elements to reduce x to zero.

Example 2:

Input: nums = [5,6,7,8,9], x = 4
Output: -1

Example 3:

Input: nums = [3,2,20,1,1,3], x = 10
Output: 5
Explanation: The optimal solution is to remove the last three elements and the first two elements (5 operations in tot al) to reduce x to zero.

Constraints:

1 <= nums.length <= 105
1 <= nums[i] <= 104
1 <= x <= 109

1659. Maximize Grid Happiness

You are given four integers, m, n, introvertsCount, and extrovertsCount. You have an m x n grid, and there are two t ypes of people: introverts and extroverts. There are introvertsCount introverts and extrovertsCount extroverts.
You should decide how many people you want to live in the grid and assign each of them one grid cell. Note that yo u do not have to have all the people living in the grid.
The happiness of each person is calculated as follows:

Introverts start with 120 happiness and lose 30 happiness for each neighbor (introvert or extrovert).
Extroverts start with 40 happiness and gain 20 happiness for each neighbor (introvert or extrovert).

Neighbors live in the directly adjacent cells north, east, south, and west of a person's cell.
The grid happiness is the sum of each person's happiness. Return the maximum possible grid happiness.

Example 1:

Input: m = 2, n = 3, introvertsCount = 1, extrovertsCount = 2
Output: 240
Explanation: Assume the grid is 1-indexed with coordinates (row, column).
We can put the introvert in cell (1,1) and put the extroverts in cells (1,3) and (2,3).
- Introvert at (1,1) happiness: 120 (starting happiness) - (0 * 30) (0 neighbors) = 120
- Extrovert at (1,3) happiness: 40 (starting happiness) + (1 * 20) (1 neighbor) = 60
- Extrovert at (2,3) happiness: 40 (starting happiness) + (1 * 20) (1 neighbor) = 60
The grid happiness is 120 + 60 + 60 = 240.

The above figure shows the grid in this example with each person's happiness. The introvert stays in the light green c ell while the extroverts live on the light purple cells.

Example 2:

Input: m = 3, n = 1, introvertsCount = 2, extrovertsCount = 1
Output: 260
Explanation: Place the two introverts in (1,1) and (3,1) and the extrovert at (2,1).
- Introvert at (1,1) happiness: 120 (starting happiness) - (1 * 30) (1 neighbor) = 90
- Extrovert at (2,1) happiness: 40 (starting happiness) + (2 * 20) (2 neighbors) = 80
- Introvert at (3,1) happiness: 120 (starting happiness) - (1 * 30) (1 neighbor) = 90
The grid happiness is 90 + 80 + 90 = 260.

Example 3:

Input: m = 2, n = 2, introvertsCount = 4, extrovertsCount = 0
Output: 240

Constraints:

1 <= m, n <= 5
0 <= introvertsCount, extrovertsCount <= min(m * n, 6)

1662. Check If Two String Arrays are Equivalent

Given two string arrays word1 and word2, return true if the two arrays represent the same string, and false otherwise .
A string is represented by an array if the array elements concatenated in order forms the string.

Example 1:

Input: word1 = ["ab", "c"], word2 = ["a", "bc"]
Output: true
Explanation:
word1 represents string "ab" + "c" -> "abc"
word2 represents string "a" + "bc" -> "abc"
The strings are the same, so return true.
Example 2:

Input: word1 = ["a", "cb"], word2 = ["ab", "c"]
Output: false

Example 3:

Input: word1 = ["abc", "d", "defg"], word2 = ["abcddefg"]

Output: true

Constraints:

1 <= word1.length, word2.length <= 103
1 <= word1[i].length, word2[i].length <= 103
1 <= sum(word1[i].length), sum(word2[i].length) <= 103
word1[i] and word2[i] consist of lowercase letters.

1663. Smallest String With A Given Numeric Value

The numeric value of a lowercase character is defined as its position (1-indexed) in the alphabet, so the numeric valu e of a is 1, the numeric value of b is 2, the numeric value of c is 3, and so on.
The numeric value of a string consisting of lowercase characters is defined as the sum of its characters' numeric valu es. For example, the numeric value of the string "abe" is equal to 1 + 2 + 5 = 8.
You are given two integers n and k. Return the lexicographically smallest string with length equal to n and numeric value equal to k.
Note that a string x is lexicographically smaller than string y if x comes before y in dictionary order, that is, either x i s a prefix of y, or if i is the first position such that x[i] != y[i], then x[i] comes before y[i] in alphabetic order.

Example 1:

Input: n = 3, k = 27
Output: "aay"
Explanation: The numeric value of the string is 1 + 1 + 25 = 27, and it is the smallest string with such a value and len gth equal to 3.

Example 2:

Input: n = 5, k = 73
Output: "aaszz"

Constraints:

1 <= n <= 105
n <= k <= 26 * n

1664. Ways to Make a Fair Array

You are given an integer array nums. You can choose exactly one index (0-indexed) and remove the element. Notice that the index of the elements may change after the removal.
For example, if nums = [6,1,7,4,1]:

Choosing to remove index 1 results in nums = [6,7,4,1].
Choosing to remove index 2 results in nums = [6,1,4,1].
Choosing to remove index 4 results in nums = [6,1,7,4].

An array is fair if the sum of the odd-indexed values equals the sum of the even-indexed values.
Return the number of indices that you could choose such that after the removal, nums is fair.

Example 1:

Input: nums = [2,1,6,4]
Output: 1
Explanation:
Remove index 0: [1,6,4] -> Even sum: 1 + 4 = 5. Odd sum: 6. Not fair.
Remove index 1: [2,6,4] -> Even sum: 2 + 4 = 6. Odd sum: 6. Fair.
Remove index 2: [2,1,4] -> Even sum: 2 + 4 = 6. Odd sum: 1. Not fair.
Remove index 3: [2,1,6] -> Even sum: 2 + 6 = 8. Odd sum: 1. Not fair.
There is 1 index that you can remove to make nums fair.

Example 2:

Input: nums = [1,1,1]
Output: 3
Explanation: You can remove any index and the remaining array is fair.

Example 3:

Input: nums = [1,2,3]
Output: 0
Explanation: You cannot make a fair array after removing any index.

Constraints:

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

1665. Minimum Initial Energy to Finish Tasks

You are given an array tasks where tasks[i] = [actuali, minimumi]:

actuali is the actual amount of energy you spend to finish the ith task.
minimumi is the minimum amount of energy you require to begin the ith task.

For example, if the task is [10, 12] and your current energy is 11, you cannot start this task. However, if your current energy is 13, you can complete this task, and your energy will be 3 after finishing it.
You can finish the tasks in any order you like.
Return the minimum initial amount of energy you will need to finish all the tasks.

Example 1:

Input: tasks = [[1,2],[2,4],[4,8]]
Output: 8
Explanation:
Starting with 8 energy, we finish the tasks in the following order:
- 3rd task. Now energy = 8 - 4 = 4.
- 2nd task. Now energy = 4 - 2 = 2.
- 1st task. Now energy = 2 - 1 = 1.

Notice that even though we have leftover energy, starting with 7 energy does not work because we cannot do the 3rd task.
Example 2:

Input: tasks = [[1,3],[2,4],[10,11],[10,12],[8,9]]
Output: 32
Explanation:
Starting with 32 energy, we finish the tasks in the following order:
- 1st task. Now energy = 32 - 1 = 31.
- 2nd task. Now energy = 31 - 2 = 29.
- 3rd task. Now energy = 29 - 10 = 19.
- 4th task. Now energy = 19 - 10 = 9.
- 5th task. Now energy = 9 - 8 = 1.

Example 3:

Input: tasks = [[1,7],[2,8],[3,9],[4,10],[5,11],[6,12]]
Output: 27
Explanation:
Starting with 27 energy, we finish the tasks in the following order:
- 5th task. Now energy = 27 - 5 = 22.
- 2nd task. Now energy = 22 - 2 = 20.
- 3rd task. Now energy = 20 - 3 = 17.
- 1st task. Now energy = 17 - 1 = 16.
- 4th task. Now energy = 16 - 4 = 12.
- 6th task. Now energy = 12 - 6 = 6.

Constraints:

1 <= tasks.length <= 105
1 <= actual i <= minimumi <= 104

1668. Maximum Repeating Substring

For a string sequence, a string word is k-repeating if word concatenated k times is a substring of sequence. The word 's maximum k-repeating value is the highest value k where word is k-repeating in sequence. If word is not a substrin g of sequence, word's maximum k-repeating value is 0.
Given strings sequence and word, return the maximum k-repeating value of word in sequence.

Example 1:

Input: sequence = "ababc", word = "ab"
Output: 2
Explanation: "abab" is a substring in "ababc".

Example 2:

Input: sequence = "ababc", word = "ba"
Output: 1
Explanation: "ba" is a substring in "ababc". "baba" is not a substring in "ababc".

Example 3:

Input: sequence = "ababc", word = "ac"
Output: 0
Explanation: "ac" is not a substring in "ababc".

Constraints:

1 <= sequence.length <= 100
1 <= word.length <= 100
sequence and word contains only lowercase English letters.

1669. Merge In Between Linked Lists

You are given two linked lists: list1 and list2 of sizes n and m respectively.
Remove list1's nodes from the ath node to the bth node, and put list2 in their place.
The blue edges and nodes in the following figure incidate the result:

Build the result list and return its head.

Example 1:

Input: list1 = [0,1,2,3,4,5], a = 3, b = 4, list2 = [1000000,1000001,1000002]
Output: [0,1,2,1000000,1000001,1000002,5]
Explanation: We remove the nodes 3 and 4 and put the entire list2 in their place. The blue edges and nodes in the ab ove figure indicate the result.

Example 2:

Input: list1 = [0,1,2,3,4,5,6], a = 2, b = 5, list2 = [1000000,1000001,1000002,1000003,1000004]
Output: [0,1,1000000,1000001,1000002,1000003,1000004,6]
Explanation: The blue edges and nodes in the above figure indicate the result.

Constraints:

3 <= list1.length <= 104
1 <= a <= b < list1.length - 1
1 <= list2.length <= 104

1670. Design Front Middle Back Queue

Design a queue that supports push and pop operations in the front, middle, and back.
Implement the FrontMiddleBack class:

FrontMiddleBack() Initializes the queue.
void pushFront(int val) Adds val to the front of the queue.
void pushMiddle(int val) Adds val to the middle of the queue.
void pushBack(int val) Adds val to the back of the queue.
int popFront() Removes the front element of the queue and returns it. If the queue is empty, return -1.
int popMiddle() Removes the middle element of the queue and returns it. If the queue is empty, return -1.
int popBack() Removes the back element of the queue and returns it. If the queue is empty, return -1.

Notice that when there are two middle position choices, the operation is performed on the frontmost middle position choice. For example:

Pushing 6 into the middle of [1, 2, 3, 4, 5] results in [1, 2, 6, 3, 4, 5].
Popping the middle from [1, 2, 3, 4, 5, 6] returns 3 and results in [1, 2, 4, 5, 6].

Example 1:

Input:
["FrontMiddleBackQueue", "pushFront", "pushBack", "pushMiddle", "pushMiddle", "popFront", "popMiddle", "pop Middle", "popBack", "popFront"]
[[], [1], [2], [3], [4], [], [], [], [], []]

Output:
[null, null, null, null, null, 1, 3, 4, 2, -1]

Explanation:
FrontMiddleBackQueue q = new FrontMiddleBackQueue();
q.pushFront(1);   // [1]
q.pushBack(2);   // [1, 2]
q.pushMiddle(3); // [1, 3, 2]
q.pushMiddle(4); // [1, 4, 3, 2]
q.popFront();   // return 1 -> [4, 3, 2]
q.popMiddle();   // return 3 -> [4, 2]
q.popMiddle();   // return 4 -> [2]
q.popBack();   // return 2 -> []
q.popFront();   // return -1 -> [] (The queue is empty)

Constraints:

1 <= val <= 109
At most 1000 calls will be made to pushFront, pushMiddle, pushBack, popFront, popMiddle, and popBack.

1671. Minimum Number of Removals to Make Mountain Array

You may recall that an array arr is a mountain array if and only if:

arr.length >= 3
There exists some index i (0-indexed) with 0 < i < arr.length - 1 such that:

arr[0] < arr[1] < ... < arr[i - 1] < arr[i]
arr[i] > arr[i + 1] > ... > arr[arr.length - 1]

Given an integer array nums , return the minimum number of elements to remove to make nums a mountain array .

Example 1:

Input: nums = [1,3,1]
Output: 0
Explanation: The array itself is a mountain array so we do not need to remove any elements.

Example 2:

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

Explanation: One solution is to remove the elements at indices 0, 1, and 5, making the array nums = [1,5,6,3,1].

Example 3:

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

Example 4:

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

Constraints:

3 <= nums.length <= 1000
1 <= nums[i] <= 109
It is guaranteed that you can make a mountain array out of nums.

1672. Richest Customer Wealth

You are given an m x n integer grid accounts where accounts[i][j] is the amount of money the i th customer has in the j th bank. Return the wealth that the richest customer has.
A customer's wealth is the amount of money they have in all their bank accounts. The richest customer is the custom er that has the maximum wealth.

Example 1:

Input: accounts = [[1,2,3],[3,2,1]]
Output: 6
Explanation:
1st customer has wealth = 1 + 2 + 3 = 6
2nd customer has wealth = 3 + 2 + 1 = 6
Both customers are considered the richest with a wealth of 6 each, so return 6.

Example 2:

Input: accounts = [[1,5],[7,3],[3,5]]
Output: 10
Explanation:
1st customer has wealth = 6
2nd customer has wealth = 10
3rd customer has wealth = 8
The 2nd customer is the richest with a wealth of 10.
Example 3:

Input: accounts = [[2,8,7],[7,1,3],[1,9,5]]
Output: 17

Constraints:

m == accounts.length
n == accounts[i].length
1 <= m, n <= 50
1 <= accounts[i][j] <= 100

1673. Find the Most Competitive Subsequence

Given an integer array nums and a positive integer k, return the most competitive subsequence of nums of size k.
An array's subsequence is a resulting sequence obtained by erasing some (possibly zero) elements from the array.
We define that a subsequence a is more competitive than a subsequence b (of the same length) if in the first position where a and b differ, subsequence a has a number less than the corresponding number in b. For example, [1,3,4] is m ore competitive than [1,3,5] because the first position they differ is at the final number, and 4 is less than 5.

Example 1:

Input: nums = [3,5,2,6], k = 2
Output: [2,6]
Explanation: Among the set of every possible subsequence: {[3,5], [3,2], [3,6], [5,2], [5,6], [2,6]}, [2,6] is the most c ompetitive.

Example 2:

Input: nums = [2,4,3,3,5,4,9,6], k = 4
Output: [2,3,3,4]

Constraints:

1 <= nums.length <= 105
0 <= nums[i] <= 109
1 <= k <= nums.length

1674. Minimum Moves to Make Array Complementary

You are given an integer array nums of even length n and an integer limit. In one move, you can replace any integer from nums with another integer between 1 and limit, inclusive.
The array nums is complementary if for all indices i (0-indexed), nums[i] + nums[n - 1 - i] equals the same number. For example, the array [1,2,3,4] is complementary because for all indices i, nums[i] + nums[n - 1 - i] = 5.
Return the minimum number of moves required to make nums complementary.

Example 1:

Input: nums = [1,2,4,3], limit = 4
Output: 1
Explanation: In 1 move, you can change nums to [1,2,2,3] (underlined elements are changed).
nums[0] + nums[3] = 1 + 3 = 4.
nums[1] + nums[2] = 2 + 2 = 4.
nums[2] + nums[1] = 2 + 2 = 4.
nums[3] + nums[0] = 3 + 1 = 4.
Therefore, nums[i] + nums[n-1-i] = 4 for every i, so nums is complementary.

Example 2:

Input: nums = [1,2,2,1], limit = 2
Output: 2
Explanation: In 2 moves, you can change nums to [2,2,2,2]. You cannot change any number to 3 since 3 > limit.

Example 3:

Input: nums = [1,2,1,2], limit = 2
Output: 0
Explanation: nums is already complementary.

Constraints:

n == nums.length
2 <= n <= 105
1 <= nums[i] <= limit <= 105
n is even.

1675. Minimize Deviation in Array

You are given an array nums of n positive integers.
You can perform two types of operations on any element of the array any number of times:

If the element is even, divide it by 2.

For example, if the array is [1,2,3,4], then you can do this operation on the last element, and the array will be [1,2,3, 2].

If the element is odd, multiply it by 2.

For example, if the array is [1,2,3,4], then you can do this operation on the first element, and the array will be [2,2,3, 4].

The deviation of the array is the maximum difference between any two elements in the array.
Return the minimum deviation the array can have after performing some number of operations.

Example 1:

Input: nums = [1,2,3,4]
Output: 1
Explanation: You can transform the array to [1,2,3,2], then to [2,2,3,2], then the deviation will be 3 - 2 = 1.

Example 2:

Input: nums = [4,1,5,20,3]
Output: 3
Explanation: You can transform the array after two operations to [4,2,5,5,3], then the deviation will be 5 - 2 = 3.

Example 3:

Input: nums = [2,10,8]
Output: 3

Constraints:

n == nums.length
2 <= n <= 105
1 <= nums[i] <= 109

1678. Goal Parser Interpretation

You own a Goal Parser that can interpret a string command. The command consists of an alphabet of "G", "()" and/o r "(al)" in some order. The Goal Parser will interpret "G" as the string "G", "()" as the string "o", and "(al)" as the stri ng "al". The interpreted strings are then concatenated in the original order.
Given the string command, return the Goal Parser's interpretation of command.

Example 1:

Input: command = "G()(al)"
Output: "Goal"
Explanation: The Goal Parser interprets the command as follows:
G -> G
() -> o
(al) -> al
The final concatenated result is "Goal".

Example 2:

Input: command = "G()()()()(al)"
Output: "Gooooal"

Example 3:

Input: command = "(al)G(al)()()G"
Output: "alGalooG"

Constraints:

1 <= command.length <= 100
command consists of "G", "()", and/or "(al)" in some order.

1679. Max Number of K-Sum Pairs

You are given an integer array nums and an integer k.
In one operation, you can pick two numbers from the array whose sum equals k and remove them from the array.
Return the maximum number of operations you can perform on the array.

Example 1:

Input: nums = [1,2,3,4], k = 5
Output: 2
Explanation: Starting with nums = [1,2,3,4]:
- Remove numbers 1 and 4, then nums = [2,3]
- Remove numbers 2 and 3, then nums = []
There are no more pairs that sum up to 5, hence a total of 2 operations.
Example 2:

Input: nums = [3,1,3,4,3], k = 6
Output: 1
Explanation: Starting with nums = [3,1,3,4,3]:
- Remove the first two 3's, then nums = [1,4,3]

There are no more pairs that sum up to 6, hence a total of 1 operation.

Constraints:

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

1680. Concatenation of Consecutive Binary Numbers

Given an integer n, return the decimal value of the binary string formed by concatenating the binary representations of 1 to n in order, modulo 109 + 7.

Example 1:

Input: n = 1
Output: 1
Explanation: "1" in binary corresponds to the decimal value 1.

Example 2:

Input: n = 3
Output: 27
Explanation: In binary, 1, 2, and 3 corresponds to "1", "10", and "11".
After concatenating them, we have "11011", which corresponds to the decimal value 27.

Example 3:

Input: n = 12
Output: 505379714
Explanation: The concatenation results in "110111001011101111000100110101010111100".
The decimal value of that is 118505380540.
After modulo 109 + 7, the result is 505379714.

Constraints:

1 <= n <= 105

1681. Minimum Incompatibility

You are given an integer array nums and an integer k. You are asked to distribute this array into k subsets of equal size such that there are no two equal elements in the same subset.
A subset's incompatibility is the difference between the maximum and minimum elements in that array.
Return the minimum possible sum of incompatibilities of the k subsets after distributing the array optimally, or retur n -1 if it is not possible.
A subset is a group integers that appear in the array with no particular order.

Example 1:

Input: nums = [1,2,1,4], k = 2
Output: 4
Explanation: The optimal distribution of subsets is [1,2] and [1,4].
The incompatibility is (2-1) + (4-1) = 4.
Note that [1,1] and [2,4] would result in a smaller sum, but the first subset contains 2 equal elements.
Example 2:

Input: nums = [6,3,8,1,3,1,2,2], k = 4
Output: 6
Explanation: The optimal distribution of subsets is [1,2], [2,3], [6,8], and [1,3].
The incompatibility is (2-1) + (3-2) + (8-6) + (3-1) = 6.

Example 3:

Input: nums = [5,3,3,6,3,3], k = 3
Output: -1
Explanation: It is impossible to distribute nums into 3 subsets where no two elements are equal in the same subset.

Constraints:

1 <= k <= nums.length <= 16
nums.length is divisible by k
1 <= nums[i] <= nums.length

1684. Count the Number of Consistent Strings

You are given a string allowed consisting of distinct characters and an array of strings words. A string is consistent if all characters in the string appear in the string allowed.
Return the number of consistent strings in the array words.

Example 1:

Input: allowed = "ab", words = ["ad","bd","aaab","baa","badab"]

Output: 2
Explanation: Strings "aaab" and "baa" are consistent since they only contain characters 'a' and 'b'.

Example 2:

Input: allowed = "abc", words = ["a","b","c","ab","ac","bc","abc"]
Output: 7
Explanation: All strings are consistent.

Example 3:

Input: allowed = "cad", words = ["cc","acd","b","ba","bac","bad","ac","d"]
Output: 4
Explanation: Strings "cc", "acd", "ac", and "d" are consistent.

Constraints:

1 <= words.length <= 104
1 <= allowed.length <= 26
1 <= words[i].length <= 10
The characters in allowed are distinct.
words[i] and allowed contain only lowercase English letters.

1685. Sum of Absolute Differences in a Sorted Array

You are given an integer array nums sorted in non-decreasing order.
Build and return an integer array result with the same length as nums such that result[i] is equal to the summation of absolute differences between nums[i] and all the other elements in the array.
In other words, result[i] is equal to sum(|nums[i]-nums[j]|) where 0 <= j < nums.length and j != i (0-indexed).

Example 1:

Input: nums = [2,3,5]
Output: [4,3,5]
Explanation: Assuming the arrays are 0-indexed, then
result[0] = |2-2| + |2-3| + |2-5| = 0 + 1 + 3 = 4,
result[1] = |3-2| + |3-3| + |3-5| = 1 + 0 + 2 = 3,
result[2] = |5-2| + |5-3| + |5-5| = 3 + 2 + 0 = 5.

Example 2:

Input: nums = [1,4,6,8,10]
Output: [24,15,13,15,21]

Constraints:

2 <= nums.length <= 105
1 <= nums[i] <= nums[i + 1] <= 104

1686. Stone Game VI

Alice and Bob take turns playing a game, with Alice starting first.
There are n stones in a pile. On each player's turn, they can remove a stone from the pile and receive points based on the stone's value. Alice and Bob may value the stones differently.
You are given two integer arrays of length n, aliceValues and bobValues. Each aliceValues[i] and bobValues[i] represents how Alice and Bob, respectively, value the ith stone.
The winner is the person with the most points after all the stones are chosen. If both players have the same amount of points, the game results in a draw. Both players will play optimally. Both players know the other's values.
Determine the result of the game, and:

If Alice wins, return 1.
If Bob wins, return -1.
If the game results in a draw, return 0.

Example 1:

Input: aliceValues = [1,3], bobValues = [2,1]
Output: 1
Explanation:
If Alice takes stone 1 (0-indexed) first, Alice will receive 3 points.
Bob can only choose stone 0, and will only receive 2 points.
Alice wins.

Example 2:

Input: aliceValues = [1,2], bobValues = [3,1]
Output: 0
Explanation:
If Alice takes stone 0, and Bob takes stone 1, they will both have 1 point.
Draw.

Example 3:

Input: aliceValues = [2,4,3], bobValues = [1,6,7]
Output: -1
Explanation:
Regardless of how Alice plays, Bob will be able to have more points than Alice.
For example, if Alice takes stone 1, Bob can take stone 2, and Alice takes stone 0, Alice will have 6 points to Bob's 7.

Bob wins.

Constraints:

n == aliceValues.length == bobValues.length
1 <= n <= 105
1 <= aliceValues[i], bobValues[i] <= 100

1687. Delivering Boxes from Storage to Ports

You have the task of delivering some boxes from storage to their ports using only one ship. However, this ship has a limit on the number of boxes and the total weight that it can carry.
You are given an array boxes, where boxes[i] = [ports i , weighti], and three integers portsCount, maxBoxes, and m axWeight.

ports i is the port where you need to deliver the ith box and weightsi is the weight of the ith box.
portsCount is the number of ports.
maxBoxes and maxWeight are the respective box and weight limits of the ship.

The boxes need to be delivered in the order they are given. The ship will follow these steps:

The ship will take some number of boxes from the boxes queue, not violating the maxBoxes and maxWeight constra ints.
For each loaded box in order, the ship will make a trip to the port the box needs to be delivered to and deliver it. If th e ship is already at the correct port, no trip is needed, and the box can immediately be delivered.
The ship then makes a return trip to storage to take more boxes from the queue.

The ship must end at storage after all the boxes have been delivered.
Return the minimum number of trips the ship needs to make to deliver all boxes to their respective ports.

Example 1:

Input: boxes = [[1,1],[2,1],[1,1]], portsCount = 2, maxBoxes = 3, maxWeight = 3
Output: 4
Explanation: The optimal strategy is as follows:
- The ship takes all the boxes in the queue, goes to port 1, then port 2, then port 1 again, then returns to storage. 4 tri ps.
So the total number of trips is 4.
Note that the first and third boxes cannot be delivered together because the boxes need to be delivered in order (i.e. t he second box needs to be delivered at port 2 before the third box).

Example 2:

Input: boxes = [[1,2],[3,3],[3,1],[3,1],[2,4]], portsCount = 3, maxBoxes = 3, maxWeight = 6
Output: 6

Explanation: The optimal strategy is as follows:
- The ship takes the first box, goes to port 1, then returns to storage. 2 trips.
- The ship takes the second, third and fourth boxes, goes to port 3, then returns to storage. 2 trips.
- The ship takes the fifth box, goes to port 3, then returns to storage. 2 trips.
So the total number of trips is 2 + 2 + 2 = 6.

Example 3:

Input: boxes = [[1,4],[1,2],[2,1],[2,1],[3,2],[3,4]], portsCount = 3, maxBoxes = 6, maxWeight = 7
Output: 6
Explanation: The optimal strategy is as follows:
- The ship takes the first and second boxes, goes to port 1, then returns to storage. 2 trips.
- The ship takes the third and fourth boxes, goes to port 2, then returns to storage. 2 trips.
- The ship takes the fifth and sixth boxes, goes to port 3, then returns to storage. 2 trips.
So the total number of trips is 2 + 2 + 2 = 6.

Example 4:

Input: boxes = [[2,4],[2,5],[3,1],[3,2],[3,7],[3,1],[4,4],[1,3],[5,2]], portsCount = 5, maxBoxes = 5, maxWeight = 7
Output: 14
Explanation: The optimal strategy is as follows:
- The ship takes the first box, goes to port 2, then storage. 2 trips.
- The ship takes the second box, goes to port 2, then storage. 2 trips.
- The ship takes the third and fourth boxes, goes to port 3, then storage. 2 trips.
- The ship takes the fifth box, goes to port 3, then storage. 2 trips.
- The ship takes the sixth and seventh boxes, goes to port 3, then port 4, then storage. 3 trips.
- The ship takes the eighth and ninth boxes, goes to port 1, then port 5, then storage. 3 trips.
So the total number of trips is 2 + 2 + 2 + 2 + 3 + 3 = 14.

Constraints:

1 <= boxes.length <= 105
1 <= portsCount, maxBoxes, maxWeight <= 105
1 <= ports i <= portsCount
1 <= weightsi <= maxWeight

1688. Count of Matches in Tournament

You are given an integer n, the number of teams in a tournament that has strange rules:

If the current number of teams is even, each team gets paired with another team. A total of n / 2 matches are played, and n / 2 teams advance to the next round.
If the current number of teams is odd, one team randomly advances in the tournament, and the rest gets paired. A tot al of (n - 1) / 2 matches are played, and (n - 1) / 2 + 1 teams advance to the next round.

Return the number of matches played in the tournament until a winner is decided.

Example 1:

Input: n = 7
Output: 6
Explanation: Details of the tournament:
- 1st Round: Teams = 7, Matches = 3, and 4 teams advance.
- 2nd Round: Teams = 4, Matches = 2, and 2 teams advance.
- 3rd Round: Teams = 2, Matches = 1, and 1 team is declared the winner.
Total number of matches = 3 + 2 + 1 = 6.

Example 2:

Input: n = 14
Output: 13
Explanation: Details of the tournament:
- 1st Round: Teams = 14, Matches = 7, and 7 teams advance.
- 2nd Round: Teams = 7, Matches = 3, and 4 teams advance.
- 3rd Round: Teams = 4, Matches = 2, and 2 teams advance.
- 4th Round: Teams = 2, Matches = 1, and 1 team is declared the winner.
Total number of matches = 7 + 3 + 2 + 1 = 13.

Constraints:

1 <= n <= 200

1689. Partitioning Into Minimum Number Of Deci-Binary Numbers

A decimal number is called deci-binary if each of its digits is either 0 or 1 without any leading zeros. For example, 1 01 and 1100 are deci-binary, while 112 and 3001 are not.
Given a string n that represents a positive decimal integer, return the minimum number of positive deci-binary numb ers needed so that they sum up to n.

Example 1:

Input: n = "32"
Output: 3
Explanation: 10 + 11 + 11 = 32

Example 2:

Input: n = "82734"
Output: 8

Example 3:

Input: n = "27346209830709182346"
Output: 9

Constraints:

1 <= n.length <= 105
n consists of only digits.
n does not contain any leading zeros and represents a positive integer.

1690. Stone Game VII

Alice and Bob take turns playing a game, with Alice starting first.
There are n stones arranged in a row. On each player's turn, they can remove either the leftmost stone or the rightmo st stone from the row and receive points equal to the sum of the remaining stones' values in the row. The winner is th e one with the higher score when there are no stones left to remove.
Bob found that he will always lose this game (poor Bob, he always loses), so he decided to minimize the score's diff erence. Alice's goal is to maximize the difference in the score.
Given an array of integers stones where stones[i] represents the value of the ith stone from the left, return the differe nce in Alice and Bob's score if they both play optimally.

Example 1:

Input: stones = [5,3,1,4,2]
Output: 6
Explanation:
- Alice removes 2 and gets 5 + 3 + 1 + 4 = 13 points. Alice = 13, Bob = 0, stones = [5,3,1,4].
- Bob removes 5 and gets 3 + 1 + 4 = 8 points. Alice = 13, Bob = 8, stones = [3,1,4].
- Alice removes 3 and gets 1 + 4 = 5 points. Alice = 18, Bob = 8, stones = [1,4].
- Bob removes 1 and gets 4 points. Alice = 18, Bob = 12, stones = [4].
- Alice removes 4 and gets 0 points. Alice = 18, Bob = 12, stones = [].
The score difference is 18 - 12 = 6.

Example 2:

Input: stones = [7,90,5,1,100,10,10,2]
Output: 122

Constraints:

n == stones.length
2 <= n <= 1000
1 <= stones[i] <= 1000

1691. Maximum Height by Stacking Cuboids

Given n cuboids where the dimensions of the ith cuboid is cuboids[i] = [widthi, lengthi, heighti] (0-indexed). Choose a subset of cuboids and place them on each other.
You can place cuboid i on cuboid j if widthi <= widthj and lengthi <= lengthj and heighti <= heightj. You can rearrange any cuboid's dimensions by rotating it to put it on another cuboid.
Return the maximum height of the stacked cuboids.

Example 1:

Input: cuboids = [[50,45,20],[95,37,53],[45,23,12]]
Output: 190
Explanation:
Cuboid 1 is placed on the bottom with the 53x37 side facing down with height 95.
Cuboid 0 is placed next with the 45x20 side facing down with height 50.
Cuboid 2 is placed next with the 23x12 side facing down with height 45.
The total height is 95 + 50 + 45 = 190.

Example 2:

Input: cuboids = [[38,25,45],[76,35,3]]
Output: 76
Explanation:
You can't place any of the cuboids on the other.
We choose cuboid 1 and rotate it so that the 35x3 side is facing down and its height is 76.

Example 3:

Input: cuboids = [[7,11,17],[7,17,11],[11,7,17],[11,17,7],[17,7,11],[17,11,7]]
Output: 102
Explanation:
After rearranging the cuboids, you can see that all cuboids have the same dimension.
You can place the 11x7 side down on all cuboids so their heights are 17.
The maximum height of stacked cuboids is 6 * 17 = 102.

Constraints:

n == cuboids.length
1 <= n <= 100
1 <= widthi, lengthi, heighti <= 100

1694. Reformat Phone Number

You are given a phone number as a string number. number consists of digits, spaces ' ', and/or dashes '-'.
You would like to reformat the phone number in a certain manner. Firstly, remove all spaces and dashes. Then, grou p the digits from left to right into blocks of length 3 until there are 4 or fewer digits. The final digits are then grouped as follows:

2 digits: A single block of length 2.
3 digits: A single block of length 3.
4 digits: Two blocks of length 2 each.

The blocks are then joined by dashes. Notice that the reformatting process should never produce any blocks of lengt h 1 and produce at most two blocks of length 2.
Return the phone number after formatting.

Example 1:

Input: number = "1-23-45 6"
Output: "123-456"
Explanation: The digits are "123456".
Step 1: There are more than 4 digits, so group the next 3 digits. The 1st block is "123".
Step 2: There are 3 digits remaining, so put them in a single block of length 3. The 2nd block is "456".
Joining the blocks gives "123-456".

Example 2:

Input: number = "123 4-567"
Output: "123-45-67"
Explanation: The digits are "1234567".
Step 1: There are more than 4 digits, so group the next 3 digits. The 1st block is "123".
Step 2: There are 4 digits left, so split them into two blocks of length 2. The blocks are "45" and "67".
Joining the blocks gives "123-45-67".

Example 3:

Input: number = "123 4-5678"
Output: "123-456-78"
Explanation: The digits are "12345678".
Step 1: The 1st block is "123".
Step 2: The 2nd block is "456".
Step 3: There are 2 digits left, so put them in a single block of length 2. The 3rd block is "78".
Joining the blocks gives "123-456-78".

Example 4:

Input: number = "12"
Output: "12"

Example 5:

Input: number = "--17-5 229 35-39475 "
Output: "175-229-353-94-75"

Constraints:

2 <= number.length <= 100
number consists of digits and the characters '-' and ' '.
There are at least two digits in number.

1695. Maximum Erasure Value

You are given an array of positive integers nums and want to erase a subarray containing unique elements. The score you get by erasing the subarray is equal to the sum of its elements.
Return the maximum score you can get by erasing exactly one subarray.
An array b is called to be a subarray of a if it forms a contiguous subsequence of a, that is, if it is equal to a[l],a[l+1],...,a[r] for some (l,r).

Example 1:

Input: nums = [4,2,4,5,6]
Output: 17
Explanation: The optimal subarray here is [2,4,5,6].

Example 2:

Input: nums = [5,2,1,2,5,2,1,2,5]
Output: 8
Explanation: The optimal subarray here is [5,2,1] or [1,2,5].

Constraints:

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

1696. Jump Game VI

You are given a 0-indexed integer array nums and an integer k.
You are initially standing at index 0. In one move, you can jump at most k steps forward without going outside the b oundaries of the array. That is, you can jump from index i to any index in the range [i + 1, min(n - 1, i + k)] inclusive .
You want to reach the last index of the array (index n - 1). Your score is the sum of all nums[j] for each index j you visited in the array.
Return the maximum score you can get.

Example 1:

Input: nums = [1,-1,-2,4,-7,3], k = 2
Output: 7
Explanation: You can choose your jumps forming the subsequence [1,-1,4,3] (underlined above). The sum is 7.

Example 2:

Input: nums = [10,-5,-2,4,0,3], k = 3
Output: 17
Explanation: You can choose your jumps forming the subsequence [10,4,3] (underlined above). The sum is 17.

Example 3:

Input: nums = [1,-5,-20,4,-1,3,-6,-3], k = 2
Output: 0

Constraints:

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

1697. Checking Existence of Edge Length Limited Paths

An undirected graph of n nodes is defined by edgeList, where edgeList[i] = [ui, vi, disi] denotes an edge between no des ui and vi with distance disi. Note that there may be multiple edges between two nodes.
Given an array queries, where queries[j] = [pj, qj, limitj], your task is to determine for each queries[j] whether there i s a path between pj and qj such that each edge on the path has a distance strictly less than limitj .
Return a boolean array answer, where answer.length == queries.length and the jth value of answer is true if there is a path for queries[j] is true, and false otherwise.

Example 1:

Input: n = 3, edgeList = [[0,1,2],[1,2,4],[2,0,8],[1,0,16]], queries = [[0,1,2],[0,2,5]]
Output: [false,true]

Explanation: The above figure shows the given graph. Note that there are two overlapping edges between 0 and 1 wi th distances 2 and 16.
For the first query, between 0 and 1 there is no path where each distance is less than 2, thus we return false for this q uery.
For the second query, there is a path (0 -> 1 -> 2) of two edges with distances less than 5, thus we return true for this query.

Example 2:

Input: n = 5, edgeList = [[0,1,10],[1,2,5],[2,3,9],[3,4,13]], queries = [[0,4,14],[1,4,13]]
Output: [true,false]
Exaplanation: The above figure shows the given graph.

Constraints:

2 <= n <= 105
1 <= edgeList.length, queries.length <= 105
edgeList[i].length == 3
queries[j].length == 3
0 <= ui, vi, pj, qj <= n - 1
ui != vi
pj != qj
1 <= disi, limitj <= 109
There may be multiple edges between two nodes.

1700. Number of Students Unable to Eat Lunch

The school cafeteria offers circular and square sandwiches at lunch break, referred to by numbers 0 and 1 respectivel y. All students stand in a queue. Each student either prefers square or circular sandwiches.
The number of sandwiches in the cafeteria is equal to the number of students. The sandwiches are placed in a stack. At each step:

If the student at the front of the queue prefers the sandwich on the top of the stack, they will take it and leave the que ue.
Otherwise, they will leave it and go to the queue's end.

This continues until none of the queue students want to take the top sandwich and are thus unable to eat.
You are given two integer arrays students and sandwiches where sandwiches[i] is the type of the i th sandwich in the stack (i = 0 is the top of the stack) and students[j] is the preference of the j th student in the initial queue (j = 0 is the front of the queue). Return the number of students that are unable to eat.

Example 1:

Input: students = [1,1,0,0], sandwiches = [0,1,0,1]

Output: 0
Explanation:
- Front student leaves the top sandwich and returns to the end of the line making students = [1,0,0,1].
- Front student leaves the top sandwich and returns to the end of the line making students = [0,0,1,1].
- Front student takes the top sandwich and leaves the line making students = [0,1,1] and sandwiches = [1,0,1].
- Front student leaves the top sandwich and returns to the end of the line making students = [1,1,0].
- Front student takes the top sandwich and leaves the line making students = [1,0] and sandwiches = [0,1].
- Front student leaves the top sandwich and returns to the end of the line making students = [0,1].
- Front student takes the top sandwich and leaves the line making students = [1] and sandwiches = [1].
- Front student takes the top sandwich and leaves the line making students = [] and sandwiches = [].
Hence all students are able to eat.

Example 2:

Input: students = [1,1,1,0,0,1], sandwiches = [1,0,0,0,1,1]
Output: 3

Constraints:

1 <= students.length, sandwiches.length <= 100
students.length == sandwiches.length
sandwiches[i] is 0 or 1.
students[i] is 0 or 1.

1701. Average Waiting Time

There is a restaurant with a single chef. You are given an array customers, where customers[i] = [arrivali, timei]:

arrivali is the arrival time of the ith customer. The arrival times are sorted in non-decreasing order.
timei is the time needed to prepare the order of the ith customer.

When a customer arrives, he gives the chef his order, and the chef starts preparing it once he is idle. The customer w aits till the chef finishes preparing his order. The chef does not prepare food for more than one customer at a time. T he chef prepares food for customers in the order they were given in the input.
Return the average waiting time of all customers. Solutions within 10-5 from the actual answer are considered accept ed.

Example 1:

Input: customers = [[1,2],[2,5],[4,3]]
Output: 5.00000
Explanation:
1) The first customer arrives at time 1, the chef takes his order and starts preparing it immediately at time 1, and finis hes at time 3, so the waiting time of the first customer is 3 - 1 = 2.
2) The second customer arrives at time 2, the chef takes his order and starts preparing it at time 3, and finishes at tim

e 8, so the waiting time of the second customer is 8 - 2 = 6.
3) The third customer arrives at time 4, the chef takes his order and starts preparing it at time 8, and finishes at time 11, so the waiting time of the third customer is 11 - 4 = 7.
So the average waiting time = (2 + 6 + 7) / 3 = 5.

Example 2:

Input: customers = [[5,2],[5,4],[10,3],[20,1]]
Output: 3.25000
Explanation:
1) The first customer arrives at time 5, the chef takes his order and starts preparing it immediately at time 5, and finis hes at time 7, so the waiting time of the first customer is 7 - 5 = 2.
2) The second customer arrives at time 5, the chef takes his order and starts preparing it at time 7, and finishes at tim e 11, so the waiting time of the second customer is 11 - 5 = 6.
3) The third customer arrives at time 10, the chef takes his order and starts preparing it at time 11, and finishes at tim e 14, so the waiting time of the third customer is 14 - 10 = 4.
4) The fourth customer arrives at time 20, the chef takes his order and starts preparing it immediately at time 20, and finishes at time 21, so the waiting time of the fourth customer is 21 - 20 = 1.
So the average waiting time = (2 + 6 + 4 + 1) / 4 = 3.25.

Constraints:

1 <= customers.length <= 105
1 <= arrivali, timei <= 104
arrivali <= arrivali+1

1702. Maximum Binary String After Change

You are given a binary string binary consisting of only 0's or 1's. You can apply each of the following operations any number of times:

Operation 1: If the number contains the substring "00", you can replace it with "10".

For example, "00010" -> "10010"

Operation 2: If the number contains the substring "10", you can replace it with "01".

For example, "00010" -> "00001"

Return the maximum binary string you can obtain after any number of operations. Binary string x is greater than bin ary string y if x's decimal representation is greater than y's decimal representation.

Example 1:

Input: binary = "000110"
Output: "111011"
Explanation: A valid transformation sequence can be:
"000110" -> "000101"
"000101" -> "100101"
"100101" -> "110101"
"110101" -> "110011"
"110011" -> "111011"

Example 2:

Input: binary = "01"
Output: "01"
Explanation: "01" cannot be transformed any further.

Constraints:

1 <= binary.length <= 105
binary consist of '0' and '1'.

1703. Minimum Adjacent Swaps for K Consecutive Ones

You are given an integer array, nums, and an integer k. nums comprises of only 0's and 1's. In one move, you can ch oose two adjacent indices and swap their values.
Return the minimum number of moves required so that nums has k consecutive 1's.

Example 1:

Input: nums = [1,0,0,1,0,1], k = 2
Output: 1
Explanation: In 1 move, nums could be [1,0,0,0,1,1] and have 2 consecutive 1's.

Example 2:

Input: nums = [1,0,0,0,0,0,1,1], k = 3
Output: 5
Explanation: In 5 moves, the leftmost 1 can be shifted right until nums = [0,0,0,0,0,1,1,1].

Example 3:

Input: nums = [1,1,0,1], k = 2
Output: 0

Explanation: nums already has 2 consecutive 1's.

Constraints:

1 <= nums.length <= 105
nums[i] is 0 or 1.
1 <= k <= sum(nums)

1704. Determine if String Halves Are Alike

You are given a string s of even length. Split this string into two halves of equal lengths, and let a be the first half an d b be the second half.
Two strings are alike if they have the same number of vowels ('a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U'). Notice that s cont ains uppercase and lowercase letters.
Return true if a and b are alike. Otherwise, return false.

Example 1:

Input: s = "book"
Output: true
Explanation: a = "bo" and b = "ok". a has 1 vowel and b has 1 vowel. Therefore, they are alike.

Example 2:

Input: s = "textbook"
Output: false
Explanation: a = "text" and b = "book". a has 1 vowel whereas b has 2. Therefore, they are not alike.
Notice that the vowel o is counted twice.

Example 3:

Input: s = "MerryChristmas"
Output: false

Example 4:

Input: s = "AbCdEfGh"
Output: true

Constraints:

2 <= s.length <= 1000
s.length is even.
s consists of uppercase and lowercase letters.

1705. Maximum Number of Eaten Apples

There is a special kind of apple tree that grows apples every day for n days. On the ith day, the tree grows apples[i] a pples that will rot after days[i] days, that is on day i + days[i] the apples will be rotten and cannot be eaten. On some days, the apple tree does not grow any apples, which are denoted by apples[i] == 0 and days[i] == 0.
You decided to eat at most one apple a day (to keep the doctors away). Note that you can keep eating after the first n days.
Given two integer arrays days and apples of length n, return the maximum number of apples you can eat.

Example 1:

Input: apples = [1,2,3,5,2], days = [3,2,1,4,2]
Output: 7
Explanation: You can eat 7 apples:
- On the first day, you eat an apple that grew on the first day.
- On the second day, you eat an apple that grew on the second day.
- On the third day, you eat an apple that grew on the second day. After this day, the apples that grew on the third day rot.
- On the fourth to the seventh days, you eat apples that grew on the fourth day.

Example 2:

Input: apples = [3,0,0,0,0,2], days = [3,0,0,0,0,2]
Output: 5
Explanation: You can eat 5 apples:
- On the first to the third day you eat apples that grew on the first day.
- Do nothing on the fouth and fifth days.
- On the sixth and seventh days you eat apples that grew on the sixth day.

Constraints:

apples.length == n
days.length == n
1 <= n <= 2 * 104
0 <= apples[i], days[i] <= 2 * 104
days[i] = 0 if and only if apples[i] = 0.

1706. Where Will the Ball Fall

You have a 2-D grid of size m x n representing a box, and you have n balls. The box is open on the top and bottom si des.
Each cell in the box has a diagonal board spanning two corners of the cell that can redirect a ball to the right or to the left.

A board that redirects the ball to the right spans the top-left corner to the bottom-right corner and is represented in th e grid as 1.
A board that redirects the ball to the left spans the top-right corner to the bottom-left corner and is represented in the grid as -1.

We drop one ball at the top of each column of the box. Each ball can get stuck in the box or fall out of the bottom. A ball gets stuck if it hits a "V" shaped pattern between two boards or if a board redirects the ball into either wall of th e box.
Return an array answer of size n where answer[i] is the column that the ball falls out of at the bottom after dropping t he ball from the ith column at the top, or -1 if the ball gets stuck in the box.

Example 1:

Input: grid = [[1,1,1,-1,-1],[1,1,1,-1,-1],[-1,-1,-1,1,1],[1,1,1,1,-1],[-1,-1,-1,-1,-1]]
Output: [1,-1,-1,-1,-1]
Explanation: This example is shown in the photo.
Ball b0 is dropped at column 0 and falls out of the box at column 1.
Ball b1 is dropped at column 1 and will get stuck in the box between column 2 and 3 and row 1.
Ball b2 is dropped at column 2 and will get stuck on the box between column 2 and 3 and row 0.
Ball b3 is dropped at column 3 and will get stuck on the box between column 2 and 3 and row 0.
Ball b4 is dropped at column 4 and will get stuck on the box between column 2 and 3 and row 1.

Example 2:

Input: grid = [[-1]]
Output: [-1]
Explanation: The ball gets stuck against the left wall.

Example 3:

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

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 100
grid[i][j] is 1 or -1.

1707. Maximum XOR With an Element From Array

You are given an array nums consisting of non-negative integers. You are also given a queries array, where queries[i] = [xi, mi].
The answer to the ith query is the maximum bitwise XOR value of xi and any element of nums that does not exceed mi. In other words, the answer is max(nums[j] XOR xi) for all j such that nums[j] <= mi. If all elements in nums are larger than mi, then the answer is -1.
Return an integer array answer where answer.length == queries.length and answer[i] is the answer to the ith query.

Example 1:

Input: nums = [0,1,2,3,4], queries = [[3,1],[1,3],[5,6]]
Output: [3,3,7]
Explanation:
1) 0 and 1 are the only two integers not greater than 1. 0 XOR 3 = 3 and 1 XOR 3 = 2. The larger of the two is 3.
2) 1 XOR 2 = 3.
3) 5 XOR 2 = 7.

Example 2:

Input: nums = [5,2,4,6,6,3], queries = [[12,4],[8,1],[6,3]]
Output: [15,-1,5]

Constraints:

1 <= nums.length, queries.length <= 105
queries[i].length == 2
0 <= nums[j], xi, mi <= 109

1710. Maximum Units on a Truck

You are assigned to put some amount of boxes onto one truck. You are given a 2D array boxTypes, where boxTypes [i] = [numberOfBoxesi, numberOfUnitsPerBoxi]:

numberOfBoxesi is the number of boxes of type i.
numberOfUnitsPerBoxi is the number of units in each box of the type i.

You are also given an integer truckSize, which is the maximum number of boxes that can be put on the truck. You c an choose any boxes to put on the truck as long as the number of boxes does not exceed truckSize.
Return the maximum total number of units that can be put on the truck.

Example 1:

Input: boxTypes = [[1,3],[2,2],[3,1]], truckSize = 4
Output: 8
Explanation: There are:
- 1 box of the first type that contains 3 units.
- 2 boxes of the second type that contain 2 units each.
- 3 boxes of the third type that contain 1 unit each.
You can take all the boxes of the first and second types, and one box of the third type.
The total number of units will be = (1 * 3) + (2 * 2) + (1 * 1) = 8.

Example 2:

Input: boxTypes = [[5,10],[2,5],[4,7],[3,9]], truckSize = 10
Output: 91

Constraints:

1 <= boxTypes.length <= 1000
1 <= numberOfBoxesi, numberOfUnitsPerBoxi <= 1000
1 <= truckSize <= 106

1711. Count Good Meals

A good meal is a meal that contains exactly two different food items with a sum of deliciousness equal to a power of two.
You can pick any two different foods to make a good meal.
Given an array of integers deliciousness where deliciousness[i] is the deliciousness of the i th item of food, return the number of different good meals you can make from this list modulo 109 + 7.
Note that items with different indices are considered different even if they have the same deliciousness value.

Example 1:

Input: deliciousness = [1,3,5,7,9]
Output: 4
Explanation: The good meals are (1,3), (1,7), (3,5) and, (7,9).
Their respective sums are 4, 8, 8, and 16, all of which are powers of 2.

Example 2:

Input: deliciousness = [1,1,1,3,3,3,7]
Output: 15
Explanation: The good meals are (1,1) with 3 ways, (1,3) with 9 ways, and (1,7) with 3 ways.

Constraints:

1 <= deliciousness.length <= 105
0 <= deliciousness[i] <= 220

1712. Ways to Split Array Into Three Subarrays

A split of an integer array is good if:

The array is split into three non-empty contiguous subarrays - named left, mid, right respectively from left to right. The sum of the elements in left is less than or equal to the sum of the elements in mid, and the sum of the elements i n mid is less than or equal to the sum of the elements in right.

Given nums, an array of non-negative integers, return the number of good ways to split nums. As the number may be too large, return it modulo 109 + 7.

Example 1:

Input: nums = [1,1,1]
Output: 1
Explanation: The only good way to split nums is [1] [1] [1].
Example 2:

Input: nums = [1,2,2,2,5,0]
Output: 3
Explanation: There are three good ways of splitting nums:
[1] [2] [2,2,5,0]
[1] [2,2] [2,5,0]
[1,2] [2,2] [5,0]

Example 3:

Input: nums = [3,2,1]
Output: 0
Explanation: There is no good way to split nums.

Constraints:

3 <= nums.length <= 105
0 <= nums[i] <= 104

1713. Minimum Operations to Make a Subsequence

You are given an array target that consists of distinct integers and another integer array arr that can have duplicates.
In one operation, you can insert any integer at any position in arr. For example, if arr = [1,4,1,2], you can add 3 in th e middle and make it [1,4,3,1,2]. Note that you can insert the integer at the very beginning or end of the array.
Return the minimum number of operations needed to make target a subsequence of arr.
A subsequence of an array is a new array generated from the original array by deleting some elements (possibly non e) without changing the remaining elements' relative order. For example, [2,7,4] is a subsequence of [4,2,3,7,2,1,4] (t he underlined elements), while [2,4,2] is not.

Example 1:

Input: target = [5,1,3], arr = [9,4,2,3,4]
Output: 2
Explanation: You can add 5 and 1 in such a way that makes arr = [5,9,4,1,2,3,4], then target will be a subsequence of arr.

Example 2:

Input: target = [6,4,8,1,3,2], arr = [4,7,6,2,3,8,6,1]
Output: 3

Constraints:

1 <= target.length, arr.length <= 105
1 <= target[i], arr[i] <= 109
target contains no duplicates.

1716. Calculate Money in Leetcode Bank

Hercy wants to save money for his first car. He puts money in the Leetcode bank every day.
He starts by putting in $1 on Monday, the first day. Every day from Tuesday to Sunday, he will put in $1 more than t he day before. On every subsequent Monday, he will put in $1 more than the previous Monday.
Given n, return the total amount of money he will have in the Leetcode bank at the end of the nth day.

Example 1:

Input: n = 4
Output: 10
Explanation: After the 4th day, the total is 1 + 2 + 3 + 4 = 10.

Example 2:

Input: n = 10

Output: 37
Explanation: After the 10th day, the total is (1 + 2 + 3 + 4 + 5 + 6 + 7) + (2 + 3 + 4) = 37. Notice that on the 2nd Mo nday, Hercy only puts in $2.

Example 3:

Input: n = 20
Output: 96
Explanation: After the 20th day, the total is (1 + 2 + 3 + 4 + 5 + 6 + 7) + (2 + 3 + 4 + 5 + 6 + 7 + 8) + (3 + 4 + 5 + 6 + 7 + 8) = 96.

Constraints:

1 <= n <= 1000

1717. Maximum Score From Removing Substrings

You are given a string s and two integers x and y. You can perform two types of operations any number of times.

Remove substring "ab" and gain x points.

For example, when removing "ab" from "cabxbae" it becomes "cxbae".

Remove substring "ba" and gain y points.

For example, when removing "ba" from "cabxbae" it becomes "cabxe".

Return the maximum points you can gain after applying the above operations on s.

Example 1:

Input: s = "cdbcbbaaabab", x = 4, y = 5
Output: 19
Explanation:
- Remove the "ba" underlined in "cdbcbbaaabab". Now, s = "cdbcbbaaab" and 5 points are added to the score.
- Remove the "ab" underlined in "cdbcbbaaab". Now, s = "cdbcbbaa" and 4 points are added to the score.
- Remove the "ba" underlined in "cdbcbbaa". Now, s = "cdbcba" and 5 points are added to the score.
- Remove the "ba" underlined in "cdbcba". Now, s = "cdbc" and 5 points are added to the score.
Total score = 5 + 4 + 5 + 5 = 19.
Example 2:

Input: s = "aabbaaxybbaabb", x = 5, y = 4
Output: 20

Constraints:

1 <= s.length <= 105
1 <= x, y <= 104
s consists of lowercase English letters.

1718. Construct the Lexicographically Largest Valid Sequence

Given an integer n, find a sequence that satisfies all of the following:

The integer 1 occurs once in the sequence.
Each integer between 2 and n occurs twice in the sequence.
For every integer i between 2 and n, the distance between the two occurrences of i is exactly i.

The distance between two numbers on the sequence, a[i] and a[j], is the absolute difference of their indices, |j - i|.
Return the lexicographically largest sequence. It is guaranteed that under the given constraints, there is always a solu tion.
A sequence a is lexicographically larger than a sequence b (of the same length) if in the first position where a and b differ, sequence a has a number greater than the corresponding number in b. For example, [0,1,9,0] is lexicographica lly larger than [0,1,5,6] because the first position they differ is at the third number, and 9 is greater than 5.

Example 1:

Input: n = 3
Output: [3,1,2,3,2]
Explanation: [2,3,2,1,3] is also a valid sequence, but [3,1,2,3,2] is the lexicographically largest valid sequence.

Example 2:

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

Constraints:

1 <= n <= 20

1719. Number Of Ways To Reconstruct A Tree

You are given an array pairs, where pairs[i] = [xi, yi], and:

There are no duplicates.
xi < yi

Let ways be the number of rooted trees that satisfy the following conditions:

The tree consists of nodes whose values appeared in pairs.
A pair [xi, yi] exists in pairs if and only if xi is an ancestor of yi or yi is an ancestor of xi.
Note: the tree does not have to be a binary tree.

Two ways are considered to be different if there is at least one node that has different parents in both ways.
Return:

0 if ways == 0
1 if ways == 1
2 if ways > 1

A rooted tree is a tree that has a single root node, and all edges are oriented to be outgoing from the root.
An ancestor of a node is any node on the path from the root to that node (excluding the node itself). The root has no ancestors.

Example 1:

Input: pairs = [[1,2],[2,3]]
Output: 1
Explanation: There is exactly one valid rooted tree, which is shown in the above figure.

Example 2:

Input: pairs = [[1,2],[2,3],[1,3]]
Output: 2
Explanation: There are multiple valid rooted trees. Three of them are shown in the above figures.

Example 3:

Input: pairs = [[1,2],[2,3],[2,4],[1,5]]
Output: 0
Explanation: There are no valid rooted trees.

Constraints:

1 <= pairs.length <= 105
1 <= xi < yi <= 500
The elements in pairs are unique.

1720. Decode XORed Array

There is a hidden integer array arr that consists of n non-negative integers.
It was encoded into another integer array encoded of length n - 1, such that encoded[i] = arr[i] XOR arr[i + 1]. For e
xample, if arr = [1,0,2,1], then encoded = [1,2,3].
You are given the encoded array. You are also given an integer first, that is the first element of arr, i.e. arr[0].
Return the original array arr. It can be proved that the answer exists and is unique.

Example 1:

Input: encoded = [1,2,3], first = 1
Output: [1,0,2,1]
Explanation: If arr = [1,0,2,1], then first = 1 and encoded = [1 XOR 0, 0 XOR 2, 2 XOR 1] = [1,2,3]

Example 2:

Input: encoded = [6,2,7,3], first = 4
Output: [4,2,0,7,4]

Constraints:

2 <= n <= 104
encoded.length == n - 1
0 <= encoded[i] <= 105
0 <= first <= 105

1721. Swapping Nodes in a Linked List

You are given the head of a linked list, and an integer k.
Return the head of the linked list after swapping the values of the kth node from the beginning and the kth node from
the end (the list is 1-indexed).

Example 1:

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

Example 2:

Input: head = [7,9,6,6,7,8,3,0,9,5], k = 5
Output: [7,9,6,6,8,7,3,0,9,5]

Example 3:

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

Example 4:

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

Example 5:

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

Constraints:

The number of nodes in the list is n.
1 <= k <= n <= 105
0 <= Node.val <= 100

1722. Minimize Hamming Distance After Swap Operations

You are given two integer arrays, source and target, both of length n. You are also given an array allowedSwaps whe re each allowedSwaps[i] = [ai, bi] indicates that you are allowed to swap the elements at index ai and index bi (0-ind exed) of array source. Note that you can swap elements at a specific pair of indices multiple times and in any order. The Hamming distance of two arrays of the same length, source and target, is the number of positions where the ele ments are different. Formally, it is the number of indices i for 0 <= i <= n-1 where source[i] != target[i] (0-indexed). Return the minimum Hamming distance of source and target after performing any amount of swap operations on arr ay source.

Example 1:

Input: source = [1,2,3,4], target = [2,1,4,5], allowedSwaps = [[0,1],[2,3]]
Output: 1
Explanation: source can be transformed the following way:
- Swap indices 0 and 1: source = [2,1,3,4]
- Swap indices 2 and 3: source = [2,1,4,3]
The Hamming distance of source and target is 1 as they differ in 1 position: index 3.

Example 2:

Input: source = [1,2,3,4], target = [1,3,2,4], allowedSwaps = []
Output: 2
Explanation: There are no allowed swaps.
The Hamming distance of source and target is 2 as they differ in 2 positions: index 1 and index 2.

Example 3:

Input: source = [5,1,2,4,3], target = [1,5,4,2,3], allowedSwaps = [[0,4],[4,2],[1,3],[1,4]]
Output: 0

Constraints:

n == source.length == target.length
1 <= n <= 105
1 <= source[i], target[i] <= 105
0 <= allowedSwaps.length <= 105
allowedSwaps[i].length == 2
0 <= ai, bi <= n - 1
ai != bi

1723. Find Minimum Time to Finish All Jobs

You are given an integer array jobs, where jobs[i] is the amount of time it takes to complete the ith job.
There are k workers that you can assign jobs to. Each job should be assigned to exactly one worker. The working tim e of a worker is the sum of the time it takes to complete all jobs assigned to them. Your goal is to devise an optimal a ssignment such that the maximum working time of any worker is minimized.
Return the minimum possible maximum working time of any assignment.

Example 1:

Input: jobs = [3,2,3], k = 3
Output: 3
Explanation: By assigning each person one job, the maximum time is 3.

Example 2:

Input: jobs = [1,2,4,7,8], k = 2
Output: 11
Explanation: Assign the jobs the following way:
Worker 1: 1, 2, 8 (working time = 1 + 2 + 8 = 11)
Worker 2: 4, 7 (working time = 4 + 7 = 11)
The maximum working time is 11.

Constraints:

1 <= k <= jobs.length <= 12
1 <= jobs[i] <= 107

1725. Number Of Rectangles That Can Form The Largest Square

You are given an array rectangles where rectangles[i] = [li, wi] represents the ith rectangle of length li and width wi.
You can cut the ith rectangle to form a square with a side length of k if both k <= li and k <= wi. For example, if you have a rectangle [4,6], you can cut it to get a square with a side length of at most 4.
Let maxLen be the side length of the largest square you can obtain from any of the given rectangles.
Return the number of rectangles that can make a square with a side length of maxLen.

Example 1:

Input: rectangles = [[5,8],[3,9],[5,12],[16,5]]
Output: 3
Explanation: The largest squares you can get from each rectangle are of lengths [5,3,5,5].
The largest possible square is of length 5, and you can get it out of 3 rectangles.

Example 2:

Input: rectangles = [[2,3],[3,7],[4,3],[3,7]]
Output: 3

Constraints:

1 <= rectangles.length <= 1000
rectangles[i].length == 2
1 <= li, wi <= 109
li != wi

1726. Tuple with Same Product

Given an array nums of distinct positive integers, return the number of tuples (a, b, c, d) such that a * b = c * d wher e a, b, c, and d are elements of nums, and a != b != c != d.

Example 1:

Input: nums = [2,3,4,6]
Output: 8
Explanation: There are 8 valid tuples:
(2,6,3,4) , (2,6,4,3) , (6,2,3,4) , (6,2,4,3)
(3,4,2,6) , (4,3,2,6) , (3,4,6,2) , (4,3,6,2)

Example 2:

Input: nums = [1,2,4,5,10]
Output: 16
Explanation: There are 16 valids tuples:
(1,10,2,5) , (1,10,5,2) , (10,1,2,5) , (10,1,5,2)
(2,5,1,10) , (2,5,10,1) , (5,2,1,10) , (5,2,10,1)
(2,10,4,5) , (2,10,5,4) , (10,2,4,5) , (10,2,5,4)
(4,5,2,10) , (4,5,10,2) , (5,4,2,10) , (5,4,10,2)

Example 3:

Input: nums = [2,3,4,6,8,12]
Output: 40

Example 4:

Input: nums = [2,3,5,7]
Output: 0

Constraints:

1 <= nums.length <= 1000
1 <= nums[i] <= 104
All elements in nums are distinct.

1727. Largest Submatrix With Rearrangements

You are given a binary matrix matrix of size m x n, and you are allowed to rearrange the columns of the matrix in an y order.
Return the area of the largest submatrix within matrix where every element of the submatrix is 1 after reordering the columns optimally.

Example 1:

Input: matrix = [[0,0,1],[1,1,1],[1,0,1]]
Output: 4

Explanation: You can rearrange the columns as shown above.
The largest submatrix of 1s, in bold, has an area of 4.

Example 2:

Input: matrix = [[1,0,1,0,1]]
Output: 3
Explanation: You can rearrange the columns as shown above.
The largest submatrix of 1s, in bold, has an area of 3.

Example 3:

Input: matrix = [[1,1,0],[1,0,1]]
Output: 2
Explanation: Notice that you must rearrange entire columns, and there is no way to make a submatrix of 1s larger tha
n an area of 2.
Example 4:

Input: matrix = [[0,0],[0,0]]
Output: 0
Explanation: As there are no 1s, no submatrix of 1s can be formed and the area is 0.

Constraints:

m == matrix.length
n == matrix[i].length
1 <= m * n <= 105
matrix[i][j] is 0 or 1.

1728. Cat and Mouse II

A game is played by a cat and a mouse named Cat and Mouse.
The environment is represented by a grid of size rows x cols, where each element is a wall, floor, player (Cat, Mouse
), or food.

Players are represented by the characters 'C'(Cat),'M'(Mouse).
Floors are represented by the character '.' and can be walked on.
Walls are represented by the character '#' and cannot be walked on.
Food is represented by the character 'F' and can be walked on.
There is only one of each character 'C', 'M', and 'F' in grid.

Mouse and Cat play according to the following rules:

Mouse moves first, then they take turns to move.
During each turn, Cat and Mouse can jump in one of the four directions (left, right, up, down). They cannot jump ov

er the wall nor outside of the grid.
catJump, mouseJump are the maximum lengths Cat and Mouse can jump at a time, respectively. Cat and Mouse can jump less than the maximum length.
Staying in the same position is allowed.
Mouse can jump over Cat.

The game can end in 4 ways:

If Cat occupies the same position as Mouse, Cat wins.
If Cat reaches the food first, Cat wins.
If Mouse reaches the food first, Mouse wins.
If Mouse cannot get to the food within 1000 turns, Cat wins.

Given a rows x cols matrix grid and two integers catJump and mouseJump, return true if Mouse can win the game if both Cat and Mouse play optimally, otherwise return false.

Example 1:

Input: grid = ["####F","#C...","M...."], catJump = 1, mouseJump = 2
Output: true
Explanation: Cat cannot catch Mouse on its turn nor can it get the food before Mouse.

Example 2:

Input: grid = ["M.C...F"], catJump = 1, mouseJump = 4
Output: true

Example 3:

Input: grid = ["M.C...F"], catJump = 1, mouseJump = 3
Output: false

Example 4:

Input: grid = ["C...#","...#F","....#","M...."], catJump = 2, mouseJump = 5
Output: false

Example 5:

Input: grid = [".M...","..#..","#..#.","C#.#.","...#F"], catJump = 3, mouseJump = 1
Output: true

Constraints:

rows == grid.length
cols = grid[i].length
1 <= rows, cols <= 8
grid[i][j] consist only of characters 'C', 'M', 'F', '.', and '#'.
There is only one of each character 'C', 'M', and 'F' in grid.
1 <= catJump, mouseJump <= 8

1732. Find the Highest Altitude

There is a biker going on a road trip. The road trip consists of n + 1 points at different altitudes. The biker starts his t rip on point 0 with altitude equal 0.
You are given an integer array gain of length n where gain[i] is the net gain in altitude between points i and i + 1 for all (0 <= i < n). Return the highest altitude of a point.

Example 1:

Input: gain = [-5,1,5,0,-7]
Output: 1
Explanation: The altitudes are [0,-5,-4,1,1,-6]. The highest is 1.

Example 2:

Input: gain = [-4,-3,-2,-1,4,3,2]
Output: 0
Explanation: The altitudes are [0,-4,-7,-9,-10,-6,-3,-1]. The highest is 0.

Constraints:

n == gain.length
1 <= n <= 100
-100 <= gain[i] <= 100

1733. Minimum Number of People to Teach

On a social network consisting of m users and some friendships between users, two users can communicate with eac h other if they know a common language.
You are given an integer n, an array languages, and an array friendships where:

There are n languages numbered 1 through n,
languages[i] is the set of languages the i th user knows, and
friendships[i] = [u i , v i] denotes a friendship between the users u i and vi.

You can choose one language and teach it to some users so that all friends can communicate with each other. Return the minimum number of users you need to teach.

Note that friendships are not transitive, meaning if x is a friend of y and y is a friend of z, this doesn't guarantee that x is a friend of z.

Example 1:

Input: n = 2, languages = [[1],[2],[1,2]], friendships = [[1,2],[1,3],[2,3]]
Output: 1
Explanation: You can either teach user 1 the second language or user 2 the first language.

Example 2:

Input: n = 3, languages = [[2],[1,3],[1,2],[3]], friendships = [[1,4],[1,2],[3,4],[2,3]]
Output: 2
Explanation: Teach the third language to users 1 and 3, yielding two users to teach.

Constraints:

2 <= n <= 500
languages.length == m
1 <= m <= 500
1 <= languages[i].length <= n
1 <= languages[i][j] <= n
1 <= u i < v i <= languages.length
1 <= friendships.length <= 500
All tuples (u i, v i) are unique
languages[i] contains only unique values

1734. Decode XORed Permutation

There is an integer array perm that is a permutation of the first n positive integers, where n is always odd.
It was encoded into another integer array encoded of length n - 1, such that encoded[i] = perm[i] XOR perm[i + 1]. F
or example, if perm = [1,3,2], then encoded = [2,1].
Given the encoded array, return the original array perm. It is guaranteed that the answer exists and is unique.

Example 1:

Input: encoded = [3,1]
Output: [1,2,3]
Explanation: If perm = [1,2,3], then encoded = [1 XOR 2,2 XOR 3] = [3,1]

Example 2:

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

Constraints:

3 <= n < 105
n is odd.
encoded.length == n - 1

1735. Count Ways to Make Array With Product

You are given a 2D integer array, queries. For each queries[i], where queries[i] = [ni, ki], find the number of differen t ways you can place positive integers into an array of size ni such that the product of the integers is ki. As the numb er of ways may be too large, the answer to the ith query is the number of ways modulo 109 + 7.
Return an integer array answer where answer.length == queries.length, and answer[i] is the answer to the ith query.

Example 1:

Input: queries = [[2,6],[5,1],[73,660]]
Output: [4,1,50734910]
Explanation: Each query is independent.
[2,6]: There are 4 ways to fill an array of size 2 that multiply to 6: [1,6], [2,3], [3,2], [6,1].
[5,1]: There is 1 way to fill an array of size 5 that multiply to 1: [1,1,1,1,1].
[73,660]: There are 1050734917 ways to fill an array of size 73 that multiply to 660. 1050734917 modulo 109 + 7 = 50734910.

Example 2:

Input: queries = [[1,1],[2,2],[3,3],[4,4],[5,5]]
Output: [1,2,3,10,5]

Constraints:

1 <= queries.length <= 104
1 <= ni, ki <= 104

1736. Latest Time by Replacing Hidden Digits

You are given a string time in the form of hh:mm, where some of the digits in the string are hidden (represented by

?).
The valid times are those inclusively between 00:00 and 23:59.
Return the latest valid time you can get from time by replacing the hidden digits.

Example 1:

Input: time = "2?:?0"
Output: "23:50"
Explanation: The latest hour beginning with the digit '2' is 23 and the latest minute ending with the digit '0' is 50.

Example 2:

Input: time = "0?:3?"
Output: "09:39"

Example 3:

Input: time = "1?:22"
Output: "19:22"

Constraints:

time is in the format hh:mm.
It is guaranteed that you can produce a valid time from the given string.

1737. Change Minimum Characters to Satisfy One of Three Conditions

You are given two strings a and b that consist of lowercase letters. In one operation, you can change any character in a or b to any lowercase letter.
Your goal is to satisfy one of the following three conditions:

Every letter in a is strictly less than every letter in b in the alphabet.
Every letter in b is strictly less than every letter in a in the alphabet.
Both a and b consist of only one distinct letter.

Return the minimum number of operations needed to achieve your goal.

Example 1:

Input: a = "aba", b = "caa"
Output: 2
Explanation: Consider the best way to make each condition true:
1) Change b to "ccc" in 2 operations, then every letter in a is less than every letter in b.
2) Change a to "bbb" and b to "aaa" in 3 operations, then every letter in b is less than every letter in a.
3) Change a to "aaa" and b to "aaa" in 2 operations, then a and b consist of one distinct letter.

The best way was done in 2 operations (either condition 1 or condition 3).

Example 2:

Input: a = "dabadd", b = "cda"
Output: 3
Explanation: The best way is to make condition 1 true by changing b to "eee".

Constraints:

1 <= a.length, b.length <= 105
a and b consist only of lowercase letters.

1738. Find Kth Largest XOR Coordinate Value

You are given a 2D matrix of size m x n, consisting of non-negative integers. You are also given an integer k.
The value of coordinate (a, b) of the matrix is the XOR of all matrix[i][j] where 0 <= i <= a < m and 0 <= j <= b < n (0-indexed).
Find the kth largest value (1-indexed) of all the coordinates of matrix.

Example 1:

Input: matrix = [[5,2],[1,6]], k = 1
Output: 7
Explanation: The value of coordinate (0,1) is 5 XOR 2 = 7, which is the largest value.
Example 2:

Input: matrix = [[5,2],[1,6]], k = 2
Output: 5
Explanation: The value of coordinate (0,0) is 5 = 5, which is the 2nd largest value.
Example 3:

Input: matrix = [[5,2],[1,6]], k = 3
Output: 4
Explanation: The value of coordinate (1,0) is 5 XOR 1 = 4, which is the 3rd largest value.
Example 4:

Input: matrix = [[5,2],[1,6]], k = 4
Output: 0
Explanation: The value of coordinate (1,1) is 5 XOR 2 XOR 1 XOR 6 = 0, which is the 4th largest value.

Constraints:

m == matrix.length
n == matrix[i].length

1 <= m, n <= 1000
0 <= matrix[i][j] <= 106
1 <= k <= m * n

1739. Building Boxes

You have a cubic storeroom where the width, length, and height of the room are all equal to n units. You are asked t o place n boxes in this room where each box is a cube of unit side length. There are however some rules to placing th e boxes:

You can place the boxes anywhere on the floor.
If box x is placed on top of the box y, then each side of the four vertical sides of the box y must either be adjacent to another box or to a wall.

Given an integer n, return the minimum possible number of boxes touching the floor.

Example 1:

Input: n = 3
Output: 3
Explanation: The figure above is for the placement of the three boxes.
These boxes are placed in the corner of the room, where the corner is on the left side.

Example 2:

Input: n = 4
Output: 3
Explanation: The figure above is for the placement of the four boxes.
These boxes are placed in the corner of the room, where the corner is on the left side.

Example 3:

Input: n = 10
Output: 6
Explanation: The figure above is for the placement of the ten boxes.
These boxes are placed in the corner of the room, where the corner is on the back side.

Constraints:

1 <= n <= 109

1742. Maximum Number of Balls in a Box

You are working in a ball factory where you have n balls numbered from lowLimit up to highLimit inclusive (i.e., n == highLimit - lowLimit + 1), and an infinite number of boxes numbered from 1 to infinity.
Your job at this factory is to put each ball in the box with a number equal to the sum of digits of the ball's number. F or example, the ball number 321 will be put in the box number 3 + 2 + 1 = 6 and the ball number 10 will be put in th e box number 1 + 0 = 1.
Given two integers lowLimit and highLimit, return the number of balls in the box with the most balls.

Example 1:

Input: lowLimit = 1, highLimit = 10
Output: 2
Explanation:
Box Number: 1 2 3 4 5 6 7 8 9 10 11 ...
Ball Count: 2 1 1 1 1 1 1 1 1 0 0 ...
Box 1 has the most number of balls with 2 balls.
Example 2:

Input: lowLimit = 5, highLimit = 15
Output: 2
Explanation:
Box Number: 1 2 3 4 5 6 7 8 9 10 11 ...
Ball Count: 1 1 1 1 2 2 1 1 1 0 0 ...
Boxes 5 and 6 have the most number of balls with 2 balls in each.

Example 3:

Input: lowLimit = 19, highLimit = 28
Output: 2
Explanation:
Box Number: 1 2 3 4 5 6 7 8 9 10 11 12 ...
Ball Count: 0 1 1 1 1 1 1 1 1 2 0 0 ...
Box 10 has the most number of balls with 2 balls.

Constraints:

1 <= lowLimit <= highLimit <= 105

1743. Restore the Array From Adjacent Pairs

There is an integer array nums that consists of n unique elements, but you have forgotten it. However, you do remember every pair of adjacent elements in nums.
You are given a 2D integer array adjacentPairs of size n - 1 where each adjacentPairs[i] = [ui, vi] indicates that the elements ui and vi are adjacent in nums.
It is guaranteed that every adjacent pair of elements nums[i] and nums[i+1] will exist in adjacentPairs, either as [nums[i], nums[i+1]] or [nums[i+1], nums[i]]. The pairs can appear in any order.
Return the original array nums. If there are multiple solutions, return any of them.

Example 1:

Input: adjacentPairs = [[2,1],[3,4],[3,2]]
Output: [1,2,3,4]
Explanation: This array has all its adjacent pairs in adjacentPairs.
Notice that adjacentPairs[i] may not be in left-to-right order.

Example 2:

Input: adjacentPairs = [[4,-2],[1,4],[-3,1]]
Output: [-2,4,1,-3]
Explanation: There can be negative numbers.
Another solution is [-3,1,4,-2], which would also be accepted.

Example 3:

Input: adjacentPairs = [[100000,-100000]]
Output: [100000,-100000]

Constraints:

nums.length == n
adjacentPairs.length == n - 1
adjacentPairs[i].length == 2
2 <= n <= 105
-105 <= nums[i], ui, vi <= 105
There exists some nums that has adjacentPairs as its pairs.

1744. Can You Eat Your Favorite Candy on Your Favorite Day?

You are given a (0-indexed) array of positive integers candiesCount where candiesCount[i] represents the number of candies of the ith type you have. You are also given a 2D array queries where queries[i] = [favoriteTypei, favoriteDayi, dailyCapi].
You play a game with the following rules:

You start eating candies on day 0.
You cannot eat any candy of type i unless you have eaten all candies of type i - 1.
You must eat at least one candy per day until you have eaten all the candies.

Construct a boolean array answer such that answer.length == queries.length and answer[i] is true if you can eat a can dy of type favoriteTypei on day favoriteDayi without eating more than dailyCapi candies on any day, and false other wise. Note that you can eat different types of candy on the same day, provided that you follow rule 2.
Return the constructed array answer.

Example 1:

Input: candiesCount = [7,4,5,3,8], queries = [[0,2,2],[4,2,4],[2,13,1000000000]]
Output: [true,false,true]
Explanation:
1- If you eat 2 candies (type 0) on day 0 and 2 candies (type 0) on day 1, you will eat a candy of type 0 on day 2.
2- You can eat at most 4 candies each day.
   If you eat 4 candies every day, you will eat 4 candies (type 0) on day 0 and 4 candies (type 0 and type 1) on day 1.
   On day 2, you can only eat 4 candies (type 1 and type 2), so you cannot eat a candy of type 4 on day 2.
3- If you eat 1 candy each day, you will eat a candy of type 2 on day 13.

Example 2:

Input: candiesCount = [5,2,6,4,1], queries = [[3,1,2],[4,10,3],[3,10,100],[4,100,30],[1,3,1]]
Output: [false,true,true,false,false]

Constraints:

1 <= candiesCount.length <= 105
1 <= candiesCount[i] <= 105
1 <= queries.length <= 105
queries[i].length == 3
0 <= favoriteTypei < candiesCount.length
0 <= favoriteDayi <= 109
1 <= dailyCapi <= 109

1745. Palindrome Partitioning IV

Given a string s, return true if it is possible to split the string s into three non-empty palindromic substrings. Otherwi se, return false.
A string is said to be palindrome if it the same string when reversed.

Example 1:

Input: s = "abcbdd"
Output: true

Explanation: "abcbdd" = "a" + "bcb" + "dd", and all three substrings are palindromes.

Example 2:

Input: s = "bcbddxy"
Output: false
Explanation: s cannot be split into 3 palindromes.

Constraints:

3 <= s.length <= 2000
s consists only of lowercase English letters.

1748. Sum of Unique Elements

You are given an integer array nums. The unique elements of an array are the elements that appear exactly once in th e array.
Return the sum of all the unique elements of nums.

Example 1:

Input: nums = [1,2,3,2]
Output: 4
Explanation: The unique elements are [1,3], and the sum is 4.

Example 2:

Input: nums = [1,1,1,1,1]
Output: 0
Explanation: There are no unique elements, and the sum is 0.

Example 3:

Input: nums = [1,2,3,4,5]
Output: 15
Explanation: The unique elements are [1,2,3,4,5], and the sum is 15.

Constraints:

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

1749. Maximum Absolute Sum of Any Subarray

You are given an integer array nums. The absolute sum of a subarray [numsl, numsl+1, ..., numsr-1, numsr] is abs(n
umsl + numsl+1 + ... + numsr-1 + numsr).
Return the maximum absolute sum of any (possibly empty) subarray of nums.
Note that abs(x) is defined as follows:

If x is a negative integer, then abs(x) = -x.
If x is a non-negative integer, then abs(x) = x.

Example 1:

Input: nums = [1,-3,2,3,-4]
Output: 5
Explanation: The subarray [2,3] has absolute sum = abs(2+3) = abs(5) = 5.

Example 2:

Input: nums = [2,-5,1,-4,3,-2]
Output: 8
Explanation: The subarray [-5,1,-4] has absolute sum = abs(-5+1-4) = abs(-8) = 8.

Constraints:

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

1750. Minimum Length of String After Deleting Similar Ends

Given a string s consisting only of characters 'a', 'b', and 'c'. You are asked to apply the following algorithm on the st
ring any number of times:

Pick a non-empty prefix from the string s where all the characters in the prefix are equal.
Pick a non-empty suffix from the string s where all the characters in this suffix are equal.
The prefix and the suffix should not intersect at any index.
The characters from the prefix and suffix must be the same.
Delete both the prefix and the suffix.

Return the minimum length of s after performing the above operation any number of times (possibly zero times).

Example 1:

Input: s = "ca"
Output: 2
Explanation: You can't remove any characters, so the string stays as is.

Example 2:

Input: s = "cabaabac"
Output: 0
Explanation: An optimal sequence of operations is:
- Take prefix = "c" and suffix = "c" and remove them, s = "abaaba".
- Take prefix = "a" and suffix = "a" and remove them, s = "baab".
- Take prefix = "b" and suffix = "b" and remove them, s = "aa".
- Take prefix = "a" and suffix = "a" and remove them, s = "".
Example 3:

Input: s = "aabccabba"
Output: 3
Explanation: An optimal sequence of operations is:
- Take prefix = "aa" and suffix = "a" and remove them, s = "bccabb".
- Take prefix = "b" and suffix = "bb" and remove them, s = "cca".

Constraints:

1 <= s.length <= 105
s only consists of characters 'a', 'b', and 'c'.

1751. Maximum Number of Events That Can Be Attended II

You are given an array of events where events[i] = [startDayi, endDayi, valuei]. The ith event starts at startDayi and ends at endDayi, and if you attend this event, you will receive a value of valuei. You are also given an integer k whic h represents the maximum number of events you can attend.
You can only attend one event at a time. If you choose to attend an event, you must attend the entire event. Note that the end day is inclusive: that is, you cannot attend two events where one of them starts and the other ends on the sa me day.
Return the maximum sum of values that you can receive by attending events.

Example 1:

Input: events = [[1,2,4],[3,4,3],[2,3,1]], k = 2
Output: 7

Explanation: Choose the green events, 0 and 1 (0-indexed) for a total value of 4 + 3 = 7.
Example 2:

Input: events = [[1,2,4],[3,4,3],[2,3,10]], k = 2
Output: 10
Explanation: Choose event 2 for a total value of 10.
Notice that you cannot attend any other event as they overlap, and that you do not have to attend k events.
Example 3:

Input: events = [[1,1,1],[2,2,2],[3,3,3],[4,4,4]], k = 3
Output: 9
Explanation: Although the events do not overlap, you can only attend 3 events. Pick the highest valued three.

Constraints:

1 <= k <= events.length
1 <= k * events.length <= 106
1 <= startDayi <= endDayi <= 109
1 <= valuei <= 106

1752. Check if Array Is Sorted and Rotated

Given an array nums, return true if the array was originally sorted in non-decreasing order, then rotated some numbe r of positions (including zero). Otherwise, return false.
There may be duplicates in the original array.
Note: An array A rotated by x positions results in an array B of the same length such that A[i] == B[(i+x) % A.lengt h], where % is the modulo operation.

Example 1:

Input: nums = [3,4,5,1,2]
Output: true
Explanation: [1,2,3,4,5] is the original sorted array.
You can rotate the array by x = 3 positions to begin on the the element of value 3: [3,4,5,1,2].

Example 2:

Input: nums = [2,1,3,4]
Output: false
Explanation: There is no sorted array once rotated that can make nums.

Example 3:

Input: nums = [1,2,3]

Output: true
Explanation: [1,2,3] is the original sorted array.
You can rotate the array by x = 0 positions (i.e. no rotation) to make nums.

Example 4:

Input: nums = [1,1,1]
Output: true
Explanation: [1,1,1] is the original sorted array.
You can rotate any number of positions to make nums.

Example 5:

Input: nums = [2,1]
Output: true
Explanation: [1,2] is the original sorted array.
You can rotate the array by x = 5 positions to begin on the element of value 2: [2,1].

Constraints:

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

1753. Maximum Score From Removing Stones

You are playing a solitaire game with three piles of stones of sizes a , b, and c respectively. Each turn you choose two different non-empty piles, take one stone from each, and add 1 point to your score. The game stops when there are fewer than two non-empty piles (meaning there are no more available moves).
Given three integers a , b, and c , return the maximum score you can get.

Example 1:

Input: a = 2, b = 4, c = 6
Output: 6
Explanation: The starting state is (2, 4, 6). One optimal set of moves is:
- Take from 1st and 3rd piles, state is now (1, 4, 5)
- Take from 1st and 3rd piles, state is now (0, 4, 4)
- Take from 2nd and 3rd piles, state is now (0, 3, 3)
- Take from 2nd and 3rd piles, state is now (0, 2, 2)
- Take from 2nd and 3rd piles, state is now (0, 1, 1)
- Take from 2nd and 3rd piles, state is now (0, 0, 0)
There are fewer than two non-empty piles, so the game ends. Total: 6 points.

Example 2:

Input: a = 4, b = 4, c = 6
Output: 7
Explanation: The starting state is (4, 4, 6). One optimal set of moves is:
- Take from 1st and 2nd piles, state is now (3, 3, 6)
- Take from 1st and 3rd piles, state is now (2, 3, 5)
- Take from 1st and 3rd piles, state is now (1, 3, 4)
- Take from 1st and 3rd piles, state is now (0, 3, 3)
- Take from 2nd and 3rd piles, state is now (0, 2, 2)
- Take from 2nd and 3rd piles, state is now (0, 1, 1)
- Take from 2nd and 3rd piles, state is now (0, 0, 0)
There are fewer than two non-empty piles, so the game ends. Total: 7 points.

Example 3:

Input: a = 1, b = 8, c = 8
Output: 8
Explanation: One optimal set of moves is to take from the 2nd and 3rd piles for 8 turns until they are empty.
After that, there are fewer than two non-empty piles, so the game ends.

Constraints:

1 <= a, b, c <= 105

1754. Largest Merge Of Two Strings

You are given two strings word1 and word2. You want to construct a string merge in the following way: while either word1 or word2 are non-empty, choose one of the following options:

If word1 is non-empty, append the first character in word1 to merge and delete it from word1.

For example, if word1 = "abc" and merge = "dv", then after choosing this operation, word1 = "bc" and merge = "dva".

If word2 is non-empty, append the first character in word2 to merge and delete it from word2.

For example, if word2 = "abc" and merge = "", then after choosing this operation, word2 = "bc" and merge = "a".

Return the lexicographically largest merge you can construct.
A string a is lexicographically larger than a string b (of the same length) if in the first position where a and b differ, a has a character strictly larger than the corresponding character in b. For example, "abcd" is lexicographically larger than "abcc" because the first position they differ is at the fourth character, and d is greater than c.

Example 1:

Input: word1 = "cabaa", word2 = "bcaaa"
Output: "cbcabaaaaa"
Explanation: One way to get the lexicographically largest merge is:
- Take from word1: merge = "c", word1 = "abaa", word2 = "bcaaa"
- Take from word2: merge = "cb", word1 = "abaa", word2 = "caaa"
- Take from word2: merge = "cbc", word1 = "abaa", word2 = "aaa"
- Take from word1: merge = "cbca", word1 = "baa", word2 = "aaa"
- Take from word1: merge = "cbcab", word1 = "aa", word2 = "aaa"
- Append the remaining 5 a's from word1 and word2 at the end of merge.

Example 2:

Input: word1 = "abcabc", word2 = "abdcaba"
Output: "abdcabcabcaba"

Constraints:

1 <= word1.length, word2.length <= 3000
word1 and word2 consist only of lowercase English letters.

1755. Closest Subsequence Sum

You are given an integer array nums and an integer goal.
You want to choose a subsequence of nums such that the sum of its elements is the closest possible to goal. That is, i f the sum of the subsequence's elements is sum, then you want to minimize the absolute difference abs(sum - goal).
Return the minimum possible value of abs(sum - goal).
Note that a subsequence of an array is an array formed by removing some elements (possibly all or none) of the origi nal array.

Example 1:

Input: nums = [5,-7,3,5], goal = 6
Output: 0
Explanation: Choose the whole array as a subsequence, with a sum of 6.
This is equal to the goal, so the absolute difference is 0.

Example 2:

Input: nums = [7,-9,15,-2], goal = -5
Output: 1
Explanation: Choose the subsequence [7,-9,-2], with a sum of -4.
The absolute difference is abs(-4 - (-5)) = abs(1) = 1, which is the minimum.

Example 3:

Input: nums = [1,2,3], goal = -7
Output: 7

Constraints:

1 <= nums.length <= 40
-107 <= nums[i] <= 107
-109 <= goal <= 109

1758. Minimum Changes To Make Alternating Binary String

You are given a string s consisting only of the characters '0' and '1'. In one operation, you can change any '0' to '1' or vice versa.
The string is called alternating if no two adjacent characters are equal. For example, the string "010" is alternating, w hile the string "0100" is not.
Return the minimum number of operations needed to make s alternating.

Example 1:

Input: s = "0100"
Output: 1
Explanation: If you change the last character to '1', s will be "0101", which is alternating.

Example 2:

Input: s = "10"
Output: 0
Explanation: s is already alternating.

Example 3:

Input: s = "1111"
Output: 2
Explanation: You need two operations to reach "0101" or "1010".

Constraints:

1 <= s.length <= 104
s[i] is either '0' or '1'.

1759. Count Number of Homogenous Substrings

Given a string s, return the number of homogenous substrings of s. Since the answer may be too large, return it mod ulo 109 + 7.
A string is homogenous if all the characters of the string are the same.
A substring is a contiguous sequence of characters within a string.

Example 1:

Input: s = "abbcccaa"
Output: 13
Explanation: The homogenous substrings are listed as below:
"a"   appears 3 times.
"aa" appears 1 time.
"b"   appears 2 times.
"bb" appears 1 time.
"c"   appears 3 times.
"cc" appears 2 times.
"ccc" appears 1 time.
3 + 1 + 2 + 1 + 3 + 2 + 1 = 13.
Example 2:

Input: s = "xy"
Output: 2
Explanation: The homogenous substrings are "x" and "y".
Example 3:

Input: s = "zzzzz"
Output: 15

Constraints:

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

1760. Minimum Limit of Balls in a Bag

You are given an integer array nums where the ith bag contains nums[i] balls. You are also given an integer maxOpe rations.

You can perform the following operation at most maxOperations times:

Take any bag of balls and divide it into two new bags with a positive number of balls.

For example, a bag of 5 balls can become two new bags of 1 and 4 balls, or two new bags of 2 and 3 balls.

Your penalty is the maximum number of balls in a bag. You want to minimize your penalty after the operations. Return the minimum possible penalty after performing the operations.

Example 1:

Input: nums = [9], maxOperations = 2
Output: 3
Explanation:
- Divide the bag with 9 balls into two bags of sizes 6 and 3. [9] -> [6,3].
- Divide the bag with 6 balls into two bags of sizes 3 and 3. [6,3] -> [3,3,3].
The bag with the most number of balls has 3 balls, so your penalty is 3 and you should return 3.

Example 2:

Input: nums = [2,4,8,2], maxOperations = 4
Output: 2
Explanation:
- Divide the bag with 8 balls into two bags of sizes 4 and 4. [2,4,8,2] -> [2,4,4,4,2].
- Divide the bag with 4 balls into two bags of sizes 2 and 2. [2,4,4,4,2] -> [2,2,2,4,4,2].
- Divide the bag with 4 balls into two bags of sizes 2 and 2. [2,2,2,4,4,2] -> [2,2,2,2,2,4,2].
- Divide the bag with 4 balls into two bags of sizes 2 and 2. [2,2,2,2,2,4,2] -> [2,2,2,2,2,2,2,2].
The bag with the most number of balls has 2 balls, so your penalty is 2 an you should return 2.

Example 3:

Input: nums = [7,17], maxOperations = 2
Output: 7

Constraints:

1 <= nums.length <= 105
1 <= maxOperations, nums[i] <= 109

1761. Minimum Degree of a Connected Trio in a Graph

You are given an undirected graph. You are given an integer n which is the number of nodes in the graph and an arra

y edges, where each edges[i] = [ui, vi] indicates that there is an undirected edge between ui and vi.
A connected trio is a set of three nodes where there is an edge between every pair of them.
The degree of a connected trio is the number of edges where one endpoint is in the trio, and the other is not.
Return the minimum degree of a connected trio in the graph, or -1 if the graph has no connected trios.

Example 1:

Input: n = 6, edges = [[1,2],[1,3],[3,2],[4,1],[5,2],[3,6]]
Output: 3
Explanation: There is exactly one trio, which is [1,2,3]. The edges that form its degree are bolded in the figure above
.

Example 2:

Input: n = 7, edges = [[1,3],[4,1],[4,3],[2,5],[5,6],[6,7],[7,5],[2,6]]
Output: 0
Explanation: There are exactly three trios:
1) [1,4,3] with degree 0.
2) [2,5,6] with degree 2.
3) [5,6,7] with degree 2.

Constraints:

2 <= n <= 400
edges[i].length == 2
1 <= edges.length <= n * (n-1) / 2
1 <= ui, vi <= n
ui != vi
There are no repeated edges.

1763. Longest Nice Substring

A string s is nice if, for every letter of the alphabet that s contains, it appears both in uppercase and lowercase. For ex
ample, "abABB" is nice because 'A' and 'a' appear, and 'B' and 'b' appear. However, "abA" is not because 'b' appears,
but 'B' does not.
Given a string s, return the longest substring of s that is nice. If there are multiple, return the substring of the earliest
occurrence. If there are none, return an empty string.

Example 1:

Input: s = "YazaAay"
Output: "aAa"
Explanation: "aAa" is a nice string because 'A/a' is the only letter of the alphabet in s, and both 'A' and 'a' appear.

"aAa" is the longest nice substring.

Example 2:

Input: s = "Bb"
Output: "Bb"
Explanation: "Bb" is a nice string because both 'B' and 'b' appear. The whole string is a substring.
Example 3:

Input: s = "c"
Output: ""
Explanation: There are no nice substrings.
Example 4:

Input: s = "dDzeE"
Output: "dD"
Explanation: Both "dD" and "eE" are the longest nice substrings.
As there are multiple longest nice substrings, return "dD" since it occurs earlier.

Constraints:

1 <= s.length <= 100
s consists of uppercase and lowercase English letters.

1764. Form Array by Concatenating Subarrays of Another Array

You are given a 2D integer array groups of length n. You are also given an integer array nums.
You are asked if you can choose n disjoint subarrays from the array nums such that the ith subarray is equal to group s[i] (0-indexed), and if i > 0, the (i-1)th subarray appears before the ith subarray in nums (i.e. the subarrays must be i n the same order as groups).
Return true if you can do this task, and false otherwise.
Note that the subarrays are disjoint if and only if there is no index k such that nums[k] belongs to more than one sub array. A subarray is a contiguous sequence of elements within an array.

Example 1:

Input: groups = [[1,-1,-1],[3,-2,0]], nums = [1,-1,0,1,-1,-1,3,-2,0]
Output: true
Explanation: You can choose the 0th subarray as [1,-1,0,1,-1,-1,3,-2,0] and the 1st one as [1,-1,0,1,-1,-1,3,-2,0].
These subarrays are disjoint as they share no common nums[k] element.

Example 2:

Input: groups = [[10,-2],[1,2,3,4]], nums = [1,2,3,4,10,-2]
Output: false
Explanation: Note that choosing the subarrays [1,2,3,4,10,-2] and [1,2,3,4,10,-2] is incorrect because they are not in t

he same order as in groups.
[10,-2] must come before [1,2,3,4].

Example 3:

Input: groups = [[1,2,3],[3,4]], nums = [7,7,1,2,3,4,7,7]
Output: false
Explanation: Note that choosing the subarrays [7,7,1,2,3,4,7,7] and [7,7,1,2,3,4,7,7] is invalid because they are not di
sjoint.
They share a common elements nums[4] (0-indexed).

Constraints:

groups.length == n
1 <= n <= 103
1 <= groups[i].length, sum(groups[i].length) <= 103
1 <= nums.length <= 103
-107 <= groups[i][j], nums[k] <= 107

1765. Map of Highest Peak

You are given an integer matrix isWater of size m x n that represents a map of land and water cells.

If isWater[i][j] == 0, cell (i, j) is a land cell.
If isWater[i][j] == 1, cell (i, j) is a water cell.

You must assign each cell a height in a way that follows these rules:

The height of each cell must be non-negative.
If the cell is a water cell, its height must be 0.
Any two adjacent cells must have an absolute height difference of at most 1. A cell is adjacent to another cell if the f
ormer is directly north, east, south, or west of the latter (i.e., their sides are touching).

Find an assignment of heights such that the maximum height in the matrix is maximized.
Return an integer matrix height of size m x n where height[i][j] is cell (i, j)'s height. If there are multiple solutions, re
turn any of them.

Example 1:

Input: isWater = [[0,1],[0,0]]
Output: [[1,0],[2,1]]
Explanation: The image shows the assigned heights of each cell.
The blue cell is the water cell, and the green cells are the land cells.

Example 2:

Input: isWater = [[0,0,1],[1,0,0],[0,0,0]]
Output: [[1,1,0],[0,1,1],[1,2,2]]
Explanation: A height of 2 is the maximum possible height of any assignment.
Any height assignment that has a maximum height of 2 while still meeting the rules will also be accepted.

Constraints:

m == isWater.length
n == isWater[i].length
1 <= m, n <= 1000
isWater[i][j] is 0 or 1.
There is at least one water cell.

1766. Tree of Coprimes

There is a tree (i.e., a connected, undirected graph that has no cycles) consisting of n nodes numbered from 0 to n - 1 and exactly n - 1 edges. Each node has a value associated with it, and the root of the tree is node 0.
To represent this tree, you are given an integer array nums and a 2D array edges. Each nums[i] represents the ith node's value, and each edges[j] = [uj, vj] represents an edge between nodes uj and vj in the tree.
Two values x and y are coprime if gcd(x, y) == 1 where gcd(x, y) is the greatest common divisor of x and y.
An ancestor of a node i is any other node on the shortest path from node i to the root. A node is not considered an ancestor of itself.
Return an array ans of size n, where ans[i] is the closest ancestor to node i such that nums[i] and nums[ans[i]] are coprime, or -1 if there is no such ancestor.

Example 1:

Input: nums = [2,3,3,2], edges = [[0,1],[1,2],[1,3]]
Output: [-1,0,0,1]
Explanation: In the above figure, each node's value is in parentheses.
- Node 0 has no coprime ancestors.
- Node 1 has only one ancestor, node 0. Their values are coprime (gcd(2,3) == 1).
- Node 2 has two ancestors, nodes 1 and 0. Node 1's value is not coprime (gcd(3,3) == 3), but node 0's value is (gcd(2,3) == 1), so node 0 is the closest valid ancestor.
- Node 3 has two ancestors, nodes 1 and 0. It is coprime with node 1 (gcd(3,2) == 1), so node 1 is its closest valid ancestor.

Example 2:

Input: nums = [5,6,10,2,3,6,15], edges = [[0,1],[0,2],[1,3],[1,4],[2,5],[2,6]]

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

Constraints:

nums.length == n
1 <= nums[i] <= 50
1 <= n <= 105
edges.length == n - 1
edges[j].length == 2
0 <= uj, vj < n
uj != vj

1768. Merge Strings Alternately

You are given two strings word1 and word2. Merge the strings by adding letters in alternating order, starting with w ord1. 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.

1769. Minimum Number of Operations to Move All Balls to Each Box

You have n boxes. You are given a binary string boxes of length n, where boxes[i] is '0' if the ith box is empty, and '1' if it contains one ball.
In one operation, you can move one ball from a box to an adjacent box. Box i is adjacent to box j if abs(i - j) == 1. Note that after doing so, there may be more than one ball in some boxes.
Return an array answer of size n, where answer[i] is the minimum number of operations needed to move all the balls to the ith box.
Each answer[i] is calculated considering the initial state of the boxes.

Example 1:

Input: boxes = "110"
Output: [1,1,3]
Explanation: The answer for each box is as follows:
1) First box: you will have to move one ball from the second box to the first box in one operation.
2) Second box: you will have to move one ball from the first box to the second box in one operation.
3) Third box: you will have to move one ball from the first box to the third box in two operations, and move one ball from the second box to the third box in one operation.

Example 2:

Input: boxes = "001011"
Output: [11,8,5,4,3,4]

Constraints:

n == boxes.length
1 <= n <= 2000
boxes[i] is either '0' or '1'.

1770. Maximum Score from Performing Multiplication Operations

You are given two integer arrays nums and multipliers of size n and m respectively, where n >= m. The arrays are 1-indexed.
You begin with a score of 0. You want to perform exactly m operations. On the ith operation (1-indexed), you will:

Choose one integer x from either the start or the end of the array nums.
Add multipliers[i] * x to your score.
Remove x from the array nums.

Return the maximum score after performing m operations.

Example 1:

Input: nums = [1,2,3], multipliers = [3,2,1]
Output: 14
Explanation: An optimal solution is as follows:
- Choose from the end, [1,2,3], adding 3 * 3 = 9 to the score.
- Choose from the end, [1,2], adding 2 * 2 = 4 to the score.
- Choose from the end, [1], adding 1 * 1 = 1 to the score.
The total score is 9 + 4 + 1 = 14.
Example 2:

Input: nums = [-5,-3,-3,-2,7,1], multipliers = [-10,-5,3,4,6]
Output: 102
Explanation: An optimal solution is as follows:
- Choose from the start, [-5,-3,-3,-2,7,1], adding -5 * -10 = 50 to the score.
- Choose from the start, [-3,-3,-2,7,1], adding -3 * -5 = 15 to the score.
- Choose from the start, [-3,-2,7,1], adding -3 * 3 = -9 to the score.
- Choose from the end, [-2,7,1], adding 1 * 4 = 4 to the score.
- Choose from the end, [-2,7], adding 7 * 6 = 42 to the score.
The total score is 50 + 15 - 9 + 4 + 42 = 102.

Constraints:

n == nums.length
m == multipliers.length
1 <= m <= 103
m <= n <= 105
-1000 <= nums[i], multipliers[i] <= 1000

1771. Maximize Palindrome Length From Subsequences

You are given two strings, word1 and word2. You want to construct a string in the following manner:

Choose some non-empty subsequence subsequence1 from word1.
Choose some non-empty subsequence subsequence2 from word2.

Concatenate the subsequences: subsequence1 + subsequence2, to make the string.

Return the length of the longest palindrome that can be constructed in the described manner. If no palindromes can b
e constructed, return 0.
A subsequence of a string s is a string that can be made by deleting some (possibly none) characters from s without c
hanging the order of the remaining characters.
A palindrome is a string that reads the same forward as well as backward.

Example 1:

Input: word1 = "cacb", word2 = "cbba"
Output: 5
Explanation: Choose "ab" from word1 and "cba" from word2 to make "abcba", which is a palindrome.
Example 2:

Input: word1 = "ab", word2 = "ab"
Output: 3
Explanation: Choose "ab" from word1 and "a" from word2 to make "aba", which is a palindrome.
Example 3:

Input: word1 = "aa", word2 = "bb"
Output: 0
Explanation: You cannot construct a palindrome from the described method, so return 0.

Constraints:

1 <= word1.length, word2.length <= 1000
word1 and word2 consist of lowercase English letters.

1773. Count Items Matching a Rule

You are given an array items, where each items[i] = [typei, colori, namei] describes the type, color, and name of the i
th item. You are also given a rule represented by two strings, ruleKey and ruleValue.
The ith item is said to match the rule if one of the following is true:

ruleKey == "type" and ruleValue == typei.
ruleKey == "color" and ruleValue == colori.
ruleKey == "name" and ruleValue == namei.

Return the number of items that match the given rule.

Example 1:

Input: items = [["phone","blue","pixel"],["computer","silver","lenovo"],["phone","gold","iphone"]], ruleKey = "colo
r", ruleValue = "silver"
Output: 1

Explanation: There is only one item matching the given rule, which is ["computer","silver","lenovo"].

Example 2:

Input: items = [["phone","blue","pixel"],["computer","silver","phone"],["phone","gold","iphone"]], ruleKey = "type", ruleValue = "phone"
Output: 2
Explanation: There are only two items matching the given rule, which are ["phone","blue","pixel"] and ["phone","gold","iphone"]. Note that the item ["computer","silver","phone"] does not match.

Constraints:

1 <= items.length <= 104
1 <= typei.length, colori.length, namei.length, ruleValue.length <= 10
ruleKey is equal to either "type", "color", or "name".
All strings consist only of lowercase letters.

1774. Closest Dessert Cost

You would like to make dessert and are preparing to buy the ingredients. You have n ice cream base flavors and m types of toppings to choose from. You must follow these rules when making your dessert:

There must be exactly one ice cream base.
You can add one or more types of topping or have no toppings at all.
There are at most two of each type of topping.

You are given three inputs:

baseCosts, an integer array of length n, where each baseCosts[i] represents the price of the ith ice cream base flavor.
toppingCosts, an integer array of length m, where each toppingCosts[i] is the price of one of the ith topping.
target, an integer representing your target price for dessert.

You want to make a dessert with a total cost as close to target as possible.
Return the closest possible cost of the dessert to target. If there are multiple, return the lower one.

Example 1:

Input: baseCosts = [1,7], toppingCosts = [3,4], target = 10
Output: 10
Explanation: Consider the following combination (all 0-indexed):
- Choose base 1: cost 7
- Take 1 of topping 0: cost 1 x 3 = 3
- Take 0 of topping 1: cost 0 x 4 = 0
Total: 7 + 3 + 0 = 10.

Example 2:

Input: baseCosts = [2,3], toppingCosts = [4,5,100], target = 18
Output: 17
Explanation: Consider the following combination (all 0-indexed):
- Choose base 1: cost 3
- Take 1 of topping 0: cost 1 x 4 = 4
- Take 2 of topping 1: cost 2 x 5 = 10
- Take 0 of topping 2: cost 0 x 100 = 0
Total: 3 + 4 + 10 + 0 = 17. You cannot make a dessert with a total cost of 18.

Example 3:

Input: baseCosts = [3,10], toppingCosts = [2,5], target = 9
Output: 8
Explanation: It is possible to make desserts with cost 8 and 10. Return 8 as it is the lower cost.

Example 4:

Input: baseCosts = [10], toppingCosts = [1], target = 1
Output: 10
Explanation: Notice that you don't have to have any toppings, but you must have exactly one base.

Constraints:

n == baseCosts.length
m == toppingCosts.length
1 <= n, m <= 10
1 <= baseCosts[i], toppingCosts[i] <= 104
1 <= target <= 104

1775. Equal Sum Arrays With Minimum Number of Operations

You are given two arrays of integers nums1 and nums2, possibly of different lengths. The values in the arrays are be tween 1 and 6, inclusive.
In one operation, you can change any integer's value in any of the arrays to any value between 1 and 6, inclusive.
Return the minimum number of operations required to make the sum of values in nums1 equal to the sum of values i n nums2. Return -1 if it is not possible to make the sum of the two arrays equal.

Example 1:

Input: nums1 = [1,2,3,4,5,6], nums2 = [1,1,2,2,2,2]
Output: 3
Explanation: You can make the sums of nums1 and nums2 equal with 3 operations. All indices are 0-indexed.
- Change nums2[0] to 6. nums1 = [1,2,3,4,5,6], nums2 = [6,1,2,2,2,2].
- Change nums1[5] to 1. nums1 = [1,2,3,4,5,1], nums2 = [6,1,2,2,2,2].
- Change nums1[2] to 2. nums1 = [1,2,2,4,5,1], nums2 = [6,1,2,2,2,2].

Example 2:

Input: nums1 = [1,1,1,1,1,1,1], nums2 = [6]
Output: -1
Explanation: There is no way to decrease the sum of nums1 or to increase the sum of nums2 to make them equal.

Example 3:

Input: nums1 = [6,6], nums2 = [1]
Output: 3
Explanation: You can make the sums of nums1 and nums2 equal with 3 operations. All indices are 0-indexed.
- Change nums1[0] to 2. nums1 = [2,6], nums2 = [1].
- Change nums1[1] to 2. nums1 = [2,2], nums2 = [1].
- Change nums2[0] to 4. nums1 = [2,2], nums2 = [4].

Constraints:

1 <= nums1.length, nums2.length <= 105
1 <= nums1[i], nums2[i] <= 6

1776. Car Fleet II

There are n cars traveling at different speeds in the same direction along a one-lane road. You are given an array cars of length n, where cars[i] = [positioni, speedi] represents:

positioni is the distance between the ith car and the beginning of the road in meters. It is guaranteed that positioni < positioni+1.
speedi is the initial speed of the ith car in meters per second.

For simplicity, cars can be considered as points moving along the number line. Two cars collide when they occupy t he same position. Once a car collides with another car, they unite and form a single car fleet. The cars in the formed fleet will have the same position and the same speed, which is the initial speed of the slowest car in the fleet.
Return an array answer, where answer[i] is the time, in seconds, at which the ith car collides with the next car, or -1 i f the car does not collide with the next car. Answers within 10-5 of the actual answers are accepted.

Example 1:

Input: cars = [[1,2],[2,1],[4,3],[7,2]]
Output: [1.00000,-1.00000,3.00000,-1.00000]
Explanation: After exactly one second, the first car will collide with the second car, and form a car fleet with speed 1 m/s. After exactly 3 seconds, the third car will collide with the fourth car, and form a car fleet with speed 2 m/s.

Example 2:

Input: cars = [[3,4],[5,4],[6,3],[9,1]]
Output: [2.00000,1.00000,1.50000,-1.00000]

Constraints:

1 <= cars.length <= 105
1 <= positioni, speedi <= 106
positioni < positioni+1

1779. Find Nearest Point That Has the Same X or Y Coordinate

You are given two integers, x and y, which represent your current location on a Cartesian grid: (x, y). You are also g iven an array points where each points[i] = [ai, bi] represents that a point exists at (ai, bi). A point is valid if it shares the same x-coordinate or the same y-coordinate as your location.
Return the index (0-indexed) of the valid point with the smallest Manhattan distance from your current location. If th ere are multiple, return the valid point with the smallest index. If there are no valid points, return -1.
The Manhattan distance between two points (x1, y1) and (x2, y2) is abs(x1 - x2) + abs(y1 - y2).

Example 1:

Input: x = 3, y = 4, points = [[1,2],[3,1],[2,4],[2,3],[4,4]]
Output: 2
Explanation: Of all the points, only [3,1], [2,4] and [4,4] are valid. Of the valid points, [2,4] and [4,4] have the small est Manhattan distance from your current location, with a distance of 1. [2,4] has the smallest index, so return 2.
Example 2:

Input: x = 3, y = 4, points = [[3,4]]
Output: 0
Explanation: The answer is allowed to be on the same location as your current location.
Example 3:

Input: x = 3, y = 4, points = [[2,3]]
Output: -1
Explanation: There are no valid points.

Constraints:

1 <= points.length <= 104
points[i].length == 2
1 <= x, y, ai, bi <= 104

1780. Check if Number is a Sum of Powers of Three

Given an integer n, return true if it is possible to represent n as the sum of distinct powers of three. Otherwise, return false.
An integer y is a power of three if there exists an integer x such that y == 3x.

Example 1:

Input: n = 12
Output: true
Explanation: 12 = 31 + 32

Example 2:

Input: n = 91
Output: true
Explanation: 91 = 30 + 32 + 34

Example 3:

Input: n = 21
Output: false

Constraints:

1 <= n <= 107

1781. Sum of Beauty of All Substrings

The beauty of a string is the difference in frequencies between the most frequent and least frequent characters.

For example, the beauty of "abaacc" is 3 - 1 = 2.

Given a string s, return the sum of beauty of all of its substrings.

Example 1:

Input: s = "aabcb"
Output: 5
Explanation: The substrings with non-zero beauty are ["aab","aabc","aabcb","abcb","bcb"], each with beauty equal t
o 1.

Example 2:

Input: s = "aabcbaa"
Output: 17

Constraints:

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

1782. Count Pairs Of Nodes

You are given an undirected graph defined by an integer n, the number of nodes, and a 2D integer array edges, the e dges in the graph, where edges[i] = [ui, vi] indicates that there is an undirected edge between ui and vi. You are also given an integer array queries.
Let incident(a, b) be defined as the number of edges that are connected to either node a or b.
The answer to the jth query is the number of pairs of nodes (a, b) that satisfy both of the following conditions:

a < b
incident(a, b) > queries[j]

Return an array answers such that answers.length == queries.length and answers[j] is the answer of the jth query.
Note that there can be multiple edges between the same two nodes.

Example 1:

Input: n = 4, edges = [[1,2],[2,4],[1,3],[2,3],[2,1]], queries = [2,3]
Output: [6,5]
Explanation: The calculations for incident(a, b) are shown in the table above.
The answers for each of the queries are as follows:
- answers[0] = 6. All the pairs have an incident(a, b) value greater than 2.
- answers[1] = 5. All the pairs except (3, 4) have an incident(a, b) value greater than 3.

Example 2:

Input: n = 5, edges = [[1,5],[1,5],[3,4],[2,5],[1,3],[5,1],[2,3],[2,5]], queries = [1,2,3,4,5]
Output: [10,10,9,8,6]

Constraints:

2 <= n <= 2 * 104
1 <= edges.length <= 105
1 <= ui, vi <= n

ui != vi
1 <= queries.length <= 20
0 <= queries[j] < edges.length

1784. Check if Binary String Has at Most One Segment of Ones

Given a binary string s without leading zeros, return true if s contains at most one contiguous segment of ones. Otherwise, return false.

Example 1:

Input: s = "1001"
Output: false
Explanation: The ones do not form a contiguous segment.

Example 2:

Input: s = "110"
Output: true

Constraints:

1 <= s.length <= 100
s[i] is either '0' or '1'.
s[0] is '1'.

1785. Minimum Elements to Add to Form a Given Sum

You are given an integer array nums and two integers limit and goal. The array nums has an interesting property that abs(nums[i]) <= limit.
Return the minimum number of elements you need to add to make the sum of the array equal to goal. The array must maintain its property that abs(nums[i]) <= limit.
Note that abs(x) equals x if x >= 0, and -x otherwise.

Example 1:

Input: nums = [1,-1,1], limit = 3, goal = -4
Output: 2

Explanation: You can add -2 and -3, then the sum of the array will be 1 - 1 + 1 - 2 - 3 = -4.

Example 2:

Input: nums = [1,-10,9,1], limit = 100, goal = 0
Output: 1

Constraints:

1 <= nums.length <= 105
1 <= limit <= 106
-limit <= nums[i] <= limit
-109 <= goal <= 109

1786. Number of Restricted Paths From First to Last Node

There is an undirected weighted connected graph. You are given a positive integer n which denotes that the graph ha s n nodes labeled from 1 to n, and an array edges where each edges[i] = [ui, vi, weighti] denotes that there is an edge between nodes ui and vi with weight equal to weighti.
A path from node start to node end is a sequence of nodes [z0, z1, z2, ..., zk] such that z0 = start and zk = end and th ere is an edge between zi and zi+1 where 0 <= i <= k-1.
The distance of a path is the sum of the weights on the edges of the path. Let distanceToLastNode(x) denote the shor test distance of a path between node n and node x. A restricted path is a path that also satisfies that distanceToLastN ode(zi) > distanceToLastNode(zi+1) where 0 <= i <= k-1.
Return the number of restricted paths from node 1 to node n. Since that number may be too large, return it modulo 1 09 + 7.

Example 1:

Input: n = 5, edges = [[1,2,3],[1,3,3],[2,3,1],[1,4,2],[5,2,2],[3,5,1],[5,4,10]]
Output: 3
Explanation: Each circle contains the node number in black and its distanceToLastNode value in blue. The three rest ricted paths are:
1) 1 --> 2 --> 5
2) 1 --> 2 --> 3 --> 5
3) 1 --> 3 --> 5

Example 2:

Input: n = 7, edges = [[1,3,1],[4,1,2],[7,3,4],[2,5,3],[5,6,1],[6,7,2],[7,5,3],[2,6,4]]
Output: 1
Explanation: Each circle contains the node number in black and its distanceToLastNode value in blue. The only restr icted path is 1 --> 3 --> 7.

Constraints:

1 <= n <= 2 * 104
n - 1 <= edges.length <= 4 * 104
edges[i].length == 3
1 <= ui, vi <= n
ui != vi
1 <= weighti <= 105
There is at most one edge between any two nodes.
There is at least one path between any two nodes.

1787. Make the XOR of All Segments Equal to Zero

You are given an array nums and an integer k . The XOR of a segment [left, right] where left <= right is the XOR of all the elements with indices between left and right, inclusive: nums[left] XOR nums[left+1] XOR ... XOR nums[right].
Return the minimum number of elements to change in the array such that the XOR of all segments of size k is equal to zero.

Example 1:

Input: nums = [1,2,0,3,0], k = 1
Output: 3
Explanation: Modify the array from [1,2,0,3,0] to from [0,0,0,0,0].

Example 2:

Input: nums = [3,4,5,2,1,7,3,4,7], k = 3
Output: 3
Explanation: Modify the array from [3,4,5,2,1,7,3,4,7] to [3,4,7,3,4,7,3,4,7].

Example 3:

Input: nums = [1,2,4,1,2,5,1,2,6], k = 3
Output: 3
Explanation: Modify the array from [1,2,4,1,2,5,1,2,6] to [1,2,3,1,2,3,1,2,3].

Constraints:

1 <= k <= nums.length <= 2000
0 <= nums[i] < 210

1790. Check if One String Swap Can Make Strings Equal

You are given two strings s1 and s2 of equal length. A string swap is an operation where you choose two indices in a string (not necessarily different) and swap the characters at these indices.
Return true if it is possible to make both strings equal by performing at most one string swap on exactly one of the st rings. Otherwise, return false.

Example 1:

Input: s1 = "bank", s2 = "kanb"
Output: true
Explanation: For example, swap the first character with the last character of s2 to make "bank".

Example 2:

Input: s1 = "attack", s2 = "defend"
Output: false
Explanation: It is impossible to make them equal with one string swap.

Example 3:

Input: s1 = "kelb", s2 = "kelb"
Output: true
Explanation: The two strings are already equal, so no string swap operation is required.

Example 4:

Input: s1 = "abcd", s2 = "dcba"
Output: false

Constraints:

1 <= s1.length, s2.length <= 100
s1.length == s2.length
s1 and s2 consist of only lowercase English letters.

1791. Find Center of Star Graph

There is an undirected star graph consisting of n nodes labeled from 1 to n. A star graph is a graph where there is one

center node and exactly n - 1 edges that connect the center node with every other node.
You are given a 2D integer array edges where each edges[i] = [ui, vi] indicates that there is an edge between the nod es ui and vi. Return the center of the given star graph.

Example 1:

Input: edges = [[1,2],[2,3],[4,2]]
Output: 2
Explanation: As shown in the figure above, node 2 is connected to every other node, so 2 is the center.

Example 2:

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

Constraints:

3 <= n <= 105
edges.length == n - 1
edges[i].length == 2
1 <= ui, vi <= n
ui != vi
The given edges represent a valid star graph.

1792. Maximum Average Pass Ratio

There is a school that has classes of students and each class will be having a final exam. You are given a 2D integer array classes, where classes[i] = [passi, totali]. You know beforehand that in the ith class, there are totali total studen ts, but only passi number of students will pass the exam.
You are also given an integer extraStudents. There are another extraStudents brilliant students that are guaranteed to pass the exam of any class they are assigned to. You want to assign each of the extraStudents students to a class in a way that maximizes the average pass ratio across all the classes.
The pass ratio of a class is equal to the number of students of the class that will pass the exam divided by the total nu mber of students of the class. The average pass ratio is the sum of pass ratios of all the classes divided by the number of the classes.
Return the maximum possible average pass ratio after assigning the extraStudents students. Answers within 10-5 of t he actual answer will be accepted.

Example 1:

Input: classes = [[1,2],[3,5],[2,2]], extraStudents = 2
Output: 0.78333
Explanation: You can assign the two extra students to the first class. The average pass ratio will be equal to (3/4 + 3/ 5 + 2/2) / 3 = 0.78333.

Example 2:

Input: classes = [[2,4],[3,9],[4,5],[2,10]], extraStudents = 4
Output: 0.53485

Constraints:

1 <= classes.length <= 105
classes[i].length == 2
1 <= passi <= totali <= 105
1 <= extraStudents <= 105

1793. Maximum Score of a Good Subarray

You are given an array of integers nums (0-indexed) and an integer k.
The score of a subarray (i, j) is defined as min(nums[i], nums[i+1], ..., nums[j]) * (j - i + 1). A good subarray is a sub array where i <= k <= j.
Return the maximum possible score of a good subarray.

Example 1:

Input: nums = [1,4,3,7,4,5], k = 3
Output: 15
Explanation: The optimal subarray is (1, 5) with a score of min(4,3,7,4,5) * (5-1+1) = 3 * 5 = 15.

Example 2:

Input: nums = [5,5,4,5,4,1,1,1], k = 0
Output: 20
Explanation: The optimal subarray is (0, 4) with a score of min(5,5,4,5,4) * (4-0+1) = 4 * 5 = 20.

Constraints:

1 <= nums.length <= 105
1 <= nums[i] <= 2 * 104
0 <= k < nums.length

1796. Second Largest Digit in a String

Given an alphanumeric string s, return the second largest numerical digit that appears in s, or -1 if it does not exist.
An alphanumeric string is a string consisting of lowercase English letters and digits.

Example 1:

Input: s = "dfa12321afd"
Output: 2
Explanation: The digits that appear in s are [1, 2, 3]. The second largest digit is 2.

Example 2:

Input: s = "abc1111"
Output: -1
Explanation: The digits that appear in s are [1]. There is no second largest digit.

Constraints:

1 <= s.length <= 500
s consists of only lowercase English letters and/or digits.

1797. Design Authentication Manager

There is an authentication system that works with authentication tokens. For each session, the user will receive a ne
w authentication token that will expire timeToLive seconds after the currentTime. If the token is renewed, the expiry
time will be extended to expire timeToLive seconds after the (potentially different) currentTime.
Implement the AuthenticationManager class:

AuthenticationManager(int timeToLive) constructs the AuthenticationManager and sets the timeToLive.
generate(string tokenId, int currentTime) generates a new token with the given tokenId at the given currentTime in s
econds.
renew(string tokenId, int currentTime) renews the unexpired token with the given tokenId at the given currentTime i
n seconds. If there are no unexpired tokens with the given tokenId, the request is ignored, and nothing happens.
countUnexpiredTokens(int currentTime) returns the number of unexpired tokens at the given currentTime.

Note that if a token expires at time t, and another action happens on time t (renew or countUnexpiredTokens), the ex
piration takes place before the other actions.

Example 1:

Input

["AuthenticationManager", "renew", "generate", "countUnexpiredTokens", "generate", "renew", "renew", "countUnexpiredTokens"]
[[5], ["aaa", 1], ["aaa", 2], [6], ["bbb", 7], ["aaa", 8], ["bbb", 10], [15]]
Output
[null, null, null, 1, null, null, null, 0]

Explanation
AuthenticationManager authenticationManager = new AuthenticationManager(5); // Constructs the AuthenticationManager with timeToLive = 5 seconds.
authenticationManager.renew("aaa", 1); // No token exists with tokenId "aaa" at time 1, so nothing happens.
authenticationManager.generate("aaa", 2); // Generates a new token with tokenId "aaa" at time 2.
authenticationManager.countUnexpiredTokens(6); // The token with tokenId "aaa" is the only unexpired one at time 6, so return 1.
authenticationManager.generate("bbb", 7); // Generates a new token with tokenId "bbb" at time 7.
authenticationManager.renew("aaa", 8); // The token with tokenId "aaa" expired at time 7, and 8 >= 7, so at time 8 the renew request is ignored, and nothing happens.
authenticationManager.renew("bbb", 10); // The token with tokenId "bbb" is unexpired at time 10, so the renew request is fulfilled and now the token will expire at time 15.
authenticationManager.countUnexpiredTokens(15); // The token with tokenId "bbb" expires at time 15, and the token with tokenId "aaa" expired at time 7, so currently no token is unexpired, so return 0.

Constraints:

1 <= timeToLive <= 108
1 <= currentTime <= 108
1 <= tokenId.length <= 5
tokenId consists only of lowercase letters.
All calls to generate will contain unique values of tokenId.
The values of currentTime across all the function calls will be strictly increasing.
At most 2000 calls will be made to all functions combined.

1798. Maximum Number of Consecutive Values You Can Make

You are given an integer array coins of length n which represents the n coins that you own. The value of the ith coin is coins[i]. You can make some value x if you can choose some of your n coins such that their values sum up to x.
Return the maximum number of consecutive integer values that you can make with your coins starting from and including 0.
Note that you may have multiple coins of the same value.

Example 1:

Input: coins = [1,3]
Output: 2
Explanation: You can make the following values:
- 0: take []

- 1: take [1]
You can make 2 consecutive integer values starting from 0.
Example 2:

Input: coins = [1,1,1,4]
Output: 8
Explanation: You can make the following values:
- 0: take []
- 1: take [1]
- 2: take [1,1]
- 3: take [1,1,1]
- 4: take [4]
- 5: take [4,1]
- 6: take [4,1,1]
- 7: take [4,1,1,1]
You can make 8 consecutive integer values starting from 0.
Example 3:

Input: nums = [1,4,10,3,1]
Output: 20

Constraints:

coins.length == n
1 <= n <= 4 * 104
1 <= coins[i] <= 4 * 104

1799. Maximize Score After N Operations

You are given nums, an array of positive integers of size 2 * n. You must perform n operations on this array.
In the ith operation (1-indexed), you will:

Choose two elements, x and y.
Receive a score of i * gcd(x, y).
Remove x and y from nums.

Return the maximum score you can receive after performing n operations.
The function gcd(x, y) is the greatest common divisor of x and y.

Example 1:

Input: nums = [1,2]
Output: 1
Explanation: The optimal choice of operations is:
(1 * gcd(1, 2)) = 1

Example 2:

Input: nums = [3,4,6,8]
Output: 11
Explanation: The optimal choice of operations is:
(1 * gcd(3, 6)) + (2 * gcd(4, 8)) = 3 + 8 = 11

Example 3:

Input: nums = [1,2,3,4,5,6]
Output: 14
Explanation: The optimal choice of operations is:
(1 * gcd(1, 5)) + (2 * gcd(2, 4)) + (3 * gcd(3, 6)) = 1 + 4 + 9 = 14

Constraints:

1 <= n <= 7
nums.length == 2 * n
1 <= nums[i] <= 106

1800. Maximum Ascending Subarray Sum

Given an array of positive integers nums, return the maximum possible sum of an ascending subarray in nums.
A subarray is defined as a contiguous sequence of numbers in an array.
A subarray [numsl, numsl+1, ..., numsr-1, numsr] is ascending if for all i where l <= i < r, numsi < numsi+1. Note th at a subarray of size 1 is ascending.

Example 1:

Input: nums = [10,20,30,5,10,50]
Output: 65
Explanation: [5,10,50] is the ascending subarray with the maximum sum of 65.

Example 2:

Input: nums = [10,20,30,40,50]
Output: 150
Explanation: [10,20,30,40,50] is the ascending subarray with the maximum sum of 150.

Example 3:

Input: nums = [12,17,15,13,10,11,12]
Output: 33
Explanation: [10,11,12] is the ascending subarray with the maximum sum of 33.

Example 4:

Input: nums = [100,10,1]
Output: 100

Constraints:

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

1801. Number of Orders in the Backlog

You are given a 2D integer array orders, where each orders[i] = [pricei, amounti, orderTypei] denotes that amounti o rders have been placed of type orderTypei at the price pricei. The orderTypei is:

0 if it is a batch of buy orders, or
1 if it is a batch of sell orders.

Note that orders[i] represents a batch of amounti independent orders with the same price and order type. All orders r epresented by orders[i] will be placed before all orders represented by orders[i+1] for all valid i.
There is a backlog that consists of orders that have not been executed. The backlog is initially empty. When an order is placed, the following happens:

If the order is a buy order, you look at the sell order with the smallest price in the backlog. If that sell order's price is smaller than or equal to the current buy order's price, they will match and be executed, and that sell order will be rem oved from the backlog. Else, the buy order is added to the backlog.
Vice versa, if the order is a sell order, you look at the buy order with the largest price in the backlog. If that buy orde r's price is larger than or equal to the current sell order's price, they will match and be executed, and that buy order w ill be removed from the backlog. Else, the sell order is added to the backlog.

Return the total amount of orders in the backlog after placing all the orders from the input. Since this number can be large, return it modulo 109 + 7.

Example 1:

Input: orders = [[10,5,0],[15,2,1],[25,1,1],[30,4,0]]
Output: 6
Explanation: Here is what happens with the orders:
- 5 orders of type buy with price 10 are placed. There are no sell orders, so the 5 orders are added to the backlog.
- 2 orders of type sell with price 15 are placed. There are no buy orders with prices larger than or equal to 15, so the 2 orders are added to the backlog.
- 1 order of type sell with price 25 is placed. There are no buy orders with prices larger than or equal to 25 in the bac klog, so this order is added to the backlog.
- 4 orders of type buy with price 30 are placed. The first 2 orders are matched with the 2 sell orders of the least price,

which is 15 and these 2 sell orders are removed from the backlog. The 3rd order is matched with the sell order of th e least price, which is 25 and this sell order is removed from the backlog. Then, there are no more sell orders in the b acklog, so the 4th order is added to the backlog.
Finally, the backlog has 5 buy orders with price 10, and 1 buy order with price 30. So the total number of orders in t he backlog is 6.

Example 2:

Input: orders = [[7,1000000000,1],[15,3,0],[5,999999995,0],[5,1,1]]
Output: 999999984
Explanation: Here is what happens with the orders:
- 109 orders of type sell with price 7 are placed. There are no buy orders, so the 109 orders are added to the backlog.
- 3 orders of type buy with price 15 are placed. They are matched with the 3 sell orders with the least price which is 7, and those 3 sell orders are removed from the backlog.
- 999999995 orders of type buy with price 5 are placed. The least price of a sell order is 7, so the 999999995 orders are added to the backlog.
- 1 order of type sell with price 5 is placed. It is matched with the buy order of the highest price, which is 5, and that buy order is removed from the backlog.
Finally, the backlog has (1000000000-3) sell orders with price 7, and (999999995-1) buy orders with price 5. So the total number of orders = 1999999991, which is equal to 999999984 % (109 + 7).

Constraints:

1 <= orders.length <= 105
orders[i].length == 3
1 <= pricei, amounti <= 109
orderTypei is either 0 or 1.

1802. Maximum Value at a Given Index in a Bounded Array

You are given three positive integers: n, index, and maxSum. You want to construct an array nums (0-indexed) that s atisfies the following conditions:

nums.length == n
nums[i] is a positive integer where 0 <= i < n.
abs(nums[i] - nums[i+1]) <= 1 where 0 <= i < n-1.
The sum of all the elements of nums does not exceed maxSum.
nums[index] is maximized.

Return nums[index] of the constructed array.
Note that abs(x) equals x if x >= 0, and -x otherwise.

Example 1:

Input: n = 4, index = 2, maxSum = 6

Output: 2
Explanation: nums = [1,2,2,1] is one array that satisfies all the conditions.
There are no arrays that satisfy all the conditions and have nums[2] == 3, so 2 is the maximum nums[2].

Example 2:

Input: n = 6, index = 1, maxSum = 10
Output: 3

Constraints:

1 <= n <= maxSum <= 109
0 <= index < n

1803. Count Pairs With XOR in a Range

Given a (0-indexed) integer array nums and two integers low and high, return the number of nice pairs.
A nice pair is a pair (i, j) where 0 <= i < j < nums.length and low <= (nums[i] XOR nums[j]) <= high.

Example 1:

Input: nums = [1,4,2,7], low = 2, high = 6
Output: 6
Explanation: All nice pairs (i, j) are as follows:
- (0, 1): nums[0] XOR nums[1] = 5
- (0, 2): nums[0] XOR nums[2] = 3
- (0, 3): nums[0] XOR nums[3] = 6
- (1, 2): nums[1] XOR nums[2] = 6
- (1, 3): nums[1] XOR nums[3] = 3
- (2, 3): nums[2] XOR nums[3] = 5

Example 2:

Input: nums = [9,8,4,2,1], low = 5, high = 14
Output: 8
Explanation: All nice pairs (i, j) are as follows:
- (0, 2): nums[0] XOR nums[2] = 13
- (0, 3): nums[0] XOR nums[3] = 11
- (0, 4): nums[0] XOR nums[4] = 8
- (1, 2): nums[1] XOR nums[2] = 12
- (1, 3): nums[1] XOR nums[3] = 10
- (1, 4): nums[1] XOR nums[4] = 9
- (2, 3): nums[2] XOR nums[3] = 6
- (2, 4): nums[2] XOR nums[4] = 5

Constraints:

1 <= nums.length <= 2 * 104
1 <= nums[i] <= 2 * 104
1 <= low <= high <= 2 * 104

1805. Number of Different Integers in a String

You are given a string word that consists of digits and lowercase English letters.
You will replace every non-digit character with a space. For example, "a123bc34d8ef34" will become " 123 34 8 3 4". Notice that you are left with some integers that are separated by at least one space: "123", "34", "8", and "34".
Return the number of different integers after performing the replacement operations on word.
Two integers are considered different if their decimal representations without any leading zeros are different.

Example 1:

Input: word = "a123bc34d8ef34"
Output: 3
Explanation: The three different integers are "123", "34", and "8". Notice that "34" is only counted once.

Example 2:

Input: word = "leet1234code234"
Output: 2

Example 3:

Input: word = "a1b01c001"
Output: 1
Explanation: The three integers "1", "01", and "001" all represent the same integer because
the leading zeros are ignored when comparing their decimal values.

Constraints:

1 <= word.length <= 1000
word consists of digits and lowercase English letters.

1806. Minimum Number of Operations to Reinitialize a Permutation

You are given an even integer n . You initially have a permutation perm of size n where perm[i] == i (0-indexed) .
In one operation, you will create a new array arr, and for each i:

If i % 2 == 0, then arr[i] = perm[i / 2].
If i % 2 == 1, then arr[i] = perm[n / 2 + (i - 1) / 2].

You will then assign arr to perm.
Return the minimum non-zero number of operations you need to perform on perm to return the permutation to its initial value.

Example 1:

Input: n = 2
Output: 1
Explanation: perm = [0,1] initially.
After the 1st operation, perm = [0,1]
So it takes only 1 operation.

Example 2:

Input: n = 4
Output: 2
Explanation: perm = [0,1,2,3] initially.
After the 1st operation, perm = [0,2,1,3]
After the 2nd operation, perm = [0,1,2,3]
So it takes only 2 operations.

Example 3:

Input: n = 6
Output: 4

Constraints:

2 <= n <= 1000
n is even.

1807. Evaluate the Bracket Pairs of a String

You are given a string s that contains some bracket pairs, with each pair containing a non-empty key.

For example, in the string "(name)is(age)yearsold", there are two bracket pairs that contain the keys "name" and "age".

You know the values of a wide range of keys. This is represented by a 2D string array knowledge where each knowl edge[i] = [keyi, valuei] indicates that key keyi has a value of valuei.
You are tasked to evaluate all of the bracket pairs. When you evaluate a bracket pair that contains some key keyi, yo u will:

Replace keyi and the bracket pair with the key's corresponding valuei.
If you do not know the value of the key, you will replace keyi and the bracket pair with a question mark "?" (without the quotation marks).

Each key will appear at most once in your knowledge. There will not be any nested brackets in s.
Return the resulting string after evaluating all of the bracket pairs.

Example 1:

Input: s = "(name)is(age)yearsold", knowledge = [["name","bob"],["age","two"]]
Output: "bobistwoyearsold"
Explanation:
The key "name" has a value of "bob", so replace "(name)" with "bob".
The key "age" has a value of "two", so replace "(age)" with "two".

Example 2:

Input: s = "hi(name)", knowledge = [["a","b"]]
Output: "hi?"
Explanation: As you do not know the value of the key "name", replace "(name)" with "?".

Example 3:

Input: s = "(a)(a)(a)aaa", knowledge = [["a","yes"]]
Output: "yesyesyesaaa"
Explanation: The same key can appear multiple times.
The key "a" has a value of "yes", so replace all occurrences of "(a)" with "yes".
Notice that the "a"s not in a bracket pair are not evaluated.

Example 4:

Input: s = "(a)(b)", knowledge = [["a","b"],["b","a"]]
Output: "ba"

Constraints:

1 <= s.length <= 105
0 <= knowledge.length <= 105
knowledge[i].length == 2
1 <= keyi.length, valuei.length <= 10
s consists of lowercase English letters and round brackets '(' and ')'.
Every open bracket '(' in s will have a corresponding close bracket ')'.
The key in each bracket pair of s will be non-empty.
There will not be any nested bracket pairs in s.
keyi and valuei consist of lowercase English letters.
Each keyi in knowledge is unique.

1808. Maximize Number of Nice Divisors

You are given a positive integer primeFactors. You are asked to construct a positive integer n that satisfies the follo wing conditions:

The number of prime factors of n (not necessarily distinct) is at most primeFactors.
The number of nice divisors of n is maximized. Note that a divisor of n is nice if it is divisible by every prime factor of n. For example, if n = 12, then its prime factors are [2,2,3], then 6 and 12 are nice divisors, while 3 and 4 are not.

Return the number of nice divisors of n. Since that number can be too large, return it modulo 109 + 7.
Note that a prime number is a natural number greater than 1 that is not a product of two smaller natural numbers. Th e prime factors of a number n is a list of prime numbers such that their product equals n.

Example 1:

Input: primeFactors = 5
Output: 6
Explanation: 200 is a valid value of n.
It has 5 prime factors: [2,2,2,5,5], and it has 6 nice divisors: [10,20,40,50,100,200].
There is not other value of n that has at most 5 prime factors and more nice divisors.

Example 2:

Input: primeFactors = 8
Output: 18

Constraints:

1 <= primeFactors <= 109

1812. Determine Color of a Chessboard Square

You are given coordinates, a string that represents the coordinates of a square of the chessboard. Below is a chessboa rd for your reference.

Return true if the square is white, and false if the square is black.
The coordinate will always represent a valid chessboard square. The coordinate will always have the letter first, and t he number second.

Example 1:

Input: coordinates = "a1"
Output: false
Explanation: From the chessboard above, the square with coordinates "a1" is black, so return false.

Example 2:

Input: coordinates = "h3"
Output: true
Explanation: From the chessboard above, the square with coordinates "h3" is white, so return true.

Example 3:

Input: coordinates = "c7"
Output: false

Constraints:

coordinates.length == 2
'a' <= coordinates[0] <= 'h'
'1' <= coordinates[1] <= '8'

1813. Sentence Similarity III

A sentence is a list of words that are separated by a single space with no leading or trailing spaces. For example, "He llo World", "HELLO", "hello world hello world" are all sentences. Words consist of only uppercase and lowercase E nglish letters.
Two sentences sentence1 and sentence2 are similar if it is possible to insert an arbitrary sentence (possibly empty) in side one of these sentences such that the two sentences become equal. For example, sentence1 = "Hello my name is J ane" and sentence2 = "Hello Jane" can be made equal by inserting "my name is" between "Hello" and "Jane" in sent ence2.
Given two sentences sentence1 and sentence2, return true if sentence1 and sentence2 are similar. Otherwise, return f alse.

Example 1:

Input: sentence1 = "My name is Haley", sentence2 = "My Haley"
Output: true
Explanation: sentence2 can be turned to sentence1 by inserting "name is" between "My" and "Haley".

Example 2:

Input: sentence1 = "of", sentence2 = "A lot of words"
Output: false

Explanation: No single sentence can be inserted inside one of the sentences to make it equal to the other.

Example 3:

Input: sentence1 = "Eating right now", sentence2 = "Eating"
Output: true
Explanation: sentence2 can be turned to sentence1 by inserting "right now" at the end of the sentence.

Example 4:

Input: sentence1 = "Luky", sentence2 = "Lucccky"
Output: false

Constraints:

1 <= sentence1.length, sentence2.length <= 100
sentence1 and sentence2 consist of lowercase and uppercase English letters and spaces.
The words in sentence1 and sentence2 are separated by a single space.

1814. Count Nice Pairs in an Array

You are given an array nums that consists of non-negative integers. Let us define rev(x) as the reverse of the non-ne gative integer x. For example, rev(123) = 321, and rev(120) = 21. A pair of indices (i, j) is nice if it satisfies all of th e following conditions:

0 <= i < j < nums.length
nums[i] + rev(nums[j]) == nums[j] + rev(nums[i])

Return the number of nice pairs of indices. Since that number can be too large, return it modulo 109 + 7.

Example 1:

Input: nums = [42,11,1,97]
Output: 2
Explanation: The two pairs are:
 - (0,3) : 42 + rev(97) = 42 + 79 = 121, 97 + rev(42) = 97 + 24 = 121.
 - (1,2) : 11 + rev(1) = 11 + 1 = 12, 1 + rev(11) = 1 + 11 = 12.

Example 2:

Input: nums = [13,10,35,24,76]
Output: 4

Constraints:

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

1815. Maximum Number of Groups Getting Fresh Donuts

There is a donuts shop that bakes donuts in batches of batchSize. They have a rule where they must serve all of the d onuts of a batch before serving any donuts of the next batch. You are given an integer batchSize and an integer array groups, where groups[i] denotes that there is a group of groups[i] customers that will visit the shop. Each customer will get exactly one donut.
When a group visits the shop, all customers of the group must be served before serving any of the following groups. A group will be happy if they all get fresh donuts. That is, the first customer of the group does not receive a donut th at was left over from the previous group.
You can freely rearrange the ordering of the groups. Return the maximum possible number of happy groups after rea rranging the groups.

Example 1:

Input: batchSize = 3, groups = [1,2,3,4,5,6]
Output: 4
Explanation: You can arrange the groups as [6,2,4,5,1,3]. Then the 1st, 2nd, 4th, and 6th groups will be happy.

Example 2:

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

Constraints:

1 <= batchSize <= 9
1 <= groups.length <= 30
1 <= groups[i] <= 109

1816. Truncate Sentence

A sentence is a list of words that are separated by a single space with no leading or trailing spaces. Each of the words consists of only uppercase and lowercase English letters (no punctuation).

For example, "Hello World", "HELLO", and "hello world hello world" are all sentences.

You are given a sentence s and an integer k . You want to truncate s such that it contains only the first k words. Return s after truncating it.

Example 1:

Input: s = "Hello how are you Contestant", k = 4
Output: "Hello how are you"
Explanation:
The words in s are ["Hello", "how" "are", "you", "Contestant"].
The first 4 words are ["Hello", "how", "are", "you"].
Hence, you should return "Hello how are you".

Example 2:

Input: s = "What is the solution to this problem", k = 4
Output: "What is the solution"
Explanation:
The words in s are ["What", "is" "the", "solution", "to", "this", "problem"].
The first 4 words are ["What", "is", "the", "solution"].
Hence, you should return "What is the solution".
Example 3:

Input: s = "chopper is not a tanuki", k = 5
Output: "chopper is not a tanuki"

Constraints:

1 <= s.length <= 500
k is in the range [1, the number of words in s].
s consist of only lowercase and uppercase English letters and spaces.
The words in s are separated by a single space.
There are no leading or trailing spaces.

1817. Finding the Users Active Minutes

You are given the logs for users' actions on LeetCode, and an integer k. The logs are represented by a 2D integer arr ay logs where each logs[i] = [IDi, timei] indicates that the user with IDi performed an action at the minute timei. Multiple users can perform actions simultaneously, and a single user can perform multiple actions in the same minut e.
The user active minutes (UAM) for a given user is defined as the number of unique minutes in which the user perfor med an action on LeetCode. A minute can only be counted once, even if multiple actions occur during it.
You are to calculate a 1-indexed array answer of size k such that, for each j (1 <= j <= k), answer[j] is the number of

users whose UAM equals j.
Return the array answer as described above.

Example 1:

Input: logs = [[0,5],[1,2],[0,2],[0,5],[1,3]], k = 5
Output: [0,2,0,0,0]
Explanation:
The user with ID=0 performed actions at minutes 5, 2, and 5 again. Hence, they have a UAM of 2 (minute 5 is only counted once).
The user with ID=1 performed actions at minutes 2 and 3. Hence, they have a UAM of 2.
Since both users have a UAM of 2, answer[2] is 2, and the remaining answer[j] values are 0.

Example 2:

Input: logs = [[1,1],[2,2],[2,3]], k = 4
Output: [1,1,0,0]
Explanation:
The user with ID=1 performed a single action at minute 1. Hence, they have a UAM of 1.
The user with ID=2 performed actions at minutes 2 and 3. Hence, they have a UAM of 2.
There is one user with a UAM of 1 and one with a UAM of 2.
Hence, answer[1] = 1, answer[2] = 1, and the remaining values are 0.

Constraints:

1 <= logs.length <= 104
0 <= IDi <= 109
1 <= timei <= 105
k is in the range [The maximum UAM for a user, 105].

1818. Minimum Absolute Sum Difference

You are given two positive integer arrays nums1 and nums2, both of length n.
The absolute sum difference of arrays nums1 and nums2 is defined as the sum of |nums1[i] - nums2[i]| for each 0 <= i < n (0-indexed).
You can replace at most one element of nums1 with any other element in nums1 to minimize the absolute sum difference.
Return the minimum absolute sum difference after replacing at most one element in the array nums1. Since the answer may be large, return it modulo 109 + 7.
|x| is defined as:

x if x >= 0, or
-x if x < 0.

Example 1:

Input: nums1 = [1,7,5], nums2 = [2,3,5]
Output: 3
Explanation: There are two possible optimal solutions:
- Replace the second element with the first: [1,7,5] => [1,1,5], or
- Replace the second element with the third: [1,7,5] => [1,5,5].
Both will yield an absolute sum difference of |1-2| + (|1-3| or |5-3|) + |5-5| = 3.

Example 2:

Input: nums1 = [2,4,6,8,10], nums2 = [2,4,6,8,10]
Output: 0
Explanation: nums1 is equal to nums2 so no replacement is needed. This will result in an
absolute sum difference of 0.

Example 3:

Input: nums1 = [1,10,4,4,2,7], nums2 = [9,3,5,1,7,4]
Output: 20
Explanation: Replace the first element with the second: [1,10,4,4,2,7] => [10,10,4,4,2,7].
This yields an absolute sum difference of |10-9| + |10-3| + |4-5| + |4-1| + |2-7| + |7-4| = 20

Constraints:

n == nums1.length
n == nums2.length
1 <= n <= 105
1 <= nums1[i], nums2[i] <= 105

1819. Number of Different Subsequences GCDs

You are given an array nums that consists of positive integers.
The GCD of a sequence of numbers is defined as the greatest integer that divides all the numbers in the sequence eve nly.

For example, the GCD of the sequence [4,6,16] is 2.

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].

Return the number of different GCDs among all non-empty subsequences of nums.

Example 1:

Input: nums = [6,10,3]
Output: 5
Explanation: The figure shows all the non-empty subsequences and their GCDs.
The different GCDs are 6, 10, 3, 2, and 1.

Example 2:

Input: nums = [5,15,40,5,6]
Output: 7

Constraints:

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

1822. Sign of the Product of an Array

There is a function signFunc(x) that returns:

1 if x is positive.
-1 if x is negative.
0 if x is equal to 0.

You are given an integer array nums. Let product be the product of all values in the array nums.
Return signFunc(product).

Example 1:

Input: nums = [-1,-2,-3,-4,3,2,1]
Output: 1
Explanation: The product of all values in the array is 144, and signFunc(144) = 1

Example 2:

Input: nums = [1,5,0,2,-3]
Output: 0
Explanation: The product of all values in the array is 0, and signFunc(0) = 0

Example 3:

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

Explanation: The product of all values in the array is -1, and signFunc(-1) = -1

Constraints:

1 <= nums.length <= 1000
-100 <= nums[i] <= 100

1823. Find the Winner of the Circular Game

There are n friends that are playing a game. The friends are sitting in a circle and are numbered from 1 to n in clock wise order. More formally, moving clockwise from the ith friend brings you to the (i+1)th friend for 1 <= i < n, and moving clockwise from the nth friend brings you to the 1st friend.
The rules of the game are as follows:

Start at the 1st friend.
Count the next k friends in the clockwise direction including the friend you started at. The counting wraps around th e circle and may count some friends more than once.
The last friend you counted leaves the circle and loses the game.
If there is still more than one friend in the circle, go back to step 2 starting from the friend immediately clockwise of the friend who just lost and repeat.
Else, the last friend in the circle wins the game.

Given the number of friends, n, and an integer k, return the winner of the game.

Example 1:

Input: n = 5, k = 2
Output: 3
Explanation: Here are the steps of the game:
1) Start at friend 1.
2) Count 2 friends clockwise, which are friends 1 and 2.
3) Friend 2 leaves the circle. Next start is friend 3.
4) Count 2 friends clockwise, which are friends 3 and 4.
5) Friend 4 leaves the circle. Next start is friend 5.
6) Count 2 friends clockwise, which are friends 5 and 1.
7) Friend 1 leaves the circle. Next start is friend 3.
8) Count 2 friends clockwise, which are friends 3 and 5.
9) Friend 5 leaves the circle. Only friend 3 is left, so they are the winner.
Example 2:

Input: n = 6, k = 5
Output: 1
Explanation: The friends leave in this order: 5, 4, 6, 2, 3. The winner is friend 1.

Constraints:

1 <= k <= n <= 500

1824. Minimum Sideway Jumps

There is a 3 lane road of length n that consists of n + 1 points labeled from 0 to n. A frog starts at point 0 in the seco nd lane and wants to jump to point n. However, there could be obstacles along the way.
You are given an array obstacles of length n + 1 where each obstacles[i] (ranging from 0 to 3) describes an obstacle on the lane obstacles[i] at point i. If obstacles[i] == 0, there are no obstacles at point i. There will be at most one obst acle in the 3 lanes at each point.

For example, if obstacles[2] == 1, then there is an obstacle on lane 1 at point 2.

The frog can only travel from point i to point i + 1 on the same lane if there is not an obstacle on the lane at point i + 1. To avoid obstacles, the frog can also perform a side jump to jump to another lane (even if they are not adjacent) at the same point if there is no obstacle on the new lane.

For example, the frog can jump from lane 3 at point 3 to lane 1 at point 3.

Return the minimum number of side jumps the frog needs to reach any lane at point n starting from lane 2 at point 0.
Note: There will be no obstacles on points 0 and n.

Example 1:

Input: obstacles = [0,1,2,3,0]
Output: 2
Explanation: The optimal solution is shown by the arrows above. There are 2 side jumps (red arrows).
Note that the frog can jump over obstacles only when making side jumps (as shown at point 2).

Example 2:

Input: obstacles = [0,1,1,3,3,0]
Output: 0
Explanation: There are no obstacles on lane 2. No side jumps are required.

Example 3:

Input: obstacles = [0,2,1,0,3,0]
Output: 2
Explanation: The optimal solution is shown by the arrows above. There are 2 side jumps.

Constraints:

obstacles.length == n + 1
1 <= n <= 5 * 105
0 <= obstacles[i] <= 3
obstacles[0] == obstacles[n] == 0

1825. Finding MK Average

You are given two integers, m and k, and a stream of integers. You are tasked to implement a data structure that calc ulates the MKAverage for the stream.
The MKAverage can be calculated using these steps:

If the number of the elements in the stream is less than m you should consider the MKAverage to be -1. Otherwise, c opy the last m elements of the stream to a separate container.
Remove the smallest k elements and the largest k elements from the container.
Calculate the average value for the rest of the elements rounded down to the nearest integer.

Implement the MKAverage class:

MKAverage(int m, int k) Initializes the MKAverage object with an empty stream and the two integers m and k.
void addElement(int num) Inserts a new element num into the stream.
int calculateMKAverage() Calculates and returns the MKAverage for the current stream rounded down to the nearest integer.

Example 1:

Input
["MKAverage", "addElement", "addElement", "calculateMKAverage", "addElement", "calculateMKAverage", "add Element", "addElement", "addElement", "calculateMKAverage"]
[[3, 1], [3], [1], [], [10], [], [5], [5], [5], []]
Output
[null, null, null, -1, null, 3, null, null, null, 5]

Explanation
MKAverage obj = new MKAverage(3, 1);
obj.addElement(3); // current elements are [3]
obj.addElement(1); // current elements are [3,1]
obj.calculateMKAverage(); // return -1, because m = 3 and only 2 elements exist.
obj.addElement(10); // current elements are [3,1,10]
obj.calculateMKAverage(); // The last 3 elements are [3,1,10].
// After removing smallest and largest 1 element the container will be [3].
// The average of [3] equals 3/1 = 3, return 3
obj.addElement(5); // current elements are [3,1,10,5]
obj.addElement(5); // current elements are [3,1,10,5,5]

obj.addElement(5); // current elements are [3,1,10,5,5,5]
obj.calculateMKAverage(); // The last 3 elements are [5,5,5].
// After removing smallest and largest 1 element the container will be [5].
// The average of [5] equals 5/1 = 5, return 5

Constraints:

3 <= m <= 105
1 <= k*2 < m
1 <= num <= 105
At most 105 calls will be made to addElement and calculateMKAverage.

1827. Minimum Operations to Make the Array Increasing

You are given an integer array nums (0-indexed). In one operation, you can choose an element of the array and incre ment it by 1.

For example, if nums = [1,2,3], you can choose to increment nums[1] to make nums = [1,3,3].

Return the minimum number of operations needed to make nums strictly increasing.
An array nums is strictly increasing if nums[i] < nums[i+1] for all 0 <= i < nums.length - 1. An array of length 1 is tr ivially strictly increasing.

Example 1:

Input: nums = [1,1,1]
Output: 3
Explanation: You can do the following operations:
1) Increment nums[2], so nums becomes [1,1,2].
2) Increment nums[1], so nums becomes [1,2,2].
3) Increment nums[2], so nums becomes [1,2,3].

Example 2:

Input: nums = [1,5,2,4,1]
Output: 14

Example 3:

Input: nums = [8]
Output: 0

Constraints:

1 <= nums.length <= 5000
1 <= nums[i] <= 104

1828. Queries on Number of Points Inside a Circle

You are given an array points where points[i] = [xi, yi] is the coordinates of the ith point on a 2D plane. Multiple poi nts can have the same coordinates.
You are also given an array queries where queries[j] = [xj, yj, rj] describes a circle centered at (xj, yj) with a radius o f rj.
For each query queries[j], compute the number of points inside the jth circle. Points on the border of the circle are co nsidered inside.
Return an array answer, where answer[j] is the answer to the jth query.

Example 1:

Input: points = [[1,3],[3,3],[5,3],[2,2]], queries = [[2,3,1],[4,3,1],[1,1,2]]
Output: [3,2,2]
Explanation: The points and circles are shown above.
queries[0] is the green circle, queries[1] is the red circle, and queries[2] is the blue circle.

Example 2:

Input: points = [[1,1],[2,2],[3,3],[4,4],[5,5]], queries = [[1,2,2],[2,2,2],[4,3,2],[4,3,3]]
Output: [2,3,2,4]
Explanation: The points and circles are shown above.
queries[0] is green, queries[1] is red, queries[2] is blue, and queries[3] is purple.

Constraints:

1 <= points.length <= 500
points[i].length == 2
0 <= x i, y i <= 500
1 <= queries.length <= 500
queries[j].length == 3
0 <= xj, yj <= 500
1 <= rj <= 500
All coordinates are integers.

Follow up: Could you find the answer for each query in better complexity than O(n)?

1829. Maximum XOR for Each Query

You are given a sorted array nums of n non-negative integers and an integer maximumBit. You want to perform the following query n times:

Find a non-negative integer k < 2maximumBit such that nums[0] XOR nums[1] XOR ... XOR nums[nums.length-1] XOR k is maximized. k is the answer to the ith query.
Remove the last element from the current array nums.

Return an array answer, where answer[i] is the answer to the ith query.

Example 1:

Input: nums = [0,1,1,3], maximumBit = 2
Output: [0,3,2,3]
Explanation: The queries are answered as follows:
1st query: nums = [0,1,1,3], k = 0 since 0 XOR 1 XOR 1 XOR 3 XOR 0 = 3.
2nd query: nums = [0,1,1], k = 3 since 0 XOR 1 XOR 1 XOR 3 = 3.
3rd query: nums = [0,1], k = 2 since 0 XOR 1 XOR 2 = 3.
4th query: nums = [0], k = 3 since 0 XOR 3 = 3.

Example 2:

Input: nums = [2,3,4,7], maximumBit = 3
Output: [5,2,6,5]
Explanation: The queries are answered as follows:
1st query: nums = [2,3,4,7], k = 5 since 2 XOR 3 XOR 4 XOR 7 XOR 5 = 7.
2nd query: nums = [2,3,4], k = 2 since 2 XOR 3 XOR 4 XOR 2 = 7.
3rd query: nums = [2,3], k = 6 since 2 XOR 3 XOR 6 = 7.
4th query: nums = [2], k = 5 since 2 XOR 5 = 7.

Example 3:

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

Constraints:

nums.length == n
1 <= n <= 105
1 <= maximumBit <= 20
0 <= nums[i] < 2maximumBit
nums is sorted in ascending order.

1830. Minimum Number of Operations to Make String Sorted

You are given a string s (0-indexed) . You are asked to perform the following operation on s until you get a so rted string:

Find the largest index i such that 1 <= i < s.length and s[i] < s[i - 1].
Find the largest index j such that i <= j < s.length and s[k] < s[i - 1] for all the possible values of k in the range [i, j] i nclusive.
Swap the two characters at indices i - 1 and j .
Reverse the suffix starting at index i .

Return the number of operations needed to make the string sorted. Since the answer can be too large, return it modul o 109 + 7.

Example 1:

Input: s = "cba"
Output: 5
Explanation: The simulation goes as follows:
Operation 1: i=2, j=2. Swap s[1] and s[2] to get s="cab", then reverse the suffix starting at 2. Now, s="cab".
Operation 2: i=1, j=2. Swap s[0] and s[2] to get s="bac", then reverse the suffix starting at 1. Now, s="bca".
Operation 3: i=2, j=2. Swap s[1] and s[2] to get s="bac", then reverse the suffix starting at 2. Now, s="bac".
Operation 4: i=1, j=1. Swap s[0] and s[1] to get s="abc", then reverse the suffix starting at 1. Now, s="acb".
Operation 5: i=2, j=2. Swap s[1] and s[2] to get s="abc", then reverse the suffix starting at 2. Now, s="abc".

Example 2:

Input: s = "aabaa"
Output: 2
Explanation: The simulation goes as follows:
Operation 1: i=3, j=4. Swap s[2] and s[4] to get s="aaaab", then reverse the substring starting at 3. Now, s="aaaba".
Operation 2: i=4, j=4. Swap s[3] and s[4] to get s="aaaab", then reverse the substring starting at 4. Now, s="aaaab".

Example 3:

Input: s = "cdbea"
Output: 63
Example 4:

Input: s = "leetcodeleetcodeleetcode"
Output: 982157772

Constraints:

1 <= s.length <= 3000
s consists only of lowercase English letters.

1832. Check if the Sentence Is Pangram

A pangram is a sentence where every letter of the English alphabet appears at least once.
Given a string sentence containing only lowercase English letters, return true if sentence is a pangram, or false other wise.

Example 1:

Input: sentence = "thequickbrownfoxjumpsoverthelazydog"
Output: true
Explanation: sentence contains at least one of every letter of the English alphabet.

Example 2:

Input: sentence = "leetcode"
Output: false

Constraints:

1 <= sentence.length <= 1000
sentence consists of lowercase English letters.

1833. Maximum Ice Cream Bars

It is a sweltering summer day, and a boy wants to buy some ice cream bars.
At the store, there are n ice cream bars. You are given an array costs of length n, where costs[i] is the price of the ith ice cream bar in coins. The boy initially has coins coins to spend, and he wants to buy as many ice cream bars as pos sible.
Return the maximum number of ice cream bars the boy can buy with coins coins.
Note: The boy can buy the ice cream bars in any order.

Example 1:

Input: costs = [1,3,2,4,1], coins = 7
Output: 4
Explanation: The boy can buy ice cream bars at indices 0,1,2,4 for a total price of 1 + 3 + 2 + 1 = 7.

Example 2:

Input: costs = [10,6,8,7,7,8], coins = 5

Output: 0
Explanation: The boy cannot afford any of the ice cream bars.

Example 3:

Input: costs = [1,6,3,1,2,5], coins = 20
Output: 6
Explanation: The boy can buy all the ice cream bars for a total price of 1 + 6 + 3 + 1 + 2 + 5 = 18.

Constraints:

costs.length == n
1 <= n <= 105
1 <= costs[i] <= 105
1 <= coins <= 108

1834. Single-Threaded CPU

You are given n tasks labeled from 0 to n - 1 represented by a 2D integer array tasks, where tasks[i] = [enqueueTimei, processingTimei] means that the i th task will be available to process at enqueueTimei and will take processingTimei to finish processing.
You have a single-threaded CPU that can process at most one task at a time and will act in the following way:

If the CPU is idle and there are no available tasks to process, the CPU remains idle.
If the CPU is idle and there are available tasks, the CPU will choose the one with the shortest processing time. If multiple tasks have the same shortest processing time, it will choose the task with the smallest index.
Once a task is started, the CPU will process the entire task without stopping.
The CPU can finish a task then start a new one instantly.

Return the order in which the CPU will process the tasks.

Example 1:

Input: tasks = [[1,2],[2,4],[3,2],[4,1]]
Output: [0,2,3,1]
Explanation: The events go as follows:
- At time = 1, task 0 is available to process. Available tasks = {0}.
- Also at time = 1, the idle CPU starts processing task 0. Available tasks = {}.
- At time = 2, task 1 is available to process. Available tasks = {1}.
- At time = 3, task 2 is available to process. Available tasks = {1, 2}.
- Also at time = 3, the CPU finishes task 0 and starts processing task 2 as it is the shortest. Available tasks = {1}.
- At time = 4, task 3 is available to process. Available tasks = {1, 3}.
- At time = 5, the CPU finishes task 2 and starts processing task 3 as it is the shortest. Available tasks = {1}.
- At time = 6, the CPU finishes task 3 and starts processing task 1. Available tasks = {}.
- At time = 10, the CPU finishes task 1 and becomes idle.

Example 2:

Input: tasks = [[7,10],[7,12],[7,5],[7,4],[7,2]]
Output: [4,3,2,0,1]
Explanation: The events go as follows:
- At time = 7, all the tasks become available. Available tasks = {0,1,2,3,4}.
- Also at time = 7, the idle CPU starts processing task 4. Available tasks = {0,1,2,3}.
- At time = 9, the CPU finishes task 4 and starts processing task 3. Available tasks = {0,1,2}.
- At time = 13, the CPU finishes task 3 and starts processing task 2. Available tasks = {0,1}.
- At time = 18, the CPU finishes task 2 and starts processing task 0. Available tasks = {1}.
- At time = 28, the CPU finishes task 0 and starts processing task 1. Available tasks = {}.
- At time = 40, the CPU finishes task 1 and becomes idle.

Constraints:

tasks.length == n
1 <= n <= 105
1 <= enqueueTimei, processingTimei <= 109

1835. Find XOR Sum of All Pairs Bitwise AND

The XOR sum of a list is the bitwise XOR of all its elements. If the list only contains one element, then its XOR sum will be equal to this element.

For example, the XOR sum of [1,2,3,4] is equal to 1 XOR 2 XOR 3 XOR 4 = 4, and the XOR sum of [3] is equal to 3.

You are given two 0-indexed arrays arr1 and arr2 that consist only of non-negative integers.
Consider the list containing the result of arr1[i] AND arr2[j] (bitwise AND) for every (i, j) pair where 0 <= i < arr1.length and 0 <= j < arr2.length.
Return the XOR sum of the aforementioned list.

Example 1:

Input: arr1 = [1,2,3], arr2 = [6,5]
Output: 0
Explanation: The list = [1 AND 6, 1 AND 5, 2 AND 6, 2 AND 5, 3 AND 6, 3 AND 5] = [0,1,2,0,2,1].
The XOR sum = 0 XOR 1 XOR 2 XOR 0 XOR 2 XOR 1 = 0.

Example 2:

Input: arr1 = [12], arr2 = [4]
Output: 4
Explanation: The list = [12 AND 4] = [4]. The XOR sum = 4.

Constraints:

1 <= arr1.length, arr2.length <= 105
0 <= arr1[i], arr2[j] <= 109

1837. Sum of Digits in Base K

Given an integer n (in base 10) and a base k, return the sum of the digits of n after converting n from base 10 to base k.
After converting, each digit should be interpreted as a base 10 number, and the sum should be returned in base 10.

Example 1:

Input: n = 34, k = 6
Output: 9
Explanation: 34 (base 10) expressed in base 6 is 54. 5 + 4 = 9.

Example 2:

Input: n = 10, k = 10
Output: 1
Explanation: n is already in base 10. 1 + 0 = 1.

Constraints:

1 <= n <= 100
2 <= k <= 10

1838. Frequency of the Most Frequent Element

The frequency of an element is the number of times it occurs in an array.
You are given an integer array nums and an integer k. In one operation, you can choose an index of nums and incre ment the element at that index by 1.
Return the maximum possible frequency of an element after performing at most k operations.

Example 1:

Input: nums = [1,2,4], k = 5
Output: 3
Explanation: Increment the first element three times and the second element two times to make nums = [4,4,4].
4 has a frequency of 3.
Example 2:

Input: nums = [1,4,8,13], k = 5
Output: 2
Explanation: There are multiple optimal solutions:
- Increment the first element three times to make nums = [4,4,8,13]. 4 has a frequency of 2.
- Increment the second element four times to make nums = [1,8,8,13]. 8 has a frequency of 2.
- Increment the third element five times to make nums = [1,4,13,13]. 13 has a frequency of 2.

Example 3:

Input: nums = [3,9,6], k = 2
Output: 1

Constraints:

1 <= nums.length <= 105
1 <= nums[i] <= 105
1 <= k <= 105

1839. Longest Substring Of All Vowels in Order

A string is considered beautiful if it satisfies the following conditions:

Each of the 5 English vowels ('a', 'e', 'i', 'o', 'u') must appear at least once in it.
The letters must be sorted in alphabetical order (i.e. all 'a's before 'e's, all 'e's before 'i's, etc.).

For example, strings "aeiou" and "aaaaaaeiiiioou" are considered beautiful, but "uaeio", "aeoiu", and "aaaeeeooo" ar e not beautiful.
Given a string word consisting of English vowels, return the length of the longest beautiful substring of word. If no s uch substring exists, return 0.
A substring is a contiguous sequence of characters in a string.

Example 1:

Input: word = "aeiaaioaaaaeiiiiouuuooaauuaeiu"
Output: 13
Explanation: The longest beautiful substring in word is "aaaaeiiiiouuu" of length 13.
Example 2:

Input: word = "aeeeiiiioooauuuaeiou"
Output: 5
Explanation: The longest beautiful substring in word is "aeiou" of length 5.

Example 3:

Input: word = "a"
Output: 0
Explanation: There is no beautiful substring, so return 0.

Constraints:

1 <= word.length <= 5 * 105
word consists of characters 'a', 'e', 'i', 'o', and 'u'.

1840. Maximum Building Height

You want to build n new buildings in a city. The new buildings will be built in a line and are labeled from 1 to n. However, there are city restrictions on the heights of the new buildings:

The height of each building must be a non-negative integer.
The height of the first building must be 0.
The height difference between any two adjacent buildings cannot exceed 1.

Additionally, there are city restrictions on the maximum height of specific buildings. These restrictions are given as a 2D integer array restrictions where restrictions[i] = [idi, maxHeighti] indicates that building idi must have a height less than or equal to maxHeighti.
It is guaranteed that each building will appear at most once in restrictions, and building 1 will not be in restrictions.
Return the maximum possible height of the tallest building.

Example 1:

Input: n = 5, restrictions = [[2,1],[4,1]]
Output: 2
Explanation: The green area in the image indicates the maximum allowed height for each building.
We can build the buildings with heights [0,1,2,1,2], and the tallest building has a height of 2.
Example 2:

Input: n = 6, restrictions = []
Output: 5
Explanation: The green area in the image indicates the maximum allowed height for each building.
We can build the buildings with heights [0,1,2,3,4,5], and the tallest building has a height of 5.

Example 3:

Input: n = 10, restrictions = [[5,3],[2,5],[7,4],[10,3]]
Output: 5
Explanation: The green area in the image indicates the maximum allowed height for each building.
We can build the buildings with heights [0,1,2,3,3,4,4,5,4,3], and the tallest building has a height of 5.

Constraints:

2 <= n <= 109
0 <= restrictions.length <= min(n - 1, 105)
2 <= idi <= n
idi is unique.
0 <= maxHeighti <= 109

1844. Replace All Digits with Characters

You are given a 0-indexed string s that has lowercase English letters in its even indices and digits in its odd indices.
There is a function shift(c, x), where c is a character and x is a digit, that returns the xth character after c.

For example, shift('a', 5) = 'f' and shift('x', 0) = 'x'.

For every odd index i, you want to replace the digit s[i] with shift(s[i-1], s[i]).
Return s after replacing all digits. It is guaranteed that shift(s[i-1], s[i]) will never exceed 'z'.

Example 1:

Input: s = "a1c1e1"
Output: "abcdef"
Explanation: The digits are replaced as follows:
- s[1] -> shift('a',1) = 'b'
- s[3] -> shift('c',1) = 'd'
- s[5] -> shift('e',1) = 'f'
Example 2:

Input: s = "a1b2c3d4e"
Output: "abbdcfdhe"
Explanation: The digits are replaced as follows:
- s[1] -> shift('a',1) = 'b'
- s[3] -> shift('b',2) = 'd'
- s[5] -> shift('c',3) = 'f'
- s[7] -> shift('d',4) = 'h'

Constraints:

1 <= s.length <= 100
s consists only of lowercase English letters and digits.
shift(s[i-1], s[i]) <= 'z' for all odd indices i.

1845. Seat Reservation Manager

Design a system that manages the reservation state of n seats that are numbered from 1 to n.
Implement the SeatManager class:

SeatManager(int n) Initializes a SeatManager object that will manage n seats numbered from 1 to n. All seats are init ially available.
int reserve() Fetches the smallest-numbered unreserved seat, reserves it, and returns its number.
void unreserve(int seatNumber) Unreserves the seat with the given seatNumber.

Example 1:

Input
["SeatManager", "reserve", "reserve", "unreserve", "reserve", "reserve", "reserve", "reserve", "unreserve"]
[[5], [], [], [2], [], [], [], [], [5]]
Output
[null, 1, 2, null, 2, 3, 4, 5, null]

Explanation
SeatManager seatManager = new SeatManager(5); // Initializes a SeatManager with 5 seats.
seatManager.reserve();   // All seats are available, so return the lowest numbered seat, which is 1.
seatManager.reserve();   // The available seats are [2,3,4,5], so return the lowest of them, which is 2.
seatManager.unreserve(2); // Unreserve seat 2, so now the available seats are [2,3,4,5].
seatManager.reserve();   // The available seats are [2,3,4,5], so return the lowest of them, which is 2.
seatManager.reserve();   // The available seats are [3,4,5], so return the lowest of them, which is 3.
seatManager.reserve();   // The available seats are [4,5], so return the lowest of them, which is 4.
seatManager.reserve();   // The only available seat is seat 5, so return 5.
seatManager.unreserve(5); // Unreserve seat 5, so now the available seats are [5].

Constraints:

1 <= n <= 105
1 <= seatNumber <= n
For each call to reserve, it is guaranteed that there will be at least one unreserved seat.
For each call to unreserve, it is guaranteed that seatNumber will be reserved.
At most 105 calls in total will be made to reserve and unreserve.

1846. Maximum Element After Decreasing and Rearranging

You are given an array of positive integers arr. Perform some operations (possibly none) on arr so that it satisfies the se conditions:

The value of the first element in arr must be 1.
The absolute difference between any 2 adjacent elements must be less than or equal to 1. In other words, abs(arr[i] - arr[i - 1]) <= 1 for each i where 1 <= i < arr.length (0-indexed). abs(x) is the absolute value of x.

There are 2 types of operations that you can perform any number of times:

Decrease the value of any element of arr to a smaller positive integer.
Rearrange the elements of arr to be in any order.

Return the maximum possible value of an element in arr after performing the operations to satisfy the conditions.

Example 1:

Input: arr = [2,2,1,2,1]
Output: 2
Explanation:
We can satisfy the conditions by rearranging arr so it becomes [1,2,2,2,1].
The largest element in arr is 2.

Example 2:

Input: arr = [100,1,1000]
Output: 3
Explanation:
One possible way to satisfy the conditions is by doing the following:
1. Rearrange arr so it becomes [1,100,1000].
2. Decrease the value of the second element to 2.
3. Decrease the value of the third element to 3.
Now arr = [1,2,3], which satisfies the conditions.
The largest element in arr is 3.

Example 3:

Input: arr = [1,2,3,4,5]
Output: 5
Explanation: The array already satisfies the conditions, and the largest element is 5.

Constraints:

1 <= arr.length <= 105
1 <= arr[i] <= 109

1847. Closest Room

There is a hotel with n rooms. The rooms are represented by a 2D integer array rooms where rooms[i] = [roomIdi, si zei] denotes that there is a room with room number roomIdi and size equal to sizei. Each roomIdi is guaranteed to be unique.
You are also given k queries in a 2D array queries where queries[j] = [preferredj, minSizej]. The answer to the jth qu ery is the room number id of a room such that:

The room has a size of at least minSizej, and
abs(id - preferredj) is minimized, where abs(x) is the absolute value of x.

If there is a tie in the absolute difference, then use the room with the smallest such id. If there is no such room, the an swer is -1.
Return an array answer of length k where answer[j] contains the answer to the jth query.

Example 1:

Input: rooms = [[2,2],[1,2],[3,2]], queries = [[3,1],[3,3],[5,2]]
Output: [3,-1,3]
Explanation: The answers to the queries are as follows:
Query = [3,1]: Room number 3 is the closest as abs(3 - 3) = 0, and its size of 2 is at least 1. The answer is 3.
Query = [3,3]: There are no rooms with a size of at least 3, so the answer is -1.
Query = [5,2]: Room number 3 is the closest as abs(3 - 5) = 2, and its size of 2 is at least 2. The answer is 3.
Example 2:

Input: rooms = [[1,4],[2,3],[3,5],[4,1],[5,2]], queries = [[2,3],[2,4],[2,5]]
Output: [2,1,3]
Explanation: The answers to the queries are as follows:
Query = [2,3]: Room number 2 is the closest as abs(2 - 2) = 0, and its size of 3 is at least 3. The answer is 2.
Query = [2,4]: Room numbers 1 and 3 both have sizes of at least 4. The answer is 1 since it is smaller.
Query = [2,5]: Room number 3 is the only room with a size of at least 5. The answer is 3.

Constraints:

n == rooms.length
1 <= n <= 105
k == queries.length
1 <= k <= 104
1 <= roomIdi, preferredj <= 107
1 <= sizei, minSizej <= 107

1848. Minimum Distance to the Target Element

Given an integer array nums (0-indexed) and two integers target and start, find an index i such that nums[i] == target and abs(i - start) is minimized. Note that abs(x) is the absolute value of x.
Return abs(i - start).
It is guaranteed that target exists in nums.

Example 1:

Input: nums = [1,2,3,4,5], target = 5, start = 3
Output: 1
Explanation: nums[4] = 5 is the only value equal to target, so the answer is abs(4 - 3) = 1.

Example 2:

Input: nums = [1], target = 1, start = 0
Output: 0
Explanation: nums[0] = 1 is the only value equal to target, so the answer is abs(0 - 0) = 0.

Example 3:

Input: nums = [1,1,1,1,1,1,1,1,1,1], target = 1, start = 0
Output: 0
Explanation: Every value of nums is 1, but nums[0] minimizes abs(i - start), which is abs(0 - 0) = 0.

Constraints:

1 <= nums.length <= 1000
1 <= nums[i] <= 104
0 <= start < nums.length
target is in nums.

1849. Splitting a String Into Descending Consecutive Values

You are given a string s that consists of only digits.
Check if we can split s into two or more non-empty substrings such that the numerical values of the substrings are in descending order and the difference between numerical values of every two adjacent substrings is equal to 1.

For example, the string s = "0090089" can be split into ["0090", "089"] with numerical values [90,89]. The values ar e in descending order and adjacent values differ by 1, so this way is valid.
Another example, the string s = "001" can be split into ["0", "01"], ["00", "1"], or ["0", "0", "1"]. However all the wa ys are invalid because they have numerical values [0,1], [0,1], and [0,0,1] respectively, all of which are not in descen

ding order.

Return true if it is possible to split s as described above, or false otherwise.
A substring is a contiguous sequence of characters in a string.

Example 1:

Input: s = "1234"
Output: false
Explanation: There is no valid way to split s.

Example 2:

Input: s = "050043"
Output: true
Explanation: s can be split into ["05", "004", "3"] with numerical values [5,4,3].
The values are in descending order with adjacent values differing by 1.

Example 3:

Input: s = "9080701"
Output: false
Explanation: There is no valid way to split s.

Example 4:

Input: s = "10009998"
Output: true
Explanation: s can be split into ["100", "099", "98"] with numerical values [100,99,98].
The values are in descending order with adjacent values differing by 1.

Constraints:

1 <= s.length <= 20
s only consists of digits.

1850. Minimum Adjacent Swaps to Reach the Kth Smallest Number

You are given a string num, representing a large integer, and an integer k.
We call some integer wonderful if it is a permutation of the digits in num and is greater in value than num. There can be many wonderful integers. However, we only care about the smallest-valued ones.

For example, when num = "5489355142":

The 1st smallest wonderful integer is "5489355214".
The 2nd smallest wonderful integer is "5489355241".
The 3rd smallest wonderful integer is "5489355412".
The 4th smallest wonderful integer is "5489355421".

Return the minimum number of adjacent digit swaps that needs to be applied to num to reach the kth smallest wonde rful integer.
The tests are generated in such a way that kth smallest wonderful integer exists.

Example 1:

Input: num = "5489355142", k = 4
Output: 2
Explanation: The 4th smallest wonderful number is "5489355421". To get this number:
- Swap index 7 with index 8: "5489355142" -> "5489355412"
- Swap index 8 with index 9: "5489355412" -> "5489355421"

Example 2:

Input: num = "11112", k = 4
Output: 4
Explanation: The 4th smallest wonderful number is "21111". To get this number:
- Swap index 3 with index 4: "11112" -> "11121"
- Swap index 2 with index 3: "11121" -> "11211"
- Swap index 1 with index 2: "11211" -> "12111"
- Swap index 0 with index 1: "12111" -> "21111"

Example 3:

Input: num = "00123", k = 1
Output: 1
Explanation: The 1st smallest wonderful number is "00132". To get this number:
- Swap index 3 with index 4: "00123" -> "00132"

Constraints:

2 <= num.length <= 1000
1 <= k <= 1000
num only consists of digits.

1851. Minimum Interval to Include Each Query

You are given a 2D integer array intervals, where intervals[i] = [lefti, righti] describes the ith interval starting at lefti

and ending at righti (inclusive). The size of an interval is defined as the number of integers it contains, or more form ally righti - lefti + 1.
You are also given an integer array queries. The answer to the jth query is the size of the smallest interval i such that lefti <= queries[j] <= righti. If no such interval exists, the answer is -1.
Return an array containing the answers to the queries.

Example 1:

Input: intervals = [[1,4],[2,4],[3,6],[4,4]], queries = [2,3,4,5]
Output: [3,3,1,4]
Explanation: The queries are processed as follows:
- Query = 2: The interval [2,4] is the smallest interval containing 2. The answer is 4 - 2 + 1 = 3.
- Query = 3: The interval [2,4] is the smallest interval containing 3. The answer is 4 - 2 + 1 = 3.
- Query = 4: The interval [4,4] is the smallest interval containing 4. The answer is 4 - 4 + 1 = 1.
- Query = 5: The interval [3,6] is the smallest interval containing 5. The answer is 6 - 3 + 1 = 4.

Example 2:

Input: intervals = [[2,3],[2,5],[1,8],[20,25]], queries = [2,19,5,22]
Output: [2,-1,4,6]
Explanation: The queries are processed as follows:
- Query = 2: The interval [2,3] is the smallest interval containing 2. The answer is 3 - 2 + 1 = 2.
- Query = 19: None of the intervals contain 19. The answer is -1.
- Query = 5: The interval [2,5] is the smallest interval containing 5. The answer is 5 - 2 + 1 = 4.
- Query = 22: The interval [20,25] is the smallest interval containing 22. The answer is 25 - 20 + 1 = 6.

Constraints:

1 <= intervals.length <= 105
1 <= queries.length <= 105
intervals[i].length == 2
1 <= lefti <= righti <= 107
1 <= queries[j] <= 107

1854. Maximum Population Year

You are given a 2D integer array logs where each logs[i] = [birthi, deathi] indicates the birth and death years of the it h person.
The population of some year x is the number of people alive during that year. The ith person is counted in year x's p opulation if x is in the inclusive range [birthi, deathi - 1]. Note that the person is not counted in the year that they die.
Return the earliest year with the maximum population.

Example 1:

Input: logs = [[1993,1999],[2000,2010]]

Output: 1993
Explanation: The maximum population is 1, and 1993 is the earliest year with this population.

Example 2:

Input: logs = [[1950,1961],[1960,1971],[1970,1981]]
Output: 1960
Explanation:
The maximum population is 2, and it had happened in years 1960 and 1970.
The earlier year between them is 1960.

Constraints:

1 <= logs.length <= 100
1950 <= birthi < deathi <= 2050

1855. Maximum Distance Between a Pair of Values

You are given two non-increasing 0-indexed integer arrays nums1 and nums2 .
A pair of indices (i, j), where 0 <= i < nums1.length and 0 <= j < nums2.length, is valid if both i <= j and nums1[i] <= nums2[j]. The distance of the pair is j - i .
Return the maximum distance of any valid pair (i, j). If there are no valid pairs, return 0.
An array arr is non-increasing if arr[i-1] >= arr[i] for every 1 <= i < arr.length.

Example 1:

Input: nums1 = [55,30,5,4,2], nums2 = [100,20,10,10,5]
Output: 2
Explanation: The valid pairs are (0,0), (2,2), (2,3), (2,4), (3,3), (3,4), and (4,4).
The maximum distance is 2 with pair (2,4).

Example 2:

Input: nums1 = [2,2,2], nums2 = [10,10,1]
Output: 1
Explanation: The valid pairs are (0,0), (0,1), and (1,1).
The maximum distance is 1 with pair (0,1).

Example 3:

Input: nums1 = [30,29,19,5], nums2 = [25,25,25,25,25]
Output: 2
Explanation: The valid pairs are (2,2), (2,3), (2,4), (3,3), and (3,4).
The maximum distance is 2 with pair (2,4).

Example 4:

Input: nums1 = [5,4], nums2 = [3,2]
Output: 0
Explanation: There are no valid pairs, so return 0.

Constraints:

1 <= nums1.length, nums2.length <= 105
1 <= nums1[i], nums2[j] <= 105
Both nums1 and nums2 are non-increasing.

1856. Maximum Subarray Min-Product

The min-product of an array is equal to the minimum value in the array multiplied by the array's sum.

For example, the array [3,2,5] (minimum value is 2) has a min-product of 2 * (3+2+5) = 2 * 10 = 20.

Given an array of integers nums, return the maximum min-product of any non-empty subarray of nums. Since the an swer may be large, return it modulo 109 + 7.
Note that the min-product should be maximized before performing the modulo operation. Testcases are generated su ch that the maximum min-product without modulo will fit in a 64-bit signed integer.
A subarray is a contiguous part of an array.

Example 1:

Input: nums = [1,2,3,2]
Output: 14
Explanation: The maximum min-product is achieved with the subarray [2,3,2] (minimum value is 2).
2 * (2+3+2) = 2 * 7 = 14.

Example 2:

Input: nums = [2,3,3,1,2]
Output: 18
Explanation: The maximum min-product is achieved with the subarray [3,3] (minimum value is 3).
3 * (3+3) = 3 * 6 = 18.

Example 3:

Input: nums = [3,1,5,6,4,2]
Output: 60
Explanation: The maximum min-product is achieved with the subarray [5,6,4] (minimum value is 4).
4 * (5+6+4) = 4 * 15 = 60.

Constraints:

1 <= nums.length <= 105
1 <= nums[i] <= 107

1857. Largest Color Value in a Directed Graph

There is a directed graph of n colored nodes and m edges. The nodes are numbered from 0 to n - 1.
You are given a string colors where colors[i] is a lowercase English letter representing the color of the ith node in thi s graph (0-indexed). You are also given a 2D array edges where edges[j] = [aj, bj] indicates that there is a directed ed ge from node aj to node bj.
A valid path in the graph is a sequence of nodes x1 -> x2 -> x3 -> ... -> xk such that there is a directed edge from xi t o xi+1 for every 1 <= i < k. The color value of the path is the number of nodes that are colored the most frequently o ccurring color along that path.
Return the largest color value of any valid path in the given graph, or -1 if the graph contains a cycle.

Example 1:

Input: colors = "abaca", edges = [[0,1],[0,2],[2,3],[3,4]]
Output: 3
Explanation: The path 0 -> 2 -> 3 -> 4 contains 3 nodes that are colored "a" (red in the above image).

Example 2:

Input: colors = "a", edges = [[0,0]]
Output: -1
Explanation: There is a cycle from 0 to 0.

Constraints:

n == colors.length
m == edges.length
1 <= n <= 105
0 <= m <= 105
colors consists of lowercase English letters.
0 <= aj, bj < n

1859. Sorting the Sentence

A sentence is a list of words that are separated by a single space with no leading or trailing spaces. Each word consis ts of lowercase and uppercase English letters.
A sentence can be shuffled by appending the 1-indexed word position to each word then rearranging the words in the sentence.

For example, the sentence "This is a sentence" can be shuffled as "sentence4 a3 is2 This1" or "is2 sentence4 This1 a 3".

Given a shuffled sentence s containing no more than 9 words, reconstruct and return the original sentence.

Example 1:

Input: s = "is2 sentence4 This1 a3"
Output: "This is a sentence"
Explanation: Sort the words in s to their original positions "This1 is2 a3 sentence4", then remove the numbers.

Example 2:

Input: s = "Myself2 Me1 I4 and3"
Output: "Me Myself and I"
Explanation: Sort the words in s to their original positions "Me1 Myself2 and3 I4", then remove the numbers.

Constraints:

2 <= s.length <= 200
s consists of lowercase and uppercase English letters, spaces, and digits from 1 to 9.
The number of words in s is between 1 and 9.
The words in s are separated by a single space.
s contains no leading or trailing spaces.

1860. Incremental Memory Leak

You are given two integers memory1 and memory2 representing the available memory in bits on two memory sticks . There is currently a faulty program running that consumes an increasing amount of memory every second.
At the ith second (starting from 1), i bits of memory are allocated to the stick with more available memory (or from t he first memory stick if both have the same available memory). If neither stick has at least i bits of available memory , the program crashes.
Return an array containing [crashTime, memory1crash, memory2crash], where crashTime is the time (in seconds) w hen the program crashed and memory1crash and memory2crash are the available bits of memory in the first and seco nd sticks respectively.

Example 1:

Input: memory1 = 2, memory2 = 2
Output: [3,1,0]
Explanation: The memory is allocated as follows:
- At the 1st second, 1 bit of memory is allocated to stick 1. The first stick now has 1 bit of available memory.
- At the 2nd second, 2 bits of memory are allocated to stick 2. The second stick now has 0 bits of available memory.
- At the 3rd second, the program crashes. The sticks have 1 and 0 bits available respectively.

Example 2:

Input: memory1 = 8, memory2 = 11
Output: [6,0,4]
Explanation: The memory is allocated as follows:
- At the 1st second, 1 bit of memory is allocated to stick 2. The second stick now has 10 bit of available memory.
- At the 2nd second, 2 bits of memory are allocated to stick 2. The second stick now has 8 bits of available memory.
- At the 3rd second, 3 bits of memory are allocated to stick 1. The first stick now has 5 bits of available memory.
- At the 4th second, 4 bits of memory are allocated to stick 2. The second stick now has 4 bits of available memory.
- At the 5th second, 5 bits of memory are allocated to stick 1. The first stick now has 0 bits of available memory.
- At the 6th second, the program crashes. The sticks have 0 and 4 bits available respectively.

Constraints:

0 <= memory1, memory2 <= 231 - 1

1861. Rotating the Box

You are given an m x n matrix of characters box representing a side-view of a box. Each cell of the box is one of the following:

A stone '#'
A stationary obstacle '*'
Empty '.'

The box is rotated 90 degrees clockwise, causing some of the stones to fall due to gravity. Each stone falls down unti l it lands on an obstacle, another stone, or the bottom of the box. Gravity does not affect the obstacles' positions, and the inertia from the box's rotation does not affect the stones' horizontal positions.
It is guaranteed that each stone in box rests on an obstacle, another stone, or the bottom of the box.
Return an n x m matrix representing the box after the rotation described above.

Example 1:

Input: box = [["#",".","#"]]
Output: [["."],
["#"],
["#"]]

Example 2:

```
Input: box = [["#",".","*","."],
              ["#","#","*","."]]
Output: [["#","."],
         ["#","#"],
         ["*","*"],
         [".","."]]
```

Example 3:

```
Input: box = [["#","#","*",".","*","."],
              ["#","#","#","*",".","."],
              ["#","#","#",".","#","."]]
Output: [[".","#","#"],
         [".","#","#"],
         ["#","#","*"],
         ["#","*","."],
         ["#",".","*"],
         ["#",".","."]]
```

Constraints:

m == box.length
n == box[i].length
1 <= m, n <= 500
box[i][j] is either '#', '*', or '.'.

1862. Sum of Floored Pairs

Given an integer array nums, return the sum of floor(nums[i] / nums[j]) for all pairs of indices 0 <= i, j < nums.length in the array. Since the answer may be too large, return it modulo 109 + 7.
The floor() function returns the integer part of the division.

Example 1:

```
Input: nums = [2,5,9]
Output: 10
Explanation:
floor(2 / 5) = floor(2 / 9) = floor(5 / 9) = 0
floor(2 / 2) = floor(5 / 5) = floor(9 / 9) = 1
floor(5 / 2) = 2
floor(9 / 2) = 4
```

floor(9 / 5) = 1
We calculate the floor of the division for every pair of indices in the array then sum them up.

Example 2:

Input: nums = [7,7,7,7,7,7,7]
Output: 49

Constraints:

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

1863. Sum of All Subset XOR Totals

The XOR total of an array is defined as the bitwise XOR of all its elements, or 0 if the array is empty.

For example, the XOR total of the array [2,5,6] is 2 XOR 5 XOR 6 = 1.

Given an array nums, return the sum of all XOR totals for every subset of nums.
Note: Subsets with the same elements should be counted multiple times.
An array a is a subset of an array b if a can be obtained from b by deleting some (possibly zero) elements of b.

Example 1:

Input: nums = [1,3]
Output: 6
Explanation: The 4 subsets of [1,3] are:
- The empty subset has an XOR total of 0.
- [1] has an XOR total of 1.
- [3] has an XOR total of 3.
- [1,3] has an XOR total of 1 XOR 3 = 2.
0 + 1 + 3 + 2 = 6

Example 2:

Input: nums = [5,1,6]
Output: 28
Explanation: The 8 subsets of [5,1,6] are:
- The empty subset has an XOR total of 0.
- [5] has an XOR total of 5.
- [1] has an XOR total of 1.
- [6] has an XOR total of 6.
- [5,1] has an XOR total of 5 XOR 1 = 4.
- [5,6] has an XOR total of 5 XOR 6 = 3.

- [1,6] has an XOR total of 1 XOR 6 = 7.
- [5,1,6] has an XOR total of 5 XOR 1 XOR 6 = 2.
0 + 5 + 1 + 6 + 4 + 3 + 7 + 2 = 28

Example 3:

Input: nums = [3,4,5,6,7,8]
Output: 480
Explanation: The sum of all XOR totals for every subset is 480.

Constraints:

1 <= nums.length <= 12
1 <= nums[i] <= 20

1864. Minimum Number of Swaps to Make the Binary String Alternating

Given a binary string s, return the minimum number of character swaps to make it alternating, or -1 if it is impossibl e.
The string is called alternating if no two adjacent characters are equal. For example, the strings "010" and "1010" are alternating, while the string "0100" is not.
Any two characters may be swapped, even if they are not adjacent.

Example 1:

Input: s = "111000"
Output: 1
Explanation: Swap positions 1 and 4: "111000" -> "101010"
The string is now alternating.

Example 2:

Input: s = "010"
Output: 0
Explanation: The string is already alternating, no swaps are needed.

Example 3:

Input: s = "1110"
Output: -1

Constraints:

1 <= s.length <= 1000

s[i] is either '0' or '1'.

1865. Finding Pairs With a Certain Sum

You are given two integer arrays nums1 and nums2. You are tasked to implement a data structure that supports queri es of two types:

Add a positive integer to an element of a given index in the array nums2.
Count the number of pairs (i, j) such that nums1[i] + nums2[j] equals a given value (0 <= i < nums1.length and 0 <= j < nums2.length).

Implement the FindSumPairs class:

FindSumPairs(int[] nums1, int[] nums2) Initializes the FindSumPairs object with two integer arrays nums1 and num s2.
void add(int index, int val) Adds val to nums2[index], i.e., apply nums2[index] += val.
int count(int tot) Returns the number of pairs (i, j) such that nums1[i] + nums2[j] == tot.

Example 1:

Input
["FindSumPairs", "count", "add", "count", "count", "add", "add", "count"]
[[[1, 1, 2, 2, 2, 3], [1, 4, 5, 2, 5, 4]], [7], [3, 2], [8], [4], [0, 1], [1, 1], [7]]
Output
[null, 8, null, 2, 1, null, null, 11]

Explanation
FindSumPairs findSumPairs = new FindSumPairs([1, 1, 2, 2, 2, 3], [1, 4, 5, 2, 5, 4]);
findSumPairs.count(7); // return 8; pairs (2,2), (3,2), (4,2), (2,4), (3,4), (4,4) make 2 + 5 and pairs (5,1), (5,5) make 3 + 4
findSumPairs.add(3, 2); // now nums2 = [1,4,5,4,5,4]
findSumPairs.count(8); // return 2; pairs (5,2), (5,4) make 3 + 5
findSumPairs.count(4); // return 1; pair (5,0) makes 3 + 1
findSumPairs.add(0, 1); // now nums2 = [2,4,5,4,5,4]
findSumPairs.add(1, 1); // now nums2 = [2,5,5,4,5,4]
findSumPairs.count(7); // return 11; pairs (2,1), (2,2), (2,4), (3,1), (3,2), (3,4), (4,1), (4,2), (4,4) make 2 + 5 and pair s (5,3), (5,5) make 3 + 4

Constraints:

1 <= nums1.length <= 1000
1 <= nums2.length <= 105
1 <= nums1[i] <= 109
1 <= nums2[i] <= 105

0 <= index < nums2.length
1 <= val <= 105
1 <= tot <= 109
At most 1000 calls are made to add and count each.

1866. Number of Ways to Rearrange Sticks With K Sticks Visible

There are n uniquely-sized sticks whose lengths are integers from 1 to n. You want to arrange the sticks such that ex actly k sticks are visible from the left. A stick is visible from the left if there are no longer sticks to the left of it.

For example, if the sticks are arranged [1,3,2,5,4], then the sticks with lengths 1, 3, and 5 are visible from the left.

Given n and k, return the number of such arrangements. Since the answer may be large, return it modulo 109 + 7.

Example 1:

Input: n = 3, k = 2
Output: 3
Explanation: [1,3,2], [2,3,1], and [2,1,3] are the only arrangements such that exactly 2 sticks are visible.
The visible sticks are underlined.

Example 2:

Input: n = 5, k = 5
Output: 1
Explanation: [1,2,3,4,5] is the only arrangement such that all 5 sticks are visible.
The visible sticks are underlined.

Example 3:

Input: n = 20, k = 11
Output: 647427950
Explanation: There are 647427950 (mod 109 + 7) ways to rearrange the sticks such that exactly 11 sticks are visible.

Constraints:

1 <= n <= 1000
1 <= k <= n

1869. Longer Contiguous Segments of Ones than Zeros

Given a binary string s, return true if the longest contiguous segment of 1's is strictly longer than the longest contiguous segment of 0's in s, or return false otherwise.

For example, in s = "110100010" the longest continuous segment of 1s has length 2, and the longest continuous segment of 0s has length 3.

Note that if there are no 0's, then the longest continuous segment of 0's is considered to have a length 0. The same applies if there is no 1's.

Example 1:

Input: s = "1101"
Output: true
Explanation:
The longest contiguous segment of 1s has length 2: "1101"
The longest contiguous segment of 0s has length 1: "1101"
The segment of 1s is longer, so return true.

Example 2:

Input: s = "111000"
Output: false
Explanation:
The longest contiguous segment of 1s has length 3: "111000"
The longest contiguous segment of 0s has length 3: "111000"
The segment of 1s is not longer, so return false.

Example 3:

Input: s = "110100010"
Output: false
Explanation:
The longest contiguous segment of 1s has length 2: "110100010"
The longest contiguous segment of 0s has length 3: "110100010"
The segment of 1s is not longer, so return false.

Constraints:

1 <= s.length <= 100
s[i] is either '0' or '1'.

1870. Minimum Speed to Arrive on Time

You are given a floating-point number hour, representing the amount of time you have to reach the office. To comm ute to the office, you must take n trains in sequential order. You are also given an integer array dist of length n, wher e dist[i] describes the distance (in kilometers) of the ith train ride.
Each train can only depart at an integer hour, so you may need to wait in between each train ride.

For example, if the 1st train ride takes 1.5 hours, you must wait for an additional 0.5 hours before you can depart on the 2nd train ride at the 2 hour mark.

Return the minimum positive integer speed (in kilometers per hour) that all the trains must travel at for you to reach t he office on time, or -1 if it is impossible to be on time.
Tests are generated such that the answer will not exceed 107 and hour will have at most two digits after the decimal point.

Example 1:

Input: dist = [1,3,2], hour = 6
Output: 1
Explanation: At speed 1:
- The first train ride takes 1/1 = 1 hour.
- Since we are already at an integer hour, we depart immediately at the 1 hour mark. The second train takes 3/1 = 3 h ours.
- Since we are already at an integer hour, we depart immediately at the 4 hour mark. The third train takes 2/1 = 2 hou rs.
- You will arrive at exactly the 6 hour mark.

Example 2:

Input: dist = [1,3,2], hour = 2.7
Output: 3
Explanation: At speed 3:
- The first train ride takes 1/3 = 0.33333 hours.
- Since we are not at an integer hour, we wait until the 1 hour mark to depart. The second train ride takes 3/3 = 1 hou r.
- Since we are already at an integer hour, we depart immediately at the 2 hour mark. The third train takes 2/3 = 0.666 67 hours.
- You will arrive at the 2.66667 hour mark.

Example 3:

Input: dist = [1,3,2], hour = 1.9
Output: -1
Explanation: It is impossible because the earliest the third train can depart is at the 2 hour mark.

Constraints:

n == dist.length
1 <= n <= 105
1 <= dist[i] <= 105
1 <= hour <= 109
There will be at most two digits after the decimal point in hour.

1871. Jump Game VII

You are given a 0-indexed binary string s and two integers minJump and maxJump. In the beginning, you are standi ng at index 0, which is equal to '0'. You can move from index i to index j if the following conditions are fulfilled:

i + minJump <= j <= min(i + maxJump, s.length - 1), and
s[j] == '0'.

Return true if you can reach index s.length - 1 in s, or false otherwise.

Example 1:

Input: s = "011010", minJump = 2, maxJump = 3
Output: true
Explanation:
In the first step, move from index 0 to index 3.
In the second step, move from index 3 to index 5.

Example 2:

Input: s = "01101110", minJump = 2, maxJump = 3
Output: false

Constraints:

2 <= s.length <= 105
s[i] is either '0' or '1'.
s[0] == '0'
1 <= minJump <= maxJump < s.length

1872. Stone Game VIII

Alice and Bob take turns playing a game, with Alice starting first.
There are n stones arranged in a row. On each player's turn, while the number of stones is more than one, they will d o the following:

Choose an integer x > 1, and remove the leftmost x stones from the row.

Add the sum of the removed stones' values to the player's score.
Place a new stone, whose value is equal to that sum, on the left side of the row.

The game stops when only one stone is left in the row.
The score difference between Alice and Bob is (Alice's score - Bob's score). Alice's goal is to maximize the score dif ference, and Bob's goal is the minimize the score difference.
Given an integer array stones of length n where stones[i] represents the value of the ith stone from the left, return the score difference between Alice and Bob if they both play optimally.

Example 1:

Input: stones = [-1,2,-3,4,-5]
Output: 5
Explanation:
- Alice removes the first 4 stones, adds (-1) + 2 + (-3) + 4 = 2 to her score, and places a stone of value 2 on the left. stones = [2,-5].
- Bob removes the first 2 stones, adds 2 + (-5) = -3 to his score, and places a stone of value -3 on the left. stones = [-3].
The difference between their scores is 2 - (-3) = 5.

Example 2:

Input: stones = [7,-6,5,10,5,-2,-6]
Output: 13
Explanation:
- Alice removes all stones, adds 7 + (-6) + 5 + 10 + 5 + (-2) + (-6) = 13 to her score, and places a stone of value 13 on the left. stones = [13].
The difference between their scores is 13 - 0 = 13.

Example 3:

Input: stones = [-10,-12]
Output: -22
Explanation:
- Alice can only make one move, which is to remove both stones. She adds (-10) + (-12) = -22 to her score and places a stone of value -22 on the left. stones = [-22].
The difference between their scores is (-22) - 0 = -22.

Constraints:

n == stones.length
2 <= n <= 105
-104 <= stones[i] <= 104

1876. Substrings of Size Three with Distinct Characters

A string is good if there are no repeated characters.
Given a string s , return the number of good substrings of length three in s .
Note that if there are multiple occurrences of the same substring, every occurrence should be counted.
A substring is a contiguous sequence of characters in a string.

Example 1:

Input: s = "xyzzaz"
Output: 1
Explanation: There are 4 substrings of size 3: "xyz", "yzz", "zza", and "zaz".
The only good substring of length 3 is "xyz".

Example 2:

Input: s = "aababcabc"
Output: 4
Explanation: There are 7 substrings of size 3: "aab", "aba", "bab", "abc", "bca", "cab", and "abc".
The good substrings are "abc", "bca", "cab", and "abc".

Constraints:

1 <= s.length <= 100
s consists of lowercase English letters.

1877. Minimize Maximum Pair Sum in Array

The pair sum of a pair (a,b) is equal to a + b. The maximum pair sum is the largest pair sum in a list of pairs.

For example, if we have pairs (1,5), (2,3), and (4,4), the maximum pair sum would be max(1+5, 2+3, 4+4) = max(6, 5, 8) = 8.

Given an array nums of even length n, pair up the elements of nums into n / 2 pairs such that:

Each element of nums is in exactly one pair, and
The maximum pair sum is minimized.

Return the minimized maximum pair sum after optimally pairing up the elements.

Example 1:

Input: nums = [3,5,2,3]
Output: 7
Explanation: The elements can be paired up into pairs (3,3) and (5,2).
The maximum pair sum is max(3+3, 5+2) = max(6, 7) = 7.

Example 2:

Input: nums = [3,5,4,2,4,6]
Output: 8
Explanation: The elements can be paired up into pairs (3,5), (4,4), and (6,2).
The maximum pair sum is max(3+5, 4+4, 6+2) = max(8, 8, 8) = 8.

Constraints:

n == nums.length
2 <= n <= 105
n is even.
1 <= nums[i] <= 105

1878. Get Biggest Three Rhombus Sums in a Grid

You are given an m x n integer matrix grid .
A rhombus sum is the sum of the elements that form the border of a regular rhombus shape in grid . The rhombus must have the shape of a square rotated 45 degrees with each of the corners centered in a grid cell. Below is an imag e of four valid rhombus shapes with the corresponding colored cells that should be included in each rhombus sum:

Note that the rhombus can have an area of 0, which is depicted by the purple rhombus in the bottom right corner.
Return the biggest three distinct rhombus sums in the grid in descending order. If there are less than three distinct val ues, return all of them.

Example 1:

Input: grid = [[3,4,5,1,3],[3,3,4,2,3],[20,30,200,40,10],[1,5,5,4,1],[4,3,2,2,5]]
Output: [228,216,211]
Explanation: The rhombus shapes for the three biggest distinct rhombus sums are depicted above.
- Blue: 20 + 3 + 200 + 5 = 228
- Red: 200 + 2 + 10 + 4 = 216
- Green: 5 + 200 + 4 + 2 = 211

Example 2:

Input: grid = [[1,2,3],[4,5,6],[7,8,9]]
Output: [20,9,8]
Explanation: The rhombus shapes for the three biggest distinct rhombus sums are depicted above.
- Blue: 4 + 2 + 6 + 8 = 20
- Red: 9 (area 0 rhombus in the bottom right corner)
- Green: 8 (area 0 rhombus in the bottom middle)

Example 3:

Input: grid = [[7,7,7]]
Output: [7]
Explanation: All three possible rhombus sums are the same, so return [7].

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 50
1 <= grid[i][j] <= 105

1879. Minimum XOR Sum of Two Arrays

You are given two integer arrays nums1 and nums2 of length n.
The XOR sum of the two integer arrays is (nums1[0] XOR nums2[0]) + (nums1[1] XOR nums2[1]) + ... + (nums1[n - 1] XOR nums2[n - 1]) (0-indexed).

For example, the XOR sum of [1,2,3] and [3,2,1] is equal to (1 XOR 3) + (2 XOR 2) + (3 XOR 1) = 2 + 0 + 2 = 4.

Rearrange the elements of nums2 such that the resulting XOR sum is minimized.
Return the XOR sum after the rearrangement.

Example 1:

Input: nums1 = [1,2], nums2 = [2,3]
Output: 2
Explanation: Rearrange nums2 so that it becomes [3,2].
The XOR sum is (1 XOR 3) + (2 XOR 2) = 2 + 0 = 2.
Example 2:

Input: nums1 = [1,0,3], nums2 = [5,3,4]
Output: 8
Explanation: Rearrange nums2 so that it becomes [5,4,3].
The XOR sum is (1 XOR 5) + (0 XOR 4) + (3 XOR 3) = 4 + 4 + 0 = 8.

Constraints:

n == nums1.length
n == nums2.length
1 <= n <= 14
0 <= nums1[i], nums2[i] <= 107

1880. Check if Word Equals Summation of Two Words

The letter value of a letter is its position in the alphabet starting from 0 (i.e. 'a' -> 0, 'b' -> 1, 'c' -> 2, etc.).
The numerical value of some string of lowercase English letters s is the concatenation of the letter values of each lett er in s, which is then converted into an integer.

For example, if s = "acb", we concatenate each letter's letter value, resulting in "021". After converting it, we get 21.

You are given three strings firstWord, secondWord, and targetWord, each consisting of lowercase English letters 'a' t hrough 'j' inclusive.
Return true if the summation of the numerical values of firstWord and secondWord equals the numerical value of tar getWord, or false otherwise.

Example 1:

Input: firstWord = "acb", secondWord = "cba", targetWord = "cdb"
Output: true
Explanation:
The numerical value of firstWord is "acb" -> "021" -> 21.
The numerical value of secondWord is "cba" -> "210" -> 210.
The numerical value of targetWord is "cdb" -> "231" -> 231.
We return true because 21 + 210 == 231.

Example 2:

Input: firstWord = "aaa", secondWord = "a", targetWord = "aab"
Output: false
Explanation:
The numerical value of firstWord is "aaa" -> "000" -> 0.
The numerical value of secondWord is "a" -> "0" -> 0.
The numerical value of targetWord is "aab" -> "001" -> 1.
We return false because 0 + 0 != 1.

Example 3:

Input: firstWord = "aaa", secondWord = "a", targetWord = "aaaa"
Output: true
Explanation:
The numerical value of firstWord is "aaa" -> "000" -> 0.
The numerical value of secondWord is "a" -> "0" -> 0.
The numerical value of targetWord is "aaaa" -> "0000" -> 0.
We return true because 0 + 0 == 0.

Constraints:

1 <= firstWord.length, secondWord.length, targetWord.length <= 8

firstWord, secondWord, and targetWord consist of lowercase English letters from 'a' to 'j' inclusive.

1881. Maximum Value after Insertion

You are given a very large integer n, represented as a string, and an integer digit x. The digits in n and the digit x are in the inclusive range [1, 9], and n may represent a negative number.
You want to maximize n's numerical value by inserting x anywhere in the decimal representation of n. You cannot insert x to the left of the negative sign.

For example, if n = 73 and x = 6, it would be best to insert it between 7 and 3, making n = 763.
If n = -55 and x = 2, it would be best to insert it before the first 5, making n = -255.

Return a string representing the maximum value of n after the insertion.

Example 1:

Input: n = "99", x = 9
Output: "999"
Explanation: The result is the same regardless of where you insert 9.

Example 2:

Input: n = "-13", x = 2
Output: "-123"
Explanation: You can make n one of {-213, -123, -132}, and the largest of those three is -123.

Constraints:

1 <= n.length <= 105
1 <= x <= 9
The digits in n are in the range [1, 9].
n is a valid representation of an integer.
In the case of a negative n, it will begin with '-'.

1882. Process Tasks Using Servers

You are given two 0-indexed integer arrays servers and tasks of lengths n and m respectively. servers[i] is the

weight of the i th server, and tasks[j] is the time needed to process the j th task in seconds.
Tasks are assigned to the servers using a task queue. Initially, all servers are free, and the queue is empty.
At second j, the jth task is inserted into the queue (starting with the 0th task being inserted at second 0). As long as t here are free servers and the queue is not empty, the task in the front of the queue will be assigned to a free server wi th the smallest weight, and in case of a tie, it is assigned to a free server with the smallest index.
If there are no free servers and the queue is not empty, we wait until a server becomes free and immediately assign t he next task. If multiple servers become free at the same time, then multiple tasks from the queue will be assigned in order of insertion following the weight and index priorities above.
A server that is assigned task j at second t will be free again at second t + tasks[j].
Build an array ans of length m, where ans[j] is the index of the server the j th task will be assigned to.
Return the array ans .

Example 1:

Input: servers = [3,3,2], tasks = [1,2,3,2,1,2]
Output: [2,2,0,2,1,2]
Explanation: Events in chronological order go as follows:
- At second 0, task 0 is added and processed using server 2 until second 1.
- At second 1, server 2 becomes free. Task 1 is added and processed using server 2 until second 3.
- At second 2, task 2 is added and processed using server 0 until second 5.
- At second 3, server 2 becomes free. Task 3 is added and processed using server 2 until second 5.
- At second 4, task 4 is added and processed using server 1 until second 5.
- At second 5, all servers become free. Task 5 is added and processed using server 2 until second 7.
Example 2:

Input: servers = [5,1,4,3,2], tasks = [2,1,2,4,5,2,1]
Output: [1,4,1,4,1,3,2]
Explanation: Events in chronological order go as follows:
- At second 0, task 0 is added and processed using server 1 until second 2.
- At second 1, task 1 is added and processed using server 4 until second 2.
- At second 2, servers 1 and 4 become free. Task 2 is added and processed using server 1 until second 4.
- At second 3, task 3 is added and processed using server 4 until second 7.
- At second 4, server 1 becomes free. Task 4 is added and processed using server 1 until second 9.
- At second 5, task 5 is added and processed using server 3 until second 7.
- At second 6, task 6 is added and processed using server 2 until second 7.

Constraints:

servers.length == n
tasks.length == m
1 <= n, m <= 2 * 105
1 <= servers[i], tasks[j] <= 2 * 105

1883. Minimum Skips to Arrive at Meeting On Time

You are given an integer hoursBefore, the number of hours you have to travel to your meeting. To arrive at your mee ting, you have to travel through n roads. The road lengths are given as an integer array dist of length n, where dist[i] describes the length of the ith road in kilometers. In addition, you are given an integer speed, which is the speed (in k m/h) you will travel at.
After you travel road i, you must rest and wait for the next integer hour before you can begin traveling on the next ro ad. Note that you do not have to rest after traveling the last road because you are already at the meeting.

For example, if traveling a road takes 1.4 hours, you must wait until the 2 hour mark before traveling the next road. I f traveling a road takes exactly 2 hours, you do not need to wait.

However, you are allowed to skip some rests to be able to arrive on time, meaning you do not need to wait for the ne xt integer hour. Note that this means you may finish traveling future roads at different hour marks.

For example, suppose traveling the first road takes 1.4 hours and traveling the second road takes 0.6 hours. Skipping the rest after the first road will mean you finish traveling the second road right at the 2 hour mark, letting you start tr aveling the third road immediately.

Return the minimum number of skips required to arrive at the meeting on time, or -1 if it is impossible.

Example 1:

Input: dist = [1,3,2], speed = 4, hoursBefore = 2
Output: 1
Explanation:
Without skipping any rests, you will arrive in (1/4 + 3/4) + (3/4 + 1/4) + (2/4) = 2.5 hours.
You can skip the first rest to arrive in ((1/4 + 0) + (3/4 + 0)) + (2/4) = 1.5 hours.
Note that the second rest is shortened because you finish traveling the second road at an integer hour due to skipping the first rest.

Example 2:

Input: dist = [7,3,5,5], speed = 2, hoursBefore = 10
Output: 2
Explanation:
Without skipping any rests, you will arrive in (7/2 + 1/2) + (3/2 + 1/2) + (5/2 + 1/2) + (5/2) = 11.5 hours.
You can skip the first and third rest to arrive in ((7/2 + 0) + (3/2 + 0)) + ((5/2 + 0) + (5/2)) = 10 hours.

Example 3:

Input: dist = [7,3,5,5], speed = 1, hoursBefore = 10
Output: -1
Explanation: It is impossible to arrive at the meeting on time even if you skip all the rests.

Constraints:

n == dist.length
1 <= n <= 1000
1 <= dist[i] <= 105
1 <= speed <= 106
1 <= hoursBefore <= 107

1884. Egg Drop With 2 Eggs and N Floors

You are given two identical eggs and you have access to a building with n floors labeled from 1 to n.
You know that there exists a floor f where 0 <= f <= n such that any egg dropped at a floor higher than f will break, and any egg dropped at or below floor f will not break.
In each move, you may take an unbroken egg and drop it from any floor x (where 1 <= x <= n). If the egg breaks, yo u can no longer use it. However, if the egg does not break, you may reuse it in future moves.
Return the minimum number of moves that you need to determine with certainty what the value of f is.

Example 1:

Input: n = 2
Output: 2
Explanation: We can drop the first egg from floor 1 and the second egg from floor 2.
If the first egg breaks, we know that f = 0.
If the second egg breaks but the first egg didn't, we know that f = 1.
Otherwise, if both eggs survive, we know that f = 2.

Example 2:

Input: n = 100
Output: 14
Explanation: One optimal strategy is:
- Drop the 1st egg at floor 9. If it breaks, we know f is between 0 and 8. Drop the 2nd egg starting from floor 1 and g oing up one at a time to find f within 8 more drops. Total drops is 1 + 8 = 9.
- If the 1st egg does not break, drop the 1st egg again at floor 22. If it breaks, we know f is between 9 and 21. Drop t he 2nd egg starting from floor 10 and going up one at a time to find f within 12 more drops. Total drops is 2 + 12 = 1 4.
- If the 1st egg does not break again, follow a similar process dropping the 1st egg from floors 34, 45, 55, 64, 72, 79, 85, 90, 94, 97, 99, and 100.
Regardless of the outcome, it takes at most 14 drops to determine f.

Constraints:

1 <= n <= 1000

1886. Determine Whether Matrix Can Be Obtained By Rotation

Given two n x n binary matrices mat and target, return true if it is possible to make mat equal to target by rotating m

at in 90-degree increments, or false otherwise.

Example 1:

Input: mat = [[0,1],[1,0]], target = [[1,0],[0,1]]
Output: true
Explanation: We can rotate mat 90 degrees clockwise to make mat equal target.

Example 2:

Input: mat = [[0,1],[1,1]], target = [[1,0],[0,1]]
Output: false
Explanation: It is impossible to make mat equal to target by rotating mat.

Example 3:

Input: mat = [[0,0,0],[0,1,0],[1,1,1]], target = [[1,1,1],[0,1,0],[0,0,0]]
Output: true
Explanation: We can rotate mat 90 degrees clockwise two times to make mat equal target.

Constraints:

n == mat.length == target.length
n == mat[i].length == target[i].length
1 <= n <= 10
mat[i][j] and target[i][j] are either 0 or 1.

1887. Reduction Operations to Make the Array Elements Equal

Given an integer array nums, your goal is to make all elements in nums equal. To complete one operation, follow the se steps:

Find the largest value in nums. Let its index be i (0-indexed) and its value be largest. If there are multiple elements w ith the largest value, pick the smallest i.
Find the next largest value in nums strictly smaller than largest. Let its value be nextLargest.
Reduce nums[i] to nextLargest.

Return the number of operations to make all elements in nums equal.

Example 1:

Input: nums = [5,1,3]

Output: 3
Explanation: It takes 3 operations to make all elements in nums equal:
1. largest = 5 at index 0. nextLargest = 3. Reduce nums[0] to 3. nums = [3,1,3].
2. largest = 3 at index 0. nextLargest = 1. Reduce nums[0] to 1. nums = [1,1,3].
3. largest = 3 at index 2. nextLargest = 1. Reduce nums[2] to 1. nums = [1,1,1].

Example 2:

Input: nums = [1,1,1]
Output: 0
Explanation: All elements in nums are already equal.

Example 3:

Input: nums = [1,1,2,2,3]
Output: 4
Explanation: It takes 4 operations to make all elements in nums equal:
1. largest = 3 at index 4. nextLargest = 2. Reduce nums[4] to 2. nums = [1,1,2,2,2].
2. largest = 2 at index 2. nextLargest = 1. Reduce nums[2] to 1. nums = [1,1,1,2,2].
3. largest = 2 at index 3. nextLargest = 1. Reduce nums[3] to 1. nums = [1,1,1,1,2].
4. largest = 2 at index 4. nextLargest = 1. Reduce nums[4] to 1. nums = [1,1,1,1,1].

Constraints:

1 <= nums.length <= 5 * 104
1 <= nums[i] <= 5 * 104

1888. Minimum Number of Flips to Make the Binary String Alternating

You are given a binary string s. You are allowed to perform two types of operations on the string in any sequence:

Type-1: Remove the character at the start of the string s and append it to the end of the string.
Type-2: Pick any character in s and flip its value, i.e., if its value is '0' it becomes '1' and vice-versa.

Return the minimum number of type-2 operations you need to perform such that s becomes alternating.
The string is called alternating if no two adjacent characters are equal.

For example, the strings "010" and "1010" are alternating, while the string "0100" is not.

Example 1:

Input: s = "111000"
Output: 2
Explanation: Use the first operation two times to make s = "100011".

Then, use the second operation on the third and sixth elements to make s = "101010".

Example 2:

Input: s = "010"
Output: 0
Explanation: The string is already alternating.

Example 3:

Input: s = "1110"
Output: 1
Explanation: Use the second operation on the second element to make s = "1010".

Constraints:

1 <= s.length <= 105
s[i] is either '0' or '1'.

1889. Minimum Space Wasted From Packaging

You have n packages that you are trying to place in boxes, one package in each box. There are m suppliers that each produce boxes of different sizes (with infinite supply). A package can be placed in a box if the size of the package is less than or equal to the size of the box.
The package sizes are given as an integer array packages, where packages[i] is the size of the ith package. The suppli ers are given as a 2D integer array boxes, where boxes[j] is an array of box sizes that the jth supplier produces.
You want to choose a single supplier and use boxes from them such that the total wasted space is minimized. For eac h package in a box, we define the space wasted to be size of the box - size of the package. The total wasted space is t he sum of the space wasted in all the boxes.

For example, if you have to fit packages with sizes [2,3,5] and the supplier offers boxes of sizes [4,8], you can fit the packages of size-2 and size-3 into two boxes of size-4 and the package with size-5 into a box of size-8. This would r esult in a waste of (4-2) + (4-3) + (8-5) = 6.

Return the minimum total wasted space by choosing the box supplier optimally, or -1 if it is impossible to fit all the packages inside boxes. Since the answer may be large, return it modulo 109 + 7.

Example 1:

Input: packages = [2,3,5], boxes = [[4,8],[2,8]]
Output: 6
Explanation: It is optimal to choose the first supplier, using two size-4 boxes and one size-8 box.
The total waste is (4-2) + (4-3) + (8-5) = 6.

Example 2:

Input: packages = [2,3,5], boxes = [[1,4],[2,3],[3,4]]
Output: -1
Explanation: There is no box that the package of size 5 can fit in.

Example 3:

Input: packages = [3,5,8,10,11,12], boxes = [[12],[11,9],[10,5,14]]
Output: 9
Explanation: It is optimal to choose the third supplier, using two size-5 boxes, two size-10 boxes, and two size-14 bo
xes.
The total waste is (5-3) + (5-5) + (10-8) + (10-10) + (14-11) + (14-12) = 9.

Constraints:

n == packages.length
m == boxes.length
1 <= n <= 105
1 <= m <= 105
1 <= packages[i] <= 105
1 <= boxes[j].length <= 105
1 <= boxes[j][k] <= 105
sum(boxes[j].length) <= 105
The elements in boxes[j] are distinct.

1893. Check if All the Integers in a Range Are Covered

You are given a 2D integer array ranges and two integers left and right. Each ranges[i] = [starti, endi] represents an i
nclusive interval between starti and endi.
Return true if each integer in the inclusive range [left, right] is covered by at least one interval in ranges. Return false
otherwise.
An integer x is covered by an interval ranges[i] = [starti, endi] if starti <= x <= endi.

Example 1:

Input: ranges = [[1,2],[3,4],[5,6]], left = 2, right = 5
Output: true
Explanation: Every integer between 2 and 5 is covered:
- 2 is covered by the first range.
- 3 and 4 are covered by the second range.
- 5 is covered by the third range.

Example 2:

Input: ranges = [[1,10],[10,20]], left = 21, right = 21

Output: false
Explanation: 21 is not covered by any range.

Constraints:

1 <= ranges.length <= 50
1 <= starti <= endi <= 50
1 <= left <= right <= 50

1894. Find the Student that Will Replace the Chalk

There are n students in a class numbered from 0 to n - 1. The teacher will give each student a problem starting with t he student number 0, then the student number 1, and so on until the teacher reaches the student number n - 1. After t hat, the teacher will restart the process, starting with the student number 0 again.
You are given a 0-indexed integer array chalk and an integer k. There are initially k pieces of chalk. When the stude nt number i is given a problem to solve, they will use chalk[i] pieces of chalk to solve that problem. However, if the current number of chalk pieces is strictly less than chalk[i], then the student number i will be asked to replace the cha lk.
Return the index of the student that will replace the chalk.

Example 1:

Input: chalk = [5,1,5], k = 22
Output: 0
Explanation: The students go in turns as follows:
- Student number 0 uses 5 chalk, so k = 17.
- Student number 1 uses 1 chalk, so k = 16.
- Student number 2 uses 5 chalk, so k = 11.
- Student number 0 uses 5 chalk, so k = 6.
- Student number 1 uses 1 chalk, so k = 5.
- Student number 2 uses 5 chalk, so k = 0.
Student number 0 does not have enough chalk, so they will have to replace it.
Example 2:

Input: chalk = [3,4,1,2], k = 25
Output: 1
Explanation: The students go in turns as follows:
- Student number 0 uses 3 chalk so k = 22.
- Student number 1 uses 4 chalk so k = 18.
- Student number 2 uses 1 chalk so k = 17.
- Student number 3 uses 2 chalk so k = 15.
- Student number 0 uses 3 chalk so k = 12.
- Student number 1 uses 4 chalk so k = 8.
- Student number 2 uses 1 chalk so k = 7.
- Student number 3 uses 2 chalk so k = 5.

- Student number 0 uses 3 chalk so k = 2.
Student number 1 does not have enough chalk, so they will have to replace it.

Constraints:

chalk.length == n
1 <= n <= 105
1 <= chalk[i] <= 105
1 <= k <= 109

1895. Largest Magic Square

A k x k magic square is a k x k grid filled with integers such that every row sum, every column sum, and both diago nal sums are all equal. The integers in the magic square do not have to be distinct. Every 1 x 1 grid is trivially a magi c square.
Given an m x n integer grid, return the size (i.e., the side length k) of the largest magic square that can be found with in this grid.

Example 1:

Input: grid = [[7,1,4,5,6],[2,5,1,6,4],[1,5,4,3,2],[1,2,7,3,4]]
Output: 3
Explanation: The largest magic square has a size of 3.
Every row sum, column sum, and diagonal sum of this magic square is equal to 12.
- Row sums: 5+1+6 = 5+4+3 = 2+7+3 = 12
- Column sums: 5+5+2 = 1+4+7 = 6+3+3 = 12
- Diagonal sums: 5+4+3 = 6+4+2 = 12

Example 2:

Input: grid = [[5,1,3,1],[9,3,3,1],[1,3,3,8]]
Output: 2

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 50
1 <= grid[i][j] <= 106

1896. Minimum Cost to Change the Final Value of Expression

You are given a valid boolean expression as a string expression consisting of the characters '1','0','&' (bitwise AND o perator),'|' (bitwise OR operator),'(', and ')'.

For example, "()1|1" and "(1)&()" are not valid while "1", "(((1))|(0))", and "1|(0&(1))" are valid expressions.

Return the minimum cost to change the final value of the expression.

For example, if expression = "1|1|(0&0)&1", its value is 1|1|(0&0)&1 = 1|1|0&1 = 1|0&1 = 1&1 = 1. We want to app ly operations so that the new expression evaluates to 0.

The cost of changing the final value of an expression is the number of operations performed on the expression. The t ypes of operations are described as follows:

Turn a '1' into a '0'.
Turn a '0' into a '1'.
Turn a '&' into a '|'.
Turn a '|' into a '&'.

Note: '&' does not take precedence over '|' in the order of calculation. Evaluate parentheses first, then in left-to-right order.

Example 1:

Input: expression = "1&(0|1)"
Output: 1
Explanation: We can turn "1&(0|1)" into "1&(0&1)" by changing the '|' to a '&' using 1 operation.
The new expression evaluates to 0.

Example 2:

Input: expression = "(0&0)&(0&0&0)"
Output: 3
Explanation: We can turn "(0&0)&(0&0&0)" into "(0|1)|(0&0&0)" using 3 operations.
The new expression evaluates to 1.

Example 3:

Input: expression = "(0|(1|0&1))"
Output: 1
Explanation: We can turn "(0|(1|0&1))" into "(0|(0|0&1))" using 1 operation.
The new expression evaluates to 0.

Constraints:

1 <= expression.length <= 105
expression only contains '1','0','&','|','(', and ')'

All parentheses are properly matched.
There will be no empty parentheses (i.e: "()" is not a substring of expression).

1897. Redistribute Characters to Make All Strings Equal

You are given an array of strings words (0-indexed).
In one operation, pick two distinct indices i and j, where words[i] is a non-empty string, and move any character fro m words[i] to any position in words[j].
Return true if you can make every string in words equal using any number of operations, and false otherwise.

Example 1:

Input: words = ["abc","aabc","bc"]
Output: true
Explanation: Move the first 'a' in words[1] to the front of words[2],
to make words[1] = "abc" and words[2] = "abc".
All the strings are now equal to "abc", so return true.

Example 2:

Input: words = ["ab","a"]
Output: false
Explanation: It is impossible to make all the strings equal using the operation.

Constraints:

1 <= words.length <= 100
1 <= words[i].length <= 100
words[i] consists of lowercase English letters.

1898. Maximum Number of Removable Characters

You are given two strings s and p where p is a subsequence of s. You are also given a distinct 0-indexed integer arra y removable containing a subset of indices of s (s is also 0-indexed).
You want to choose an integer k (0 <= k <= removable.length) such that, after removing k characters from s using th e first k indices in removable, p is still a subsequence of s. More formally, you will mark the character at s[removabl e[i]] for each 0 <= i < k, then remove all marked characters and check if p is still a subsequence.

Return the maximum k you can choose such that p is still a subsequence of s after the removals.
A subsequence of a string is a new string generated from the original string with some characters (can be none) delet ed without changing the relative order of the remaining characters.

Example 1:

Input: s = "abcacb", p = "ab", removable = [3,1,0]
Output: 2
Explanation: After removing the characters at indices 3 and 1, "abcacb" becomes "accb".
"ab" is a subsequence of "accb".
If we remove the characters at indices 3, 1, and 0, "abcacb" becomes "ccb", and "ab" is no longer a subsequence.
Hence, the maximum k is 2.

Example 2:

Input: s = "abcbddddd", p = "abcd", removable = [3,2,1,4,5,6]
Output: 1
Explanation: After removing the character at index 3, "abcbddddd" becomes "abcddddd".
"abcd" is a subsequence of "abcddddd".

Example 3:

Input: s = "abcab", p = "abc", removable = [0,1,2,3,4]
Output: 0
Explanation: If you remove the first index in the array removable, "abc" is no longer a subsequence.

Constraints:

1 <= p.length <= s.length <= 105
0 <= removable.length < s.length
0 <= removable[i] < s.length
p is a subsequence of s.
s and p both consist of lowercase English letters.
The elements in removable are distinct.

1899. Merge Triplets to Form Target Triplet

A triplet is an array of three integers. You are given a 2D integer array triplets, where triplets[i] = [ai, bi, ci] describe s the ith triplet. You are also given an integer array target = [x, y, z] that describes the triplet you want to obtain.
To obtain target, you may apply the following operation on triplets any number of times (possibly zero):

Choose two indices (0-indexed) i and j (i != j) and update triplets[j] to become [max(ai, aj), max(bi, bj), max(ci, cj)].

For example, if triplets[i] = [2, 5, 3] and triplets[j] = [1, 7, 5], triplets[j] will be updated to [max(2, 1), max(5, 7), ma

x(3, 5)] = [2, 7, 5].

Return true if it is possible to obtain the target triplet [x, y, z] as an element of triplets, or false otherwise.

Example 1:

Input: triplets = [[2,5,3],[1,8,4],[1,7,5]], target = [2,7,5]
Output: true
Explanation: Perform the following operations:
- Choose the first and last triplets [[2,5,3],[1,8,4],[1,7,5]]. Update the last triplet to be [max(2,1), max(5,7), max(3,5)] = [2,7,5]. triplets = [[2,5,3],[1,8,4],[2,7,5]]
The target triplet [2,7,5] is now an element of triplets.

Example 2:

Input: triplets = [[1,3,4],[2,5,8]], target = [2,5,8]
Output: true
Explanation: The target triplet [2,5,8] is already an element of triplets.

Example 3:

Input: triplets = [[2,5,3],[2,3,4],[1,2,5],[5,2,3]], target = [5,5,5]
Output: true
Explanation: Perform the following operations:
- Choose the first and third triplets [[2,5,3],[2,3,4],[1,2,5],[5,2,3]]. Update the third triplet to be [max(2,1), max(5,2), max(3,5)] = [2,5,5]. triplets = [[2,5,3],[2,3,4],[2,5,5],[5,2,3]].
- Choose the third and fourth triplets [[2,5,3],[2,3,4],[2,5,5],[5,2,3]]. Update the fourth triplet to be [max(2,5), max(5,2), max(5,3)] = [5,5,5]. triplets = [[2,5,3],[2,3,4],[2,5,5],[5,5,5]].
The target triplet [5,5,5] is now an element of triplets.

Example 4:

Input: triplets = [[3,4,5],[4,5,6]], target = [3,2,5]
Output: false
Explanation: It is impossible to have [3,2,5] as an element because there is no 2 in any of the triplets.

Constraints:

1 <= triplets.length <= 105
triplets[i].length == target.length == 3
1 <= ai, bi, ci, x, y, z <= 1000

1900. The Earliest and Latest Rounds Where Players Compete

There is a tournament where n players are participating. The players are standing in a single row and are numbered f rom 1 to n based on their initial standing position (player 1 is the first player in the row, player 2 is the second player in the row, etc.).
The tournament consists of multiple rounds (starting from round number 1). In each round, the ith player from the fr ont of the row competes against the ith player from the end of the row, and the winner advances to the next round. W hen the number of players is odd for the current round, the player in the middle automatically advances to the next ro und.

For example, if the row consists of players 1, 2, 4, 6, 7

Player 1 competes against player 7.
Player 2 competes against player 6.
Player 4 automatically advances to the next round.

After each round is over, the winners are lined back up in the row based on the original ordering assigned to them ini tially (ascending order).
The players numbered firstPlayer and secondPlayer are the best in the tournament. They can win against any other pl ayer before they compete against each other. If any two other players compete against each other, either of them mig ht win, and thus you may choose the outcome of this round.
Given the integers n, firstPlayer, and secondPlayer, return an integer array containing two values, the earliest possibl e round number and the latest possible round number in which these two players will compete against each other, res pectively.

Example 1:

Input: n = 11, firstPlayer = 2, secondPlayer = 4
Output: [3,4]
Explanation:
One possible scenario which leads to the earliest round number:
First round: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Second round: 2, 3, 4, 5, 6, 11
Third round: 2, 3, 4
One possible scenario which leads to the latest round number:
First round: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
Second round: 1, 2, 3, 4, 5, 6
Third round: 1, 2, 4
Fourth round: 2, 4

Example 2:

Input: n = 5, firstPlayer = 1, secondPlayer = 5
Output: [1,1]
Explanation: The players numbered 1 and 5 compete in the first round.
There is no way to make them compete in any other round.

Constraints:

$2 <= n <= 28$
$1 <= firstPlayer < secondPlayer <= n$

1901. Find a Peak Element II

A peak element in a 2D grid is an element that is strictly greater than all of its adjacent neighbors to the left, right, to p, and bottom.
Given a 0-indexed m x n matrix mat where no two adjacent cells are equal, find any peak element mat[i][j] and retur n the length 2 array [i,j].
You may assume that the entire matrix is surrounded by an outer perimeter with the value -1 in each cell.
You must write an algorithm that runs in O(m log(n)) or O(n log(m)) time.

Example 1:

Input: mat = [[1,4],[3,2]]
Output: [0,1]
Explanation: Both 3 and 4 are peak elements so [1,0] and [0,1] are both acceptable answers.

Example 2:

Input: mat = [[10,20,15],[21,30,14],[7,16,32]]
Output: [1,1]
Explanation: Both 30 and 32 are peak elements so [1,1] and [2,2] are both acceptable answers.

Constraints:

m == mat.length
n == mat[i].length
1 <= m, n <= 500
1 <= mat[i][j] <= 105
No two adjacent cells are equal.

1903. Largest Odd Number in String

You are given a string num, representing a large integer. Return the largest-valued odd integer (as a string) that is a n on-empty substring of num, or an empty string "" if no odd integer exists.
A substring is a contiguous sequence of characters within a string.

Example 1:

Input: num = "52"
Output: "5"
Explanation: The only non-empty substrings are "5", "2", and "52". "5" is the only odd number.

Example 2:

Input: num = "4206"
Output: ""
Explanation: There are no odd numbers in "4206".

Example 3:

Input: num = "35427"
Output: "35427"
Explanation: "35427" is already an odd number.

Constraints:

1 <= num.length <= 105
num only consists of digits and does not contain any leading zeros.

1904. The Number of Full Rounds You Have Played

A new online video game has been released, and in this video game, there are 15-minute rounds scheduled every qua rter-hour period. This means that at HH:00, HH:15, HH:30 and HH:45, a new round starts, where HH represents an i nteger number from 00 to 23. A 24-hour clock is used, so the earliest time in the day is 00:00 and the latest is 23:59. Given two strings startTime and finishTime in the format "HH:MM" representing the exact time you started and fini shed playing the game, respectively, calculate the number of full rounds that you played during your game session.

For example, if startTime = "05:20" and finishTime = "05:59" this means you played only one full round from 05:30 to 05:45. You did not play the full round from 05:15 to 05:30 because you started after the round began, and you did not play the full round from 05:45 to 06:00 because you stopped before the round ended.

If finishTime is earlier than startTime, this means you have played overnight (from startTime to the midnight and fro m midnight to finishTime).
Return the number of full rounds that you have played if you had started playing at startTime and finished at finishTi me.

Example 1:

Input: startTime = "12:01", finishTime = "12:44"
Output: 1
Explanation: You played one full round from 12:15 to 12:30.

You did not play the full round from 12:00 to 12:15 because you started playing at 12:01 after it began.
You did not play the full round from 12:30 to 12:45 because you stopped playing at 12:44 before it ended.

Example 2:

Input: startTime = "20:00", finishTime = "06:00"
Output: 40
Explanation: You played 16 full rounds from 20:00 to 00:00 and 24 full rounds from 00:00 to 06:00.
16 + 24 = 40.

Example 3:

Input: startTime = "00:00", finishTime = "23:59"
Output: 95
Explanation: You played 4 full rounds each hour except for the last hour where you played 3 full rounds.

Constraints:

startTime and finishTime are in the format HH:MM.
00 <= HH <= 23
00 <= MM <= 59
startTime and finishTime are not equal.

1905. Count Sub Islands

You are given two m x n binary matrices grid1 and grid2 containing only 0's (representing water) and 1's (representi ng land). An island is a group of 1's connected 4-directionally (horizontal or vertical). Any cells outside of the grid a re considered water cells.
An island in grid2 is considered a sub-island if there is an island in grid1 that contains all the cells that make up this i sland in grid2.
Return the number of islands in grid2 that are considered sub-islands.

Example 1:

Input: grid1 = [[1,1,1,0,0],[0,1,1,1,1],[0,0,0,0,0],[1,0,0,0,0],[1,1,0,1,1]], grid2 = [[1,1,1,0,0],[0,0,1,1,1],[0,1,0,0,0],[1, 0,1,1,0],[0,1,0,1,0]]
Output: 3
Explanation: In the picture above, the grid on the left is grid1 and the grid on the right is grid2.
The 1s colored red in grid2 are those considered to be part of a sub-island. There are three sub-islands.

Example 2:

Input: grid1 = [[1,0,1,0,1],[1,1,1,1,1],[0,0,0,0,0],[1,1,1,1,1],[1,0,1,0,1]], grid2 = [[0,0,0,0,0],[1,1,1,1,1],[0,1,0,1,0],[0,

1,0,1,0],[1,0,0,0,1]]
Output: 2
Explanation: In the picture above, the grid on the left is grid1 and the grid on the right is grid2.
The 1s colored red in grid2 are those considered to be part of a sub-island. There are two sub-islands.

Constraints:

m == grid1.length == grid2.length
n == grid1[i].length == grid2[i].length
1 <= m, n <= 500
grid1[i][j] and grid2[i][j] are either 0 or 1.

1906. Minimum Absolute Difference Queries

The minimum absolute difference of an array a is defined as the minimum value of |a[i] - a[j]|, where 0 <= i < j < a.length and a[i] != a[j]. If all elements of a are the same, the minimum absolute difference is -1.

For example, the minimum absolute difference of the array [5,2,3,7,2] is |2 - 3| = 1. Note that it is not 0 because a[i] and a[j] must be different.

You are given an integer array nums and the array queries where queries[i] = [li, ri]. For each query i, compute the minimum absolute difference of the subarray nums[li...ri] containing the elements of nums between the 0-based indices li and ri (inclusive).
Return an array ans where ans[i] is the answer to the ith query.
A subarray is a contiguous sequence of elements in an array.
The value of |x| is defined as:

x if x >= 0.
-x if x < 0.

Example 1:

Input: nums = [1,3,4,8], queries = [[0,1],[1,2],[2,3],[0,3]]
Output: [2,1,4,1]
Explanation: The queries are processed as follows:
- queries[0] = [0,1]: The subarray is [1,3] and the minimum absolute difference is |1-3| = 2.
- queries[1] = [1,2]: The subarray is [3,4] and the minimum absolute difference is |3-4| = 1.
- queries[2] = [2,3]: The subarray is [4,8] and the minimum absolute difference is |4-8| = 4.
- queries[3] = [0,3]: The subarray is [1,3,4,8] and the minimum absolute difference is |3-4| = 1.

Example 2:

Input: nums = [4,5,2,2,7,10], queries = [[2,3],[0,2],[0,5],[3,5]]
Output: [-1,1,1,3]

Explanation: The queries are processed as follows:
- queries[0] = [2,3]: The subarray is [2,2] and the minimum absolute difference is -1 because all the elements are the same.
- queries[1] = [0,2]: The subarray is [4,5,2] and the minimum absolute difference is |4-5| = 1.
- queries[2] = [0,5]: The subarray is [4,5,2,2,7,10] and the minimum absolute difference is |4-5| = 1.
- queries[3] = [3,5]: The subarray is [2,7,10] and the minimum absolute difference is |7-10| = 3.

Constraints:

2 <= nums.length <= 105
1 <= nums[i] <= 100
1 <= queries.length <= 2 * 104
0 <= li < ri < nums.length

1909. Remove One Element to Make the Array Strictly Increasing

Given a 0-indexed integer array nums, return true if it can be made strictly increasing after removing exactly one element, or false otherwise. If the array is already strictly increasing, return true.
The array nums is strictly increasing if nums[i - 1] < nums[i] for each index (1 <= i < nums.length).

Example 1:

Input: nums = [1,2,10,5,7]
Output: true
Explanation: By removing 10 at index 2 from nums, it becomes [1,2,5,7].
[1,2,5,7] is strictly increasing, so return true.

Example 2:

Input: nums = [2,3,1,2]
Output: false
Explanation:
[3,1,2] is the result of removing the element at index 0.
[2,1,2] is the result of removing the element at index 1.
[2,3,2] is the result of removing the element at index 2.
[2,3,1] is the result of removing the element at index 3.
No resulting array is strictly increasing, so return false.
Example 3:

Input: nums = [1,1,1]
Output: false
Explanation: The result of removing any element is [1,1].
[1,1] is not strictly increasing, so return false.

Example 4:

Input: nums = [1,2,3]
Output: true
Explanation: [1,2,3] is already strictly increasing, so return true.

Constraints:

2 <= nums.length <= 1000
1 <= nums[i] <= 1000

1910. Remove All Occurrences of a Substring

Given two strings s and part, perform the following operation on s until all occurrences of the substring part are rem oved:

Find the leftmost occurrence of the substring part and remove it from s.

Return s after removing all occurrences of part.
A substring is a contiguous sequence of characters in a string.

Example 1:

Input: s = "daabcbaabcbc", part = "abc"
Output: "dab"
Explanation: The following operations are done:
- s = "daabcbaabcbc", remove "abc" starting at index 2, so s = "dabaabcbc".
- s = "dabaabcbc", remove "abc" starting at index 4, so s = "dababc".
- s = "dababc", remove "abc" starting at index 3, so s = "dab".
Now s has no occurrences of "abc".

Example 2:

Input: s = "axxxxyyyyb", part = "xy"
Output: "ab"
Explanation: The following operations are done:
- s = "axxxxyyyyb", remove "xy" starting at index 4 so s = "axxxyyyb".
- s = "axxxyyyb", remove "xy" starting at index 3 so s = "axxyyb".
- s = "axxyyb", remove "xy" starting at index 2 so s = "axyb".
- s = "axyb", remove "xy" starting at index 1 so s = "ab".
Now s has no occurrences of "xy".

Constraints:

1 <= s.length <= 1000

1 <= part.length <= 1000
s and part consists of lowercase English letters.

1911. Maximum Alternating Subsequence Sum

The alternating sum of a 0-indexed array is defined as the sum of the elements at even indices minus the sum of the e lements at odd indices.

For example, the alternating sum of [4,2,5,3] is (4 + 5) - (2 + 3) = 4.

Given an array nums, return the maximum alternating sum of any subsequence of nums (after reindexing the element s of the subsequence).

A subsequence of an array is a new array generated from the original array by deleting some elements (possibly non e) without changing the remaining elements' relative order. For example, [2,7,4] is a subsequence of [4,2,3,7,2,1,4] (t he underlined elements), while [2,4,2] is not.

Example 1:

Input: nums = [4,2,5,3]
Output: 7
Explanation: It is optimal to choose the subsequence [4,2,5] with alternating sum (4 + 5) - 2 = 7.

Example 2:

Input: nums = [5,6,7,8]
Output: 8
Explanation: It is optimal to choose the subsequence [8] with alternating sum 8.

Example 3:

Input: nums = [6,2,1,2,4,5]
Output: 10
Explanation: It is optimal to choose the subsequence [6,1,5] with alternating sum (6 + 5) - 1 = 10.

Constraints:

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

1912. Design Movie Rental System

You have a movie renting company consisting of n shops. You want to implement a renting system that supports sea rching for, booking, and returning movies. The system should also support generating a report of the currently rented movies.
Each movie is given as a 2D integer array entries where entries[i] = [shopi, moviei, pricei] indicates that there is a co py of movie moviei at shop shopi with a rental price of pricei. Each shop carries at most one copy of a movie moviei .
The system should support the following functions:

Search: Finds the cheapest 5 shops that have an unrented copy of a given movie. The shops should be sorted by price in ascending order, and in case of a tie, the one with the smaller shopi should appear first. If there are less than 5 ma tching shops, then all of them should be returned. If no shop has an unrented copy, then an empty list should be retur ned.
Rent: Rents an unrented copy of a given movie from a given shop.
Drop: Drops off a previously rented copy of a given movie at a given shop.
Report: Returns the cheapest 5 rented movies (possibly of the same movie ID) as a 2D list res where res[j] = [shopj, moviej] describes that the jth cheapest rented movie moviej was rented from the shop shopj. The movies in res shoul d be sorted by price in ascending order, and in case of a tie, the one with the smaller shopj should appear first, and if there is still tie, the one with the smaller moviej should appear first. If there are fewer than 5 rented movies, then all of them should be returned. If no movies are currently being rented, then an empty list should be returned.

Implement the MovieRentingSystem class:

MovieRentingSystem(int n, int[][] entries) Initializes the MovieRentingSystem object with n shops and the movies i n entries.
List<Integer> search(int movie) Returns a list of shops that have an unrented copy of the given movie as described a bove.
void rent(int shop, int movie) Rents the given movie from the given shop.
void drop(int shop, int movie) Drops off a previously rented movie at the given shop.
List<List<Integer>> report() Returns a list of cheapest rented movies as described above.

Note: The test cases will be generated such that rent will only be called if the shop has an unrented copy of the movi e, and drop will only be called if the shop had previously rented out the movie.

Example 1:

Input
["MovieRentingSystem", "search", "rent", "rent", "report", "drop", "search"]
[[3, [[0, 1, 5], [0, 2, 6], [0, 3, 7], [1, 1, 4], [1, 2, 7], [2, 1, 5]]], [1], [0, 1], [1, 2], [], [1, 2], [2]]
Output
[null, [1, 0, 2], null, null, [[0, 1], [1, 2]], null, [0, 1]]

Explanation
MovieRentingSystem movieRentingSystem = new MovieRentingSystem(3, [[0, 1, 5], [0, 2, 6], [0, 3, 7], [1, 1, 4], [1, 2, 7], [2, 1, 5]]);
movieRentingSystem.search(1); // return [1, 0, 2], Movies of ID 1 are unrented at shops 1, 0, and 2. Shop 1 is cheap est; shop 0 and 2 are the same price, so order by shop number.
movieRentingSystem.rent(0, 1); // Rent movie 1 from shop 0. Unrented movies at shop 0 are now [2,3].
movieRentingSystem.rent(1, 2); // Rent movie 2 from shop 1. Unrented movies at shop 1 are now [1].
movieRentingSystem.report();   // return [[0, 1], [1, 2]]. Movie 1 from shop 0 is cheapest, followed by movie 2 from

shop 1.
movieRentingSystem.drop(1, 2); // Drop off movie 2 at shop 1. Unrented movies at shop 1 are now [1,2].
movieRentingSystem.search(2); // return [0, 1]. Movies of ID 2 are unrented at shops 0 and 1. Shop 0 is cheapest, fo
llowed by shop 1.

Constraints:

1 <= n <= 3 * 105
1 <= entries.length <= 105
0 <= shopi < n
1 <= moviei, pricei <= 104
Each shop carries at most one copy of a movie moviei.
At most 105 calls in total will be made to search, rent, drop and report.

1913. Maximum Product Difference Between Two Pairs

The product difference between two pairs (a, b) and (c, d) is defined as (a * b) - (c * d).

For example, the product difference between (5, 6) and (2, 7) is (5 * 6) - (2 * 7) = 16.

Given an integer array nums, choose four distinct indices w, x, y, and z such that the product difference between pair
s (nums[w], nums[x]) and (nums[y], nums[z]) is maximized.
Return the maximum such product difference.

Example 1:

Input: nums = [5,6,2,7,4]
Output: 34
Explanation: We can choose indices 1 and 3 for the first pair (6, 7) and indices 2 and 4 for the second pair (2, 4).
The product difference is (6 * 7) - (2 * 4) = 34.

Example 2:

Input: nums = [4,2,5,9,7,4,8]
Output: 64
Explanation: We can choose indices 3 and 6 for the first pair (9, 8) and indices 1 and 5 for the second pair (2, 4).
The product difference is (9 * 8) - (2 * 4) = 64.

Constraints:

4 <= nums.length <= 104
1 <= nums[i] <= 104

1914. Cyclically Rotating a Grid

You are given an m x n integer matrix grid , where m and n are both even integers, and an integer k.
The matrix is composed of several layers, which is shown in the below image, where each color is its own layer:

A cyclic rotation of the matrix is done by cyclically rotating each layer in the matrix. To cyclically rotate a layer onc e, each element in the layer will take the place of the adjacent element in the counter-clockwise direction. An exampl e rotation is shown below:

Return the matrix after applying k cyclic rotations to it.

Example 1:

Input: grid = [[40,10],[30,20]], k = 1
Output: [[10,20],[40,30]]
Explanation: The figures above represent the grid at every state.

Example 2:

Input: grid = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]], k = 2
Output: [[3,4,8,12],[2,11,10,16],[1,7,6,15],[5,9,13,14]]
Explanation: The figures above represent the grid at every state.

Constraints:

m == grid.length
n == grid[i].length
2 <= m, n <= 50
Both m and n are even integers.
1 <= grid[i][j] <= 5000
1 <= k <= 109

1915. Number of Wonderful Substrings

A wonderful string is a string where at most one letter appears an odd number of times.

For example, "ccjjc" and "abab" are wonderful, but "ab" is not.

Given a string word that consists of the first ten lowercase English letters ('a' through 'j'), return the number of wond

erful non-empty substrings in word. If the same substring appears multiple times in word, then count each occurrenc
e separately.
A substring is a contiguous sequence of characters in a string.

Example 1:

Input: word = "aba"
Output: 4
Explanation: The four wonderful substrings are underlined below:
- "aba" -> "a"
- "aba" -> "b"
- "aba" -> "a"
- "aba" -> "aba"

Example 2:

Input: word = "aabb"
Output: 9
Explanation: The nine wonderful substrings are underlined below:
- "aabb" -> "a"
- "aabb" -> "aa"
- "aabb" -> "aab"
- "aabb" -> "aabb"
- "aabb" -> "a"
- "aabb" -> "abb"
- "aabb" -> "b"
- "aabb" -> "bb"
- "aabb" -> "b"

Example 3:

Input: word = "he"
Output: 2
Explanation: The two wonderful substrings are underlined below:
- "he" -> "h"
- "he" -> "e"

Constraints:

1 <= word.length <= 105
word consists of lowercase English letters from 'a' to 'j'.

1916. Count Ways to Build Rooms in an Ant Colony

You are an ant tasked with adding n new rooms numbered 0 to n-1 to your colony. You are given the expansion plan
as a 0-indexed integer array of length n, prevRoom, where prevRoom[i] indicates that you must build room prevRo

om[i] before building room i, and these two rooms must be connected directly. Room 0 is already built, so prevRoo m[0] = -1. The expansion plan is given such that once all the rooms are built, every room will be reachable from roo m 0.
You can only build one room at a time, and you can travel freely between rooms you have already built only if they are connected. You can choose to build any room as long as its previous room is already built.
Return the number of different orders you can build all the rooms in. Since the answer may be large, return it modul o 109 + 7.

Example 1:

Input: prevRoom = [-1,0,1]
Output: 1
Explanation: There is only one way to build the additional rooms: 0 → 1 → 2

Example 2:

Input: prevRoom = [-1,0,0,1,2]
Output: 6
Explanation:
The 6 ways are:
0 → 1 → 3 → 2 → 4
0 → 2 → 4 → 1 → 3
0 → 1 → 2 → 3 → 4
0 → 1 → 2 → 4 → 3
0 → 2 → 1 → 3 → 4
0 → 2 → 1 → 4 → 3

Constraints:

n == prevRoom.length
2 <= n <= 105
prevRoom[0] == -1
0 <= prevRoom[i] < n for all 1 <= i < n
Every room is reachable from room 0 once all the rooms are built.

1920. Build Array from Permutation

Given a zero-based permutation nums (0-indexed), build an array ans of the same length where ans[i] = nums[nums[i]] for each 0 <= i < nums.length and return it.
A zero-based permutation nums is an array of distinct integers from 0 to nums.length - 1 (inclusive).

Example 1:

Input: nums = [0,2,1,5,3,4]

Output: [0,1,2,4,5,3]
Explanation: The array ans is built as follows:
ans = [nums[nums[0]], nums[nums[1]], nums[nums[2]], nums[nums[3]], nums[nums[4]], nums[nums[5]]]
= [nums[0], nums[2], nums[1], nums[5], nums[3], nums[4]]
= [0,1,2,4,5,3]
Example 2:

Input: nums = [5,0,1,2,3,4]
Output: [4,5,0,1,2,3]
Explanation: The array ans is built as follows:
ans = [nums[nums[0]], nums[nums[1]], nums[nums[2]], nums[nums[3]], nums[nums[4]], nums[nums[5]]]
= [nums[5], nums[0], nums[1], nums[2], nums[3], nums[4]]
= [4,5,0,1,2,3]

Constraints:

1 <= nums.length <= 1000
0 <= nums[i] < nums.length
The elements in nums are distinct.

Follow-up: Can you solve it without using an extra space (i.e., O(1) memory)?

1921. Eliminate Maximum Number of Monsters

You are playing a video game where you are defending your city from a group of n monsters. You are given a 0-inde xed integer array dist of size n, where dist[i] is the initial distance in kilometers of the ith monster from the city.
The monsters walk toward the city at a constant speed. The speed of each monster is given to you in an integer array speed of size n, where speed[i] is the speed of the ith monster in kilometers per minute.
You have a weapon that, once fully charged, can eliminate a single monster. However, the weapon takes one minute to charge.The weapon is fully charged at the very start.
You lose when any monster reaches your city. If a monster reaches the city at the exact moment the weapon is fully charged, it counts as a loss, and the game ends before you can use your weapon.
Return the maximum number of monsters that you can eliminate before you lose, or n if you can eliminate all the mo nsters before they reach the city.

Example 1:

Input: dist = [1,3,4], speed = [1,1,1]
Output: 3
Explanation:
In the beginning, the distances of the monsters are [1,3,4]. You eliminate the first monster.
After a minute, the distances of the monsters are [X,2,3]. You eliminate the second monster.
After a minute, the distances of the monsters are [X,X,2]. You eliminate the thrid monster.
All 3 monsters can be eliminated.
Example 2:

Input: dist = [1,1,2,3], speed = [1,1,1,1]
Output: 1
Explanation:
In the beginning, the distances of the monsters are [1,1,2,3]. You eliminate the first monster.
After a minute, the distances of the monsters are [X,0,1,2], so you lose.
You can only eliminate 1 monster.

Example 3:

Input: dist = [3,2,4], speed = [5,3,2]
Output: 1
Explanation:
In the beginning, the distances of the monsters are [3,2,4]. You eliminate the first monster.
After a minute, the distances of the monsters are [X,0,2], so you lose.
You can only eliminate 1 monster.

Constraints:

n == dist.length == speed.length
1 <= n <= 105
1 <= dist[i], speed[i] <= 105

1922. Count Good Numbers

A digit string is good if the digits (0-indexed) at even indices are even and the digits at odd indices are prime (2, 3, 5, or 7).

For example, "2582" is good because the digits (2 and 8) at even positions are even and the digits (5 and 2) at odd po sitions are prime. However, "3245" is not good because 3 is at an even index but is not even.

Given an integer n, return the total number of good digit strings of length n. Since the answer may be large, return it modulo 109 + 7.
A digit string is a string consisting of digits 0 through 9 that may contain leading zeros.

Example 1:

Input: n = 1
Output: 5
Explanation: The good numbers of length 1 are "0", "2", "4", "6", "8".

Example 2:

Input: n = 4
Output: 400

Example 3:

Input: n = 50
Output: 564908303

Constraints:

1 <= n <= 1015

1923. Longest Common Subpath

There is a country of n cities numbered from 0 to n - 1. In this country, there is a road connecting every pair of cities. There are m friends numbered from 0 to m - 1 who are traveling through the country. Each one of them will take a p ath consisting of some cities. Each path is represented by an integer array that contains the visited cities in order. The path may contain a city more than once, but the same city will not be listed consecutively.
Given an integer n and a 2D integer array paths where paths[i] is an integer array representing the path of the ith frie nd, return the length of the longest common subpath that is shared by every friend's path, or 0 if there is no common subpath at all.
A subpath of a path is a contiguous sequence of cities within that path.

Example 1:

Input: n = 5, paths = [[0,1,2,3,4],
[2,3,4],
[4,0,1,2,3]]
Output: 2
Explanation: The longest common subpath is [2,3].

Example 2:

Input: n = 3, paths = [[0],[1],[2]]
Output: 0
Explanation: There is no common subpath shared by the three paths.

Example 3:

Input: n = 5, paths = [[0,1,2,3,4],
[4,3,2,1,0]]
Output: 1
Explanation: The possible longest common subpaths are [0], [1], [2], [3], and [4]. All have a length of 1.

Constraints:

1 <= n <= 105
m == paths.length

2 <= m <= 105
sum(paths[i].length) <= 105
0 <= paths[i][j] < n
The same city is not listed multiple times consecutively in paths[i].

1925. Count Square Sum Triples

A square triple (a,b,c) is a triple where a, b, and c are integers and a2 + b2 = c2.
Given an integer n, return the number of square triples such that 1 <= a, b, c <= n.

Example 1:

Input: n = 5
Output: 2
Explanation: The square triples are (3,4,5) and (4,3,5).

Example 2:

Input: n = 10
Output: 4
Explanation: The square triples are (3,4,5), (4,3,5), (6,8,10), and (8,6,10).

Constraints:

1 <= n <= 250

1926. Nearest Exit from Entrance in Maze

You are given an m x n matrix maze (0-indexed) with empty cells (represented as '.') and walls (represented as '+'). You are also given the entrance of the maze, where entrance = [entrancerow, entrancecol] denotes the row and colu mn of the cell you are initially standing at.
In one step, you can move one cell up, down, left, or right. You cannot step into a cell with a wall, and you cannot st ep outside the maze. Your goal is to find the nearest exit from the entrance. An exit is defined as an empty cell that is at the border of the maze. The entrance does not count as an exit.
Return the number of steps in the shortest path from the entrance to the nearest exit, or -1 if no such path exists.

Example 1:

Input: maze = [["+","+",".","+"],[".",".",".","+"],["+","+","+","."]], entrance = [1,2]
Output: 1
Explanation: There are 3 exits in this maze at [1,0], [0,2], and [2,3].
Initially, you are at the entrance cell [1,2].
- You can reach [1,0] by moving 2 steps left.
- You can reach [0,2] by moving 1 step up.
It is impossible to reach [2,3] from the entrance.
Thus, the nearest exit is [0,2], which is 1 step away.

Example 2:

Input: maze = [["+","+","+"],[".",".","."],["+","+","+"]], entrance = [1,0]
Output: 2
Explanation: There is 1 exit in this maze at [1,2].
[1,0] does not count as an exit since it is the entrance cell.
Initially, you are at the entrance cell [1,0].
- You can reach [1,2] by moving 2 steps right.
Thus, the nearest exit is [1,2], which is 2 steps away.

Example 3:

Input: maze = [[".","+"]], entrance = [0,0]
Output: -1
Explanation: There are no exits in this maze.

Constraints:

maze.length == m
maze[i].length == n
1 <= m, n <= 100
maze[i][j] is either '.' or '+'.
entrance.length == 2
0 <= entrancerow < m
0 <= entrancecol < n
entrance will always be an empty cell.

1927. Sum Game

Alice and Bob take turns playing a game, with Alice starting first.
You are given a string num of even length consisting of digits and '?' characters. On each turn, a player will do the fo llowing if there is still at least one '?' in num:

Choose an index i where num[i] == '?'.
Replace num[i] with any digit between '0' and '9'.

The game ends when there are no more '?' characters in num.
For Bob to win, the sum of the digits in the first half of num must be equal to the sum of the digits in the second half.
For Alice to win, the sums must not be equal.

For example, if the game ended with num = "243801", then Bob wins because 2+4+3 = 8+0+1. If the game ended wi th num = "243803", then Alice wins because 2+4+3 != 8+0+3.

Assuming Alice and Bob play optimally, return true if Alice will win and false if Bob will win.

Example 1:

Input: num = "5023"
Output: false
Explanation: There are no moves to be made.
The sum of the first half is equal to the sum of the second half: 5 + 0 = 2 + 3.

Example 2:

Input: num = "25??"
Output: true
Explanation: Alice can replace one of the '?'s with '9' and it will be impossible for Bob to make the sums equal.

Example 3:

Input: num = "?3295???"
Output: false
Explanation: It can be proven that Bob will always win. One possible outcome is:
- Alice replaces the first '?' with '9'. num = "93295???".
- Bob replaces one of the '?' in the right half with '9'. num = "932959??".
- Alice replaces one of the '?' in the right half with '2'. num = "9329592?".
- Bob replaces the last '?' in the right half with '7'. num = "93295927".
Bob wins because 9 + 3 + 2 + 9 = 5 + 9 + 2 + 7.

Constraints:

2 <= num.length <= 105
num.length is even.
num consists of only digits and '?'.

1928. Minimum Cost to Reach Destination in Time

There is a country of n cities numbered from 0 to n - 1 where all the cities are connected by bi-directional roads. The roads are represented as a 2D integer array edges where edges[i] = [xi, yi, timei] denotes a road between cities xi and yi that takes timei minutes to travel. There may be multiple roads of differing travel times connecting the same two cities, but no road connects a city to itself.
Each time you pass through a city, you must pay a passing fee. This is represented as a 0-indexed integer array passingFees of length n where passingFees[j] is the amount of dollars you must pay when you pass through city j.
In the beginning, you are at city 0 and want to reach city n - 1 in maxTime minutes or less. The cost of your journey is the summation of passing fees for each city that you passed through at some moment of your journey (including the source and destination cities).
Given maxTime, edges, and passingFees, return the minimum cost to complete your journey, or -1 if you cannot complete it within maxTime minutes.

Example 1:

Input: maxTime = 30, edges = [[0,1,10],[1,2,10],[2,5,10],[0,3,1],[3,4,10],[4,5,15]], passingFees = [5,1,2,20,20,3]
Output: 11
Explanation: The path to take is 0 -> 1 -> 2 -> 5, which takes 30 minutes and has $11 worth of passing fees.

Example 2:

Input: maxTime = 29, edges = [[0,1,10],[1,2,10],[2,5,10],[0,3,1],[3,4,10],[4,5,15]], passingFees = [5,1,2,20,20,3]
Output: 48
Explanation: The path to take is 0 -> 3 -> 4 -> 5, which takes 26 minutes and has $48 worth of passing fees.
You cannot take path 0 -> 1 -> 2 -> 5 since it would take too long.

Example 3:

Input: maxTime = 25, edges = [[0,1,10],[1,2,10],[2,5,10],[0,3,1],[3,4,10],[4,5,15]], passingFees = [5,1,2,20,20,3]
Output: -1
Explanation: There is no way to reach city 5 from city 0 within 25 minutes.

Constraints:

1 <= maxTime <= 1000
n == passingFees.length
2 <= n <= 1000
n - 1 <= edges.length <= 1000
0 <= xi, yi <= n - 1
1 <= timei <= 1000
1 <= passingFees[j] <= 1000
The graph may contain multiple edges between two nodes.
The graph does not contain self loops.

1929. Concatenation of Array

Given an integer array nums of length n, you want to create an array ans of length 2n where ans[i] == nums[i] and an s[i + n] == nums[i] for 0 <= i < n (0-indexed).
Specifically, ans is the concatenation of two nums arrays.
Return the array ans.

Example 1:

Input: nums = [1,2,1]
Output: [1,2,1,1,2,1]
Explanation: The array ans is formed as follows:
- ans = [nums[0],nums[1],nums[2],nums[0],nums[1],nums[2]]
- ans = [1,2,1,1,2,1]
Example 2:

Input: nums = [1,3,2,1]
Output: [1,3,2,1,1,3,2,1]
Explanation: The array ans is formed as follows:
- ans = [nums[0],nums[1],nums[2],nums[3],nums[0],nums[1],nums[2],nums[3]]
- ans = [1,3,2,1,1,3,2,1]

Constraints:

n == nums.length
1 <= n <= 1000
1 <= nums[i] <= 1000

1930. Unique Length-3 Palindromic Subsequences

Given a string s, return the number of unique palindromes of length three that are a subsequence of s.
Note that even if there are multiple ways to obtain the same subsequence, it is still only counted once.
A palindrome is a string that reads the same forwards and backwards.
A subsequence of a string is a new string generated from the original string with some characters (can be none) delet ed without changing the relative order of the remaining characters.

For example, "ace" is a subsequence of "abcde".

Example 1:

Input: s = "aabca"
Output: 3
Explanation: The 3 palindromic subsequences of length 3 are:
- "aba" (subsequence of "aabca")

- "aaa" (subsequence of "aabca")
- "aca" (subsequence of "aabca")

Example 2:

Input: s = "adc"
Output: 0
Explanation: There are no palindromic subsequences of length 3 in "adc".

Example 3:

Input: s = "bbcbaba"
Output: 4
Explanation: The 4 palindromic subsequences of length 3 are:
- "bbb" (subsequence of "bbcbaba")
- "bcb" (subsequence of "bbcbaba")
- "bab" (subsequence of "bbcbaba")
- "aba" (subsequence of "bbcbaba")

Constraints:

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

1931. Painting a Grid With Three Different Colors

You are given two integers m and n. Consider an m x n grid where each cell is initially white. You can paint each cel l red, green, or blue. All cells must be painted.
Return the number of ways to color the grid with no two adjacent cells having the same color. Since the answer can be very large, return it modulo 109 + 7.

Example 1:

Input: m = 1, n = 1
Output: 3
Explanation: The three possible colorings are shown in the image above.

Example 2:

Input: m = 1, n = 2
Output: 6
Explanation: The six possible colorings are shown in the image above.

Example 3:

Input: m = 5, n = 5
Output: 580986

Constraints:

1 <= m <= 5
1 <= n <= 1000

1932. Merge BSTs to Create Single BST

You are given n BST (binary search tree) root nodes for n separate BSTs stored in an array trees (0-indexed). Each B ST in trees has at most 3 nodes, and no two roots have the same value. In one operation, you can:

Select two distinct indices i and j such that the value stored at one of the leaves of trees[i] is equal to the root value o f trees[j].
Replace the leaf node in trees[i] with trees[j].
Remove trees[j] from trees.

Return the root of the resulting BST if it is possible to form a valid BST after performing n - 1 operations, or null if i t is impossible to create a valid BST.
A BST (binary search tree) is a binary tree where each node satisfies the following property:

Every node in the node's left subtree has a value strictly less than the node's value.
Every node in the node's right subtree has a value strictly greater than the node's value.

A leaf is a node that has no children.

Example 1:

Input: trees = [[2,1],[3,2,5],[5,4]]
Output: [3,2,5,1,null,4]
Explanation:
In the first operation, pick i=1 and j=0, and merge trees[0] into trees[1].
Delete trees[0], so trees = [[3,2,5,1],[5,4]].

In the second operation, pick i=0 and j=1, and merge trees[1] into trees[0].
Delete trees[1], so trees = [[3,2,5,1,null,4]].

The resulting tree, shown above, is a valid BST, so return its root.
Example 2:

Input: trees = [[5,3,8],[3,2,6]]
Output: []
Explanation:
Pick i=0 and j=1 and merge trees[1] into trees[0].
Delete trees[1], so trees = [[5,3,8,2,6]].

The resulting tree is shown above. This is the only valid operation that can be performed, but the resulting tree is not a valid BST, so return null.

Example 3:

Input: trees = [[5,4],[3]]
Output: []
Explanation: It is impossible to perform any operations.

Example 4:

Input: trees = [[2,1,3]]
Output: [2,1,3]
Explanation: There is only one tree, and it is already a valid BST, so return its root.

Constraints:

n == trees.length
1 <= n <= 5 * 104
The number of nodes in each tree is in the range [1, 3].
Each node in the input may have children but no grandchildren.
No two roots of trees have the same value.
All the trees in the input are valid BSTs.
1 <= TreeNode.val <= 5 * 104.

1935. Maximum Number of Words You Can Type

There is a malfunctioning keyboard where some letter keys do not work. All other keys on the keyboard work proper ly.
Given a string text of words separated by a single space (no leading or trailing spaces) and a string brokenLetters of all distinct letter keys that are broken, return the number of words in text you can fully type using this keyboard.

Example 1:

Input: text = "hello world", brokenLetters = "ad"
Output: 1
Explanation: We cannot type "world" because the 'd' key is broken.

Example 2:

Input: text = "leet code", brokenLetters = "lt"
Output: 1
Explanation: We cannot type "leet" because the 'l' and 't' keys are broken.

Example 3:

Input: text = "leet code", brokenLetters = "e"
Output: 0
Explanation: We cannot type either word because the 'e' key is broken.

Constraints:

1 <= text.length <= 104
0 <= brokenLetters.length <= 26
text consists of words separated by a single space without any leading or trailing spaces.
Each word only consists of lowercase English letters.
brokenLetters consists of distinct lowercase English letters.

1936. Add Minimum Number of Rungs

You are given a strictly increasing integer array rungs that represents the height of rungs on a ladder. You are current ly on the floor at height 0, and you want to reach the last rung.
You are also given an integer dist. You can only climb to the next highest rung if the distance between where you ar e currently at (the floor or on a rung) and the next rung is at most dist. You are able to insert rungs at any positive int eger height if a rung is not already there.
Return the minimum number of rungs that must be added to the ladder in order for you to climb to the last rung.

Example 1:

Input: rungs = [1,3,5,10], dist = 2
Output: 2
Explanation:
You currently cannot reach the last rung.
Add rungs at heights 7 and 8 to climb this ladder.
The ladder will now have rungs at [1,3,5,7,8,10].

Example 2:

Input: rungs = [3,6,8,10], dist = 3
Output: 0
Explanation:
This ladder can be climbed without adding additional rungs.

Example 3:

Input: rungs = [3,4,6,7], dist = 2
Output: 1
Explanation:
You currently cannot reach the first rung from the ground.
Add a rung at height 1 to climb this ladder.
The ladder will now have rungs at [1,3,4,6,7].

Example 4:

Input: rungs = [5], dist = 10
Output: 0
Explanation:
This ladder can be climbed without adding additional rungs.

Constraints:

1 <= rungs.length <= 105
1 <= rungs[i] <= 109
1 <= dist <= 109
rungs is strictly increasing.

1937. Maximum Number of Points with Cost

You are given an m x n integer matrix points (0-indexed). Starting with 0 points, you want to maximize the number of points you can get from the matrix.
To gain points, you must pick one cell in each row. Picking the cell at coordinates (r, c) will add points[r][c] to your score.
However, you will lose points if you pick a cell too far from the cell that you picked in the previous row. For every t wo adjacent rows r and r + 1 (where 0 <= r < m - 1), picking cells at coordinates (r, c1) and (r + 1, c2) will subtract a bs(c1 - c2) from your score.
Return the maximum number of points you can achieve.
abs(x) is defined as:

x for x >= 0.
-x for x < 0.

Example 1:

Input: points = [[1,2,3],[1,5,1],[3,1,1]]
Output: 9

Explanation:
The blue cells denote the optimal cells to pick, which have coordinates (0, 2), (1, 1), and (2, 0).
You add 3 + 5 + 3 = 11 to your score.
However, you must subtract abs(2 - 1) + abs(1 - 0) = 2 from your score.
Your final score is 11 - 2 = 9.

Example 2:

Input: points = [[1,5],[2,3],[4,2]]
Output: 11
Explanation:
The blue cells denote the optimal cells to pick, which have coordinates (0, 1), (1, 1), and (2, 0).
You add 5 + 3 + 4 = 12 to your score.
However, you must subtract abs(1 - 1) + abs(1 - 0) = 1 from your score.
Your final score is 12 - 1 = 11.

Constraints:

m == points.length
n == points[r].length
1 <= m, n <= 105
1 <= m * n <= 105
0 <= points[r][c] <= 105

1938. Maximum Genetic Difference Query

There is a rooted tree consisting of n nodes numbered 0 to n - 1. Each node's number denotes its unique genetic valu e (i.e. the genetic value of node x is x). The genetic difference between two genetic values is defined as the bitwise-XOR of their values. You are given the integer array parents, where parents[i] is the parent for node i. If node x is th e root of the tree, then parents[x] == -1.
You are also given the array queries where queries[i] = [nodei, vali]. For each query i, find the maximum genetic dif ference between vali and pi, where pi is the genetic value of any node that is on the path between nodei and the root (including nodei and the root). More formally, you want to maximize vali XOR pi.
Return an array ans where ans[i] is the answer to the ith query.

Example 1:

Input: parents = [-1,0,1,1], queries = [[0,2],[3,2],[2,5]]
Output: [2,3,7]
Explanation: The queries are processed as follows:
- [0,2]: The node with the maximum genetic difference is 0, with a difference of 2 XOR 0 = 2.
- [3,2]: The node with the maximum genetic difference is 1, with a difference of 2 XOR 1 = 3.
- [2,5]: The node with the maximum genetic difference is 2, with a difference of 5 XOR 2 = 7.

Example 2:

Input: parents = [3,7,-1,2,0,7,0,2], queries = [[4,6],[1,15],[0,5]]
Output: [6,14,7]
Explanation: The queries are processed as follows:
- [4,6]: The node with the maximum genetic difference is 0, with a difference of 6 XOR 0 = 6.
- [1,15]: The node with the maximum genetic difference is 1, with a difference of 15 XOR 1 = 14.
- [0,5]: The node with the maximum genetic difference is 2, with a difference of 5 XOR 2 = 7.

Constraints:

2 <= parents.length <= 105
0 <= parents[i] <= parents.length - 1 for every node i that is not the root.
parents[root] == -1
1 <= queries.length <= 3 * 104
0 <= nodei <= parents.length - 1
0 <= vali <= 2 * 105

1941. Check if All Characters Have Equal Number of Occurrences

Given a string s, return true if s is a good string, or false otherwise.
A string s is good if all the characters that appear in s have the same number of occurrences (i.e., the same frequency).

Example 1:

Input: s = "abacbc"
Output: true
Explanation: The characters that appear in s are 'a', 'b', and 'c'. All characters occur 2 times in s.

Example 2:

Input: s = "aaabb"
Output: false
Explanation: The characters that appear in s are 'a' and 'b'.
'a' occurs 3 times while 'b' occurs 2 times, which is not the same number of times.

Constraints:

1 <= s.length <= 1000
s consists of lowercase English letters.

1942. The Number of the Smallest Unoccupied Chair

There is a party where n friends numbered from 0 to n - 1 are attending. There is an infinite number of chairs in this party that are numbered from 0 to infinity. When a friend arrives at the party, they sit on the unoccupied chair with t he smallest number.

For example, if chairs 0, 1, and 5 are occupied when a friend comes, they will sit on chair number 2.

When a friend leaves the party, their chair becomes unoccupied at the moment they leave. If another friend arrives at that same moment, they can sit in that chair.
You are given a 0-indexed 2D integer array times where times[i] = [arrivali, leavingi], indicating the arrival and leav ing times of the ith friend respectively, and an integer targetFriend. All arrival times are distinct.
Return the chair number that the friend numbered targetFriend will sit on.

Example 1:

Input: times = [[1,4],[2,3],[4,6]], targetFriend = 1
Output: 1
Explanation:
- Friend 0 arrives at time 1 and sits on chair 0.
- Friend 1 arrives at time 2 and sits on chair 1.
- Friend 1 leaves at time 3 and chair 1 becomes empty.
- Friend 0 leaves at time 4 and chair 0 becomes empty.
- Friend 2 arrives at time 4 and sits on chair 0.
Since friend 1 sat on chair 1, we return 1.

Example 2:

Input: times = [[3,10],[1,5],[2,6]], targetFriend = 0
Output: 2
Explanation:
- Friend 1 arrives at time 1 and sits on chair 0.
- Friend 2 arrives at time 2 and sits on chair 1.
- Friend 0 arrives at time 3 and sits on chair 2.
- Friend 1 leaves at time 5 and chair 0 becomes empty.
- Friend 2 leaves at time 6 and chair 1 becomes empty.
- Friend 0 leaves at time 10 and chair 2 becomes empty.
Since friend 0 sat on chair 2, we return 2.

Constraints:

n == times.length
2 <= n <= 104
times[i].length == 2
1 <= arrivali < leavingi <= 105

0 <= targetFriend <= n - 1
Each arrivali time is distinct.

1943. Describe the Painting

There is a long and thin painting that can be represented by a number line. The painting was painted with multiple ov erlapping segments where each segment was painted with a unique color. You are given a 2D integer array segments , where segments[i] = [starti, endi, colori] represents the half-closed segment [starti, endi) with colori as the color. The colors in the overlapping segments of the painting were mixed when it was painted. When two or more colors m ix, they form a new color that can be represented as a set of mixed colors.

For example, if colors 2, 4, and 6 are mixed, then the resulting mixed color is {2,4,6}.

For the sake of simplicity, you should only output the sum of the elements in the set rather than the full set.
You want to describe the painting with the minimum number of non-overlapping half-closed segments of these mixe d colors. These segments can be represented by the 2D array painting where painting[j] = [leftj, rightj, mixj] describe s a half-closed segment [leftj, rightj) with the mixed color sum of mixj.

For example, the painting created with segments = [[1,4,5],[1,7,7]] can be described by painting = [[1,4,12],[4,7,7]] because:

[1,4) is colored {5,7} (with a sum of 12) from both the first and second segments.
[4,7) is colored {7} from only the second segment.

Return the 2D array painting describing the finished painting (excluding any parts that are not painted). You may ret urn the segments in any order.
A half-closed segment [a, b) is the section of the number line between points a and b including point a and not includ ing point b.

Example 1:

Input: segments = [[1,4,5],[4,7,7],[1,7,9]]
Output: [[1,4,14],[4,7,16]]
Explanation: The painting can be described as follows:
- [1,4) is colored {5,9} (with a sum of 14) from the first and third segments.
- [4,7) is colored {7,9} (with a sum of 16) from the second and third segments.

Example 2:

Input: segments = [[1,7,9],[6,8,15],[8,10,7]]
Output: [[1,6,9],[6,7,24],[7,8,15],[8,10,7]]

Explanation: The painting can be described as follows:
- [1,6) is colored 9 from the first segment.
- [6,7) is colored {9,15} (with a sum of 24) from the first and second segments.
- [7,8) is colored 15 from the second segment.
- [8,10) is colored 7 from the third segment.

Example 3:

Input: segments = [[1,4,5],[1,4,7],[4,7,1],[4,7,11]]
Output: [[1,4,12],[4,7,12]]
Explanation: The painting can be described as follows:
- [1,4) is colored {5,7} (with a sum of 12) from the first and second segments.
- [4,7) is colored {1,11} (with a sum of 12) from the third and fourth segments.
Note that returning a single segment [1,7) is incorrect because the mixed color sets are different.

Constraints:

1 <= segments.length <= 2 * 104
segments[i].length == 3
1 <= starti < endi <= 105
1 <= colori <= 109
Each colori is distinct.

1944. Number of Visible People in a Queue

There are n people standing in a queue, and they numbered from 0 to n - 1 in left to right order. You are given an arr ay heights of distinct integers where heights[i] represents the height of the ith person.
A person can see another person to their right in the queue if everybody in between is shorter than both of them. Mor e formally, the ith person can see the jth person if i < j and min(heights[i], heights[j]) > max(heights[i+1], heights[i+ 2], ..., heights[j-1]).
Return an array answer of length n where answer[i] is the number of people the ith person can see to their right in th e queue.

Example 1:

Input: heights = [10,6,8,5,11,9]
Output: [3,1,2,1,1,0]
Explanation:
Person 0 can see person 1, 2, and 4.
Person 1 can see person 2.
Person 2 can see person 3 and 4.
Person 3 can see person 4.
Person 4 can see person 5.

Person 5 can see no one since nobody is to the right of them.

Example 2:

Input: heights = [5,1,2,3,10]
Output: [4,1,1,1,0]

Constraints:

n == heights.length
1 <= n <= 105
1 <= heights[i] <= 105
All the values of heights are unique.

1945. Sum of Digits of String After Convert

You are given a string s consisting of lowercase English letters, and an integer k.
First, convert s into an integer by replacing each letter with its position in the alphabet (i.e., replace 'a' with 1, 'b' with 2, ..., 'z' with 26). Then, transform the integer by replacing it with the sum of its digits. Repeat the transform operation k times in total.
For example, if s = "zbax" and k = 2, then the resulting integer would be 8 by the following operations:

Convert: "zbax" □ "(26)(2)(1)(24)" □ "262124" □ 262124
Transform #1: 262124 □ 2 + 6 + 2 + 1 + 2 + 4 □ 17
Transform #2: 17 □ 1 + 7 □ 8

Return the resulting integer after performing the operations described above.

Example 1:

Input: s = "iiii", k = 1
Output: 36
Explanation: The operations are as follows:
- Convert: "iiii" □ "(9)(9)(9)(9)" □ "9999" □ 9999
- Transform #1: 9999 □ 9 + 9 + 9 + 9 □ 36
Thus the resulting integer is 36.

Example 2:

Input: s = "leetcode", k = 2
Output: 6
Explanation: The operations are as follows:
- Convert: "leetcode" □ "(12)(5)(5)(20)(3)(15)(4)(5)" □ "12552031545" □ 12552031545
- Transform #1: 12552031545 □ 1 + 2 + 5 + 5 + 2 + 0 + 3 + 1 + 5 + 4 + 5 □ 33
- Transform #2: 33 □ 3 + 3 □ 6

Thus the resulting integer is 6.

Example 3:

Input: s = "zbax", k = 2
Output: 8

Constraints:

1 <= s.length <= 100
1 <= k <= 10
s consists of lowercase English letters.

1946. Largest Number After Mutating Substring

You are given a string num, which represents a large integer. You are also given a 0-indexed integer array change of length 10 that maps each digit 0-9 to another digit. More formally, digit d maps to digit change[d].
You may choose to mutate a single substring of num. To mutate a substring, replace each digit num[i] with the digit it maps to in change (i.e. replace num[i] with change[num[i]]).
Return a string representing the largest possible integer after mutating (or choosing not to) a single substring of num.
A substring is a contiguous sequence of characters within the string.

Example 1:

Input: num = "132", change = [9,8,5,0,3,6,4,2,6,8]
Output: "832"
Explanation: Replace the substring "1":
- 1 maps to change[1] = 8.
Thus, "132" becomes "832".
"832" is the largest number that can be created, so return it.

Example 2:

Input: num = "021", change = [9,4,3,5,7,2,1,9,0,6]
Output: "934"
Explanation: Replace the substring "021":
- 0 maps to change[0] = 9.
- 2 maps to change[2] = 3.
- 1 maps to change[1] = 4.
Thus, "021" becomes "934".
"934" is the largest number that can be created, so return it.

Example 3:

Input: num = "5", change = [1,4,7,5,3,2,5,6,9,4]

Output: "5"
Explanation: "5" is already the largest number that can be created, so return it.

Constraints:

1 <= num.length <= 105
num consists of only digits 0-9.
change.length == 10
0 <= change[d] <= 9

1947. Maximum Compatibility Score Sum

There is a survey that consists of n questions where each question's answer is either 0 (no) or 1 (yes).
The survey was given to m students numbered from 0 to m - 1 and m mentors numbered from 0 to m - 1. The answer s of the students are represented by a 2D integer array students where students[i] is an integer array that contains the answers of the ith student (0-indexed). The answers of the mentors are represented by a 2D integer array mentors wh ere mentors[j] is an integer array that contains the answers of the jth mentor (0-indexed).
Each student will be assigned to one mentor, and each mentor will have one student assigned to them. The compatibi lity score of a student-mentor pair is the number of answers that are the same for both the student and the mentor.

For example, if the student's answers were [1, 0, 1] and the mentor's answers were [0, 0, 1], then their compatibility s core is 2 because only the second and the third answers are the same.

You are tasked with finding the optimal student-mentor pairings to maximize the sum of the compatibility scores. Given students and mentors, return the maximum compatibility score sum that can be achieved.

Example 1:

Input: students = [[1,1,0],[1,0,1],[0,0,1]], mentors = [[1,0,0],[0,0,1],[1,1,0]]
Output: 8
Explanation: We assign students to mentors in the following way:
- student 0 to mentor 2 with a compatibility score of 3.
- student 1 to mentor 0 with a compatibility score of 2.
- student 2 to mentor 1 with a compatibility score of 3.
The compatibility score sum is 3 + 2 + 3 = 8.

Example 2:

Input: students = [[0,0],[0,0],[0,0]], mentors = [[1,1],[1,1],[1,1]]
Output: 0
Explanation: The compatibility score of any student-mentor pair is 0.

Constraints:

m == students.length == mentors.length
n == students[i].length == mentors[j].length
1 <= m, n <= 8
students[i][k] is either 0 or 1.
mentors[j][k] is either 0 or 1.

1948. Delete Duplicate Folders in System

Due to a bug, there are many duplicate folders in a file system. You are given a 2D array paths, where paths[i] is an array representing an absolute path to the ith folder in the file system.

For example, ["one", "two", "three"] represents the path "/one/two/three".

Two folders (not necessarily on the same level) are identical if they contain the same non-empty set of identical subf olders and underlying subfolder structure. The folders do not need to be at the root level to be identical. If two or mo re folders are identical, then mark the folders as well as all their subfolders.

For example, folders "/a" and "/b" in the file structure below are identical. They (as well as their subfolders) should a ll be marked:

/a
/a/x
/a/x/y
/a/z
/b
/b/x
/b/x/y
/b/z

However, if the file structure also included the path "/b/w", then the folders "/a" and "/b" would not be identical. Not e that "/a/x" and "/b/x" would still be considered identical even with the added folder.

Once all the identical folders and their subfolders have been marked, the file system will delete all of them. The file system only runs the deletion once, so any folders that become identical after the initial deletion are not deleted. Return the 2D array ans containing the paths of the remaining folders after deleting all the marked folders. The paths may be returned in any order.

Example 1:

Input: paths = [["a"],["c"],["d"],["a","b"],["c","b"],["d","a"]]
Output: [["d"],["d","a"]]
Explanation: The file structure is as shown.
Folders "/a" and "/c" (and their subfolders) are marked for deletion because they both contain an empty

folder named "b".

Example 2:

Input: paths = [["a"],["c"],["a","b"],["c","b"],["a","b","x"],["a","b","x","y"],["w"],["w","y"]]
Output: [["c"],["c","b"],["a"],["a","b"]]
Explanation: The file structure is as shown.
Folders "/a/b/x" and "/w" (and their subfolders) are marked for deletion because they both contain an empty folder n amed "y".
Note that folders "/a" and "/c" are identical after the deletion, but they are not deleted because they were not marked beforehand.

Example 3:

Input: paths = [["a","b"],["c","d"],["c"],["a"]]
Output: [["c"],["c","d"],["a"],["a","b"]]
Explanation: All folders are unique in the file system.
Note that the returned array can be in a different order as the order does not matter.

Example 4:

Input: paths = [["a"],["a","x"],["a","x","y"],["a","z"],["b"],["b","x"],["b","x","y"],["b","z"]]
Output: []
Explanation: The file structure is as shown.
Folders "/a/x" and "/b/x" (and their subfolders) are marked for deletion because they both contain an empty folder named "y".
Folders "/a" and "/b" (and their subfolders) are marked for deletion because they both contain an empty folder "z" and the folder "x" described above.

Example 5:

Input: paths = [["a"],["a","x"],["a","x","y"],["a","z"],["b"],["b","x"],["b","x","y"],["b","z"],["b","w"]]
Output: [["b"],["b","w"],["b","z"],["a"],["a","z"]]
Explanation: This has the same structure as the previous example, except with the added "/b/w".
Folders "/a/x" and "/b/x" are still marked, but "/a" and "/b" are no longer marked because "/b" has the empty folder named "w" and "/a" does not.
Note that "/a/z" and "/b/z" are not marked because the set of identical subfolders must be non-empty, but these folder s are empty.

Constraints:

1 <= paths.length <= 2 * 104
1 <= paths[i].length <= 500
1 <= paths[i][j].length <= 10
1 <= sum(paths[i][j].length) <= 2 * 105
path[i][j] consists of lowercase English letters.
No two paths lead to the same folder.
For any folder not at the root level, its parent folder will also be in the input.

1952. Three Divisors

Given an integer n, return true if n has exactly three positive divisors. Otherwise, return false.
An integer m is a divisor of n if there exists an integer k such that n = k * m.

Example 1:

Input: n = 2
Output: false
Explantion: 2 has only two divisors: 1 and 2.

Example 2:

Input: n = 4
Output: true
Explantion: 4 has three divisors: 1, 2, and 4.

Constraints:

1 <= n <= 104

1953. Maximum Number of Weeks for Which You Can Work

There are n projects numbered from 0 to n - 1. You are given an integer array milestones where each milestones[i] d enotes the number of milestones the ith project has.
You can work on the projects following these two rules:

Every week, you will finish exactly one milestone of one project. You must work every week.
You cannot work on two milestones from the same project for two consecutive weeks.

Once all the milestones of all the projects are finished, or if the only milestones that you can work on will cause you to violate the above rules, you will stop working. Note that you may not be able to finish every project's milestones d ue to these constraints.
Return the maximum number of weeks you would be able to work on the projects without violating the rules mentio ned above.

Example 1:

Input: milestones = [1,2,3]
Output: 6
Explanation: One possible scenario is:
- During the 1st week, you will work on a milestone of project 0.
- During the 2nd week, you will work on a milestone of project 2.
- During the 3rd week, you will work on a milestone of project 1.
- During the 4th week, you will work on a milestone of project 2.
- During the 5th week, you will work on a milestone of project 1.
- During the 6th week, you will work on a milestone of project 2.
The total number of weeks is 6.

Example 2:

Input: milestones = [5,2,1]
Output: 7
Explanation: One possible scenario is:
- During the 1st week, you will work on a milestone of project 0.
- During the 2nd week, you will work on a milestone of project 1.
- During the 3rd week, you will work on a milestone of project 0.
- During the 4th week, you will work on a milestone of project 1.
- During the 5th week, you will work on a milestone of project 0.
- During the 6th week, you will work on a milestone of project 2.
- During the 7th week, you will work on a milestone of project 0.
The total number of weeks is 7.
Note that you cannot work on the last milestone of project 0 on 8th week because it would violate the rules.
Thus, one milestone in project 0 will remain unfinished.

Constraints:

n == milestones.length
1 <= n <= 105
1 <= milestones[i] <= 109

1954. Minimum Garden Perimeter to Collect Enough Apples

In a garden represented as an infinite 2D grid, there is an apple tree planted at every integer coordinate. The apple tre e planted at an integer coordinate (i, j) has |i| + |j| apples growing on it.
You will buy an axis-aligned square plot of land that is centered at (0, 0).
Given an integer neededApples, return the minimum perimeter of a plot such that at least neededApples apples are in side or on the perimeter of that plot.
The value of |x| is defined as:

x if x >= 0
-x if x < 0

Example 1:

Input: neededApples = 1
Output: 8
Explanation: A square plot of side length 1 does not contain any apples.
However, a square plot of side length 2 has 12 apples inside (as depicted in the image above).
The perimeter is 2 * 4 = 8.

Example 2:

Input: neededApples = 13
Output: 16

Example 3:

Input: neededApples = 1000000000
Output: 5040

Constraints:

1 <= neededApples <= 1015

1955. Count Number of Special Subsequences

A sequence is special if it consists of a positive number of 0s, followed by a positive number of 1s, then a positive n umber of 2s.

For example, [0,1,2] and [0,0,1,1,1,2] are special.
In contrast, [2,1,0], [1], and [0,1,2,0] are not special.

Given an array nums (consisting of only integers 0, 1, and 2), return the number of different subsequences that are sp ecial. Since the answer may be very large, return it modulo 109 + 7.
A subsequence of an array is a sequence that can be derived from the array by deleting some or no elements without changing the order of the remaining elements. Two subsequences are different if the set of indices chosen are differe nt.

Example 1:

Input: nums = [0,1,2,2]
Output: 3
Explanation: The special subsequences are bolded [0,1,2,2], [0,1,2,2], and [0,1,2,2].

Example 2:

Input: nums = [2,2,0,0]
Output: 0
Explanation: There are no special subsequences in [2,2,0,0].

Example 3:

Input: nums = [0,1,2,0,1,2]
Output: 7
Explanation: The special subsequences are bolded:
- [0,1,2,0,1,2]
- [0,1,2,0,1,2]
- [0,1,2,0,1,2]
- [0,1,2,0,1,2]
- [0,1,2,0,1,2]
- [0,1,2,0,1,2]
- [0,1,2,0,1,2]

Constraints:

1 <= nums.length <= 105
0 <= nums[i] <= 2

1957. Delete Characters to Make Fancy String

A fancy string is a string where no three consecutive characters are equal.
Given a string s, delete the minimum possible number of characters from s to make it fancy.
Return the final string after the deletion. It can be shown that the answer will always be unique.

Example 1:

Input: s = "leeetcode"
Output: "leetcode"
Explanation:
Remove an 'e' from the first group of 'e's to create "leetcode".
No three consecutive characters are equal, so return "leetcode".

Example 2:

Input: s = "aaabaaaa"
Output: "aabaa"
Explanation:
Remove an 'a' from the first group of 'a's to create "aabaaaa".
Remove two 'a's from the second group of 'a's to create "aabaa".

No three consecutive characters are equal, so return "aabaa".

Example 3:

Input: s = "aab"
Output: "aab"
Explanation: No three consecutive characters are equal, so return "aab".

Constraints:

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

1958. Check if Move is Legal

You are given a 0-indexed 8 x 8 grid board, where board[r][c] represents the cell (r, c) on a game board. On the boar d, free cells are represented by '.', white cells are represented by 'W', and black cells are represented by 'B'.
Each move in this game consists of choosing a free cell and changing it to the color you are playing as (either white or black). However, a move is only legal if, after changing it, the cell becomes the endpoint of a good line (horizonta l, vertical, or diagonal).
A good line is a line of three or more cells (including the endpoints) where the endpoints of the line are one color, an d the remaining cells in the middle are the opposite color (no cells in the line are free). You can find examples for go od lines in the figure below:

Given two integers rMove and cMove and a character color representing the color you are playing as (white or black), return true if changing cell (rMove, cMove) to color color is a legal move, or false if it is not legal.

Example 1:

Input: board = [[".",".",".","B",".",".",".",".","."],[".",".",".",".","W",".",".",".",".","."],[".",".",".",".","W",".",".",".",".","."],["W","B","B",".","W","W","W","B"],[".",".",".",".","B",".",".",".",".","."],[".",".",".",".","B",".",".",".",".","."],[".",".",".",".","W",".",".",".",".","."]], rMove = 4, cMove = 3, color = "B"
Output: true
Explanation: '.', 'W', and 'B' are represented by the colors blue, white, and black respectively, and cell (rMove, cMov e) is marked with an 'X'.
The two good lines with the chosen cell as an endpoint are annotated above with the red rectangles.

Example 2:

Input: board = [[".",".",".",".",".",".",".",".","."],[".","B",".",".",".","W",".",".","."],[".",".",".","W",".",".",".",".","."],[".",".",".","W","B",".",".",".","."],[".",".",".",".",".",".",".",".","."],[".",".",".",".",".",".","B","W",".",".","."],[".",".",".",".",".",".",".",".","W","."],[".",".",".",".",".",".",".",".","B"]], rMove = 4, cMove = 4, color = "W"
Output: false

Explanation: While there are good lines with the chosen cell as a middle cell, there are no good lines with the chosen cell as an endpoint.

Constraints:

board.length == board[r].length == 8
0 <= rMove, cMove < 8
board[rMove][cMove] == '.'
color is either 'B' or 'W'.

1959. Minimum Total Space Wasted With K Resizing Operations

You are currently designing a dynamic array. You are given a 0-indexed integer array nums, where nums[i] is the number of elements that will be in the array at time i. In addition, you are given an integer k, the maximum number of times you can resize the array (to any size).
The size of the array at time t, sizet, must be at least nums[t] because there needs to be enough space in the array to hold all the elements. The space wasted at time t is defined as sizet - nums[t], and the total space wasted is the sum of the space wasted across every time t where 0 <= t < nums.length.
Return the minimum total space wasted if you can resize the array at most k times.
Note: The array can have any size at the start and does not count towards the number of resizing operations.

Example 1:

Input: nums = [10,20], k = 0
Output: 10
Explanation: size = [20,20].
We can set the initial size to be 20.
The total wasted space is (20 - 10) + (20 - 20) = 10.

Example 2:

Input: nums = [10,20,30], k = 1
Output: 10
Explanation: size = [20,20,30].
We can set the initial size to be 20 and resize to 30 at time 2.
The total wasted space is (20 - 10) + (20 - 20) + (30 - 30) = 10.

Example 3:

Input: nums = [10,20,15,30,20], k = 2
Output: 15
Explanation: size = [10,20,20,30,30].
We can set the initial size to 10, resize to 20 at time 1, and resize to 30 at time 3.
The total wasted space is (10 - 10) + (20 - 20) + (20 - 15) + (30 - 30) + (30 - 20) = 15.

Constraints:

1 <= nums.length <= 200
1 <= nums[i] <= 106
0 <= k <= nums.length - 1

1960. Maximum Product of the Length of Two Palindromic Substrings

You are given a 0-indexed string s and are tasked with finding two non-intersecting palindromic substrings of odd le ngth such that the product of their lengths is maximized.
More formally, you want to choose four integers i, j, k, l such that 0 <= i <= j < k <= l < s.length and both the substri ngs s[i...j] and s[k...l] are palindromes and have odd lengths. s[i...j] denotes a substring from index i to index j inclusi ve.
Return the maximum possible product of the lengths of the two non-intersecting palindromic substrings.
A palindrome is a string that is the same forward and backward. A substring is a contiguous sequence of characters i n a string.

Example 1:

Input: s = "ababbb"
Output: 9
Explanation: Substrings "aba" and "bbb" are palindromes with odd length. product = 3 * 3 = 9.

Example 2:

Input: s = "zaaaxbbby"
Output: 9
Explanation: Substrings "aaa" and "bbb" are palindromes with odd length. product = 3 * 3 = 9.

Constraints:

2 <= s.length <= 105
s consists of lowercase English letters.

1961. Check If String Is a Prefix of Array

Given a string s and an array of strings words, determine whether s is a prefix string of words.
A string s is a prefix string of words if s can be made by concatenating the first k strings in words for some positive k no larger than words.length.
Return true if s is a prefix string of words, or false otherwise.

Example 1:

Input: s = "iloveleetcode", words = ["i","love","leetcode","apples"]
Output: true
Explanation:
s can be made by concatenating "i", "love", and "leetcode" together.

Example 2:

Input: s = "iloveleetcode", words = ["apples","i","love","leetcode"]
Output: false
Explanation:
It is impossible to make s using a prefix of arr.

Constraints:

1 <= words.length <= 100
1 <= words[i].length <= 20
1 <= s.length <= 1000
words[i] and s consist of only lowercase English letters.

1962. Remove Stones to Minimize the Total

You are given a 0-indexed integer array piles, where piles[i] represents the number of stones in the ith pile, and an in teger k. You should apply the following operation exactly k times:

Choose any piles[i] and remove floor(piles[i] / 2) stones from it.

Notice that you can apply the operation on the same pile more than once.
Return the minimum possible total number of stones remaining after applying the k operations.
floor(x) is the greatest integer that is smaller than or equal to x (i.e., rounds x down).

Example 1:

Input: piles = [5,4,9], k = 2
Output: 12
Explanation: Steps of a possible scenario are:
- Apply the operation on pile 2. The resulting piles are [5,4,5].
- Apply the operation on pile 0. The resulting piles are [3,4,5].
The total number of stones in [3,4,5] is 12.

Example 2:

Input: piles = [4,3,6,7], k = 3
Output: 12
Explanation: Steps of a possible scenario are:
- Apply the operation on pile 2. The resulting piles are [4,3,3,7].
- Apply the operation on pile 3. The resulting piles are [4,3,3,4].
- Apply the operation on pile 0. The resulting piles are [2,3,3,4].
The total number of stones in [2,3,3,4] is 12.

Constraints:

1 <= piles.length <= 105
1 <= piles[i] <= 104
1 <= k <= 105

1963. Minimum Number of Swaps to Make the String Balanced

You are given a 0-indexed string s of even length n. The string consists of exactly n / 2 opening brackets '[' and n / 2 closing brackets ']'.
A string is called balanced if and only if:

It is the empty string, or
It can be written as AB, where both A and B are balanced strings, or
It can be written as [C], where C is a balanced string.

You may swap the brackets at any two indices any number of times.
Return the minimum number of swaps to make s balanced.

Example 1:

Input: s = "][]["
Output: 1
Explanation: You can make the string balanced by swapping index 0 with index 3.
The resulting string is "[[]]".

Example 2:

Input: s = "]]][[["
Output: 2
Explanation: You can do the following to make the string balanced:
- Swap index 0 with index 4. s = "[]][][".
- Swap index 1 with index 5. s = "[[][]]".
The resulting string is "[[][]]".

Example 3:

Input: s = "[]"
Output: 0
Explanation: The string is already balanced.

Constraints:

n == s.length
2 <= n <= 106
n is even.
s[i] is either '[' or ']'.
The number of opening brackets '[' equals n / 2, and the number of closing brackets ']' equals n / 2.

1964. Find the Longest Valid Obstacle Course at Each Position

You want to build some obstacle courses. You are given a 0-indexed integer array obstacles of length n, where obsta cles[i] describes the height of the ith obstacle.
For every index i between 0 and n - 1 (inclusive), find the length of the longest obstacle course in obstacles such that :

You choose any number of obstacles between 0 and i inclusive.
You must include the ith obstacle in the course.
You must put the chosen obstacles in the same order as they appear in obstacles.
Every obstacle (except the first) is taller than or the same height as the obstacle immediately before it.

Return an array ans of length n, where ans[i] is the length of the longest obstacle course for index i as described abov e.

Example 1:

Input: obstacles = [1,2,3,2]
Output: [1,2,3,3]
Explanation: The longest valid obstacle course at each position is:
- i = 0: [1], [1] has length 1.
- i = 1: [1,2], [1,2] has length 2.
- i = 2: [1,2,3], [1,2,3] has length 3.
- i = 3: [1,2,3,2], [1,2,2] has length 3.

Example 2:

Input: obstacles = [2,2,1]
Output: [1,2,1]
Explanation: The longest valid obstacle course at each position is:
- i = 0: [2], [2] has length 1.

- i = 1: [2,2], [2,2] has length 2.
- i = 2: [2,2,1], [1] has length 1.

Example 3:

Input: obstacles = [3,1,5,6,4,2]
Output: [1,1,2,3,2,2]
Explanation: The longest valid obstacle course at each position is:
- i = 0: [3], [3] has length 1.
- i = 1: [3,1], [1] has length 1.
- i = 2: [3,1,5], [3,5] has length 2. [1,5] is also valid.
- i = 3: [3,1,5,6], [3,5,6] has length 3. [1,5,6] is also valid.
- i = 4: [3,1,5,6,4], [3,4] has length 2. [1,4] is also valid.
- i = 5: [3,1,5,6,4,2], [1,2] has length 2.

Constraints:

n == obstacles.length
1 <= n <= 105
1 <= obstacles[i] <= 107

1967. Number of Strings That Appear as Substrings in Word

Given an array of strings patterns and a string word, return the number of strings in patterns that exist as a substring i
n word.
A substring is a contiguous sequence of characters within a string.

Example 1:

Input: patterns = ["a","abc","bc","d"], word = "abc"
Output: 3
Explanation:
- "a" appears as a substring in "abc".
- "abc" appears as a substring in "abc".
- "bc" appears as a substring in "abc".
- "d" does not appear as a substring in "abc".
3 of the strings in patterns appear as a substring in word.

Example 2:

Input: patterns = ["a","b","c"], word = "aaaaabbbbb"
Output: 2
Explanation:
- "a" appears as a substring in "aaaaabbbbb".
- "b" appears as a substring in "aaaaabbbbb".

- "c" does not appear as a substring in "aaaaabbbbb".
2 of the strings in patterns appear as a substring in word.

Example 3:

Input: patterns = ["a","a","a"], word = "ab"
Output: 3
Explanation: Each of the patterns appears as a substring in word "ab".

Constraints:

1 <= patterns.length <= 100
1 <= patterns[i].length <= 100
1 <= word.length <= 100
patterns[i] and word consist of lowercase English letters.

1968. Array With Elements Not Equal to Average of Neighbors

You are given a 0-indexed array nums of distinct integers. You want to rearrange the elements in the array such that every element in the rearranged array is not equal to the average of its neighbors.
More formally, the rearranged array should have the property such that for every i in the range 1 <= i < nums.length - 1, (nums[i-1] + nums[i+1]) / 2 is not equal to nums[i].
Return any rearrangement of nums that meets the requirements.

Example 1:

Input: nums = [1,2,3,4,5]
Output: [1,2,4,5,3]
Explanation:
When i=1, nums[i] = 2, and the average of its neighbors is (1+4) / 2 = 2.5.
When i=2, nums[i] = 4, and the average of its neighbors is (2+5) / 2 = 3.5.
When i=3, nums[i] = 5, and the average of its neighbors is (4+3) / 2 = 3.5.

Example 2:

Input: nums = [6,2,0,9,7]
Output: [9,7,6,2,0]
Explanation:
When i=1, nums[i] = 7, and the average of its neighbors is (9+6) / 2 = 7.5.
When i=2, nums[i] = 6, and the average of its neighbors is (7+2) / 2 = 4.5.
When i=3, nums[i] = 2, and the average of its neighbors is (6+0) / 2 = 3.

Constraints:

3 <= nums.length <= 105
0 <= nums[i] <= 105

1969. Minimum Non-Zero Product of the Array Elements

You are given a positive integer p. Consider an array nums (1-indexed) that consists of the integers in the inclusive r ange [1, 2p - 1] in their binary representations. You are allowed to do the following operation any number of times:

Choose two elements x and y from nums.
Choose a bit in x and swap it with its corresponding bit in y. Corresponding bit refers to the bit that is in the same po sition in the other integer.

For example, if x = 1101 and y = 0011, after swapping the 2nd bit from the right, we have x = 1111 and y = 0001.
Find the minimum non-zero product of nums after performing the above operation any number of times. Return this product modulo 109 + 7.
Note: The answer should be the minimum product before the modulo operation is done.

Example 1:

Input: p = 1
Output: 1
Explanation: nums = [1].
There is only one element, so the product equals that element.

Example 2:

Input: p = 2
Output: 6
Explanation: nums = [01, 10, 11].
Any swap would either make the product 0 or stay the same.
Thus, the array product of 1 * 2 * 3 = 6 is already minimized.

Example 3:

Input: p = 3
Output: 1512
Explanation: nums = [001, 010, 011, 100, 101, 110, 111]
- In the first operation we can swap the leftmost bit of the second and fifth elements.
  - The resulting array is [001, 110, 011, 100, 001, 110, 111].
- In the second operation we can swap the middle bit of the third and fourth elements.
  - The resulting array is [001, 110, 001, 110, 001, 110, 111].
The array product is 1 * 6 * 1 * 6 * 1 * 6 * 7 = 1512, which is the minimum possible product.

Constraints:

1 <= p <= 60

1970. Last Day Where You Can Still Cross

There is a 1-based binary matrix where 0 represents land and 1 represents water. You are given integers row and col representing the number of rows and columns in the matrix, respectively.
Initially on day 0, the entire matrix is land. However, each day a new cell becomes flooded with water. You are give n a 1-based 2D array cells, where cells[i] = [ri, ci] represents that on the ith day, the cell on the rith row and cith colu mn (1-based coordinates) will be covered with water (i.e., changed to 1).
You want to find the last day that it is possible to walk from the top to the bottom by only walking on land cells. Yo u can start from any cell in the top row and end at any cell in the bottom row. You can only travel in the four cardina l directions (left, right, up, and down).
Return the last day where it is possible to walk from the top to the bottom by only walking on land cells.

Example 1:

Input: row = 2, col = 2, cells = [[1,1],[2,1],[1,2],[2,2]]
Output: 2
Explanation: The above image depicts how the matrix changes each day starting from day 0.
The last day where it is possible to cross from top to bottom is on day 2.

Example 2:

Input: row = 2, col = 2, cells = [[1,1],[1,2],[2,1],[2,2]]
Output: 1
Explanation: The above image depicts how the matrix changes each day starting from day 0.
The last day where it is possible to cross from top to bottom is on day 1.

Example 3:

Input: row = 3, col = 3, cells = [[1,2],[2,1],[3,3],[2,2],[1,1],[1,3],[2,3],[3,2],[3,1]]
Output: 3
Explanation: The above image depicts how the matrix changes each day starting from day 0.
The last day where it is possible to cross from top to bottom is on day 3.

Constraints:

2 <= row, col <= 2 * 104
4 <= row * col <= 2 * 104
cells.length == row * col
1 <= ri <= row
1 <= ci <= col

All the values of cells are unique.

1971. Find if Path Exists in Graph

There is a bi-directional graph with n vertices, where each vertex is labeled from 0 to n - 1 (inclusive). The edges in t he graph are represented as a 2D integer array edges, where each edges[i] = [ui, vi] denotes a bi-directional edge bet ween vertex ui and vertex vi. Every vertex pair is connected by at most one edge, and no vertex has an edge to itself. You want to determine if there is a valid path that exists from vertex start to vertex end.
Given edges and the integers n, start, and end, return true if there is a valid path from start to end, or false otherwise.

Example 1:

Input: n = 3, edges = [[0,1],[1,2],[2,0]], start = 0, end = 2
Output: true
Explanation: There are two paths from vertex 0 to vertex 2:
- 0 → 1 → 2
- 0 → 2

Example 2:

Input: n = 6, edges = [[0,1],[0,2],[3,5],[5,4],[4,3]], start = 0, end = 5
Output: false
Explanation: There is no path from vertex 0 to vertex 5.

Constraints:

1 <= n <= 2 * 105
0 <= edges.length <= 2 * 105
edges[i].length == 2
0 <= ui, vi <= n - 1
ui != vi
0 <= start, end <= n - 1
There are no duplicate edges.
There are no self edges.

1974. Minimum Time to Type Word Using Special Typewriter

There is a special typewriter with lowercase English letters 'a' to 'z' arranged in a circle with a pointer. A character ca n only be typed if the pointer is pointing to that character. The pointer is initially pointing to the character 'a'.

Each second, you may perform one of the following operations:

Move the pointer one character counterclockwise or clockwise.
Type the character the pointer is currently on.

Given a string word, return the minimum number of seconds to type out the characters in word.

Example 1:

Input: word = "abc"
Output: 5
Explanation:
The characters are printed as follows:
- Type the character 'a' in 1 second since the pointer is initially on 'a'.
- Move the pointer clockwise to 'b' in 1 second.
- Type the character 'b' in 1 second.
- Move the pointer clockwise to 'c' in 1 second.
- Type the character 'c' in 1 second.

Example 2:

Input: word = "bza"
Output: 7
Explanation:
The characters are printed as follows:
- Move the pointer clockwise to 'b' in 1 second.
- Type the character 'b' in 1 second.
- Move the pointer counterclockwise to 'z' in 2 seconds.
- Type the character 'z' in 1 second.
- Move the pointer clockwise to 'a' in 1 second.
- Type the character 'a' in 1 second.

Example 3:

Input: word = "zjpc"
Output: 34
Explanation:
The characters are printed as follows:
- Move the pointer counterclockwise to 'z' in 1 second.
- Type the character 'z' in 1 second.
- Move the pointer clockwise to 'j' in 10 seconds.
- Type the character 'j' in 1 second.
- Move the pointer clockwise to 'p' in 6 seconds.
- Type the character 'p' in 1 second.
- Move the pointer counterclockwise to 'c' in 13 seconds.
- Type the character 'c' in 1 second.

Constraints:

1 <= word.length <= 100
word consists of lowercase English letters.

1975. Maximum Matrix Sum

You are given an n x n integer matrix. You can do the following operation any number of times:

Choose any two adjacent elements of matrix and multiply each of them by -1.

Two elements are considered adjacent if and only if they share a border.
Your goal is to maximize the summation of the matrix's elements. Return the maximum sum of the matrix's elements using the operation mentioned above.

Example 1:

Input: matrix = [[1,-1],[-1,1]]
Output: 4
Explanation: We can follow the following steps to reach sum equals 4:
- Multiply the 2 elements in the first row by -1.
- Multiply the 2 elements in the first column by -1.

Example 2:

Input: matrix = [[1,2,3],[-1,-2,-3],[1,2,3]]
Output: 16
Explanation: We can follow the following step to reach sum equals 16:
- Multiply the 2 last elements in the second row by -1.

Constraints:

n == matrix.length == matrix[i].length
2 <= n <= 250
-105 <= matrix[i][j] <= 105

1976. Number of Ways to Arrive at Destination

You are in a city that consists of n intersections numbered from 0 to n - 1 with bi-directional roads between some int ersections. The inputs are generated such that you can reach any intersection from any other intersection and that the re is at most one road between any two intersections.
You are given an integer n and a 2D integer array roads where roads[i] = [ui, vi, timei] means that there is a road bet ween intersections ui and vi that takes timei minutes to travel. You want to know in how many ways you can travel f rom intersection 0 to intersection n - 1 in the shortest amount of time.
Return the number of ways you can arrive at your destination in the shortest amount of time. Since the answer may b e large, return it modulo 109 + 7.

Example 1:

Input: n = 7, roads = [[0,6,7],[0,1,2],[1,2,3],[1,3,3],[6,3,3],[3,5,1],[6,5,1],[2,5,1],[0,4,5],[4,6,2]]
Output: 4
Explanation: The shortest amount of time it takes to go from intersection 0 to intersection 6 is 7 minutes.
The four ways to get there in 7 minutes are:
- 0 □ 6
- 0 □ 4 □ 6
- 0 □ 1 □ 2 □ 5 □ 6
- 0 □ 1 □ 3 □ 5 □ 6

Example 2:

Input: n = 2, roads = [[1,0,10]]
Output: 1
Explanation: There is only one way to go from intersection 0 to intersection 1, and it takes 10 minutes.

Constraints:

1 <= n <= 200
n - 1 <= roads.length <= n * (n - 1) / 2
roads[i].length == 3
0 <= ui, vi <= n - 1
1 <= timei <= 109
ui != vi
There is at most one road connecting any two intersections.
You can reach any intersection from any other intersection.

1977. Number of Ways to Separate Numbers

You wrote down many positive integers in a string called num. However, you realized that you forgot to add comma s to seperate the different numbers. You remember that the list of integers was non-decreasing and that no integer ha d leading zeros.

Return the number of possible lists of integers that you could have written down to get the string num. Since the ans wer may be large, return it modulo 109 + 7.

Example 1:

Input: num = "327"
Output: 2
Explanation: You could have written down the numbers:
3, 27
327

Example 2:

Input: num = "094"
Output: 0
Explanation: No numbers can have leading zeros and all numbers must be positive.

Example 3:

Input: num = "0"
Output: 0
Explanation: No numbers can have leading zeros and all numbers must be positive.

Example 4:

Input: num = "9999999999999"
Output: 101

Constraints:

1 <= num.length <= 3500
num consists of digits '0' through '9'.

1979. Find Greatest Common Divisor of Array

Given an integer array nums, return the greatest common divisor of the smallest number and largest number in nums .
The greatest common divisor of two numbers is the largest positive integer that evenly divides both numbers.

Example 1:

Input: nums = [2,5,6,9,10]
Output: 2
Explanation:
The smallest number in nums is 2.

The largest number in nums is 10.
The greatest common divisor of 2 and 10 is 2.

Example 2:

Input: nums = [7,5,6,8,3]
Output: 1
Explanation:
The smallest number in nums is 3.
The largest number in nums is 8.
The greatest common divisor of 3 and 8 is 1.

Example 3:

Input: nums = [3,3]
Output: 3
Explanation:
The smallest number in nums is 3.
The largest number in nums is 3.
The greatest common divisor of 3 and 3 is 3.

Constraints:

2 <= nums.length <= 1000
1 <= nums[i] <= 1000

1980. Find Unique Binary String

Given an array of strings nums containing n unique binary strings each of length n, return a binary string of length n that does not appear in nums. If there are multiple answers, you may return any of them.

Example 1:

Input: nums = ["01","10"]
Output: "11"
Explanation: "11" does not appear in nums. "00" would also be correct.

Example 2:

Input: nums = ["00","01"]
Output: "11"
Explanation: "11" does not appear in nums. "10" would also be correct.

Example 3:

Input: nums = ["111","011","001"]
Output: "101"
Explanation: "101" does not appear in nums. "000", "010", "100", and "110" would also be correct.

Constraints:

n == nums.length
1 <= n <= 16
nums[i].length == n
nums[i] is either '0' or '1'.
All the strings of nums are unique.

1981. Minimize the Difference Between Target and Chosen Elements

You are given an m x n integer matrix mat and an integer target.
Choose one integer from each row in the matrix such that the absolute difference between target and the sum of the c
hosen elements is minimized.
Return the minimum absolute difference.
The absolute difference between two numbers a and b is the absolute value of a - b.

Example 1:

Input: mat = [[1,2,3],[4,5,6],[7,8,9]], target = 13
Output: 0
Explanation: One possible choice is to:
- Choose 1 from the first row.
- Choose 5 from the second row.
- Choose 7 from the third row.
The sum of the chosen elements is 13, which equals the target, so the absolute difference is 0.

Example 2:

Input: mat = [[1],[2],[3]], target = 100
Output: 94
Explanation: The best possible choice is to:
- Choose 1 from the first row.
- Choose 2 from the second row.
- Choose 3 from the third row.
The sum of the chosen elements is 6, and the absolute difference is 94.

Example 3:

Input: mat = [[1,2,9,8,7]], target = 6
Output: 1
Explanation: The best choice is to choose 7 from the first row.
The absolute difference is 1.

Constraints:

m == mat.length
n == mat[i].length
1 <= m, n <= 70
1 <= mat[i][j] <= 70
1 <= target <= 800

1982. Find Array Given Subset Sums

You are given an integer n representing the length of an unknown array that you are trying to recover. You are also g iven an array sums containing the values of all 2n subset sums of the unknown array (in no particular order).
Return the array ans of length n representing the unknown array. If multiple answers exist, return any of them.
An array sub is a subset of an array arr if sub can be obtained from arr by deleting some (possibly zero or all) elemen ts of arr. The sum of the elements in sub is one possible subset sum of arr. The sum of an empty array is considered t o be 0.
Note: Test cases are generated such that there will always be at least one correct answer.

Example 1:

Input: n = 3, sums = [-3,-2,-1,0,0,1,2,3]
Output: [1,2,-3]
Explanation: [1,2,-3] is able to achieve the given subset sums:
- []: sum is 0
- [1]: sum is 1
- [2]: sum is 2
- [1,2]: sum is 3
- [-3]: sum is -3
- [1,-3]: sum is -2
- [2,-3]: sum is -1
- [1,2,-3]: sum is 0
Note that any permutation of [1,2,-3] and also any permutation of [-1,-2,3] will also be accepted.

Example 2:

Input: n = 2, sums = [0,0,0,0]
Output: [0,0]
Explanation: The only correct answer is [0,0].

Example 3:

Input: n = 4, sums = [0,0,5,5,4,-1,4,9,9,-1,4,3,4,8,3,8]
Output: [0,-1,4,5]
Explanation: [0,-1,4,5] is able to achieve the given subset sums.

Constraints:

1 <= n <= 15
sums.length == 2n
-104 <= sums[i] <= 104

1984. Minimum Difference Between Highest and Lowest of K Scores

You are given a 0-indexed integer array nums, where nums[i] represents the score of the ith student. You are also gi
ven an integer k.
Pick the scores of any k students from the array so that the difference between the highest and the lowest of the k sco
res is minimized.
Return the minimum possible difference.

Example 1:

Input: nums = [90], k = 1
Output: 0
Explanation: There is one way to pick score(s) of one student:
- [90]. The difference between the highest and lowest score is 90 - 90 = 0.
The minimum possible difference is 0.

Example 2:

Input: nums = [9,4,1,7], k = 2
Output: 2
Explanation: There are six ways to pick score(s) of two students:
- [9,4,1,7]. The difference between the highest and lowest score is 9 - 4 = 5.
- [9,4,1,7]. The difference between the highest and lowest score is 9 - 1 = 8.
- [9,4,1,7]. The difference between the highest and lowest score is 9 - 7 = 2.
- [9,4,1,7]. The difference between the highest and lowest score is 4 - 1 = 3.
- [9,4,1,7]. The difference between the highest and lowest score is 7 - 4 = 3.
- [9,4,1,7]. The difference between the highest and lowest score is 7 - 1 = 6.
The minimum possible difference is 2.

Constraints:

1 <= k <= nums.length <= 1000
0 <= nums[i] <= 105

1985. Find the Kth Largest Integer in the Array

You are given an array of strings nums and an integer k. Each string in nums represents an integer without leading z eros.
Return the string that represents the kth largest integer in nums.
Note: Duplicate numbers should be counted distinctly. For example, if nums is ["1","2","2"], "2" is the first largest i nteger, "2" is the second-largest integer, and "1" is the third-largest integer.

Example 1:

Input: nums = ["3","6","7","10"], k = 4
Output: "3"
Explanation:
The numbers in nums sorted in non-decreasing order are ["3","6","7","10"].
The 4th largest integer in nums is "3".

Example 2:

Input: nums = ["2","21","12","1"], k = 3
Output: "2"
Explanation:
The numbers in nums sorted in non-decreasing order are ["1","2","12","21"].
The 3rd largest integer in nums is "2".

Example 3:

Input: nums = ["0","0"], k = 2
Output: "0"
Explanation:
The numbers in nums sorted in non-decreasing order are ["0","0"].
The 2nd largest integer in nums is "0".

Constraints:

1 <= k <= nums.length <= 104
1 <= nums[i].length <= 100
nums[i] consists of only digits.
nums[i] will not have any leading zeros.

1986. Minimum Number of Work Sessions to Finish the Tasks

There are n tasks assigned to you. The task times are represented as an integer array tasks of length n, where the ith t
ask takes tasks[i] hours to finish. A work session is when you work for at most sessionTime consecutive hours and t
hen take a break.
You should finish the given tasks in a way that satisfies the following conditions:

If you start a task in a work session, you must complete it in the same work session.
You can start a new task immediately after finishing the previous one.
You may complete the tasks in any order.

Given tasks and sessionTime, return the minimum number of work sessions needed to finish all the tasks following t
he conditions above.
The tests are generated such that sessionTime is greater than or equal to the maximum element in tasks[i].

Example 1:

Input: tasks = [1,2,3], sessionTime = 3
Output: 2
Explanation: You can finish the tasks in two work sessions.
- First work session: finish the first and the second tasks in 1 + 2 = 3 hours.
- Second work session: finish the third task in 3 hours.

Example 2:

Input: tasks = [3,1,3,1,1], sessionTime = 8
Output: 2
Explanation: You can finish the tasks in two work sessions.
- First work session: finish all the tasks except the last one in 3 + 1 + 3 + 1 = 8 hours.
- Second work session: finish the last task in 1 hour.

Example 3:

Input: tasks = [1,2,3,4,5], sessionTime = 15
Output: 1
Explanation: You can finish all the tasks in one work session.

Constraints:

n == tasks.length
1 <= n <= 14
1 <= tasks[i] <= 10
max(tasks[i]) <= sessionTime <= 15

1987. Number of Unique Good Subsequences

You are given a binary string binary. A subsequence of binary is considered good if it is not empty and has no leadin g zeros (with the exception of "0").
Find the number of unique good subsequences of binary.

For example, if binary = "001", then all the good subsequences are ["0", "0", "1"], so the unique good subsequences are "0" and "1". Note that subsequences "00", "01", and "001" are not good because they have leading zeros.

Return the number of unique good subsequences of binary. Since the answer may be very large, return it modulo 109 + 7.
A subsequence is a sequence that can be derived from another sequence by deleting some or no elements without ch anging the order of the remaining elements.

Example 1:

Input: binary = "001"
Output: 2
Explanation: The good subsequences of binary are ["0", "0", "1"].
The unique good subsequences are "0" and "1".

Example 2:

Input: binary = "11"
Output: 2
Explanation: The good subsequences of binary are ["1", "1", "11"].
The unique good subsequences are "1" and "11".
Example 3:

Input: binary = "101"
Output: 5
Explanation: The good subsequences of binary are ["1", "0", "1", "10", "11", "101"].
The unique good subsequences are "0", "1", "10", "11", and "101".

Constraints:

1 <= binary.length <= 105
binary consists of only '0's and '1's.

1991. Find the Middle Index in Array

Given a 0-indexed integer array nums, find the leftmost middleIndex (i.e., the smallest amongst all the possible ones).
A middleIndex is an index where nums[0] + nums[1] + ... + nums[middleIndex-1] == nums[middleIndex+1] + nums

[middleIndex+2] + ... + nums[nums.length-1].
If middleIndex == 0, the left side sum is considered to be 0. Similarly, if middleIndex == nums.length - 1, the right s ide sum is considered to be 0.
Return the leftmost middleIndex that satisfies the condition, or -1 if there is no such index.

Example 1:

Input: nums = [2,3,-1,8,4]
Output: 3
Explanation:
The sum of the numbers before index 3 is: 2 + 3 + -1 = 4
The sum of the numbers after index 3 is: 4 = 4

Example 2:

Input: nums = [1,-1,4]
Output: 2
Explanation:
The sum of the numbers before index 2 is: 1 + -1 = 0
The sum of the numbers after index 2 is: 0

Example 3:

Input: nums = [2,5]
Output: -1
Explanation:
There is no valid middleIndex.

Example 4:

Input: nums = [1]
Output: 0
Explantion:
The sum of the numbers before index 0 is: 0
The sum of the numbers after index 0 is: 0

Constraints:

1 <= nums.length <= 100
-1000 <= nums[i] <= 1000

Note: This question is the same as 724: https://leetcode.com/problems/find-pivot-index/

1992. Find All Groups of Farmland

You are given a 0-indexed m x n binary matrix land where a 0 represents a hectare of forested land and a 1 represent s a hectare of farmland.
To keep the land organized, there are designated rectangular areas of hectares that consist entirely of farmland. Thes e rectangular areas are called groups. No two groups are adjacent, meaning farmland in one group is not four-directi onally adjacent to another farmland in a different group.
land can be represented by a coordinate system where the top left corner of land is (0, 0) and the bottom right corner of land is (m-1, n-1). Find the coordinates of the top left and bottom right corner of each group of farmland. A group of farmland with a top left corner at (r1, c1) and a bottom right corner at (r2, c2) is represented by the 4-length array [r1, c1, r2, c2].
Return a 2D array containing the 4-length arrays described above for each group of farmland in land. If there are no groups of farmland, return an empty array. You may return the answer in any order.

Example 1:

Input: land = [[1,0,0],[0,1,1],[0,1,1]]
Output: [[0,0,0,0],[1,1,2,2]]
Explanation:
The first group has a top left corner at land[0][0] and a bottom right corner at land[0][0].
The second group has a top left corner at land[1][1] and a bottom right corner at land[2][2].

Example 2:

Input: land = [[1,1],[1,1]]
Output: [[0,0,1,1]]
Explanation:
The first group has a top left corner at land[0][0] and a bottom right corner at land[1][1].

Example 3:

Input: land = [[0]]
Output: []
Explanation:
There are no groups of farmland.

Constraints:

m == land.length
n == land[i].length
1 <= m, n <= 300
land consists of only 0's and 1's.
Groups of farmland are rectangular in shape.

1993. Operations on Tree

You are given a tree with n nodes numbered from 0 to n - 1 in the form of a parent array parent where parent[i] is the parent of the ith node. The root of the tree is node 0, so parent[0] = -1 since it has no parent. You want to design a d ata structure that allows users to lock, unlock, and upgrade nodes in the tree.
The data structure should support the following functions:

Lock: Locks the given node for the given user and prevents other users from locking the same node. You may only l ock a node using this function if the node is unlocked.
Unlock: Unlocks the given node for the given user. You may only unlock a node using this function if it is currently locked by the same user.
Upgrade: Locks the given node for the given user and unlocks all of its descendants regardless of who locked it. You may only upgrade a node if all 3 conditions are true:

The node is unlocked,
It has at least one locked descendant (by any user), and
It does not have any locked ancestors.

Implement the LockingTree class:

LockingTree(int[] parent) initializes the data structure with the parent array.
lock(int num, int user) returns true if it is possible for the user with id user to lock the node num, or false otherwise. I f it is possible, the node num will become locked by the user with id user.
unlock(int num, int user) returns true if it is possible for the user with id user to unlock the node num, or false otherw ise. If it is possible, the node num will become unlocked.
upgrade(int num, int user) returns true if it is possible for the user with id user to upgrade the node num, or false othe rwise. If it is possible, the node num will be upgraded.

Example 1:

Input
["LockingTree", "lock", "unlock", "unlock", "lock", "upgrade", "lock"]
[[[-1, 0, 0, 1, 1, 2, 2]], [2, 2], [2, 3], [2, 2], [4, 5], [0, 1], [0, 1]]
Output
[null, true, false, true, true, true, false]

Explanation
LockingTree lockingTree = new LockingTree([-1, 0, 0, 1, 1, 2, 2]);
lockingTree.lock(2, 2); // return true because node 2 is unlocked.
                         // Node 2 will now be locked by user 2.
lockingTree.unlock(2, 3); // return false because user 3 cannot unlock a node locked by user 2.
lockingTree.unlock(2, 2); // return true because node 2 was previously locked by user 2.
                         // Node 2 will now be unlocked.
lockingTree.lock(4, 5); // return true because node 4 is unlocked.
                         // Node 4 will now be locked by user 5.
lockingTree.upgrade(0, 1); // return true because node 0 is unlocked and has at least one locked descendant (node 4).
                         // Node 0 will now be locked by user 1 and node 4 will now be unlocked.
lockingTree.lock(0, 1); // return false because node 0 is already locked.

Constraints:

n == parent.length
2 <= n <= 2000
0 <= parent[i] <= n - 1 for i != 0
parent[0] == -1
0 <= num <= n - 1
1 <= user <= 104
parent represents a valid tree.
At most 2000 calls in total will be made to lock, unlock, and upgrade.

1994. The Number of Good Subsets

You are given an integer array nums. We call a subset of nums good if its product can be represented as a product of one or more distinct prime numbers.

For example, if nums = [1, 2, 3, 4]:

[2, 3], [1, 2, 3], and [1, 3] are good subsets with products 6 = 2*3, 6 = 2*3, and 3 = 3 respectively.
[1, 4] and [4] are not good subsets with products 4 = 2*2 and 4 = 2*2 respectively.

Return the number of different good subsets in nums modulo 109 + 7.
A subset of nums is any array that can be obtained by deleting some (possibly none or all) elements from nums. Two subsets are different if and only if the chosen indices to delete are different.

Example 1:

Input: nums = [1,2,3,4]
Output: 6
Explanation: The good subsets are:
- [1,2]: product is 2, which is the product of distinct prime 2.
- [1,2,3]: product is 6, which is the product of distinct primes 2 and 3.
- [1,3]: product is 3, which is the product of distinct prime 3.
- [2]: product is 2, which is the product of distinct prime 2.
- [2,3]: product is 6, which is the product of distinct primes 2 and 3.
- [3]: product is 3, which is the product of distinct prime 3.

Example 2:

Input: nums = [4,2,3,15]
Output: 5
Explanation: The good subsets are:
- [2]: product is 2, which is the product of distinct prime 2.

- [2,3]: product is 6, which is the product of distinct primes 2 and 3.
- [2,15]: product is 30, which is the product of distinct primes 2, 3, and 5.
- [3]: product is 3, which is the product of distinct prime 3.
- [15]: product is 15, which is the product of distinct primes 3 and 5.

Constraints:

1 <= nums.length <= 105
1 <= nums[i] <= 30

1995. Count Special Quadruplets

Given a 0-indexed integer array nums, return the number of distinct quadruplets (a, b, c, d) such that:

nums[a] + nums[b] + nums[c] == nums[d], and
a < b < c < d

Example 1:

Input: nums = [1,2,3,6]
Output: 1
Explanation: The only quadruplet that satisfies the requirement is (0, 1, 2, 3) because 1 + 2 + 3 == 6.

Example 2:

Input: nums = [3,3,6,4,5]
Output: 0
Explanation: There are no such quadruplets in [3,3,6,4,5].

Example 3:

Input: nums = [1,1,1,3,5]
Output: 4
Explanation: The 4 quadruplets that satisfy the requirement are:
- (0, 1, 2, 3): 1 + 1 + 1 == 3
- (0, 1, 3, 4): 1 + 1 + 3 == 5
- (0, 2, 3, 4): 1 + 1 + 3 == 5
- (1, 2, 3, 4): 1 + 1 + 3 == 5

Constraints:

4 <= nums.length <= 50
1 <= nums[i] <= 100

1996. The Number of Weak Characters in the Game

You are playing a game that contains multiple characters, and each of the characters has two main properties: attack and defense. You are given a 2D integer array properties where properties[i] = [attacki, defensei] represents the prop erties of the ith character in the game.
A character is said to be weak if any other character has both attack and defense levels strictly greater than this chara cter's attack and defense levels. More formally, a character i is said to be weak if there exists another character j whe re attackj > attacki and defensej > defensei.
Return the number of weak characters.

Example 1:

Input: properties = [[5,5],[6,3],[3,6]]
Output: 0
Explanation: No character has strictly greater attack and defense than the other.

Example 2:

Input: properties = [[2,2],[3,3]]
Output: 1
Explanation: The first character is weak because the second character has a strictly greater attack and defense.

Example 3:

Input: properties = [[1,5],[10,4],[4,3]]
Output: 1
Explanation: The third character is weak because the second character has a strictly greater attack and defense.

Constraints:

2 <= properties.length <= 105
properties[i].length == 2
1 <= attacki, defensei <= 105

1997. First Day Where You Have Been in All the Rooms

There are n rooms you need to visit, labeled from 0 to n - 1. Each day is labeled, starting from 0. You will go in and visit one room a day.
Initially on day 0, you visit room 0. The order you visit the rooms for the coming days is determined by the followin g rules and a given 0-indexed array nextVisit of length n:

Assuming that on a day, you visit room i,
if you have been in room i an odd number of times (including the current visit), on the next day you will visit a room with a lower or equal room number specified by nextVisit[i] where 0 <= nextVisit[i] <= i;
if you have been in room i an even number of times (including the current visit), on the next day you will visit room (i + 1) mod n.

Return the label of the first day where you have been in all the rooms. It can be shown that such a day exists. Since t he answer may be very large, return it modulo 109 + 7.

Example 1:

Input: nextVisit = [0,0]
Output: 2
Explanation:
- On day 0, you visit room 0. The total times you have been in room 0 is 1, which is odd.
  On the next day you will visit room nextVisit[0] = 0
- On day 1, you visit room 0, The total times you have been in room 0 is 2, which is even.
  On the next day you will visit room (0 + 1) mod 2 = 1
- On day 2, you visit room 1. This is the first day where you have been in all the rooms.

Example 2:

Input: nextVisit = [0,0,2]
Output: 6
Explanation:
Your room visiting order for each day is: [0,0,1,0,0,1,2,...].
Day 6 is the first day where you have been in all the rooms.

Example 3:

Input: nextVisit = [0,1,2,0]
Output: 6
Explanation:
Your room visiting order for each day is: [0,0,1,1,2,2,3,...].
Day 6 is the first day where you have been in all the rooms.

Constraints:

n == nextVisit.length
2 <= n <= 105
0 <= nextVisit[i] <= i

1998. GCD Sort of an Array

You are given an integer array nums, and you can perform the following operation any number of times on nums:

Swap the positions of two elements nums[i] and nums[j] if gcd(nums[i], nums[j]) > 1 where gcd(nums[i], nums[j]) is the greatest common divisor of nums[i] and nums[j].

Return true if it is possible to sort nums in non-decreasing order using the above swap method, or false otherwise.

Example 1:

Input: nums = [7,21,3]
Output: true
Explanation: We can sort [7,21,3] by performing the following operations:
- Swap 7 and 21 because gcd(7,21) = 7. nums = [21,7,3]
- Swap 21 and 3 because gcd(21,3) = 3. nums = [3,7,21]

Example 2:

Input: nums = [5,2,6,2]
Output: false
Explanation: It is impossible to sort the array because 5 cannot be swapped with any other element.

Example 3:

Input: nums = [10,5,9,3,15]
Output: true
We can sort [10,5,9,3,15] by performing the following operations:
- Swap 10 and 15 because gcd(10,15) = 5. nums = [15,5,9,3,10]
- Swap 15 and 3 because gcd(15,3) = 3. nums = [3,5,9,15,10]
- Swap 10 and 15 because gcd(10,15) = 5. nums = [3,5,9,10,15]

Constraints:

1 <= nums.length <= 3 * 104
2 <= nums[i] <= 105

2000. Reverse Prefix of Word

Given a 0-indexed string word and a character ch, reverse the segment of word that starts at index 0 and ends at the i ndex of the first occurrence of ch (inclusive). If the character ch does not exist in word, do nothing.

For example, if word = "abcdefd" and ch = "d", then you should reverse the segment that starts at 0 and ends at 3 (in

clusive). The resulting string will be "dcbaefd".

Return the resulting string.

Example 1:

Input: word = "abcdefd", ch = "d"
Output: "dcbaefd"
Explanation: The first occurrence of "d" is at index 3.
Reverse the part of word from 0 to 3 (inclusive), the resulting string is "dcbaefd".

Example 2:

Input: word = "xyxzxe", ch = "z"
Output: "zxyxxe"
Explanation: The first and only occurrence of "z" is at index 3.
Reverse the part of word from 0 to 3 (inclusive), the resulting string is "zxyxxe".

Example 3:

Input: word = "abcd", ch = "z"
Output: "abcd"
Explanation: "z" does not exist in word.
You should not do any reverse operation, the resulting string is "abcd".

Constraints:

1 <= word.length <= 250
word consists of lowercase English letters.
ch is a lowercase English letter.

2001. Number of Pairs of Interchangeable Rectangles

You are given n rectangles represented by a 0-indexed 2D integer array rectangles, where rectangles[i] = [widthi, hei
ghti] denotes the width and height of the ith rectangle.
Two rectangles i and j (i < j) are considered interchangeable if they have the same width-to-height ratio. More forma
lly, two rectangles are interchangeable if widthi/heighti == widthj/heightj (using decimal division, not integer divisio
n).
Return the number of pairs of interchangeable rectangles in rectangles.

Example 1:

Input: rectangles = [[4,8],[3,6],[10,20],[15,30]]
Output: 6
Explanation: The following are the interchangeable pairs of rectangles by index (0-indexed):

- Rectangle 0 with rectangle 1: 4/8 == 3/6.
- Rectangle 0 with rectangle 2: 4/8 == 10/20.
- Rectangle 0 with rectangle 3: 4/8 == 15/30.
- Rectangle 1 with rectangle 2: 3/6 == 10/20.
- Rectangle 1 with rectangle 3: 3/6 == 15/30.
- Rectangle 2 with rectangle 3: 10/20 == 15/30.

Example 2:

Input: rectangles = [[4,5],[7,8]]
Output: 0
Explanation: There are no interchangeable pairs of rectangles.

Constraints:

n == rectangles.length
1 <= n <= 105
rectangles[i].length == 2
1 <= widthi, heighti <= 105

2002. Maximum Product of the Length of Two Palindromic Subsequences

Given a string s, find two disjoint palindromic subsequences of s such that the product of their lengths is maximized. The two subsequences are disjoint if they do not both pick a character at the same index.
Return the maximum possible product of the lengths of the two palindromic subsequences.
A subsequence is a string that can be derived from another string by deleting some or no characters without changin g the order of the remaining characters. A string is palindromic if it reads the same forward and backward.

Example 1:

Input: s = "leetcodecom"
Output: 9
Explanation: An optimal solution is to choose "ete" for the 1st subsequence and "cdc" for the 2nd subsequence.
The product of their lengths is: 3 * 3 = 9.

Example 2:

Input: s = "bb"
Output: 1
Explanation: An optimal solution is to choose "b" (the first character) for the 1st subsequence and "b" (the second ch aracter) for the 2nd subsequence.
The product of their lengths is: 1 * 1 = 1.

Example 3:

Input: s = "accbcaxxcxx"
Output: 25
Explanation: An optimal solution is to choose "accca" for the 1st subsequence and "xxcxx" for the 2nd subsequence. The product of their lengths is: 5 * 5 = 25.

Constraints:

2 <= s.length <= 12
s consists of lowercase English letters only.

2003. Smallest Missing Genetic Value in Each Subtree

There is a family tree rooted at 0 consisting of n nodes numbered 0 to n - 1. You are given a 0-indexed integer array parents, where parents[i] is the parent for node i. Since node 0 is the root, parents[0] == -1.
There are 105 genetic values, each represented by an integer in the inclusive range [1, 105]. You are given a 0-index ed integer array nums, where nums[i] is a distinct genetic value for node i.
Return an array ans of length n where ans[i] is the smallest genetic value that is missing from the subtree rooted at n ode i.
The subtree rooted at a node x contains node x and all of its descendant nodes.

Example 1:

Input: parents = [-1,0,0,2], nums = [1,2,3,4]
Output: [5,1,1,1]
Explanation: The answer for each subtree is calculated as follows:
- 0: The subtree contains nodes [0,1,2,3] with values [1,2,3,4]. 5 is the smallest missing value.
- 1: The subtree contains only node 1 with value 2. 1 is the smallest missing value.
- 2: The subtree contains nodes [2,3] with values [3,4]. 1 is the smallest missing value.
- 3: The subtree contains only node 3 with value 4. 1 is the smallest missing value.

Example 2:

Input: parents = [-1,0,1,0,3,3], nums = [5,4,6,2,1,3]
Output: [7,1,1,4,2,1]
Explanation: The answer for each subtree is calculated as follows:
- 0: The subtree contains nodes [0,1,2,3,4,5] with values [5,4,6,2,1,3]. 7 is the smallest missing value.
- 1: The subtree contains nodes [1,2] with values [4,6]. 1 is the smallest missing value.
- 2: The subtree contains only node 2 with value 6. 1 is the smallest missing value.
- 3: The subtree contains nodes [3,4,5] with values [2,1,3]. 4 is the smallest missing value.
- 4: The subtree contains only node 4 with value 1. 2 is the smallest missing value.
- 5: The subtree contains only node 5 with value 3. 1 is the smallest missing value.

Example 3:

Input: parents = [-1,2,3,0,2,4,1], nums = [2,3,4,5,6,7,8]
Output: [1,1,1,1,1,1,1]
Explanation: The value 1 is missing from all the subtrees.

Constraints:

n == parents.length == nums.length
2 <= n <= 105
0 <= parents[i] <= n - 1 for i != 0
parents[0] == -1
parents represents a valid tree.
1 <= nums[i] <= 105
Each nums[i] is distinct.

2006. Count Number of Pairs With Absolute Difference K

Given an integer array nums and an integer k, return the number of pairs (i, j) where i < j such that |nums[i] - nums[j] | == k.
The value of |x| is defined as:

x if x >= 0.
-x if x < 0.

Example 1:

Input: nums = [1,2,2,1], k = 1
Output: 4
Explanation: The pairs with an absolute difference of 1 are:
- [1,2,2,1]
- [1,2,2,1]
- [1,2,2,1]
- [1,2,2,1]

Example 2:

Input: nums = [1,3], k = 3
Output: 0
Explanation: There are no pairs with an absolute difference of 3.

Example 3:

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

Output: 3
Explanation: The pairs with an absolute difference of 2 are:
- [3,2,1,5,4]
- [3,2,1,5,4]
- [3,2,1,5,4]

Constraints:

1 <= nums.length <= 200
1 <= nums[i] <= 100
1 <= k <= 99

2007. Find Original Array From Doubled Array

An integer array original is transformed into a doubled array changed by appending twice the value of every element in original, and then randomly shuffling the resulting array.
Given an array changed, return original if changed is a doubled array. If changed is not a doubled array, return an em pty array. The elements in original may be returned in any order.

Example 1:

Input: changed = [1,3,4,2,6,8]
Output: [1,3,4]
Explanation: One possible original array could be [1,3,4]:
- Twice the value of 1 is 1 * 2 = 2.
- Twice the value of 3 is 3 * 2 = 6.
- Twice the value of 4 is 4 * 2 = 8.
Other original arrays could be [4,3,1] or [3,1,4].

Example 2:

Input: changed = [6,3,0,1]
Output: []
Explanation: changed is not a doubled array.

Example 3:

Input: changed = [1]
Output: []
Explanation: changed is not a doubled array.

Constraints:

1 <= changed.length <= 105

0 <= changed[i] <= 105

2008. Maximum Earnings From Taxi

There are n points on a road you are driving your taxi on. The n points on the road are labeled from 1 to n in the dire ction you are going, and you want to drive from point 1 to point n to make money by picking up passengers. You ca nnot change the direction of the taxi.
The passengers are represented by a 0-indexed 2D integer array rides, where rides[i] = [starti, endi, tipi] denotes the i th passenger requesting a ride from point starti to point endi who is willing to give a tipi dollar tip.
For each passenger i you pick up, you earn endi - starti + tipi dollars. You may only drive at most one passenger at a time.
Given n and rides, return the maximum number of dollars you can earn by picking up the passengers optimally.
Note: You may drop off a passenger and pick up a different passenger at the same point.

Example 1:

Input: n = 5, rides = [[2,5,4],[1,5,1]]
Output: 7
Explanation: We can pick up passenger 0 to earn 5 - 2 + 4 = 7 dollars.

Example 2:

Input: n = 20, rides = [[1,6,1],[3,10,2],[10,12,3],[11,12,2],[12,15,2],[13,18,1]]
Output: 20
Explanation: We will pick up the following passengers:
- Drive passenger 1 from point 3 to point 10 for a profit of 10 - 3 + 2 = 9 dollars.
- Drive passenger 2 from point 10 to point 12 for a profit of 12 - 10 + 3 = 5 dollars.
- Drive passenger 5 from point 13 to point 18 for a profit of 18 - 13 + 1 = 6 dollars.
We earn 9 + 5 + 6 = 20 dollars in total.

Constraints:

1 <= n <= 105
1 <= rides.length <= 3 * 104
rides[i].length == 3
1 <= starti < endi <= n
1 <= tipi <= 105

2009. Minimum Number of Operations to Make Array Continuous

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

2011. Final Value of Variable After Performing Operations

There is a programming language with only four operations and one variable X:

++X and X++ increments the value of the variable X by 1.
--X and X-- decrements the value of the variable X by 1.

Initially, the value of X is 0.
Given an array of strings operations containing a list of operations, return the final value of X after performing all th e operations.

Example 1:

Input: operations = ["--X","X++","X++"]
Output: 1
Explanation: The operations are performed as follows:
Initially, X = 0.
--X: X is decremented by 1, X = 0 - 1 = -1.
X++: X is incremented by 1, X = -1 + 1 = 0.
X++: X is incremented by 1, X = 0 + 1 = 1.

Example 2:

Input: operations = ["++X","++X","X++"]
Output: 3
Explanation: The operations are performed as follows:
Initially, X = 0.
++X: X is incremented by 1, X = 0 + 1 = 1.
++X: X is incremented by 1, X = 1 + 1 = 2.
X++: X is incremented by 1, X = 2 + 1 = 3.

Example 3:

Input: operations = ["X++","++X","--X","X--"]
Output: 0
Explanation: The operations are performed as follows:
Initially, X = 0.
X++: X is incremented by 1, X = 0 + 1 = 1.
++X: X is incremented by 1, X = 1 + 1 = 2.
--X: X is decremented by 1, X = 2 - 1 = 1.
X--: X is decremented by 1, X = 1 - 1 = 0.

Constraints:

1 <= operations.length <= 100
operations[i] will be either "++X", "X++", "--X", or "X--".

2012. Sum of Beauty in the Array

You are given a 0-indexed integer array nums. For each index i (1 <= i <= nums.length - 2) the beauty of nums[i] eq uals:

2, if nums[j] < nums[i] < nums[k], for all 0 <= j < i and for all i < k <= nums.length - 1.
1, if nums[i - 1] < nums[i] < nums[i + 1], and the previous condition is not satisfied.
0, if none of the previous conditions holds.

Return the sum of beauty of all nums[i] where 1 <= i <= nums.length - 2.

Example 1:

Input: nums = [1,2,3]
Output: 2
Explanation: For each index i in the range 1 <= i <= 1:
- The beauty of nums[1] equals 2.

Example 2:

Input: nums = [2,4,6,4]
Output: 1
Explanation: For each index i in the range 1 <= i <= 2:
- The beauty of nums[1] equals 1.
- The beauty of nums[2] equals 0.

Example 3:

Input: nums = [3,2,1]
Output: 0
Explanation: For each index i in the range 1 <= i <= 1:
- The beauty of nums[1] equals 0.

Constraints:

3 <= nums.length <= 105
1 <= nums[i] <= 105

2013. Detect Squares

You are given a stream of points on the X-Y plane. Design an algorithm that:

Adds new points from the stream into a data structure. Duplicate points are allowed and should be treated as differen t points.
Given a query point, counts the number of ways to choose three points from the data structure such that the three poi nts and the query point form an axis-aligned square with positive area.

An axis-aligned square is a square whose edges are all the same length and are either parallel or perpendicular to the x-axis and y-axis.

Implement the DetectSquares class:

DetectSquares() Initializes the object with an empty data structure.
void add(int[] point) Adds a new point point = [x, y] to the data structure.
int count(int[] point) Counts the number of ways to form axis-aligned squares with point point = [x, y] as described a bove.

Example 1:

Input
["DetectSquares", "add", "add", "add", "count", "count", "add", "count"]
[[], [[3, 10]], [[11, 2]], [[3, 2]], [[11, 10]], [[14, 8]], [[11, 2]], [[11, 10]]]
Output
[null, null, null, null, 1, 0, null, 2]

Explanation
DetectSquares detectSquares = new DetectSquares();
detectSquares.add([3, 10]);
detectSquares.add([11, 2]);
detectSquares.add([3, 2]);
detectSquares.count([11, 10]); // return 1. You can choose:
// - The first, second, and third points
detectSquares.count([14, 8]); // return 0. The query point cannot form a square with any points in the data structure.
detectSquares.add([11, 2]); // Adding duplicate points is allowed.
detectSquares.count([11, 10]); // return 2. You can choose:
// - The first, second, and third points
// - The first, third, and fourth points

Constraints:

point.length == 2
0 <= x, y <= 1000
At most 3000 calls in total will be made to add and count.

2014. Longest Subsequence Repeated k Times

You are given a string s of length n, and an integer k. You are tasked to find the longest subsequence repeated k time s in string s.
A subsequence is a string that can be derived from another string by deleting some or no characters without changin g the order of the remaining characters.
A subsequence seq is repeated k times in the string s if seq * k is a subsequence of s, where seq * k represents a strin g constructed by concatenating seq k times.

For example, "bba" is repeated 2 times in the string "bababcba", because the string "bbabba", constructed by concate nating "bba" 2 times, is a subsequence of the string "bababcba".

Return the longest subsequence repeated k times in string s. If multiple such subsequences are found, return the lexic ographically largest one. If there is no such subsequence, return an empty string.

Example 1:

Input: s = "letsleetcode", k = 2
Output: "let"
Explanation: There are two longest subsequences repeated 2 times: "let" and "ete".
"let" is the lexicographically largest one.

Example 2:

Input: s = "bb", k = 2
Output: "b"
Explanation: The longest subsequence repeated 2 times is "b".

Example 3:

Input: s = "ab", k = 2
Output: ""
Explanation: There is no subsequence repeated 2 times. Empty string is returned.

Example 4:

Input: s = "bbabbabbbbabaababab", k = 3
Output: "bbbb"
Explanation: The longest subsequence "bbbb" is repeated 3 times in "bbabbabbbbabaababab".

Constraints:

n == s.length
2 <= n, k <= 2000
2 <= n < k * 8
s consists of lowercase English letters.

2016. Maximum Difference Between Increasing Elements

Given a 0-indexed integer array nums of size n, find the maximum difference between nums[i] and nums[j] (i.e., nu ms[j] - nums[i]), such that 0 <= i < j < n and nums[i] < nums[j].
Return the maximum difference. If no such i and j exists, return -1.

Example 1:

Input: nums = [7,1,5,4]
Output: 4
Explanation:
The maximum difference occurs with i = 1 and j = 2, nums[j] - nums[i] = 5 - 1 = 4.
Note that with i = 1 and j = 0, the difference nums[j] - nums[i] = 7 - 1 = 6, but i > j, so it is not valid.

Example 2:

Input: nums = [9,4,3,2]
Output: -1
Explanation:
There is no i and j such that i < j and nums[i] < nums[j].

Example 3:

Input: nums = [1,5,2,10]
Output: 9
Explanation:
The maximum difference occurs with i = 0 and j = 3, nums[j] - nums[i] = 10 - 1 = 9.

Constraints:

n == nums.length
2 <= n <= 1000
1 <= nums[i] <= 109

2017. Grid Game

You are given a 0-indexed 2D array grid of size 2 x n, where grid[r][c] represents the number of points at position (r, c) on the matrix. Two robots are playing a game on this matrix.
Both robots initially start at (0, 0) and want to reach (1, n-1). Each robot may only move to the right ((r, c) to (r, c + 1)) or down ((r, c) to (r + 1, c)).
At the start of the game, the first robot moves from (0, 0) to (1, n-1), collecting all the points from the cells on its pat h. For all cells (r, c) traversed on the path, grid[r][c] is set to 0. Then, the second robot moves from (0, 0) to (1, n-1), collecting the points on its path. Note that their paths may intersect with one another.
The first robot wants to minimize the number of points collected by the second robot. In contrast, the second robot w ants to maximize the number of points it collects. If both robots play optimally, return the number of points collected by the second robot.

Example 1:

Input: grid = [[2,5,4],[1,5,1]]

Output: 4
Explanation: The optimal path taken by the first robot is shown in red, and the optimal path taken by the second robo
t is shown in blue.
The cells visited by the first robot are set to 0.
The second robot will collect 0 + 0 + 4 + 0 = 4 points.

Example 2:

Input: grid = [[3,3,1],[8,5,2]]
Output: 4
Explanation: The optimal path taken by the first robot is shown in red, and the optimal path taken by the second robo
t is shown in blue.
The cells visited by the first robot are set to 0.
The second robot will collect 0 + 3 + 1 + 0 = 4 points.

Example 3:

Input: grid = [[1,3,1,15],[1,3,3,1]]
Output: 7
Explanation: The optimal path taken by the first robot is shown in red, and the optimal path taken by the second robo
t is shown in blue.
The cells visited by the first robot are set to 0.
The second robot will collect 0 + 1 + 3 + 3 + 0 = 7 points.

Constraints:

grid.length == 2
n == grid[r].length
1 <= n <= 5 * 104
1 <= grid[r][c] <= 105

2018. Check if Word Can Be Placed In Crossword

You are given an m x n matrix board, representing the current state of a crossword puzzle. The crossword contains l
owercase English letters (from solved words), ' ' to represent any empty cells, and '#' to represent any blocked cells.
A word can be placed horizontally (left to right or right to left) or vertically (top to bottom or bottom to top) in the b
oard if:

It does not occupy a cell containing the character '#'.
The cell each letter is placed in must either be ' ' (empty) or match the letter already on the board.
There must not be any empty cells ' ' or other lowercase letters directly left or right of the word if the word was place
d horizontally.
There must not be any empty cells ' ' or other lowercase letters directly above or below the word if the word was plac

ed vertically.

Given a string word, return true if word can be placed in board, or false otherwise.

Example 1:

Input: board = [["#", " ", "#"], [" ", " ", "#"], ["#", "c", " "]], word = "abc"
Output: true
Explanation: The word "abc" can be placed as shown above (top to bottom).

Example 2:

Input: board = [[" ", "#", "a"], [" ", "#", "c"], [" ", "#", "a"]], word = "ac"
Output: false
Explanation: It is impossible to place the word because there will always be a space/letter above or below it.
Example 3:

Input: board = [["#", " ", "#"], [" ", " ", "#"], ["#", " ", "c"]], word = "ca"
Output: true
Explanation: The word "ca" can be placed as shown above (right to left).

Constraints:

m == board.length
n == board[i].length
1 <= m * n <= 2 * 105
board[i][j] will be ' ', '#', or a lowercase English letter.
1 <= word.length <= max(m, n)
word will contain only lowercase English letters.

2019. The Score of Students Solving Math Expression

You are given a string s that contains digits 0-9, addition symbols '+', and multiplication symbols '*' only, representi ng a valid math expression of single digit numbers (e.g., 3+5*2). This expression was given to n elementary school s tudents. The students were instructed to get the answer of the expression by following this order of operations:

Compute multiplication, reading from left to right; Then,
Compute addition, reading from left to right.

You are given an integer array answers of length n, which are the submitted answers of the students in no particular order. You are asked to grade the answers, by following these rules:

If an answer equals the correct answer of the expression, this student will be rewarded 5 points;
Otherwise, if the answer could be interpreted as if the student applied the operators in the wrong order but had correc t arithmetic, this student will be rewarded 2 points;
Otherwise, this student will be rewarded 0 points.

Return the sum of the points of the students.

Example 1:

Input: s = "7+3*1*2", answers = [20,13,42]
Output: 7
Explanation: As illustrated above, the correct answer of the expression is 13, therefore one student is rewarded 5 poi nts: [20,13,42]
A student might have applied the operators in this wrong order: ((7+3)*1)*2 = 20. Therefore one student is rewarded 2 points: [20,13,42]
The points for the students are: [2,5,0]. The sum of the points is 2+5+0=7.

Example 2:

Input: s = "3+5*2", answers = [13,0,10,13,13,16,16]
Output: 19
Explanation: The correct answer of the expression is 13, therefore three students are rewarded 5 points each: [13,0,1 0,13,13,16,16]
A student might have applied the operators in this wrong order: ((3+5)*2 = 16. Therefore two students are rewarded 2 points: [13,0,10,13,13,16,16]
The points for the students are: [5,0,0,5,5,2,2]. The sum of the points is 5+0+0+5+5+2+2=19.

Example 3:

Input: s = "6+0*1", answers = [12,9,6,4,8,6]
Output: 10
Explanation: The correct answer of the expression is 6.
If a student had incorrectly done (6+0)*1, the answer would also be 6.
By the rules of grading, the students will still be rewarded 5 points (as they got the correct answer), not 2 points.
The points for the students are: [0,0,5,0,0,5]. The sum of the points is 10.

Example 4:

Input: s = "1+2*3+4", answers = [13,21,11,15]
Output: 11
Explanation: The correct answer of the expression is 11.
Every other student was rewarded 2 points because they could have applied the operators as follows:
- ((1+2)*3)+4 = 13
- (1+2)*(3+4) = 21
- 1+(2*(3+4)) = 15
The points for the students are: [2,2,5,2]. The sum of the points is 11.

Constraints:

3 <= s.length <= 31
s represents a valid expression that contains only digits 0-9, '+', and '*' only.
All the integer operands in the expression are in the inclusive range [0, 9].

1 <= The count of all operators ('+' and '*') in the math expression <= 15
Test data are generated such that the correct answer of the expression is in the range of [0, 1000].
n == answers.length
1 <= n <= 104
0 <= answers[i] <= 1000

2022. Convert 1D Array Into 2D Array

You are given a 0-indexed 1-dimensional (1D) integer array original, and two integers, m and n. You are tasked with creating a 2-dimensional (2D) array with m rows and n columns using all the elements from original.
The elements from indices 0 to n - 1 (inclusive) of original should form the first row of the constructed 2D array, the elements from indices n to 2 * n - 1 (inclusive) should form the second row of the constructed 2D array, and so on.
Return an m x n 2D array constructed according to the above procedure, or an empty 2D array if it is impossible.

Example 1:

Input: original = [1,2,3,4], m = 2, n = 2
Output: [[1,2],[3,4]]
Explanation:
The constructed 2D array should contain 2 rows and 2 columns.
The first group of n=2 elements in original, [1,2], becomes the first row in the constructed 2D array.
The second group of n=2 elements in original, [3,4], becomes the second row in the constructed 2D array.

Example 2:

Input: original = [1,2,3], m = 1, n = 3
Output: [[1,2,3]]
Explanation:
The constructed 2D array should contain 1 row and 3 columns.
Put all three elements in original into the first row of the constructed 2D array.

Example 3:

Input: original = [1,2], m = 1, n = 1
Output: []
Explanation:
There are 2 elements in original.
It is impossible to fit 2 elements in a 1x1 2D array, so return an empty 2D array.

Example 4:

Input: original = [3], m = 1, n = 2
Output: []
Explanation:
There is 1 element in original.

It is impossible to make 1 element fill all the spots in a 1x2 2D array, so return an empty 2D array.

Constraints:

1 <= original.length <= 5 * 104
1 <= original[i] <= 105
1 <= m, n <= 4 * 104

2023. Number of Pairs of Strings With Concatenation Equal to Target

Given an array of digit strings nums and a digit string target, return the number of pairs of indices (i, j) (where i != j) such that the concatenation of nums[i] + nums[j] equals target.

Example 1:

Input: nums = ["777","7","77","77"], target = "7777"
Output: 4
Explanation: Valid pairs are:
- (0, 1): "777" + "7"
- (1, 0): "7" + "777"
- (2, 3): "77" + "77"
- (3, 2): "77" + "77"

Example 2:

Input: nums = ["123","4","12","34"], target = "1234"
Output: 2
Explanation: Valid pairs are:
- (0, 1): "123" + "4"
- (2, 3): "12" + "34"

Example 3:

Input: nums = ["1","1","1"], target = "11"
Output: 6
Explanation: Valid pairs are:
- (0, 1): "1" + "1"
- (1, 0): "1" + "1"
- (0, 2): "1" + "1"
- (2, 0): "1" + "1"
- (1, 2): "1" + "1"
- (2, 1): "1" + "1"

Constraints:

2 <= nums.length <= 100
1 <= nums[i].length <= 100
2 <= target.length <= 100
nums[i] and target consist of digits.
nums[i] and target do not have leading zeros.

2024. Maximize the Confusion of an Exam

A teacher is writing a test with n true/false questions, with 'T' denoting true and 'F' denoting false. He wants to confu se the students by maximizing the number of consecutive questions with the same answer (multiple trues or multiple falses in a row).
You are given a string answerKey, where answerKey[i] is the original answer to the ith question. In addition, you ar e given an integer k, the maximum number of times you may perform the following operation:

Change the answer key for any question to 'T' or 'F' (i.e., set answerKey[i] to 'T' or 'F').

Return the maximum number of consecutive 'T's or 'F's in the answer key after performing the operation at most k ti mes.

Example 1:

Input: answerKey = "TTFF", k = 2
Output: 4
Explanation: We can replace both the 'F's with 'T's to make answerKey = "TTTT".
There are four consecutive 'T's.

Example 2:

Input: answerKey = "TFFT", k = 1
Output: 3
Explanation: We can replace the first 'T' with an 'F' to make answerKey = "FFFT".
Alternatively, we can replace the second 'T' with an 'F' to make answerKey = "TFFF".
In both cases, there are three consecutive 'F's.

Example 3:

Input: answerKey = "TTFTTFTT", k = 1
Output: 5
Explanation: We can replace the first 'F' to make answerKey = "TTTTTFTT"
Alternatively, we can replace the second 'F' to make answerKey = "TTFTTTTT".
In both cases, there are five consecutive 'T's.

Constraints:

n == answerKey.length
1 <= n <= 5 * 104
answerKey[i] is either 'T' or 'F'
1 <= k <= n

2025. Maximum Number of Ways to Partition an Array

You are given a 0-indexed integer array nums of length n. The number of ways to partition nums is the number of pi vot indices that satisfy both conditions:

1 <= pivot < n
nums[0] + nums[1] + ... + nums[pivot - 1] == nums[pivot] + nums[pivot + 1] + ... + nums[n - 1]

You are also given an integer k. You can choose to change the value of one element of nums to k, or to leave the arra y unchanged.
Return the maximum possible number of ways to partition nums to satisfy both conditions after changing at most on e element.

Example 1:

Input: nums = [2,-1,2], k = 3
Output: 1
Explanation: One optimal approach is to change nums[0] to k. The array becomes [3,-1,2].
There is one way to partition the array:
- For pivot = 2, we have the partition [3,-1 | 2]: 3 + -1 == 2.

Example 2:

Input: nums = [0,0,0], k = 1
Output: 2
Explanation: The optimal approach is to leave the array unchanged.
There are two ways to partition the array:
- For pivot = 1, we have the partition [0 | 0,0]: 0 == 0 + 0.
- For pivot = 2, we have the partition [0,0 | 0]: 0 + 0 == 0.

Example 3:

Input: nums = [22,4,-25,-20,-15,15,-16,7,19,-10,0,-13,-14], k = -33
Output: 4
Explanation: One optimal approach is to change nums[2] to k. The array becomes [22,4,-33,-20,-15,15,-16,7,19,-10, 0,-13,-14].
There are four ways to partition the array.

Constraints:

n == nums.length
2 <= n <= 105
-105 <= k, nums[i] <= 105

2027. Minimum Moves to Convert String

You are given a string s consisting of n characters which are either 'X' or 'O'.
A move is defined as selecting three consecutive characters of s and converting them to 'O'. Note that if a move is ap plied to the character 'O', it will stay the same.
Return the minimum number of moves required so that all the characters of s are converted to 'O'.

Example 1:

Input: s = "XXX"
Output: 1
Explanation: XXX -> OOO
We select all the 3 characters and convert them in one move.

Example 2:

Input: s = "XXOX"
Output: 2
Explanation: XXOX -> OOOX -> OOOO
We select the first 3 characters in the first move, and convert them to 'O'.
Then we select the last 3 characters and convert them so that the final string contains all 'O's.
Example 3:

Input: s = "OOOO"
Output: 0
Explanation: There are no 'X's in s to convert.

Constraints:

3 <= s.length <= 1000
s[i] is either 'X' or 'O'.

2028. Find Missing Observations

You have observations of n + m 6-sided dice rolls with each face numbered from 1 to 6. n of the observations went missing, and you only have the observations of m rolls. Fortunately, you have also calculated the average value of th e n + m rolls.
You are given an integer array rolls of length m where rolls[i] is the value of the ith observation. You are also given t he two integers mean and n.
Return an array of length n containing the missing observations such that the average value of the n + m rolls is exac tly mean. If there are multiple valid answers, return any of them. If no such array exists, return an empty array.
The average value of a set of k numbers is the sum of the numbers divided by k.
Note that mean is an integer, so the sum of the n + m rolls should be divisible by n + m.

Example 1:

Input: rolls = [3,2,4,3], mean = 4, n = 2
Output: [6,6]
Explanation: The mean of all n + m rolls is (3 + 2 + 4 + 3 + 6 + 6) / 6 = 4.

Example 2:

Input: rolls = [1,5,6], mean = 3, n = 4
Output: [2,3,2,2]
Explanation: The mean of all n + m rolls is (1 + 5 + 6 + 2 + 3 + 2 + 2) / 7 = 3.

Example 3:

Input: rolls = [1,2,3,4], mean = 6, n = 4
Output: []
Explanation: It is impossible for the mean to be 6 no matter what the 4 missing rolls are.

Example 4:

Input: rolls = [1], mean = 3, n = 1
Output: [5]
Explanation: The mean of all n + m rolls is (1 + 5) / 2 = 3.

Constraints:

m == rolls.length
1 <= n, m <= 105
1 <= rolls[i], mean <= 6

2029. Stone Game IX

Alice and Bob continue their games with stones. There is a row of n stones, and each stone has an associated value. You are given an integer array stones, where stones[i] is the value of the ith stone.

Alice and Bob take turns, with Alice starting first. On each turn, the player may remove any stone from stones. The player who removes a stone loses if the sum of the values of all removed stones is divisible by 3. Bob will win auto matically if there are no remaining stones (even if it is Alice's turn).
Assuming both players play optimally, return true if Alice wins and false if Bob wins.

Example 1:

Input: stones = [2,1]
Output: true
Explanation: The game will be played as follows:
- Turn 1: Alice can remove either stone.
- Turn 2: Bob removes the remaining stone.
The sum of the removed stones is 1 + 2 = 3 and is divisible by 3. Therefore, Bob loses and Alice wins the game.

Example 2:

Input: stones = [2]
Output: false
Explanation: Alice will remove the only stone, and the sum of the values on the removed stones is 2.
Since all the stones are removed and the sum of values is not divisible by 3, Bob wins the game.

Example 3:

Input: stones = [5,1,2,4,3]
Output: false
Explanation: Bob will always win. One possible way for Bob to win is shown below:
- Turn 1: Alice can remove the second stone with value 1. Sum of removed stones = 1.
- Turn 2: Bob removes the fifth stone with value 3. Sum of removed stones = 1 + 3 = 4.
- Turn 3: Alices removes the fourth stone with value 4. Sum of removed stones = 1 + 3 + 4 = 8.
- Turn 4: Bob removes the third stone with value 2. Sum of removed stones = 1 + 3 + 4 + 2 = 10.
- Turn 5: Alice removes the first stone with value 5. Sum of removed stones = 1 + 3 + 4 + 2 + 5 = 15.
Alice loses the game because the sum of the removed stones (15) is divisible by 3. Bob wins the game.

Constraints:

1 <= stones.length <= 105
1 <= stones[i] <= 104

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

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 ti mes. 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 changin g 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 t hat 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 t imes.

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.

2032. Two Out of Three

Given three integer arrays nums1, nums2, and nums3, return a distinct array containing all the values that are present in at least two out of the three arrays. You may return the values in any order.

Example 1:

Input: nums1 = [1,1,3,2], nums2 = [2,3], nums3 = [3]
Output: [3,2]
Explanation: The values that are present in at least two arrays are:
- 3, in all three arrays.

- 2, in nums1 and nums2.

Example 2:

Input: nums1 = [3,1], nums2 = [2,3], nums3 = [1,2]
Output: [2,3,1]
Explanation: The values that are present in at least two arrays are:
- 2, in nums2 and nums3.
- 3, in nums1 and nums2.
- 1, in nums1 and nums3.

Example 3:

Input: nums1 = [1,2,2], nums2 = [4,3,3], nums3 = [5]
Output: []
Explanation: No value is present in at least two arrays.

Constraints:

1 <= nums1.length, nums2.length, nums3.length <= 100
1 <= nums1[i], nums2[j], nums3[k] <= 100

2033. Minimum Operations to Make a Uni-Value Grid

You are given a 2D integer grid of size m x n and an integer x. In one operation, you can add x to or subtract x from any element in the grid.
A uni-value grid is a grid where all the elements of it are equal.
Return the minimum number of operations to make the grid uni-value. If it is not possible, return -1.

Example 1:

Input: grid = [[2,4],[6,8]], x = 2
Output: 4
Explanation: We can make every element equal to 4 by doing the following:
- Add x to 2 once.
- Subtract x from 6 once.
- Subtract x from 8 twice.
A total of 4 operations were used.

Example 2:

Input: grid = [[1,5],[2,3]], x = 1
Output: 5

Explanation: We can make every element equal to 3.

Example 3:

Input: grid = [[1,2],[3,4]], x = 2
Output: -1
Explanation: It is impossible to make every element equal.

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 105
1 <= m * n <= 105
1 <= x, grid[i][j] <= 104

2034. Stock Price Fluctuation

You are given a stream of records about a particular stock. Each record contains a timestamp and the corresponding price of the stock at that timestamp.
Unfortunately due to the volatile nature of the stock market, the records do not come in order. Even worse, some rec ords may be incorrect. Another record with the same timestamp may appear later in the stream correcting the price o f the previous wrong record.
Design an algorithm that:

Updates the price of the stock at a particular timestamp, correcting the price from any previous records at the timesta mp.
Finds the latest price of the stock based on the current records. The latest price is the price at the latest timestamp rec orded.
Finds the maximum price the stock has been based on the current records.
Finds the minimum price the stock has been based on the current records.

Implement the StockPrice class:

StockPrice() Initializes the object with no price records.
void update(int timestamp, int price) Updates the price of the stock at the given timestamp.
int current() Returns the latest price of the stock.
int maximum() Returns the maximum price of the stock.
int minimum() Returns the minimum price of the stock.

Example 1:

Input

["StockPrice", "update", "update", "current", "maximum", "update", "maximum", "update", "minimum"]
[[], [1, 10], [2, 5], [], [], [1, 3], [], [4, 2], []]
Output
[null, null, null, 5, 10, null, 5, null, 2]

Explanation
StockPrice stockPrice = new StockPrice();
stockPrice.update(1, 10); // Timestamps are [1] with corresponding prices [10].
stockPrice.update(2, 5); // Timestamps are [1,2] with corresponding prices [10,5].
stockPrice.current();   // return 5, the latest timestamp is 2 with the price being 5.
stockPrice.maximum();   // return 10, the maximum price is 10 at timestamp 1.
stockPrice.update(1, 3); // The previous timestamp 1 had the wrong price, so it is updated to 3.
                          // Timestamps are [1,2] with corresponding prices [3,5].
stockPrice.maximum();   // return 5, the maximum price is 5 after the correction.
stockPrice.update(4, 2); // Timestamps are [1,2,4] with corresponding prices [3,5,2].
stockPrice.minimum();   // return 2, the minimum price is 2 at timestamp 4.

Constraints:

1 <= timestamp, price <= 109
At most 105 calls will be made in total to update, current, maximum, and minimum.
current, maximum, and minimum will be called only after update has been called at least once.

2035. Partition Array Into Two Arrays to Minimize Sum Difference

You are given an integer array nums of 2 * n integers. You need to partition nums into two arrays of length n to mini mize the absolute difference of the sums of the arrays. To partition nums, put each element of nums into one of the t wo arrays.
Return the minimum possible absolute difference.

Example 1:

Input: nums = [3,9,7,3]
Output: 2
Explanation: One optimal partition is: [3,9] and [7,3].
The absolute difference between the sums of the arrays is abs((3 + 9) - (7 + 3)) = 2.

Example 2:

Input: nums = [-36,36]
Output: 72
Explanation: One optimal partition is: [-36] and [36].
The absolute difference between the sums of the arrays is abs((-36) - (36)) = 72.

Example 3:

Input: nums = [2,-1,0,4,-2,-9]
Output: 0
Explanation: One optimal partition is: [2,4,-9] and [-1,0,-2].
The absolute difference between the sums of the arrays is abs((2 + 4 + -9) - (-1 + 0 + -2)) = 0.

Constraints:

1 <= n <= 15
nums.length == 2 * n
-107 <= nums[i] <= 107

2037. Minimum Number of Moves to Seat Everyone

There are n seats and n students in a room. You are given an array seats of length n, where seats[i] is the position of t he ith seat. You are also given the array students of length n, where students[j] is the position of the jth student.
You may perform the following move any number of times:

Increase or decrease the position of the ith student by 1 (i.e., moving the ith student from position x to x + 1 or x - 1)

Return the minimum number of moves required to move each student to a seat such that no two students are in the sa me seat.
Note that there may be multiple seats or students in the same position at the beginning.

Example 1:

Input: seats = [3,1,5], students = [2,7,4]
Output: 4
Explanation: The students are moved as follows:
- The first student is moved from from position 2 to position 1 using 1 move.
- The second student is moved from from position 7 to position 5 using 2 moves.
- The third student is moved from from position 4 to position 3 using 1 move.
In total, 1 + 2 + 1 = 4 moves were used.

Example 2:

Input: seats = [4,1,5,9], students = [1,3,2,6]
Output: 7
Explanation: The students are moved as follows:
- The first student is not moved.
- The second student is moved from from position 3 to position 4 using 1 move.
- The third student is moved from from position 2 to position 5 using 3 moves.
- The fourth student is moved from from position 6 to position 9 using 3 moves.

In total, 0 + 1 + 3 + 3 = 7 moves were used.

Example 3:

Input: seats = [2,2,6,6], students = [1,3,2,6]
Output: 4
Explanation: Note that there are two seats at position 2 and two seats at position 6.
The students are moved as follows:
- The first student is moved from from position 1 to position 2 using 1 move.
- The second student is moved from from position 3 to position 6 using 3 moves.
- The third student is not moved.
- The fourth student is not moved.
In total, 1 + 3 + 0 + 0 = 4 moves were used.

Constraints:

n == seats.length == students.length
1 <= n <= 100
1 <= seats[i], students[j] <= 100

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

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 o f 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, Alic e 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 re move 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 rem ove 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'

2039. The Time When the Network Becomes Idle

There is a network of n servers, labeled from 0 to n - 1. You are given a 2D integer array edges, where edges[i] = [ui, vi] indicates there is a message channel between servers ui and vi, and they can pass any number of messages to eac h other directly in one second. You are also given a 0-indexed integer array patience of length n.
All servers are connected, i.e., a message can be passed from one server to any other server(s) directly or indirectly t hrough the message channels.
The server labeled 0 is the master server. The rest are data servers. Each data server needs to send its message to the master server for processing and wait for a reply. Messages move between servers optimally, so every message takes the least amount of time to arrive at the master server. The master server will process all newly arrived messages ins tantly and send a reply to the originating server via the reversed path the message had gone through.

At the beginning of second 0, each data server sends its message to be processed. Starting from second 1, at the begi nning of every second, each data server will check if it has received a reply to the message it sent (including any new ly arrived replies) from the master server:

If it has not, it will resend the message periodically. The data server i will resend the message every patience[i] seco nd(s), i.e., the data server i will resend the message if patience[i] second(s) have elapsed since the last time the messa ge was sent from this server.
Otherwise, no more resending will occur from this server.

The network becomes idle when there are no messages passing between servers or arriving at servers.
Return the earliest second starting from which the network becomes idle.

Example 1:

Input: edges = [[0,1],[1,2]], patience = [0,2,1]
Output: 8
Explanation:
At (the beginning of) second 0,
- Data server 1 sends its message (denoted 1A) to the master server.
- Data server 2 sends its message (denoted 2A) to the master server.

At second 1,
- Message 1A arrives at the master server. Master server processes message 1A instantly and sends a reply 1A back.
- Server 1 has not received any reply. 1 second (1 < patience[1] = 2) elapsed since this server has sent the message, t herefore it does not resend the message.
- Server 2 has not received any reply. 1 second (1 == patience[2] = 1) elapsed since this server has sent the message, therefore it resends the message (denoted 2B).

At second 2,
- The reply 1A arrives at server 1. No more resending will occur from server 1.
- Message 2A arrives at the master server. Master server processes message 2A instantly and sends a reply 2A back.
- Server 2 resends the message (denoted 2C).
...
At second 4,
- The reply 2A arrives at server 2. No more resending will occur from server 2.
...
At second 7, reply 2D arrives at server 2.

Starting from the beginning of the second 8, there are no messages passing between servers or arriving at servers.
This is the time when the network becomes idle.

Example 2:

Input: edges = [[0,1],[0,2],[1,2]], patience = [0,10,10]
Output: 3
Explanation: Data servers 1 and 2 receive a reply back at the beginning of second 2.
From the beginning of the second 3, the network becomes idle.

Constraints:

n == patience.length

2 <= n <= 105
patience[0] == 0
1 <= patience[i] <= 105 for 1 <= i < n
1 <= edges.length <= min(105, n * (n - 1) / 2)
edges[i].length == 2
0 <= ui, vi < n
ui != vi
There are no duplicate edges.
Each server can directly or indirectly reach another server.

2040. Kth Smallest Product of Two Sorted Arrays

Given two sorted 0-indexed integer arrays nums1 and nums2 as well as an integer k, return the kth (1-based) smalles t product of nums1[i] * nums2[j] where 0 <= i < nums1.length and 0 <= j < nums2.length.

Example 1:

Input: nums1 = [2,5], nums2 = [3,4], k = 2
Output: 8
Explanation: The 2 smallest products are:
- nums1[0] * nums2[0] = 2 * 3 = 6
- nums1[0] * nums2[1] = 2 * 4 = 8
The 2nd smallest product is 8.

Example 2:

Input: nums1 = [-4,-2,0,3], nums2 = [2,4], k = 6
Output: 0
Explanation: The 6 smallest products are:
- nums1[0] * nums2[1] = (-4) * 4 = -16
- nums1[0] * nums2[0] = (-4) * 2 = -8
- nums1[1] * nums2[1] = (-2) * 4 = -8
- nums1[1] * nums2[0] = (-2) * 2 = -4
- nums1[2] * nums2[0] = 0 * 2 = 0
- nums1[2] * nums2[1] = 0 * 4 = 0
The 6th smallest product is 0.

Example 3:

Input: nums1 = [-2,-1,0,1,2], nums2 = [-3,-1,2,4,5], k = 3
Output: -6
Explanation: The 3 smallest products are:
- nums1[0] * nums2[4] = (-2) * 5 = -10
- nums1[0] * nums2[3] = (-2) * 4 = -8
- nums1[4] * nums2[0] = 2 * (-3) = -6
The 3rd smallest product is -6.

Constraints:

1 <= nums1.length, nums2.length <= 5 * 104
-105 <= nums1[i], nums2[j] <= 105
1 <= k <= nums1.length * nums2.length
nums1 and nums2 are sorted.

2042. Check if Numbers Are Ascending in a Sentence

A sentence is a list of tokens separated by a single space with no leading or trailing spaces. Every token is either a po sitive number consisting of digits 0-9 with no leading zeros, or a word consisting of lowercase English letters.

For example, "a puppy has 2 eyes 4 legs" is a sentence with seven tokens: "2" and "4" are numbers and the other tok ens such as "puppy" are words.

Given a string s representing a sentence, you need to check if all the numbers in s are strictly increasing from left to r ight (i.e., other than the last number, each number is strictly smaller than the number on its right in s).
Return true if so, or false otherwise.

Example 1:

Input: s = "1 box has 3 blue 4 red 6 green and 12 yellow marbles"
Output: true
Explanation: The numbers in s are: 1, 3, 4, 6, 12.
They are strictly increasing from left to right: 1 < 3 < 4 < 6 < 12.

Example 2:

Input: s = "hello world 5 x 5"
Output: false
Explanation: The numbers in s are: 5, 5. They are not strictly increasing.

Example 3:

Input: s = "sunset is at 7 51 pm overnight lows will be in the low 50 and 60 s"
Output: false
Explanation: The numbers in s are: 7, 51, 50, 60. They are not strictly increasing.

Example 4:

Input: s = "4 5 11 26"
Output: true

Explanation: The numbers in s are: 4, 5, 11, 26.
They are strictly increasing from left to right: 4 < 5 < 11 < 26.

Constraints:

3 <= s.length <= 200
s consists of lowercase English letters, spaces, and digits from 0 to 9, inclusive.
The number of tokens in s is between 2 and 100, inclusive.
The tokens in s are separated by a single space.
There are at least two numbers in s.
Each number in s is a positive number less than 100, with no leading zeros.
s contains no leading or trailing spaces.

2043. Simple Bank System

You have been tasked with writing a program for a popular bank that will automate all its incoming transactions (tra nsfer, deposit, and withdraw). The bank has n accounts numbered from 1 to n. The initial balance of each account is stored in a 0-indexed integer array balance, with the (i + 1)th account having an initial balance of balance[i].
Execute all the valid transactions. A transaction is valid if:

The given account number(s) are between 1 and n, and
The amount of money withdrawn or transferred from is less than or equal to the balance of the account.

Implement the Bank class:

Bank(long[] balance) Initializes the object with the 0-indexed integer array balance.
boolean transfer(int account1, int account2, long money) Transfers money dollars from the account numbered accou nt1 to the account numbered account2. Return true if the transaction was successful, false otherwise.
boolean deposit(int account, long money) Deposit money dollars into the account numbered account. Return true if t he transaction was successful, false otherwise.
boolean withdraw(int account, long money) Withdraw money dollars from the account numbered account. Return tr ue if the transaction was successful, false otherwise.

Example 1:

Input
["Bank", "withdraw", "transfer", "deposit", "transfer", "withdraw"]
[[[10, 100, 20, 50, 30]], [3, 10], [5, 1, 20], [5, 20], [3, 4, 15], [10, 50]]
Output
[null, true, true, true, false, false]

Explanation
Bank bank = new Bank([10, 100, 20, 50, 30]);
bank.withdraw(3, 10);   // return true, account 3 has a balance of $20, so it is valid to withdraw $10.

```
                    // Account 3 has $20 - $10 = $10.
bank.transfer(5, 1, 20); // return true, account 5 has a balance of $30, so it is valid to transfer $20.
                    // Account 5 has $30 - $20 = $10, and account 1 has $10 + $20 = $30.
bank.deposit(5, 20);     // return true, it is valid to deposit $20 to account 5.
                    // Account 5 has $10 + $20 = $30.
bank.transfer(3, 4, 15); // return false, the current balance of account 3 is $10,
                    // so it is invalid to transfer $15 from it.
bank.withdraw(10, 50);   // return false, it is invalid because account 10 does not exist.
```

Constraints:

n == balance.length
1 <= n, account, account1, account2 <= 105
0 <= balance[i], money <= 1012
At most 104 calls will be made to each function transfer, deposit, withdraw.

2044. Count Number of Maximum Bitwise-OR Subsets

Given an integer array nums, find the maximum possible bitwise OR of a subset of nums and return the number of di fferent non-empty subsets with the maximum bitwise OR.
An array a is a subset of an array b if a can be obtained from b by deleting some (possibly zero) elements of b. Two s ubsets are considered different if the indices of the elements chosen are different.
The bitwise OR of an array a is equal to a[0] OR a[1] OR ... OR a[a.length - 1] (0-indexed).

Example 1:

Input: nums = [3,1]
Output: 2
Explanation: The maximum possible bitwise OR of a subset is 3. There are 2 subsets with a bitwise OR of 3:
- [3]
- [3,1]

Example 2:

Input: nums = [2,2,2]
Output: 7
Explanation: All non-empty subsets of [2,2,2] have a bitwise OR of 2. There are 23 - 1 = 7 total subsets.

Example 3:

Input: nums = [3,2,1,5]
Output: 6
Explanation: The maximum possible bitwise OR of a subset is 7. There are 6 subsets with a bitwise OR of 7:
- [3,5]
- [3,1,5]

- [3,2,5]
- [3,2,1,5]
- [2,5]
- [2,1,5]

Constraints:

1 <= nums.length <= 16
1 <= nums[i] <= 105

2045. Second Minimum Time to Reach Destination

A city is represented as a bi-directional connected graph with n vertices where each vertex is labeled from 1 to n (inc lusive). The edges in the graph are represented as a 2D integer array edges, where each edges[i] = [ui, vi] denotes a b i-directional edge between vertex ui and vertex vi. Every vertex pair is connected by at most one edge, and no vertex has an edge to itself. The time taken to traverse any edge is time minutes.
Each vertex has a traffic signal which changes its color from green to red and vice versa every change minutes. All si gnals change at the same time. You can enter a vertex at any time, but can leave a vertex only when the signal is gree n. You cannot wait at a vertex if the signal is green.
The second minimum value is defined as the smallest value strictly larger than the minimum value.

For example the second minimum value of [2, 3, 4] is 3, and the second minimum value of [2, 2, 4] is 4.

Given n, edges, time, and change, return the second minimum time it will take to go from vertex 1 to vertex n.
Notes:

You can go through any vertex any number of times, including 1 and n.
You can assume that when the journey starts, all signals have just turned green.

Example 1:

Input: n = 5, edges = [[1,2],[1,3],[1,4],[3,4],[4,5]], time = 3, change = 5
Output: 13
Explanation:
The figure on the left shows the given graph.
The blue path in the figure on the right is the minimum time path.
The time taken is:
- Start at 1, time elapsed=0
- 1 -> 4: 3 minutes, time elapsed=3
- 4 -> 5: 3 minutes, time elapsed=6
Hence the minimum time needed is 6 minutes.

The red path shows the path to get the second minimum time.
- Start at 1, time elapsed=0

- 1 -> 3: 3 minutes, time elapsed=3
- 3 -> 4: 3 minutes, time elapsed=6
- Wait at 4 for 4 minutes, time elapsed=10
- 4 -> 5: 3 minutes, time elapsed=13
Hence the second minimum time is 13 minutes.

Example 2:

Input: n = 2, edges = [[1,2]], time = 3, change = 2
Output: 11
Explanation:
The minimum time path is 1 -> 2 with time = 3 minutes.
The second minimum time path is 1 -> 2 -> 1 -> 2 with time = 11 minutes.

Constraints:

2 <= n <= 104
n - 1 <= edges.length <= min(2 * 104, n * (n - 1) / 2)
edges[i].length == 2
1 <= ui, vi <= n
ui != vi
There are no duplicate edges.
Each vertex can be reached directly or indirectly from every other vertex.
1 <= time, change <= 103

2047. Number of Valid Words in a Sentence

A sentence consists of lowercase letters ('a' to 'z'), digits ('0' to '9'), hyphens ('-'), punctuation marks ('!', '.', and ','), an d spaces (' ') only. Each sentence can be broken down into one or more tokens separated by one or more spaces ' '.
A token is a valid word if all three of the following are true:

It only contains lowercase letters, hyphens, and/or punctuation (no digits).
There is at most one hyphen '-'. If present, it must be surrounded by lowercase characters ("a-b" is valid, but "-ab" an d "ab-" are not valid).
There is at most one punctuation mark. If present, it must be at the end of the token ("ab,", "cd!", and "." are valid, b ut "a!b" and "c.," are not valid).

Examples of valid words include "a-b.", "afad", "ba-c", "a!", and "!".
Given a string sentence, return the number of valid words in sentence.

Example 1:

Input: sentence = "cat and dog"
Output: 3
Explanation: The valid words in the sentence are "cat", "and", and "dog".

Example 2:

Input: sentence = "!this 1-s b8d!"
Output: 0
Explanation: There are no valid words in the sentence.
"!this" is invalid because it starts with a punctuation mark.
"1-s" and "b8d" are invalid because they contain digits.

Example 3:

Input: sentence = "alice and bob are playing stone-game10"
Output: 5
Explanation: The valid words in the sentence are "alice", "and", "bob", "are", and "playing".
"stone-game10" is invalid because it contains digits.

Example 4:

Input: sentence = "he bought 2 pencils, 3 erasers, and 1 pencil-sharpener."
Output: 6
Explanation: The valid words in the sentence are "he", "bought", "pencils,", "erasers,", "and", and "pencil-sharpener.
".

Constraints:

1 <= sentence.length <= 1000
sentence only contains lowercase English letters, digits, ' ', '-', '!', '.', and ','.
There will be at least 1 token.

2048. Next Greater Numerically Balanced Number

An integer x is numerically balanced if for every digit d in the number x, there are exactly d occurrences of that digit in x.
Given an integer n, return the smallest numerically balanced number strictly greater than n.

Example 1:

Input: n = 1
Output: 22
Explanation:
22 is numerically balanced since:
- The digit 2 occurs 2 times.
It is also the smallest numerically balanced number strictly greater than 1.

Example 2:

Input: n = 1000
Output: 1333
Explanation:
1333 is numerically balanced since:
- The digit 1 occurs 1 time.
- The digit 3 occurs 3 times.
It is also the smallest numerically balanced number strictly greater than 1000.
Note that 1022 cannot be the answer because 0 appeared more than 0 times.

Example 3:

Input: n = 3000
Output: 3133
Explanation:
3133 is numerically balanced since:
- The digit 1 occurs 1 time.
- The digit 3 occurs 3 times.
It is also the smallest numerically balanced number strictly greater than 3000.

Constraints:

0 <= n <= 106

2049. Count Nodes With the Highest Score

There is a binary tree rooted at 0 consisting of n nodes. The nodes are labeled from 0 to n - 1. You are given a 0-inde xed integer array parents representing the tree, where parents[i] is the parent of node i. Since node 0 is the root, pare nts[0] == -1.
Each node has a score. To find the score of a node, consider if the node and the edges connected to it were removed. The tree would become one or more non-empty subtrees. The size of a subtree is the number of the nodes in it. The s core of the node is the product of the sizes of all those subtrees.
Return the number of nodes that have the highest score.

Example 1:

Input: parents = [-1,2,0,2,0]
Output: 3
Explanation:
- The score of node 0 is: 3 * 1 = 3
- The score of node 1 is: 4 = 4
- The score of node 2 is: 1 * 1 * 2 = 2
- The score of node 3 is: 4 = 4
- The score of node 4 is: 4 = 4

The highest score is 4, and three nodes (node 1, node 3, and node 4) have the highest score.

Example 2:

Input: parents = [-1,2,0]
Output: 2
Explanation:
- The score of node 0 is: 2 = 2
- The score of node 1 is: 2 = 2
- The score of node 2 is: 1 * 1 = 1
The highest score is 2, and two nodes (node 0 and node 1) have the highest score.

Constraints:

n == parents.length
2 <= n <= 105
parents[0] == -1
0 <= parents[i] <= n - 1 for i != 0
parents represents a valid binary tree.

2050. Parallel Courses III

You are given an integer n, which indicates that there are n courses labeled from 1 to n. You are also given a 2D inte ger array relations where relations[j] = [prevCoursej, nextCoursej] denotes that course prevCoursej has to be comple ted before course nextCoursej (prerequisite relationship). Furthermore, you are given a 0-indexed integer array time where time[i] denotes how many months it takes to complete the (i+1)th course.
You must find the minimum number of months needed to complete all the courses following these rules:

You may start taking a course at any time if the prerequisites are met.
Any number of courses can be taken at the same time.

Return the minimum number of months needed to complete all the courses.
Note: The test cases are generated such that it is possible to complete every course (i.e., the graph is a directed acycli c graph).

Example 1:

Input: n = 3, relations = [[1,3],[2,3]], time = [3,2,5]
Output: 8
Explanation: The figure above represents the given graph and the time required to complete each course.
We start course 1 and course 2 simultaneously at month 0.
Course 1 takes 3 months and course 2 takes 2 months to complete respectively.
Thus, the earliest time we can start course 3 is at month 3, and the total time required is 3 + 5 = 8 months.

Example 2:

Input: n = 5, relations = [[1,5],[2,5],[3,5],[3,4],[4,5]], time = [1,2,3,4,5]
Output: 12
Explanation: The figure above represents the given graph and the time required to complete each course.
You can start courses 1, 2, and 3 at month 0.
You can complete them after 1, 2, and 3 months respectively.
Course 4 can be taken only after course 3 is completed, i.e., after 3 months. It is completed after 3 + 4 = 7 months.
Course 5 can be taken only after courses 1, 2, 3, and 4 have been completed, i.e., after max(1,2,3,7) = 7 months.
Thus, the minimum time needed to complete all the courses is 7 + 5 = 12 months.

Constraints:

1 <= n <= 5 * 104
0 <= relations.length <= min(n * (n - 1) / 2, 5 * 104)
relations[j].length == 2
1 <= prevCoursej, nextCoursej <= n
prevCoursej != nextCoursej
All the pairs [prevCoursej, nextCoursej] are unique.
time.length == n
1 <= time[i] <= 104
The given graph is a directed acyclic graph.

2053. Kth Distinct String in an Array

A distinct string is a string that is present only once in an array.
Given an array of strings arr, and an integer k, return the kth distinct string present in arr. If there are fewer than k di stinct strings, return an empty string "".
Note that the strings are considered in the order in which they appear in the array.

Example 1:

Input: arr = ["d","b","c","b","c","a"], k = 2
Output: "a"
Explanation:
The only distinct strings in arr are "d" and "a".
"d" appears 1st, so it is the 1st distinct string.
"a" appears 2nd, so it is the 2nd distinct string.
Since k == 2, "a" is returned.

Example 2:

Input: arr = ["aaa","aa","a"], k = 1

Output: "aaa"
Explanation:
All strings in arr are distinct, so the 1st string "aaa" is returned.

Example 3:

Input: arr = ["a","b","a"], k = 3
Output: ""
Explanation:
The only distinct string is "b". Since there are fewer than 3 distinct strings, we return an empty string "".

Constraints:

1 <= k <= arr.length <= 1000
1 <= arr[i].length <= 5
arr[i] consists of lowercase English letters.

2054. Two Best Non-Overlapping Events

You are given a 0-indexed 2D integer array of events where events[i] = [startTimei, endTimei, valuei]. The ith event starts at startTimei and ends at endTimei, and if you attend this event, you will receive a value of valuei. You can ch oose at most two non-overlapping events to attend such that the sum of their values is maximized.
Return this maximum sum.
Note that the start time and end time is inclusive: that is, you cannot attend two events where one of them starts and t he other ends at the same time. More specifically, if you attend an event with end time t, the next event must start at or after t + 1.

Example 1:

Input: events = [[1,3,2],[4,5,2],[2,4,3]]
Output: 4
Explanation: Choose the green events, 0 and 1 for a sum of 2 + 2 = 4.

Example 2:

Input: events = [[1,3,2],[4,5,2],[1,5,5]]
Output: 5
Explanation: Choose event 2 for a sum of 5.

Example 3:

Input: events = [[1,5,3],[1,5,1],[6,6,5]]

Output: 8
Explanation: Choose events 0 and 2 for a sum of 3 + 5 = 8.

Constraints:

2 <= events.length <= 105
events[i].length == 3
1 <= startTimei <= endTimei <= 109
1 <= valuei <= 106

2055. Plates Between Candles

There is a long table with a line of plates and candles arranged on top of it. You are given a 0-indexed string s consis ting of characters '*' and '|' only, where a '*' represents a plate and a '|' represents a candle.
You are also given a 0-indexed 2D integer array queries where queries[i] = [lefti, righti] denotes the substring s[lefti. ..righti] (inclusive). For each query, you need to find the number of plates between candles that are in the substring. A plate is considered between candles if there is at least one candle to its left and at least one candle to its right in the substring.

For example, s = "||**||**|*", and a query [3, 8] denotes the substring "*||**|". The number of plates between candles in this substring is 2, as each of the two plates has at least one candle in the substring to its left and right.

Return an integer array answer where answer[i] is the answer to the ith query.

Example 1:

Input: s = "**|**|***|", queries = [[2,5],[5,9]]
Output: [2,3]
Explanation:
- queries[0] has two plates between candles.
- queries[1] has three plates between candles.

Example 2:

Input: s = "***|**|*****|**||**|*", queries = [[1,17],[4,5],[14,17],[5,11],[15,16]]
Output: [9,0,0,0,0]
Explanation:
- queries[0] has nine plates between candles.
- The other queries have zero plates between candles.

Constraints:

3 <= s.length <= 105

s consists of '*' and '|' characters.
1 <= queries.length <= 105
queries[i].length == 2
0 <= lefti <= righti < s.length

2056. Number of Valid Move Combinations On Chessboard

There is an 8 x 8 chessboard containing n pieces (rooks, queens, or bishops). You are given a string array pieces of l ength n, where pieces[i] describes the type (rook, queen, or bishop) of the ith piece. In addition, you are given a 2D i nteger array positions also of length n, where positions[i] = [ri, ci] indicates that the ith piece is currently at the 1-bas ed coordinate (ri, ci) on the chessboard.
When making a move for a piece, you choose a destination square that the piece will travel toward and stop on.

A rook can only travel horizontally or vertically from (r, c) to the direction of (r+1, c), (r-1, c), (r, c+1), or (r, c-1).
A queen can only travel horizontally, vertically, or diagonally from (r, c) to the direction of (r+1, c), (r-1, c), (r, c+1), (r, c-1), (r+1, c+1), (r+1, c-1), (r-1, c+1), (r-1, c-1).
A bishop can only travel diagonally from (r, c) to the direction of (r+1, c+1), (r+1, c-1), (r-1, c+1), (r-1, c-1).

You must make a move for every piece on the board simultaneously. A move combination consists of all the moves performed on all the given pieces. Every second, each piece will instantaneously travel one square towards their dest ination if they are not already at it. All pieces start traveling at the 0th second. A move combination is invalid if, at a given time, two or more pieces occupy the same square.
Return the number of valid move combinations .
Notes:

No two pieces will start in the same square.
You may choose the square a piece is already on as its destination.
If two pieces are directly adjacent to each other, it is valid for them to move past each other and swap positions in on e second.

Example 1:

Input: pieces = ["rook"], positions = [[1,1]]
Output: 15
Explanation: The image above shows the possible squares the piece can move to.

Example 2:

Input: pieces = ["queen"], positions = [[1,1]]
Output: 22
Explanation: The image above shows the possible squares the piece can move to.

Example 3:

Input: pieces = ["bishop"], positions = [[4,3]]
Output: 12
Explanation: The image above shows the possible squares the piece can move to.

Example 4:

Input: pieces = ["rook","rook"], positions = [[1,1],[8,8]]
Output: 223
Explanation: There are 15 moves for each rook which results in 15 * 15 = 225 move combinations.
However, there are two invalid move combinations:
- Move both rooks to (8, 1), where they collide.
- Move both rooks to (1, 8), where they collide.
Thus there are 225 - 2 = 223 valid move combinations.
Note that there are two valid move combinations that would result in one rook at (1, 8) and the other at (8, 1).
Even though the board state is the same, these two move combinations are considered different since the moves the mselves are different.

Example 5:

Input: pieces = ["queen","bishop"], positions = [[5,7],[3,4]]
Output: 281
Explanation: There are 12 * 24 = 288 move combinations.
However, there are several invalid move combinations:
- If the queen stops at (6, 7), it blocks the bishop from moving to (6, 7) or (7, 8).
- If the queen stops at (5, 6), it blocks the bishop from moving to (5, 6), (6, 7), or (7, 8).
- If the bishop stops at (5, 2), it blocks the queen from moving to (5, 2) or (5, 1).
Of the 288 move combinations, 281 are valid.

Constraints:

n == pieces.length
n == positions.length
1 <= n <= 4
pieces only contains the strings "rook", "queen", and "bishop".
There will be at most one queen on the chessboard.
1 <= xi, yi <= 8
Each positions[i] is distinct.

2057. Smallest Index With Equal Value

Given a 0-indexed integer array nums, return the smallest index i of nums such that i mod 10 == nums[i], or -1 if suc

h index does not exist.
x mod y denotes the remainder when x is divided by y.

Example 1:

Input: nums = [0,1,2]
Output: 0
Explanation:
i=0: 0 mod 10 = 0 == nums[0].
i=1: 1 mod 10 = 1 == nums[1].
i=2: 2 mod 10 = 2 == nums[2].
All indices have i mod 10 == nums[i], so we return the smallest index 0.

Example 2:

Input: nums = [4,3,2,1]
Output: 2
Explanation:
i=0: 0 mod 10 = 0 != nums[0].
i=1: 1 mod 10 = 1 != nums[1].
i=2: 2 mod 10 = 2 == nums[2].
i=3: 3 mod 10 = 3 != nums[3].
2 is the only index which has i mod 10 == nums[i].

Example 3:

Input: nums = [1,2,3,4,5,6,7,8,9,0]
Output: -1
Explanation: No index satisfies i mod 10 == nums[i].

Example 4:

Input: nums = [2,1,3,5,2]
Output: 1
Explanation: 1 is the only index with i mod 10 == nums[i].

Constraints:

1 <= nums.length <= 100
0 <= nums[i] <= 9

2058. Find the Minimum and Maximum Number of Nodes Between Critical Points

A critical point in a linked list is defined as either a local maxima or a local minima.
A node is a local maxima if the current node has a value strictly greater than the previous node and the next node.

A node is a local minima if the current node has a value strictly smaller than the previous node and the next node.
Note that a node can only be a local maxima/minima if there exists both a previous node and a next node.
Given a linked list head, return an array of length 2 containing [minDistance, maxDistance] where minDistance is th e minimum distance between any two distinct critical points and maxDistance is the maximum distance between any two distinct critical points. If there are fewer than two critical points, return [-1, -1].

Example 1:

Input: head = [3,1]
Output: [-1,-1]
Explanation: There are no critical points in [3,1].

Example 2:

Input: head = [5,3,1,2,5,1,2]
Output: [1,3]
Explanation: There are three critical points:
- [5,3,1,2,5,1,2]: The third node is a local minima because 1 is less than 3 and 2.
- [5,3,1,2,5,1,2]: The fifth node is a local maxima because 5 is greater than 2 and 1.
- [5,3,1,2,5,1,2]: The sixth node is a local minima because 1 is less than 5 and 2.
The minimum distance is between the fifth and the sixth node. minDistance = 6 - 5 = 1.
The maximum distance is between the third and the sixth node. maxDistance = 6 - 3 = 3.

Example 3:

Input: head = [1,3,2,2,3,2,2,2,7]
Output: [3,3]
Explanation: There are two critical points:
- [1,3,2,2,3,2,2,2,7]: The second node is a local maxima because 3 is greater than 1 and 2.
- [1,3,2,2,3,2,2,2,7]: The fifth node is a local maxima because 3 is greater than 2 and 2.
Both the minimum and maximum distances are between the second and the fifth node.
Thus, minDistance and maxDistance is 5 - 2 = 3.
Note that the last node is not considered a local maxima because it does not have a next node.

Example 4:

Input: head = [2,3,3,2]
Output: [-1,-1]
Explanation: There are no critical points in [2,3,3,2].

Constraints:

The number of nodes in the list is in the range [2, 105].
1 <= Node.val <= 105

2059. Minimum Operations to Convert Number

You are given a 0-indexed integer array nums containing distinct numbers, an integer start, and an integer goal. Ther e is an integer x that is initially set to start, and you want to perform operations on x such that it is converted to goal. You can perform the following operation repeatedly on the number x:
If 0 <= x <= 1000, then for any index i in the array (0 <= i < nums.length), you can set x to any of the following:

x + nums[i]
x - nums[i]
x ^ nums[i] (bitwise-XOR)

Note that you can use each nums[i] any number of times in any order. Operations that set x to be out of the range 0 < = x <= 1000 are valid, but no more operations can be done afterward.
Return the minimum number of operations needed to convert x = start into goal, and -1 if it is not possible.

Example 1:

Input: nums = [1,3], start = 6, goal = 4
Output: 2
Explanation:
We can go from 6 → 7 → 4 with the following 2 operations.
- 6 ^ 1 = 7
- 7 ^ 3 = 4

Example 2:

Input: nums = [2,4,12], start = 2, goal = 12
Output: 2
Explanation:
We can go from 2 → 14 → 12 with the following 2 operations.
- 2 + 12 = 14
- 14 - 2 = 12

Example 3:

Input: nums = [3,5,7], start = 0, goal = -4
Output: 2
Explanation:
We can go from 0 → 3 → -4 with the following 2 operations.
- 0 + 3 = 3
- 3 - 7 = -4
Note that the last operation sets x out of the range 0 <= x <= 1000, which is valid.

Example 4:

Input: nums = [2,8,16], start = 0, goal = 1
Output: -1
Explanation:
There is no way to convert 0 into 1.
Example 5:

Input: nums = [1], start = 0, goal = 3
Output: 3
Explanation:
We can go from 0 → 1 → 2 → 3 with the following 3 operations.
- 0 + 1 = 1
- 1 + 1 = 2
- 2 + 1 = 3

Constraints:

1 <= nums.length <= 1000
-109 <= nums[i], goal <= 109
0 <= start <= 1000
start != goal
All the integers in nums are distinct.

2060. Check if an Original String Exists Given Two Encoded Strings

An original string, consisting of lowercase English letters, can be encoded by the following steps:

Arbitrarily split it into a sequence of some number of non-empty substrings.
Arbitrarily choose some elements (possibly none) of the sequence, and replace each with its length (as a numeric stri ng).
Concatenate the sequence as the encoded string.

For example, one way to encode an original string "abcdefghijklmnop" might be:

Split it as a sequence: ["ab", "cdefghijklmn", "o", "p"].
Choose the second and third elements to be replaced by their lengths, respectively. The sequence becomes ["ab", "12 ", "1", "p"].
Concatenate the elements of the sequence to get the encoded string: "ab121p".

Given two encoded strings s1 and s2, consisting of lowercase English letters and digits 1-9 (inclusive), return true if there exists an original string that could be encoded as both s1 and s2. Otherwise, return false.
Note: The test cases are generated such that the number of consecutive digits in s1 and s2 does not exceed 3.

Example 1:

Input: s1 = "internationalization", s2 = "i18n"
Output: true
Explanation: It is possible that "internationalization" was the original string.
- "internationalization"
  -> Split: ["internationalization"]
  -> Do not replace any element

  -> Concatenate: "internationalization", which is s1.
- "internationalization"
  -> Split: ["i", "nternationalizatio", "n"]
  -> Replace: ["i", "18",   "n"]
  -> Concatenate: "i18n", which is s2

Example 2:

Input: s1 = "l123e", s2 = "44"
Output: true
Explanation: It is possible that "leetcode" was the original string.
- "leetcode"
  -> Split: ["l", "e", "et", "cod", "e"]
  -> Replace: ["l", "1", "2", "3",   "e"]
  -> Concatenate: "l123e", which is s1.
- "leetcode"
  -> Split: ["leet", "code"]
  -> Replace: ["4",   "4"]
  -> Concatenate: "44", which is s2.

Example 3:

Input: s1 = "a5b", s2 = "c5b"
Output: false
Explanation: It is impossible.
- The original string encoded as s1 must start with the letter 'a'.
- The original string encoded as s2 must start with the letter 'c'.

Example 4:

Input: s1 = "112s", s2 = "g841"
Output: true
Explanation: It is possible that "gaaaaaaaaaaas" was the original string
- "gaaaaaaaaaaas"
  -> Split: ["g", "aaaaaaaaaaa", "s"]
  -> Replace: ["1", "12",   "s"]
  -> Concatenate: "112s", which is s1.
- "gaaaaaaaaaaas"
  -> Split: ["g", "aaaaaaaa", "aaaa", "s"]
  -> Replace: ["g", "8",   "4",   "1"]
  -> Concatenate: "g841", which is s2.

Example 5:

Input: s1 = "ab", s2 = "a2"
Output: false
Explanation: It is impossible.
- The original string encoded as s1 has two letters.
- The original string encoded as s2 has three letters.

Constraints:

1 <= s1.length, s2.length <= 40

s1 and s2 consist of digits 1-9 (inclusive), and lowercase English letters only.
The number of consecutive digits in s1 and s2 does not exceed 3.

2062. Count Vowel Substrings of a String

A substring is a contiguous (non-empty) sequence of characters within a string.
A vowel substring is a substring that only consists of vowels ('a', 'e', 'i', 'o', and 'u') and has all five vowels present in it.
Given a string word, return the number of vowel substrings in word.

Example 1:

Input: word = "aeiouu"
Output: 2
Explanation: The vowel substrings of word are as follows (underlined):
- "aeiouu"
- "aeiouu"

Example 2:

Input: word = "unicornarihan"
Output: 0
Explanation: Not all 5 vowels are present, so there are no vowel substrings.

Example 3:

Input: word = "cuaieuouac"
Output: 7
Explanation: The vowel substrings of word are as follows (underlined):
- "cuaieuouac"
- "cuaieuouac"
- "cuaieuouac"
- "cuaieuouac"
- "cuaieuouac"
- "cuaieuouac"
- "cuaieuouac"
Example 4:

Input: word = "bbaeixoubb"
Output: 0
Explanation: The only substrings that contain all five vowels also contain consonants, so there are no vowel substrin gs.

Constraints:

1 <= word.length <= 100
word consists of lowercase English letters only.

2063. Vowels of All Substrings

Given a string word, return the sum of the number of vowels ('a', 'e', 'i', 'o', and 'u') in every substring of word.
A substring is a contiguous (non-empty) sequence of characters within a string.
Note: Due to the large constraints, the answer may not fit in a signed 32-bit integer. Please be careful during the calc ulations.

Example 1:

Input: word = "aba"
Output: 6
Explanation:
All possible substrings are: "a", "ab", "aba", "b", "ba", and "a".
- "b" has 0 vowels in it
- "a", "ab", "ba", and "a" have 1 vowel each
- "aba" has 2 vowels in it
Hence, the total sum of vowels = 0 + 1 + 1 + 1 + 1 + 2 = 6.

Example 2:

Input: word = "abc"
Output: 3
Explanation:
All possible substrings are: "a", "ab", "abc", "b", "bc", and "c".
- "a", "ab", and "abc" have 1 vowel each
- "b", "bc", and "c" have 0 vowels each
Hence, the total sum of vowels = 1 + 1 + 1 + 0 + 0 + 0 = 3.
Example 3:

Input: word = "ltcd"
Output: 0
Explanation: There are no vowels in any substring of "ltcd".
Example 4:

Input: word = "noosabasboosa"
Output: 237
Explanation: There are a total of 237 vowels in all the substrings.

Constraints:

1 <= word.length <= 105
word consists of lowercase English letters.

2064. Minimized Maximum of Products Distributed to Any Store

You are given an integer n indicating there are n specialty retail stores. There are m product types of varying amount s, which are given as a 0-indexed integer array quantities, where quantities[i] represents the number of products of th e ith product type.
You need to distribute all products to the retail stores following these rules:

A store can only be given at most one product type but can be given any amount of it.
After distribution, each store will have been given some number of products (possibly 0). Let x represent the maxim um number of products given to any store. You want x to be as small as possible, i.e., you want to minimize the max imum number of products that are given to any store.

Return the minimum possible x.

Example 1:

Input: n = 6, quantities = [11,6]
Output: 3
Explanation: One optimal way is:
- The 11 products of type 0 are distributed to the first four stores in these amounts: 2, 3, 3, 3
- The 6 products of type 1 are distributed to the other two stores in these amounts: 3, 3
The maximum number of products given to any store is max(2, 3, 3, 3, 3, 3) = 3.

Example 2:

Input: n = 7, quantities = [15,10,10]
Output: 5
Explanation: One optimal way is:
- The 15 products of type 0 are distributed to the first three stores in these amounts: 5, 5, 5
- The 10 products of type 1 are distributed to the next two stores in these amounts: 5, 5
- The 10 products of type 2 are distributed to the last two stores in these amounts: 5, 5
The maximum number of products given to any store is max(5, 5, 5, 5, 5, 5, 5) = 5.

Example 3:

Input: n = 1, quantities = [100000]
Output: 100000
Explanation: The only optimal way is:
- The 100000 products of type 0 are distributed to the only store.
The maximum number of products given to any store is max(100000) = 100000.

Constraints:

m == quantities.length

1 <= m <= n <= 105
1 <= quantities[i] <= 105

2065. Maximum Path Quality of a Graph

There is an undirected graph with n nodes numbered from 0 to n - 1 (inclusive). You are given a 0-indexed integer ar ray values where values[i] is the value of the ith node. You are also given a 0-indexed 2D integer array edges, where each edges[j] = [uj, vj, timej] indicates that there is an undirected edge between the nodes uj and vj, and it takes time j seconds to travel between the two nodes. Finally, you are given an integer maxTime.
A valid path in the graph is any path that starts at node 0, ends at node 0, and takes at most maxTime seconds to com plete. You may visit the same node multiple times. The quality of a valid path is the sum of the values of the unique nodes visited in the path (each node's value is added at most once to the sum).
Return the maximum quality of a valid path.
Note: There are at most four edges connected to each node.

Example 1:

Input: values = [0,32,10,43], edges = [[0,1,10],[1,2,15],[0,3,10]], maxTime = 49
Output: 75
Explanation:
One possible path is 0 -> 1 -> 0 -> 3 -> 0. The total time taken is 10 + 10 + 10 + 10 = 40 <= 49.
The nodes visited are 0, 1, and 3, giving a maximal path quality of 0 + 32 + 43 = 75.

Example 2:

Input: values = [5,10,15,20], edges = [[0,1,10],[1,2,10],[0,3,10]], maxTime = 30
Output: 25
Explanation:
One possible path is 0 -> 3 -> 0. The total time taken is 10 + 10 = 20 <= 30.
The nodes visited are 0 and 3, giving a maximal path quality of 5 + 20 = 25.

Example 3:

Input: values = [1,2,3,4], edges = [[0,1,10],[1,2,11],[2,3,12],[1,3,13]], maxTime = 50
Output: 7
Explanation:
One possible path is 0 -> 1 -> 3 -> 1 -> 0. The total time taken is 10 + 13 + 13 + 10 = 46 <= 50.
The nodes visited are 0, 1, and 3, giving a maximal path quality of 1 + 2 + 4 = 7.
Example 4:

Input: values = [0,1,2], edges = [[1,2,10]], maxTime = 10
Output: 0

Explanation:
The only path is 0. The total time taken is 0.
The only node visited is 0, giving a maximal path quality of 0.

Constraints:

n == values.length
1 <= n <= 1000
0 <= values[i] <= 108
0 <= edges.length <= 2000
edges[j].length == 3
0 <= uj < vj <= n - 1
10 <= timej, maxTime <= 100
All the pairs [uj, vj] are unique.
There are at most four edges connected to each node.
The graph may not be connected.

2068. Check Whether Two Strings are Almost Equivalent

Two strings word1 and word2 are considered almost equivalent if the differences between the frequencies of each let ter from 'a' to 'z' between word1 and word2 is at most 3.
Given two strings word1 and word2, each of length n, return true if word1 and word2 are almost equivalent, or false otherwise.
The frequency of a letter x is the number of times it occurs in the string.

Example 1:

Input: word1 = "aaaa", word2 = "bccb"
Output: false
Explanation: There are 4 'a's in "aaaa" but 0 'a's in "bccb".
The difference is 4, which is more than the allowed 3.

Example 2:

Input: word1 = "abcdeef", word2 = "abaaacc"
Output: true
Explanation: The differences between the frequencies of each letter in word1 and word2 are at most 3:
- 'a' appears 1 time in word1 and 4 times in word2. The difference is 3.
- 'b' appears 1 time in word1 and 1 time in word2. The difference is 0.
- 'c' appears 1 time in word1 and 2 times in word2. The difference is 1.
- 'd' appears 1 time in word1 and 0 times in word2. The difference is 1.
- 'e' appears 2 times in word1 and 0 times in word2. The difference is 2.
- 'f' appears 1 time in word1 and 0 times in word2. The difference is 1.

Example 3:

Input: word1 = "cccddabba", word2 = "babababab"
Output: true
Explanation: The differences between the frequencies of each letter in word1 and word2 are at most 3:
- 'a' appears 2 times in word1 and 4 times in word2. The difference is 2.
- 'b' appears 2 times in word1 and 5 times in word2. The difference is 3.
- 'c' appears 3 times in word1 and 0 times in word2. The difference is 3.
- 'd' appears 2 times in word1 and 0 times in word2. The difference is 2.

Constraints:

n == word1.length == word2.length
1 <= n <= 100
word1 and word2 consist only of lowercase English letters.

2069. Walking Robot Simulation II

A width x height grid is on an XY-plane with the bottom-left cell at (0, 0) and the top-right cell at (width - 1, height - 1). The grid is aligned with the four cardinal directions ("North", "East", "South", and "West"). A robot is initially at cell (0, 0) facing direction "East".
The robot can be instructed to move for a specific number of steps. For each step, it does the following.

Attempts to move forward one cell in the direction it is facing.
If the cell the robot is moving to is out of bounds, the robot instead turns 90 degrees counterclockwise and retries the step.

After the robot finishes moving the number of steps required, it stops and awaits the next instruction.
Implement the Robot class:

Robot(int width, int height) Initializes the width x height grid with the robot at (0, 0) facing "East".
void step(int num) Instructs the robot to move forward num steps.
int[] getPos() Returns the current cell the robot is at, as an array of length 2, [x, y].
String getDir() Returns the current direction of the robot, "North", "East", "South", or "West".

Example 1:

Input
["Robot", "move", "move", "getPos", "getDir", "move", "move", "move", "getPos", "getDir"]
[[6, 3], [2], [2], [], [], [2], [1], [4], [], []]
Output
[null, null, null, [4, 0], "East", null, null, null, [1, 2], "West"]

Explanation

```
Robot robot = new Robot(6, 3); // Initialize the grid and the robot at (0, 0) facing East.
robot.move(2);  // It moves two steps East to (2, 0), and faces East.
robot.move(2);  // It moves two steps East to (4, 0), and faces East.
robot.getPos(); // return [4, 0]
robot.getDir(); // return "East"
robot.move(2);  // It moves one step East to (5, 0), and faces East.
                // Moving the next step East would be out of bounds, so it turns and faces North.
                // Then, it moves one step North to (5, 1), and faces North.
robot.move(1);  // It moves one step North to (5, 2), and faces North (not West).
robot.move(4);  // Moving the next step North would be out of bounds, so it turns and faces West.
                // Then, it moves four steps West to (1, 2), and faces West.
robot.getPos(); // return [1, 2]
robot.getDir(); // return "West"
```

Constraints:

2 <= width, height <= 100
1 <= num <= 105
At most 104 calls in total will be made to step, getPos, and getDir.

2070. Most Beautiful Item for Each Query

You are given a 2D integer array items where items[i] = [pricei, beautyi] denotes the price and beauty of an item res pectively.
You are also given a 0-indexed integer array queries. For each queries[j], you want to determine the maximum beaut y of an item whose price is less than or equal to queries[j]. If no such item exists, then the answer to this query is 0.
Return an array answer of the same length as queries where answer[j] is the answer to the jth query.

Example 1:

Input: items = [[1,2],[3,2],[2,4],[5,6],[3,5]], queries = [1,2,3,4,5,6]
Output: [2,4,5,5,6,6]
Explanation:
- For queries[0]=1, [1,2] is the only item which has price <= 1. Hence, the answer for this query is 2.
- For queries[1]=2, the items which can be considered are [1,2] and [2,4].
  The maximum beauty among them is 4.
- For queries[2]=3 and queries[3]=4, the items which can be considered are [1,2], [3,2], [2,4], and [3,5].
  The maximum beauty among them is 5.
- For queries[4]=5 and queries[5]=6, all items can be considered.
  Hence, the answer for them is the maximum beauty of all items, i.e., 6.

Example 2:

Input: items = [[1,2],[1,2],[1,3],[1,4]], queries = [1]

Output: [4]
Explanation:
The price of every item is equal to 1, so we choose the item with the maximum beauty 4.
Note that multiple items can have the same price and/or beauty.

Example 3:

Input: items = [[10,1000]], queries = [5]
Output: [0]
Explanation:
No item has a price less than or equal to 5, so no item can be chosen.
Hence, the answer to the query is 0.

Constraints:

1 <= items.length, queries.length <= 105
items[i].length == 2
1 <= pricei, beautyi, queries[j] <= 109

2071. Maximum Number of Tasks You Can Assign

You have n tasks and m workers. Each task has a strength requirement stored in a 0-indexed integer array tasks, with the ith task requiring tasks[i] strength to complete. The strength of each worker is stored in a 0-indexed integer array workers, with the jth worker having workers[j] strength. Each worker can only be assigned to a single task and must have a strength greater than or equal to the task's strength requirement (i.e., workers[j] >= tasks[i]).
Additionally, you have pills magical pills that will increase a worker's strength by strength. You can decide which w orkers receive the magical pills, however, you may only give each worker at most one magical pill.
Given the 0-indexed integer arrays tasks and workers and the integers pills and strength, return the maximum numbe r of tasks that can be completed.

Example 1:

Input: tasks = [3,2,1], workers = [0,3,3], pills = 1, strength = 1
Output: 3
Explanation:
We can assign the magical pill and tasks as follows:
- Give the magical pill to worker 0.
- Assign worker 0 to task 2 (0 + 1 >= 1)
- Assign worker 1 to task 1 (3 >= 2)
- Assign worker 2 to task 0 (3 >= 3)

Example 2:

Input: tasks = [5,4], workers = [0,0,0], pills = 1, strength = 5
Output: 1

Explanation:
We can assign the magical pill and tasks as follows:
- Give the magical pill to worker 0.
- Assign worker 0 to task 0 (0 + 5 >= 5)

Example 3:

Input: tasks = [10,15,30], workers = [0,10,10,10,10], pills = 3, strength = 10
Output: 2
Explanation:
We can assign the magical pills and tasks as follows:
- Give the magical pill to worker 0 and worker 1.
- Assign worker 0 to task 0 (0 + 10 >= 10)
- Assign worker 1 to task 1 (10 + 10 >= 15)

Example 4:

Input: tasks = [5,9,8,5,9], workers = [1,6,4,2,6], pills = 1, strength = 5
Output: 3
Explanation:
We can assign the magical pill and tasks as follows:
- Give the magical pill to worker 2.
- Assign worker 1 to task 0 (6 >= 5)
- Assign worker 2 to task 2 (4 + 5 >= 8)
- Assign worker 4 to task 3 (6 >= 5)

Constraints:

n == tasks.length
m == workers.length
1 <= n, m <= 5 * 104
0 <= pills <= m
0 <= tasks[i], workers[j], strength <= 109

2073. Time Needed to Buy Tickets

There are n people in a line queuing to buy tickets, where the 0th person is at the front of the line and the (n - 1)th pe rson is at the back of the line.
You are given a 0-indexed integer array tickets of length n where the number of tickets that the ith person would like to buy is tickets[i].
Each person takes exactly 1 second to buy a ticket. A person can only buy 1 ticket at a time and has to go back to the end of the line (which happens instantaneously) in order to buy more tickets. If a person does not have any tickets le ft to buy, the person will leave the line.
Return the time taken for the person at position k (0-indexed) to finish buying tickets.

Example 1:

Input: tickets = [2,3,2], k = 2
Output: 6
Explanation:
- In the first pass, everyone in the line buys a ticket and the line becomes [1, 2, 1].
- In the second pass, everyone in the line buys a ticket and the line becomes [0, 1, 0].
The person at position 2 has successfully bought 2 tickets and it took 3 + 3 = 6 seconds.

Example 2:

Input: tickets = [5,1,1,1], k = 0
Output: 8
Explanation:
- In the first pass, everyone in the line buys a ticket and the line becomes [4, 0, 0, 0].
- In the next 4 passes, only the person in position 0 is buying tickets.
The person at position 0 has successfully bought 5 tickets and it took 4 + 1 + 1 + 1 + 1 = 8 seconds.

Constraints:

n == tickets.length
1 <= n <= 100
1 <= tickets[i] <= 100
0 <= k < n

2074. Reverse Nodes in Even Length Groups

You are given the head of a linked list.
The nodes in the linked list are sequentially assigned to non-empty groups whose lengths form the sequence of the n atural numbers (1, 2, 3, 4, ...). The length of a group is the number of nodes assigned to it. In other words,

The 1st node is assigned to the first group.
The 2nd and the 3rd nodes are assigned to the second group.
The 4th, 5th, and 6th nodes are assigned to the third group, and so on.

Note that the length of the last group may be less than or equal to 1 + the length of the second to last group.
Reverse the nodes in each group with an even length, and return the head of the modified linked list.

Example 1:

Input: head = [5,2,6,3,9,1,7,3,8,4]
Output: [5,6,2,3,9,1,4,8,3,7]
Explanation:
- The length of the first group is 1, which is odd, hence no reversal occurrs.

- The length of the second group is 2, which is even, hence the nodes are reversed.
- The length of the third group is 3, which is odd, hence no reversal occurrs.
- The length of the last group is 4, which is even, hence the nodes are reversed.

Example 2:

Input: head = [1,1,0,6]
Output: [1,0,1,6]
Explanation:
- The length of the first group is 1. No reversal occurrs.
- The length of the second group is 2. The nodes are reversed.
- The length of the last group is 1. No reversal occurrs.

Example 3:

Input: head = [1,1,0,6,5]
Output: [1,0,1,5,6]
Explanation:
- The length of the first group is 1. No reversal occurrs.
- The length of the second group is 2. The nodes are reversed.
- The length of the last group is 2. The nodes are reversed.

Example 4:

Input: head = [2,1]
Output: [2,1]
Explanation:
- The length of the first group is 1. No reversal occurrs.
- The length of the last group is 1. No reversal occurrs.

Example 5:

Input: head = [8]
Output: [8]
Explanation: There is only one group whose length is 1. No reversal occurrs.

Constraints:

The number of nodes in the list is in the range [1, 105].
0 <= Node.val <= 105

2075. Decode the Slanted Ciphertext

A string originalText is encoded using a slanted transposition cipher to a string encodedText with the help of a matri
x having a fixed number of rows rows.
originalText is placed first in a top-left to bottom-right manner.

The blue cells are filled first, followed by the red cells, then the yellow cells, and so on, until we reach the end of ori
ginalText. The arrow indicates the order in which the cells are filled. All empty cells are filled with ' '. The number o
f columns is chosen such that the rightmost column will not be empty after filling in originalText.
encodedText is then formed by appending all characters of the matrix in a row-wise fashion.

The characters in the blue cells are appended first to encodedText, then the red cells, and so on, and finally the yello
w cells. The arrow indicates the order in which the cells are accessed.
For example, if originalText = "cipher" and rows = 3, then we encode it in the following manner:

The blue arrows depict how originalText is placed in the matrix, and the red arrows denote the order in which encod
edText is formed. In the above example, encodedText = "ch ie pr".
Given the encoded string encodedText and number of rows rows, return the original string originalText.
Note: originalText does not have any trailing spaces ' '. The test cases are generated such that there is only one possi
ble originalText.

Example 1:

Input: encodedText = "ch ie pr", rows = 3
Output: "cipher"
Explanation: This is the same example described in the problem description.

Example 2:

Input: encodedText = "iveo eed l te olc", rows = 4
Output: "i love leetcode"
Explanation: The figure above denotes the matrix that was used to encode originalText.
The blue arrows show how we can find originalText from encodedText.

Example 3:

Input: encodedText = "coding", rows = 1
Output: "coding"
Explanation: Since there is only 1 row, both originalText and encodedText are the same.

Example 4:

Input: encodedText = " b ac", rows = 2
Output: " abc"
Explanation: originalText cannot have trailing spaces, but it may be preceded by one or more spaces.

Constraints:

0 <= encodedText.length <= 106
encodedText consists of lowercase English letters and ' ' only.
encodedText is a valid encoding of some originalText that does not have trailing spaces.

1 <= rows <= 1000
The testcases are generated such that there is only one possible originalText.

2076. Process Restricted Friend Requests

You are given an integer n indicating the number of people in a network. Each person is labeled from 0 to n - 1.
You are also given a 0-indexed 2D integer array restrictions, where restrictions[i] = [xi, yi] means that person xi and person yi cannot become friends, either directly or indirectly through other people.
Initially, no one is friends with each other. You are given a list of friend requests as a 0-indexed 2D integer array req uests, where requests[j] = [uj, vj] is a friend request between person uj and person vj.
A friend request is successful if uj and vj can be friends. Each friend request is processed in the given order (i.e., req uests[j] occurs before requests[j + 1]), and upon a successful request, uj and vj become direct friends for all future fri end requests.
Return a boolean array result, where each result[j] is true if the jth friend request is successful or false if it is not.
Note: If uj and vj are already direct friends, the request is still successful.

Example 1:

Input: n = 3, restrictions = [[0,1]], requests = [[0,2],[2,1]]
Output: [true,false]
Explanation:
Request 0: Person 0 and person 2 can be friends, so they become direct friends.
Request 1: Person 2 and person 1 cannot be friends since person 0 and person 1 would be indirect friends (1--2--0).

Example 2:

Input: n = 3, restrictions = [[0,1]], requests = [[1,2],[0,2]]
Output: [true,false]
Explanation:
Request 0: Person 1 and person 2 can be friends, so they become direct friends.
Request 1: Person 0 and person 2 cannot be friends since person 0 and person 1 would be indirect friends (0--2--1).

Example 3:

Input: n = 5, restrictions = [[0,1],[1,2],[2,3]], requests = [[0,4],[1,2],[3,1],[3,4]]
Output: [true,false,true,false]
Explanation:
Request 0: Person 0 and person 4 can be friends, so they become direct friends.
Request 1: Person 1 and person 2 cannot be friends since they are directly restricted.
Request 2: Person 3 and person 1 can be friends, so they become direct friends.
Request 3: Person 3 and person 4 cannot be friends since person 0 and person 1 would be indirect friends (0--4--3--1).

Constraints:

2 <= n <= 1000
0 <= restrictions.length <= 1000
restrictions[i].length == 2
0 <= xi, yi <= n - 1
xi != yi
1 <= requests.length <= 1000
requests[j].length == 2
0 <= uj, vj <= n - 1
uj != vj

2078. Two Furthest Houses With Different Colors

There are n houses evenly lined up on the street, and each house is beautifully painted. You are given a 0-indexed int eger array colors of length n, where colors[i] represents the color of the ith house.
Return the maximum distance between two houses with different colors.
The distance between the ith and jth houses is abs(i - j), where abs(x) is the absolute value of x.

Example 1:

Input: colors = [1,1,1,6,1,1,1]
Output: 3
Explanation: In the above image, color 1 is blue, and color 6 is red.
The furthest two houses with different colors are house 0 and house 3.
House 0 has color 1, and house 3 has color 6. The distance between them is abs(0 - 3) = 3.
Note that houses 3 and 6 can also produce the optimal answer.

Example 2:

Input: colors = [1,8,3,8,3]
Output: 4
Explanation: In the above image, color 1 is blue, color 8 is yellow, and color 3 is green.
The furthest two houses with different colors are house 0 and house 4.
House 0 has color 1, and house 4 has color 3. The distance between them is abs(0 - 4) = 4.

Example 3:

Input: colors = [0,1]
Output: 1
Explanation: The furthest two houses with different colors are house 0 and house 1.
House 0 has color 0, and house 1 has color 1. The distance between them is abs(0 - 1) = 1.

Constraints:

n == colors.length
2 <= n <= 100
0 <= colors[i] <= 100
Test data are generated such that at least two houses have different colors.

2079. Watering Plants

You want to water n plants in your garden with a watering can. The plants are arranged in a row and are labeled fro m 0 to n - 1 from left to right where the ith plant is located at x = i. There is a river at x = -1 that you can refill your watering can at.
Each plant needs a specific amount of water. You will water the plants in the following way:

Water the plants in order from left to right.
After watering the current plant, if you do not have enough water to completely water the next plant, return to the riv er to fully refill the watering can.
You cannot refill the watering can early.

You are initially at the river (i.e., x = -1). It takes one step to move one unit on the x-axis.
Given a 0-indexed integer array plants of n integers, where plants[i] is the amount of water the ith plant needs, and a n integer capacity representing the watering can capacity, return the number of steps needed to water all the plants.

Example 1:

Input: plants = [2,2,3,3], capacity = 5
Output: 14
Explanation: Start at the river with a full watering can:
- Walk to plant 0 (1 step) and water it. Watering can has 3 units of water.
- Walk to plant 1 (1 step) and water it. Watering can has 1 unit of water.
- Since you cannot completely water plant 2, walk back to the river to refill (2 steps).
- Walk to plant 2 (3 steps) and water it. Watering can has 2 units of water.
- Since you cannot completely water plant 3, walk back to the river to refill (3 steps).
- Walk to plant 3 (4 steps) and water it.
Steps needed = 1 + 1 + 2 + 3 + 3 + 4 = 14.

Example 2:

Input: plants = [1,1,1,4,2,3], capacity = 4
Output: 30
Explanation: Start at the river with a full watering can:
- Water plants 0, 1, and 2 (3 steps). Return to river (3 steps).
- Water plant 3 (4 steps). Return to river (4 steps).
- Water plant 4 (5 steps). Return to river (5 steps).
- Water plant 5 (6 steps).
Steps needed = 3 + 3 + 4 + 4 + 5 + 5 + 6 = 30.

Example 3:

Input: plants = [7,7,7,7,7,7,7], capacity = 8
Output: 49
Explanation: You have to refill before watering each plant.
Steps needed = 1 + 1 + 2 + 2 + 3 + 3 + 4 + 4 + 5 + 5 + 6 + 6 + 7 = 49.

Constraints:

n == plants.length
1 <= n <= 1000
1 <= plants[i] <= 106
max(plants[i]) <= capacity <= 109

2080. Range Frequency Queries

Design a data structure to find the frequency of a given value in a given subarray.
The frequency of a value in a subarray is the number of occurrences of that value in the subarray.
Implement the RangeFreqQuery class:

RangeFreqQuery(int[] arr) Constructs an instance of the class with the given 0-indexed integer array arr.
int query(int left, int right, int value) Returns the frequency of value in the subarray arr[left...right].

A subarray is a contiguous sequence of elements within an array. arr[left...right] denotes the subarray that contains th
e elements of nums between indices left and right (inclusive).

Example 1:

Input
["RangeFreqQuery", "query", "query"]
[[[12, 33, 4, 56, 22, 2, 34, 33, 22, 12, 34, 56]], [1, 2, 4], [0, 11, 33]]
Output
[null, 1, 2]

Explanation
RangeFreqQuery rangeFreqQuery = new RangeFreqQuery([12, 33, 4, 56, 22, 2, 34, 33, 22, 12, 34, 56]);
rangeFreqQuery.query(1, 2, 4); // return 1. The value 4 occurs 1 time in the subarray [33, 4]
rangeFreqQuery.query(0, 11, 33); // return 2. The value 33 occurs 2 times in the whole array.

Constraints:

1 <= arr.length <= 105
1 <= arr[i], value <= 104
0 <= left <= right < arr.length
At most 105 calls will be made to query

2081. Sum of k-Mirror Numbers

A k-mirror number is a positive integer without leading zeros that reads the same both forward and backward in base-10 as well as in base-k.

For example, 9 is a 2-mirror number. The representation of 9 in base-10 and base-2 are 9 and 1001 respectively, whi ch read the same both forward and backward.
On the contrary, 4 is not a 2-mirror number. The representation of 4 in base-2 is 100, which does not read the same b oth forward and backward.

Given the base k and the number n, return the sum of the n smallest k-mirror numbers.

Example 1:

Input: k = 2, n = 5
Output: 25
Explanation:
The 5 smallest 2-mirror numbers and their representations in base-2 are listed as follows:

base-10	base-2
1	1
3	11
5	101
7	111
9	1001

Their sum = 1 + 3 + 5 + 7 + 9 = 25.

Example 2:

Input: k = 3, n = 7
Output: 499
Explanation:
The 7 smallest 3-mirror numbers are and their representations in base-3 are listed as follows:

base-10	base-3
1	1
2	2
4	11
8	22
121	11111
151	12121
212	21212

Their sum = 1 + 2 + 4 + 8 + 121 + 151 + 212 = 499.

Example 3:

Input: k = 7, n = 17

Output: 20379000
Explanation: The 17 smallest 7-mirror numbers are:
1, 2, 3, 4, 5, 6, 8, 121, 171, 242, 292, 16561, 65656, 2137312, 4602064, 6597956, 6958596

Constraints:

2 <= k <= 9
1 <= n <= 30

2085. Count Common Words With One Occurrence

Given two string arrays words1 and words2, return the number of strings that appear exactly once in each of the two arrays.

Example 1:

Input: words1 = ["leetcode","is","amazing","as","is"], words2 = ["amazing","leetcode","is"]
Output: 2
Explanation:
- "leetcode" appears exactly once in each of the two arrays. We count this string.
- "amazing" appears exactly once in each of the two arrays. We count this string.
- "is" appears in each of the two arrays, but there are 2 occurrences of it in words1. We do not count this string.
- "as" appears once in words1, but does not appear in words2. We do not count this string.
Thus, there are 2 strings that appear exactly once in each of the two arrays.

Example 2:

Input: words1 = ["b","bb","bbb"], words2 = ["a","aa","aaa"]
Output: 0
Explanation: There are no strings that appear in each of the two arrays.

Example 3:

Input: words1 = ["a","ab"], words2 = ["a","a","a","ab"]
Output: 1
Explanation: The only string that appears exactly once in each of the two arrays is "ab".

Constraints:

1 <= words1.length, words2.length <= 1000
1 <= words1[i].length, words2[j].length <= 30
words1[i] and words2[j] consists only of lowercase English letters.

2086. Minimum Number of Buckets Required to Collect Rainwater from Houses

You are given a 0-indexed string street. Each character in street is either 'H' representing a house or '.' representing a n empty space.
You can place buckets on the empty spaces to collect rainwater that falls from the adjacent houses. The rainwater fro m a house at index i is collected if a bucket is placed at index i - 1 and/or index i + 1. A single bucket, if placed adjac ent to two houses, can collect the rainwater from both houses.
Return the minimum number of buckets needed so that for every house, there is at least one bucket collecting rainwa ter from it, or -1 if it is impossible.

Example 1:

Input: street = "H..H"
Output: 2
Explanation:
We can put buckets at index 1 and index 2.
"H..H" -> "HBBH" ('B' denotes where a bucket is placed).
The house at index 0 has a bucket to its right, and the house at index 3 has a bucket to its left.
Thus, for every house, there is at least one bucket collecting rainwater from it.

Example 2:

Input: street = ".H.H."
Output: 1
Explanation:
We can put a bucket at index 2.
".H.H." -> ".HBH." ('B' denotes where a bucket is placed).
The house at index 1 has a bucket to its right, and the house at index 3 has a bucket to its left.
Thus, for every house, there is at least one bucket collecting rainwater from it.

Example 3:

Input: street = ".HHH."
Output: -1
Explanation:
There is no empty space to place a bucket to collect the rainwater from the house at index 2.
Thus, it is impossible to collect the rainwater from all the houses.

Example 4:

Input: street = "H"
Output: -1
Explanation:
There is no empty space to place a bucket.
Thus, it is impossible to collect the rainwater from the house.

Example 5:

Input: street = "."
Output: 0
Explanation:
There is no house to collect water from.
Thus, 0 buckets are needed.

Constraints:

1 <= street.length <= 105
street[i] is either'H' or '.'.

2087. Minimum Cost Homecoming of a Robot in a Grid

There is an m x n grid, where (0, 0) is the top-left cell and (m - 1, n - 1) is the bottom-right cell. You are given an int eger array startPos where startPos = [startrow, startcol] indicates that initially, a robot is at the cell (startrow, startcol). You are also given an integer array homePos where homePos = [homerow, homecol] indicates that its home is at t he cell (homerow, homecol).
The robot needs to go to its home. It can move one cell in four directions: left, right, up, or down, and it can not mov e outside the boundary. Every move incurs some cost. You are further given two 0-indexed integer arrays: rowCosts of length m and colCosts of length n.

If the robot moves up or down into a cell whose row is r, then this move costs rowCosts[r].
If the robot moves left or right into a cell whose column is c, then this move costs colCosts[c].

Return the minimum total cost for this robot to return home.

Example 1:

Input: startPos = [1, 0], homePos = [2, 3], rowCosts = [5, 4, 3], colCosts = [8, 2, 6, 7]
Output: 18
Explanation: One optimal path is that:
Starting from (1, 0)
-> It goes down to (2, 0). This move costs rowCosts[2] = 3.
-> It goes right to (2, 1). This move costs colCosts[1] = 2.
-> It goes right to (2, 2). This move costs colCosts[2] = 6.
-> It goes right to (2, 3). This move costs colCosts[3] = 7.
The total cost is 3 + 2 + 6 + 7 = 18
Example 2:

Input: startPos = [0, 0], homePos = [0, 0], rowCosts = [5], colCosts = [26]
Output: 0
Explanation: The robot is already at its home. Since no moves occur, the total cost is 0.

Constraints:

m == rowCosts.length
n == colCosts.length
1 <= m, n <= 105
0 <= rowCosts[r], colCosts[c] <= 104
startPos.length == 2
homePos.length == 2
0 <= startrow, homerow < m
0 <= startcol, homecol < n

2088. Count Fertile Pyramids in a Land

A farmer has a rectangular grid of land with m rows and n columns that can be divided into unit cells. Each cell is eit her fertile (represented by a 1) or barren (represented by a 0). All cells outside the grid are considered barren.
A pyramidal plot of land can be defined as a set of cells with the following criteria:

The number of cells in the set has to be greater than 1 and all cells must be fertile.
The apex of a pyramid is the topmost cell of the pyramid. The height of a pyramid is the number of rows it covers. L et (r, c) be the apex of the pyramid, and its height be h. Then, the plot comprises of cells (i, j) where r <= i <= r + h - 1 and c - (i - r) <= j <= c + (i - r).

An inverse pyramidal plot of land can be defined as a set of cells with similar criteria:

The number of cells in the set has to be greater than 1 and all cells must be fertile.
The apex of an inverse pyramid is the bottommost cell of the inverse pyramid. The height of an inverse pyramid is th e number of rows it covers. Let (r, c) be the apex of the pyramid, and its height be h. Then, the plot comprises of cell s (i, j) where r - h + 1 <= i <= r and c - (r - i) <= j <= c + (r - i).

Some examples of valid and invalid pyramidal (and inverse pyramidal) plots are shown below. Black cells indicate f ertile cells.

Given a 0-indexed m x n binary matrix grid representing the farmland, return the total number of pyramidal and inve rse pyramidal plots that can be found in grid.

Example 1:

Input: grid = [[0,1,1,0],[1,1,1,1]]
Output: 2
Explanation:
The 2 possible pyramidal plots are shown in blue and red respectively.
There are no inverse pyramidal plots in this grid.
Hence total number of pyramidal and inverse pyramidal plots is 2 + 0 = 2.

Example 2:

Input: grid = [[1,1,1],[1,1,1]]
Output: 2
Explanation:
The pyramidal plot is shown in blue, and the inverse pyramidal plot is shown in red.
Hence the total number of plots is 1 + 1 = 2.

Example 3:

Input: grid = [[1,0,1],[0,0,0],[1,0,1]]
Output: 0
Explanation:
There are no pyramidal or inverse pyramidal plots in the grid.

Example 4:

Input: grid = [[1,1,1,1,0],[1,1,1,1,1],[1,1,1,1,1],[0,1,0,0,1]]
Output: 13
Explanation:
There are 7 pyramidal plots, 3 of which are shown in the 2nd and 3rd figures.
There are 6 inverse pyramidal plots, 2 of which are shown in the last figure.
The total number of plots is 7 + 6 = 13.

Constraints:

m == grid.length
n == grid[i].length
1 <= m, n <= 1000
1 <= m * n <= 105
grid[i][j] is either 0 or 1.

2089. Find Target Indices After Sorting Array

You are given a 0-indexed integer array nums and a target element target.
A target index is an index i such that nums[i] == target.
Return a list of the target indices of nums after sorting nums in non-decreasing order. If there are no target indices, re turn an empty list. The returned list must be sorted in increasing order.

Example 1:

Input: nums = [1,2,5,2,3], target = 2

Output: [1,2]
Explanation: After sorting, nums is [1,2,2,3,5].
The indices where nums[i] == 2 are 1 and 2.

Example 2:

Input: nums = [1,2,5,2,3], target = 3
Output: [3]
Explanation: After sorting, nums is [1,2,2,3,5].
The index where nums[i] == 3 is 3.

Example 3:

Input: nums = [1,2,5,2,3], target = 5
Output: [4]
Explanation: After sorting, nums is [1,2,2,3,5].
The index where nums[i] == 5 is 4.

Example 4:

Input: nums = [1,2,5,2,3], target = 4
Output: []
Explanation: There are no elements in nums with value 4.

Constraints:

1 <= nums.length <= 100
1 <= nums[i], target <= 100

2090. K Radius Subarray Averages

You are given a 0-indexed array nums of n integers, and an integer k.
The k-radius average for a subarray of nums centered at some index i with the radius k is the average of all elements in nums between the indices i - k and i + k (inclusive). If there are less than k elements before or after the index i, the n the k-radius average is -1.
Build and return an array avgs of length n where avgs[i] is the k-radius average for the subarray centered at index i.
The average of x elements is the sum of the x elements divided by x, using integer division. The integer division trun cates toward zero, which means losing its fractional part.

For example, the average of four elements 2, 3, 1, and 5 is (2 + 3 + 1 + 5) / 4 = 11 / 4 = 2.75, which truncates to 2.

Example 1:

Input: nums = [7,4,3,9,1,8,5,2,6], k = 3
Output: [-1,-1,-1,5,4,4,-1,-1,-1]
Explanation:
- avg[0], avg[1], and avg[2] are -1 because there are less than k elements before each index.
- The sum of the subarray centered at index 3 with radius 3 is: 7 + 4 + 3 + 9 + 1 + 8 + 5 = 37.
  Using integer division, avg[3] = 37 / 7 = 5.
- For the subarray centered at index 4, avg[4] = (4 + 3 + 9 + 1 + 8 + 5 + 2) / 7 = 4.
- For the subarray centered at index 5, avg[5] = (3 + 9 + 1 + 8 + 5 + 2 + 6) / 7 = 4.
- avg[6], avg[7], and avg[8] are -1 because there are less than k elements after each index.

Example 2:

Input: nums = [100000], k = 0
Output: [100000]
Explanation:
- The sum of the subarray centered at index 0 with radius 0 is: 100000.
  avg[0] = 100000 / 1 = 100000.

Example 3:

Input: nums = [8], k = 100000
Output: [-1]
Explanation:
- avg[0] is -1 because there are less than k elements before and after index 0.

Constraints:

n == nums.length
1 <= n <= 105
0 <= nums[i], k <= 105

2091. Removing Minimum and Maximum From Array

You are given a 0-indexed array of distinct integers nums.
There is an element in nums that has the lowest value and an element that has the highest value. We call them the mi nimum and maximum respectively. Your goal is to remove both these elements from the array.
A deletion is defined as either removing an element from the front of the array or removing an element from the bac k of the array.
Return the minimum number of deletions it would take to remove both the minimum and maximum element from th e array.

Example 1:

Input: nums = [2,10,7,5,4,1,8,6]
Output: 5

Explanation:
The minimum element in the array is nums[5], which is 1.
The maximum element in the array is nums[1], which is 10.
We can remove both the minimum and maximum by removing 2 elements from the front and 3 elements from the ba
ck.
This results in 2 + 3 = 5 deletions, which is the minimum number possible.

Example 2:

Input: nums = [0,-4,19,1,8,-2,-3,5]
Output: 3
Explanation:
The minimum element in the array is nums[1], which is -4.
The maximum element in the array is nums[2], which is 19.
We can remove both the minimum and maximum by removing 3 elements from the front.
This results in only 3 deletions, which is the minimum number possible.

Example 3:

Input: nums = [101]
Output: 1
Explanation:
There is only one element in the array, which makes it both the minimum and maximum element.
We can remove it with 1 deletion.

Constraints:

1 <= nums.length <= 105
-105 <= nums[i] <= 105
The integers in nums are distinct.

2092. Find All People With Secret

You are given an integer n indicating there are n people numbered from 0 to n - 1. You are also given a 0-indexed 2
D integer array meetings where meetings[i] = [xi, yi, timei] indicates that person xi and person yi have a meeting at t
imei. A person may attend multiple meetings at the same time. Finally, you are given an integer firstPerson.
Person 0 has a secret and initially shares the secret with a person firstPerson at time 0. This secret is then shared ever
y time a meeting takes place with a person that has the secret. More formally, for every meeting, if a person xi has th
e secret at timei, then they will share the secret with person yi, and vice versa.
The secrets are shared instantaneously. That is, a person may receive the secret and share it with people in other mee
tings within the same time frame.
Return a list of all the people that have the secret after all the meetings have taken place. You may return the answer
in any order.

Example 1:

Input: n = 6, meetings = [[1,2,5],[2,3,8],[1,5,10]], firstPerson = 1
Output: [0,1,2,3,5]
Explanation:
At time 0, person 0 shares the secret with person 1.
At time 5, person 1 shares the secret with person 2.
At time 8, person 2 shares the secret with person 3.
At time 10, person 1 shares the secret with person 5.
Thus, people 0, 1, 2, 3, and 5 know the secret after all the meetings.

Example 2:

Input: n = 4, meetings = [[3,1,3],[1,2,2],[0,3,3]], firstPerson = 3
Output: [0,1,3]
Explanation:
At time 0, person 0 shares the secret with person 3.
At time 2, neither person 1 nor person 2 know the secret.
At time 3, person 3 shares the secret with person 0 and person 1.
Thus, people 0, 1, and 3 know the secret after all the meetings.

Example 3:

Input: n = 5, meetings = [[3,4,2],[1,2,1],[2,3,1]], firstPerson = 1
Output: [0,1,2,3,4]
Explanation:
At time 0, person 0 shares the secret with person 1.
At time 1, person 1 shares the secret with person 2, and person 2 shares the secret with person 3.
Note that person 2 can share the secret at the same time as receiving it.
At time 2, person 3 shares the secret with person 4.
Thus, people 0, 1, 2, 3, and 4 know the secret after all the meetings.

Example 4:

Input: n = 6, meetings = [[0,2,1],[1,3,1],[4,5,1]], firstPerson = 1
Output: [0,1,2,3]
Explanation:
At time 0, person 0 shares the secret with person 1.
At time 1, person 0 shares the secret with person 2, and person 1 shares the secret with person 3.
Thus, people 0, 1, 2, and 3 know the secret after all the meetings.

Constraints:

2 <= n <= 105
1 <= meetings.length <= 105
meetings[i].length == 3
0 <= xi, yi <= n - 1
xi != yi
1 <= timei <= 105
1 <= firstPerson <= n - 1
