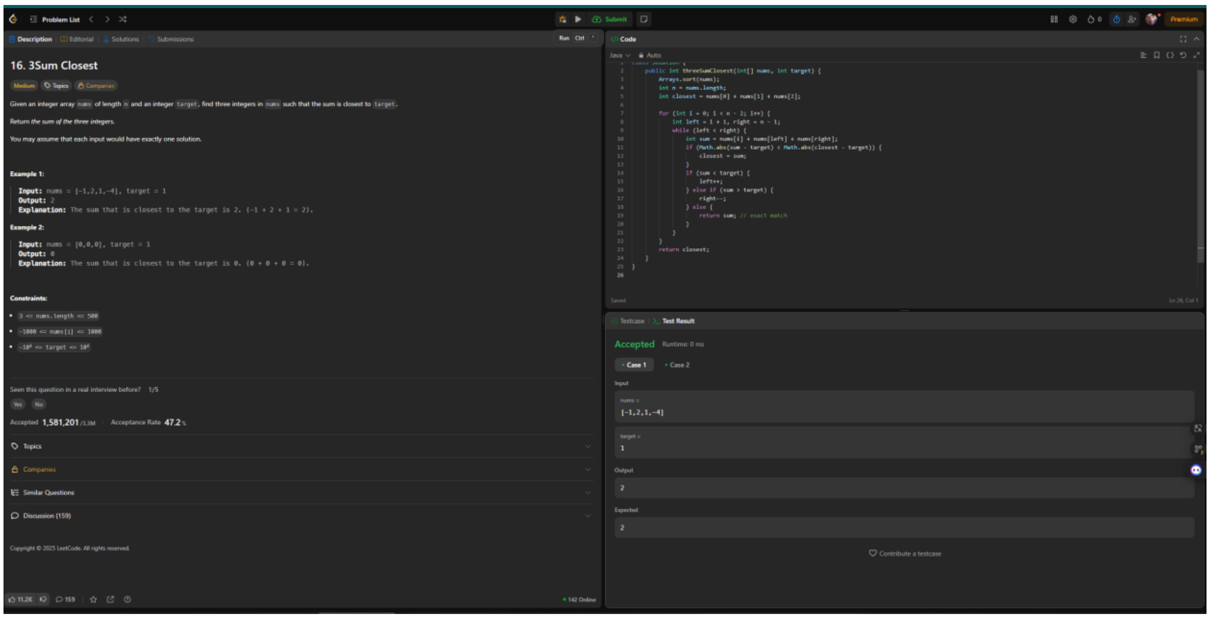
16 3Sum Closest.JAVA



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
Explanation: The sum that is closest to the target is 0. (0 + 0 + 0 = 0).
```

Constraints:

- `3 <= nums.length <= 500`
- `-1000 <= nums[i] <= 1000`
- `-10^4 <= target <= 10^4`

Seen this question in a real interview before? 1/5

Accepted 1,581,201 /3.3M Acceptance Rate 47.2 %

```java
class Solution {
    public int threeSumClosest(int[] nums, int target) {
        Arrays.sort(nums);
        int n = nums.length;
        int closest = nums[0] + nums[1] + nums[2];

        for (int i = 0; i < n - 2; i++) {
            int left = i + 1, right = n - 1;
            while (left < right) {
                int sum = nums[i] + nums[left] + nums[right];
                if (Math.abs(sum - target) < Math.abs(closest - target)) {
                    closest = sum;
                }
                if (sum < target) {
                    left++;
                } else if (sum > target) {
                    right--;
                } else {
                    return sum; // exact match
                }
            }
        }
        return closest;
    }
}
```

Accepted Runtime: 0 ms

Input

```
nums =
[-1,2,1,-4]
target =
1
```

Output

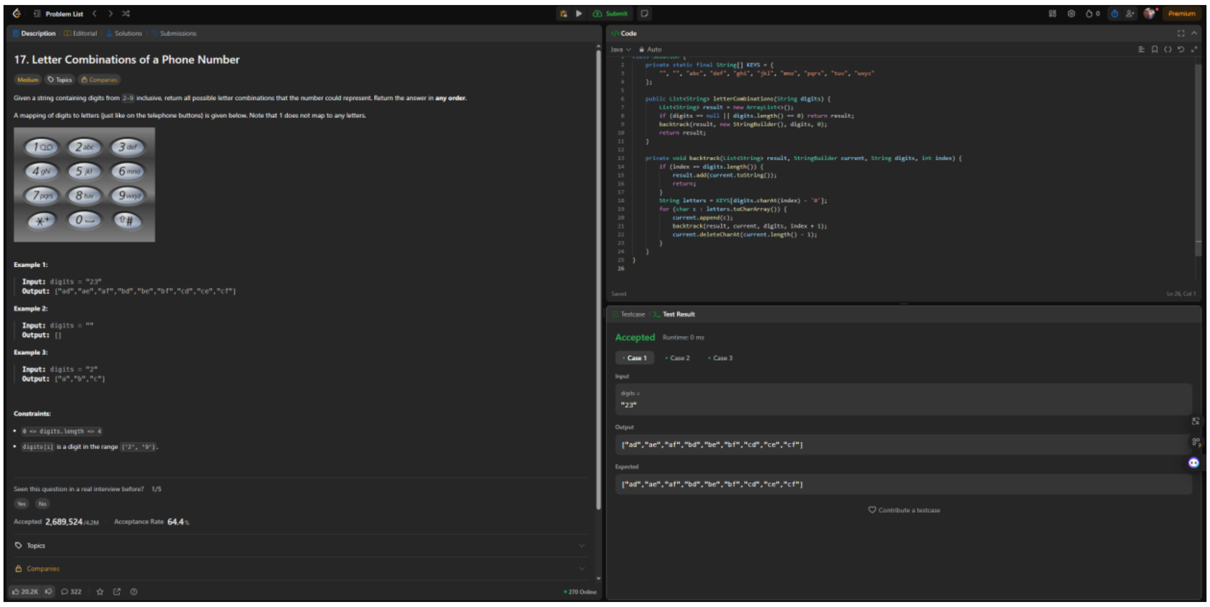
```
2
```

Expected

```
2
```

Copyright © 2025 LeetCode. All rights reserved.

17. Letter Combinations of a Phone Number.JAVA



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 represent. Return the answer in **any order**.

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

Example 1:

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

Example 2:

```
Input: digits = ""
Output: []
```

Example 3:

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

Constraints:

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

Seen this question in a real interview before? 1/5

Accepted 2,689,524 /4.2M Acceptance Rate 64.4 %

```java
class Solution {
    private static final String[] KEYS = {
        "", "", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"
    };

    public List<String> letterCombinations(String digits) {
        List<String> result = new ArrayList<>();
        if (digits == null || digits.length() == 0) return result;
        backtrack(result, new StringBuilder(), digits, 0);
        return result;
    }

    private void backtrack(List<String> result, StringBuilder current, String digits, int index) {
        if (index == digits.length()) {
            result.add(current.toString());
            return;
        }
        String letters = KEYS[digits.charAt(index) - '0'];
        for (char c : letters.toCharArray()) {
            current.append(c);
            backtrack(result, current, digits, index + 1);
            current.deleteCharAt(current.length() - 1);
        }
    }
}
```

Accepted Runtime: 0 ms

Input

```
digits =
"23"
```

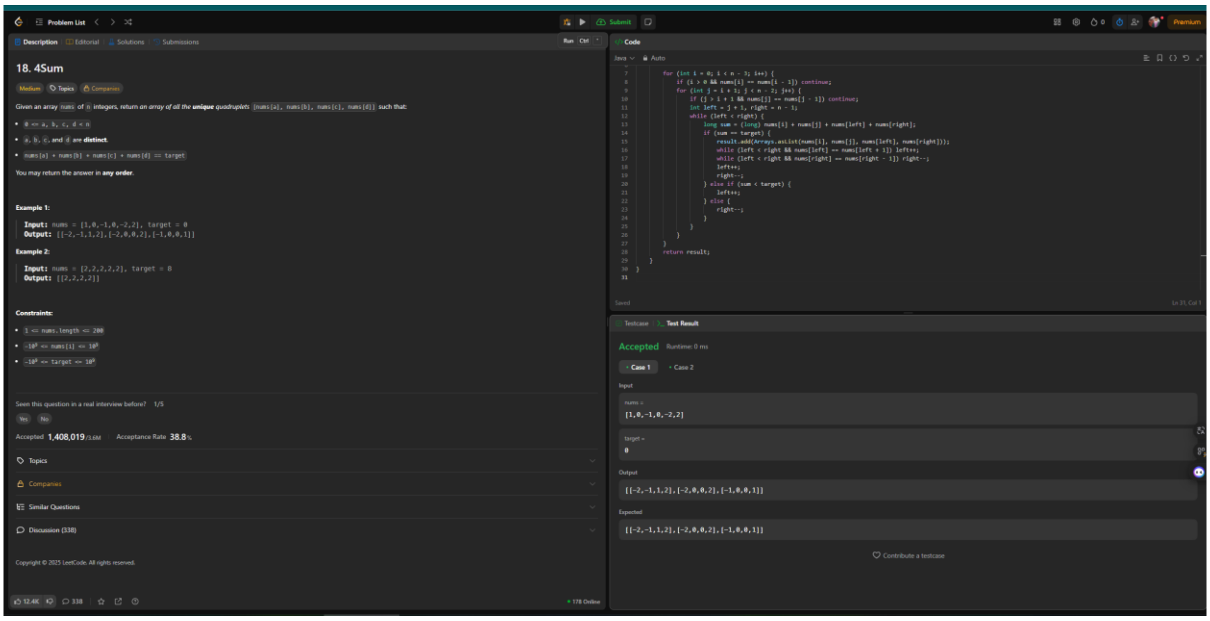
Output

```
["ad","ae","af","bd","be","bf","cd","ce","cf"]
```

Expected

```
["ad","ae","af","bd","be","bf","cd","ce","cf"]
```

18. 4Sum.JAVA


Problem List

Description | Editorial | Solutions | Submissions

18. 4Sum

Medium | Topics | Companies

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`
- -10^9 `<= nums[i] <=` 10^9
- -10^9 `<= target <=` 10^9

Seen this question in a real interview before? 1/5

Yes No

Accepted 1,408,019/3.6M | Acceptance Rate 38.8%

Topics

Companies

Similar Questions

Discussion (338)

Copyright © 2025 LeetCode. All rights reserved.

12.4K | 338 | 178 Online

Code

Java | Auto

```java
        for (int i = 0; i < n - 3; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) continue;
            for (int j = i + 1; j < n - 2; j++) {
                if (j > i + 1 && nums[j] == nums[j - 1]) continue;
                int left = j + 1, right = n - 1;
                while (left < right) {
                    long sum = (long) nums[i] + nums[j] + nums[left] + nums[right];
                    if (sum == target) {
                        result.add(Arrays.asList(nums[i], nums[j], nums[left], nums[right]));
                        while (left < right && nums[left] == nums[left + 1]) left++;
                        while (left < right && nums[right] == nums[right - 1]) right--;
                        left++;
                        right--;
                    } else if (sum < target) {
                        left++;
                    } else {
                        right--;
                    }
                }
            }
        }
        return result;
    }
}
```

Saved | Ln 31, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

nums =

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

target =

```
0
```

Output

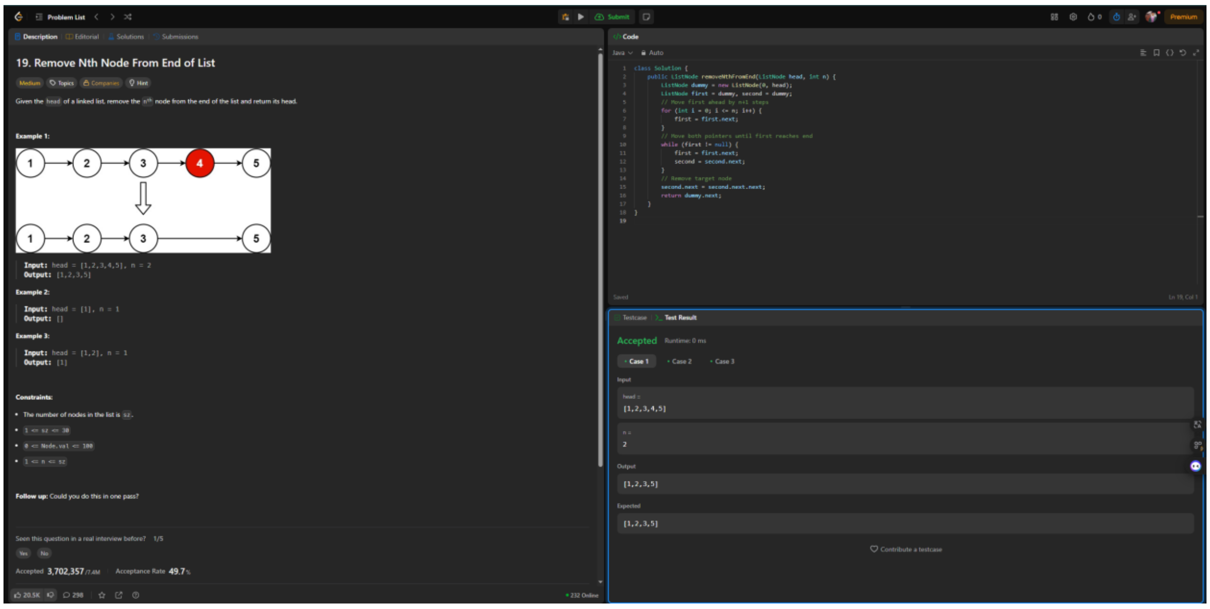
```
[[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]
```

Expected

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

Contribute a testcase

19. Remove Nth Node From End of List.JAVA


Problem List

Description | Editorial | Solutions | Submissions

19. Remove Nth Node From End of List

Medium | Topics | Companies | Hint

Given the `head` of a linked list, remove the `n`th 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?

Seen this question in a real interview before? 1/5

Yes No

Accepted 3,702,357/7.4M | Acceptance Rate 49.7%

20.5K | 298 | 232 Online

Code

Java | Auto

```java
class Solution {
    public ListNode removeNthFromEnd(ListNode head, int n) {
        ListNode dummy = new ListNode(0, head);
        ListNode first = dummy, second = dummy;
        // Move first ahead by n+1 steps
        for (int i = 0; i <= n; i++) {
            first = first.next;
        }
        // Move both pointers until first reaches end
        while (first != null) {
            first = first.next;
            second = second.next;
        }
        // Remove target node
        second.next = second.next.next;
        return dummy.next;
    }
}
```

Saved | Ln 19, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

head =

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

n =

```
2
```

Output

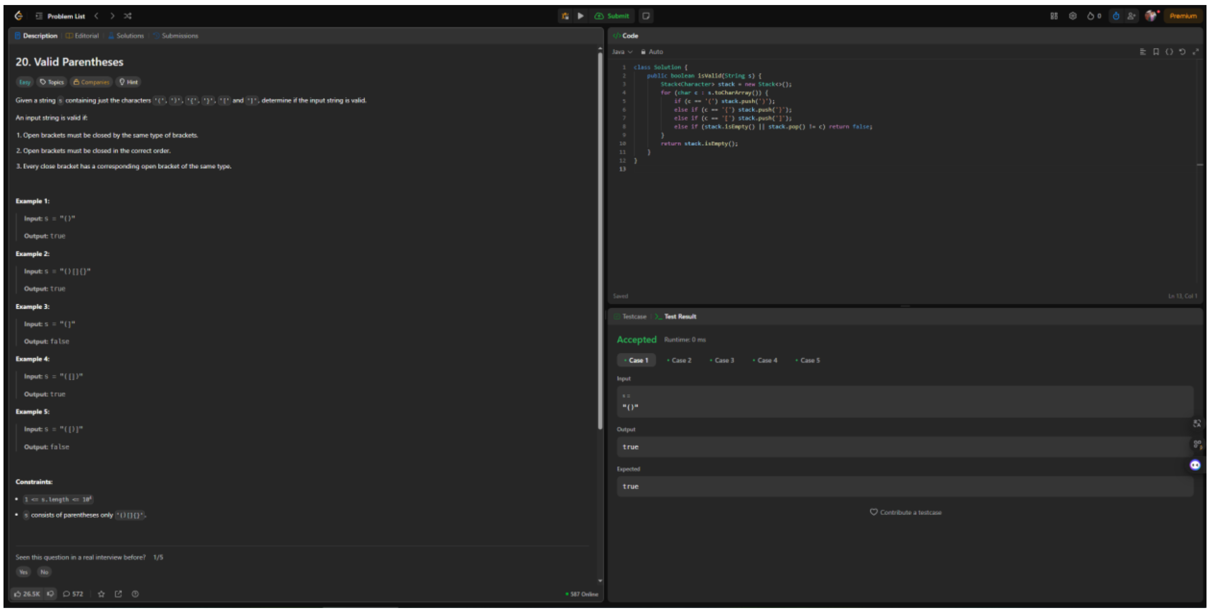
```
[1,2,3,5]
```

Expected

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

Contribute a testcase

20. Valid Parentheses.JAVA


Problem List
Description | Editorial | Solutions | Submissions
Easy Topics Companies Hint
Submit
Code
Java Auto

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:

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

Example 1:

Input: s = "()"

Output: true

Example 2:

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

Output: true

Example 3:

Input: s = "(]"

Output: false

Example 4:

Input: s = "([])"

Output: true

Example 5:

Input: s = "([)]"

Output: false

Constraints:

- `1 <= s.length <= 10^4`
- `s` consists of parentheses only `'()[]{}'`.

Seen this question in a real interview before? 1/5

Yes No

26.5K 572

```java
class Solution {
    public boolean isValid(String s) {
        Stack<Character> stack = new Stack<>();
        for (char c : s.toCharArray()) {
            if (c == '(') stack.push(')');
            else if (c == '{') stack.push('}');
            else if (c == '[') stack.push(']');
            else if (stack.isEmpty() || stack.pop() != c) return false;
        }
        return stack.isEmpty();
    }
}
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3 Case 4 Case 5

Input

s =

```
"()"
```

Output

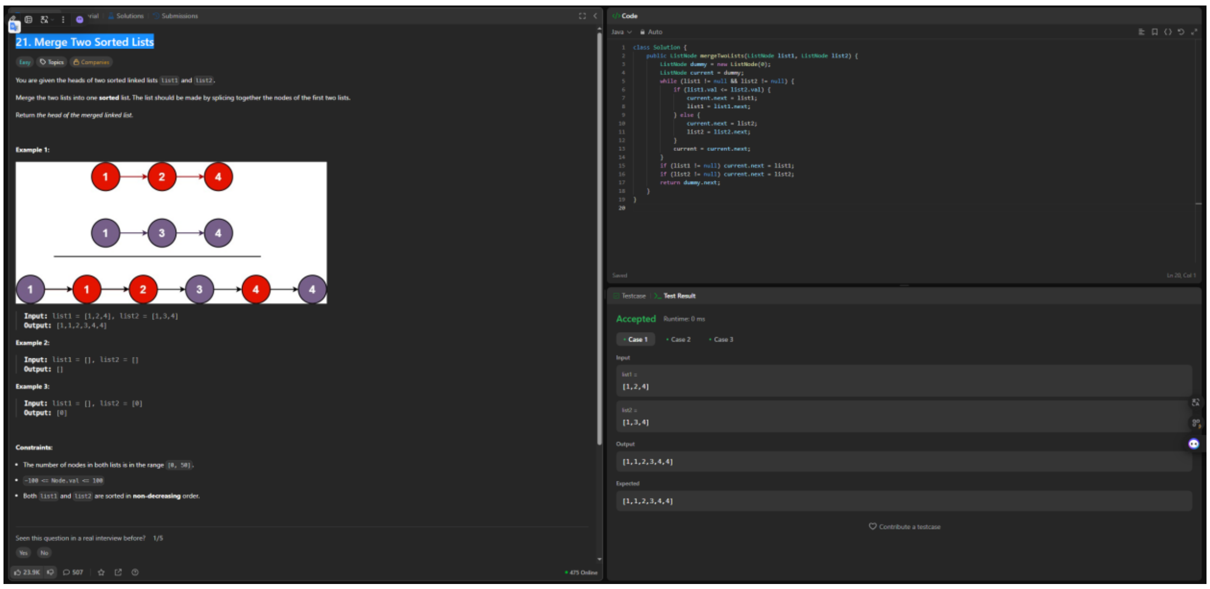
```
true
```

Expected

```
true
```

Contribute a testcase

21. Merge Two Sorted Lists.JAVA



21. Merge Two Sorted Lists

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

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

Return *the head of the merged linked list.*

Example 1:

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

Example 2:

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

Example 3:

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

Constraints:

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

Seen this question in a real interview before? 1/5

Yes No

23.9K 507

```java
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        ListNode dummy = new ListNode(0);
        ListNode current = dummy;
        while (list1 != null && list2 != null) {
            if (list1.val <= list2.val) {
                current.next = list1;
                list1 = list1.next;
            } else {
                current.next = list2;
                list2 = list2.next;
            }
            current = current.next;
        }
        if (list1 != null) current.next = list1;
        if (list2 != null) current.next = list2;
        return dummy.next;
    }
}
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

list1 =

```
[1,2,4]
```

list2 =

```
[1,3,4]
```

Output

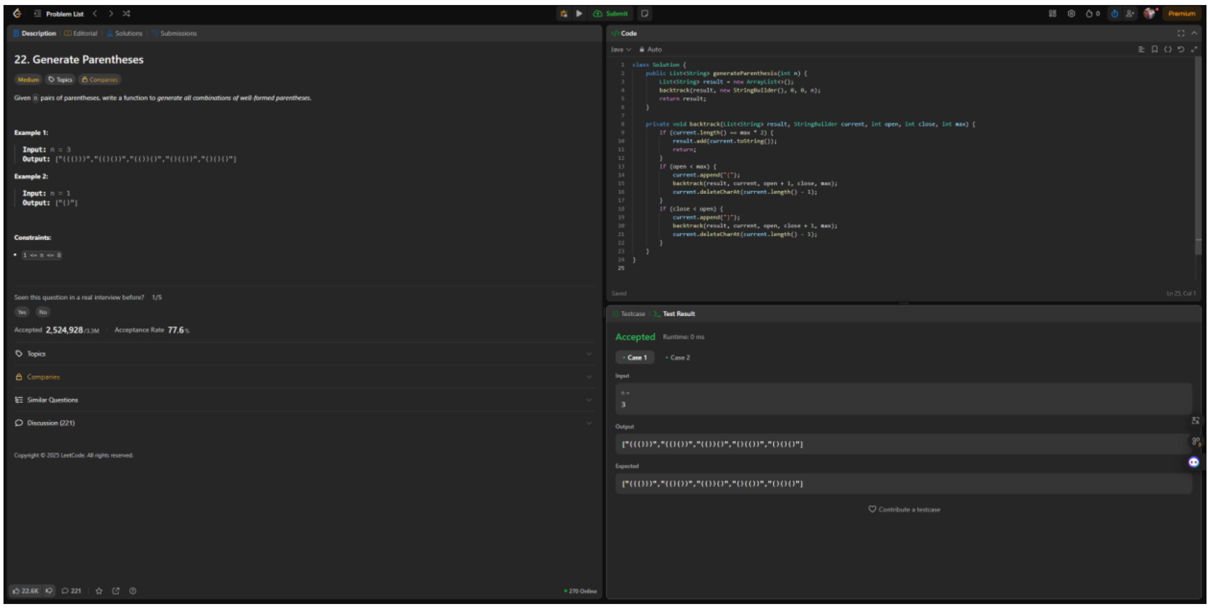
```
[1,1,2,3,4,4]
```

Expected

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

Contribute a testcase

22. Generate Parentheses.JAVA



Problem List

Description | Editorial | Solutions | Submissions

22. Generate Parentheses

Medium | Topics | Companies

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`

Seen this question in a real interview before? 1/5

Yes No

Accepted 2,524,928/3.3M | Acceptance Rate 77.6%

Topics

Companies

Similar Questions

Discussion (221)

Copyright © 2025 LeetCode. All rights reserved.

22.6K | 221 | • 270 Online

Code

Java | Auto

```java
class Solution {
    public List<String> generateParenthesis(int n) {
        List<String> result = new ArrayList<>();
        backtrack(result, new StringBuilder(), 0, 0, n);
        return result;
    }

    private void backtrack(List<String> result, StringBuilder current, int open, int close, int max) {
        if (current.length() == max * 2) {
            result.add(current.toString());
            return;
        }
        if (open < max) {
            current.append("(");
            backtrack(result, current, open + 1, close, max);
            current.deleteCharAt(current.length() - 1);
        }
        if (close < open) {
            current.append(")");
            backtrack(result, current, open, close + 1, max);
            current.deleteCharAt(current.length() - 1);
        }
    }
}
```

Saved | Ln 25, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

```
n =
3
```

Output

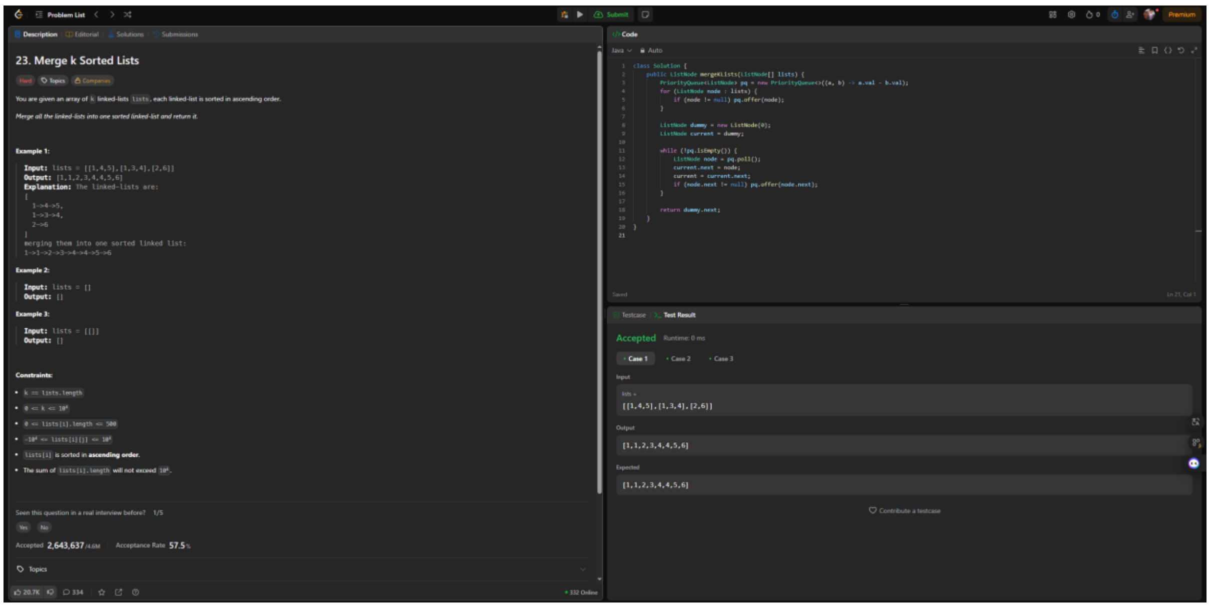
```
["((()))","(()())","(())()","()(())","()()()"]
```

Expected

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

Contribute a testcase

23. Merge k Sorted Lists.JAVA



Problem List

Description | Editorial | Solutions | Submissions

23. Merge k Sorted Lists

Hard | Topics | Companies

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 linked 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` will not exceed `10^4`.

Seen this question in a real interview before? 1/5

Yes No

Accepted 2,643,637/4.6M | Acceptance Rate 57.5%

Topics

20.7K | 334 | • 332 Online

Code

Java | Auto

```java
class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        PriorityQueue<ListNode> pq = new PriorityQueue<>((a, b) -> a.val - b.val);
        for (ListNode node : lists) {
            if (node != null) pq.offer(node);
        }

        ListNode dummy = new ListNode(0);
        ListNode current = dummy;

        while (!pq.isEmpty()) {
            ListNode node = pq.poll();
            current.next = node;
            current = current.next;
            if (node.next != null) pq.offer(node.next);
        }

        return dummy.next;
    }
}
```

Saved | Ln 21, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

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

Output

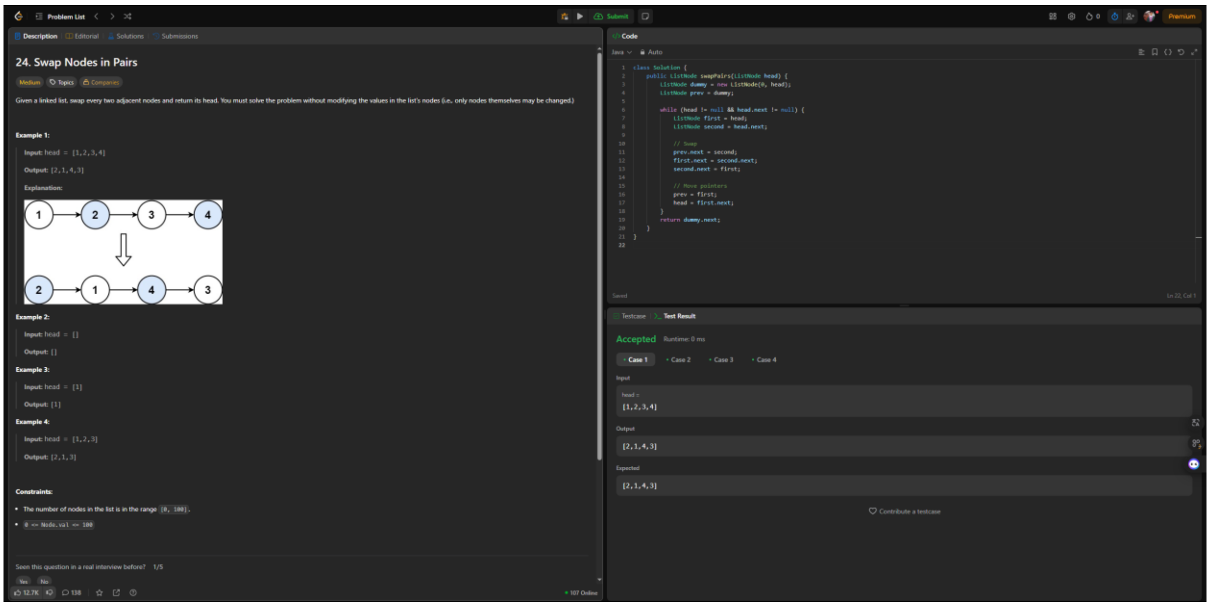
```
[1,1,2,3,4,4,5,6]
```

Expected

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

Contribute a testcase

24. Swap Nodes in Pairs.JAVA



24. Swap Nodes in Pairs

Medium · Topics · Companies

Given a linked list, swap every two adjacent nodes and return its head. You must solve the problem without modifying 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]

Explanation:

Example 2:

Input: head = []

Output: []

Example 3:

Input: head = [1]

Output: [1]

Example 4:

Input: head = [1,2,3]

Output: [2,1,3]

Constraints:

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

Seen this question in a real interview before? 1/5

Yes No

12.7K · 138 · 107 Online

```java
class Solution {
    public ListNode swapPairs(ListNode head) {
        ListNode dummy = new ListNode(0, head);
        ListNode prev = dummy;

        while (head != null && head.next != null) {
            ListNode first = head;
            ListNode second = head.next;

            // Swap
            prev.next = second;
            first.next = second.next;
            second.next = first;

            // Move pointers
            prev = first;
            head = first.next;
        }
        return dummy.next;
    }
}

```

Saved · Ln 22, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3 · Case 4

Input

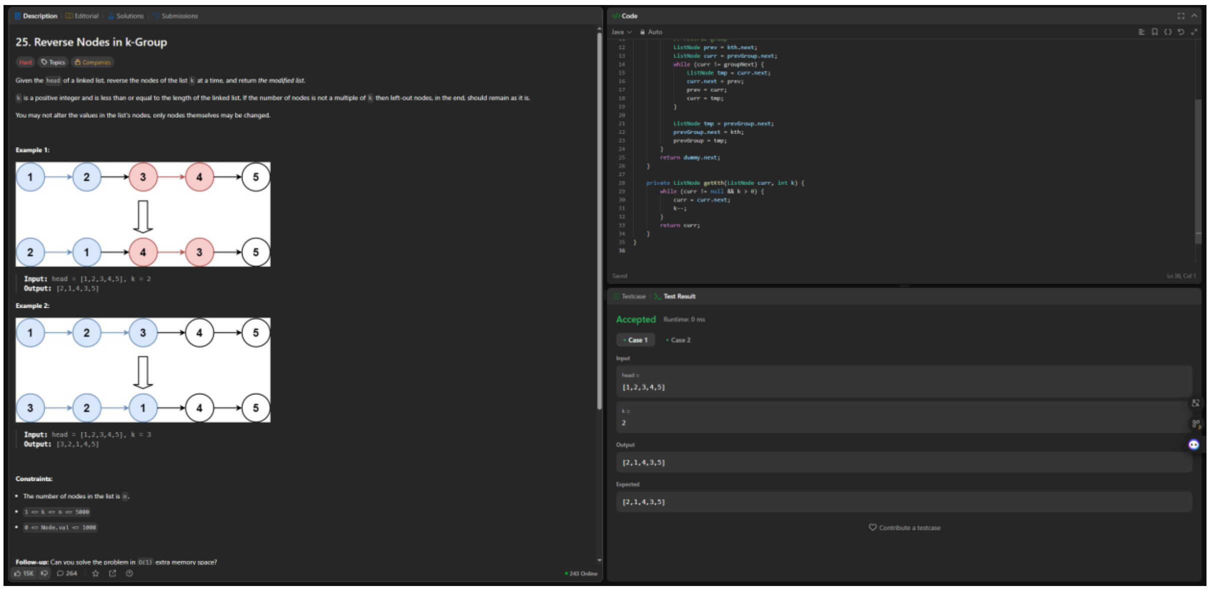
head =
```
[1,2,3,4]
```

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

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

Contribute a testcase

25. Reverse Nodes in k-Group.JAVA



25. Reverse Nodes in k-Group

Hard · Topics · Companies

Given the `head` of a linked list, reverse the nodes of the list `k` at a time, and return *the 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]
```

Constraints:

- The number of nodes in the list is `n`.
- `1 <= k <= n <= 5000`
- `0 <= Node.val <= 1000`

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

15K · 264 · 243 Online

```java
            ListNode prev = kth.next;
            ListNode curr = prevGroup.next;
            while (curr != groupNext) {
                ListNode tmp = curr.next;
                curr.next = prev;
                prev = curr;
                curr = tmp;
            }

            ListNode tmp = prevGroup.next;
            prevGroup.next = kth;
            prevGroup = tmp;
        }
        return dummy.next;
    }

    private ListNode getKth(ListNode curr, int k) {
        while (curr != null && k > 0) {
            curr = curr.next;
            k--;
        }
        return curr;
    }
}

```

Saved · Ln 36, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

head =
```
[1,2,3,4,5]
```

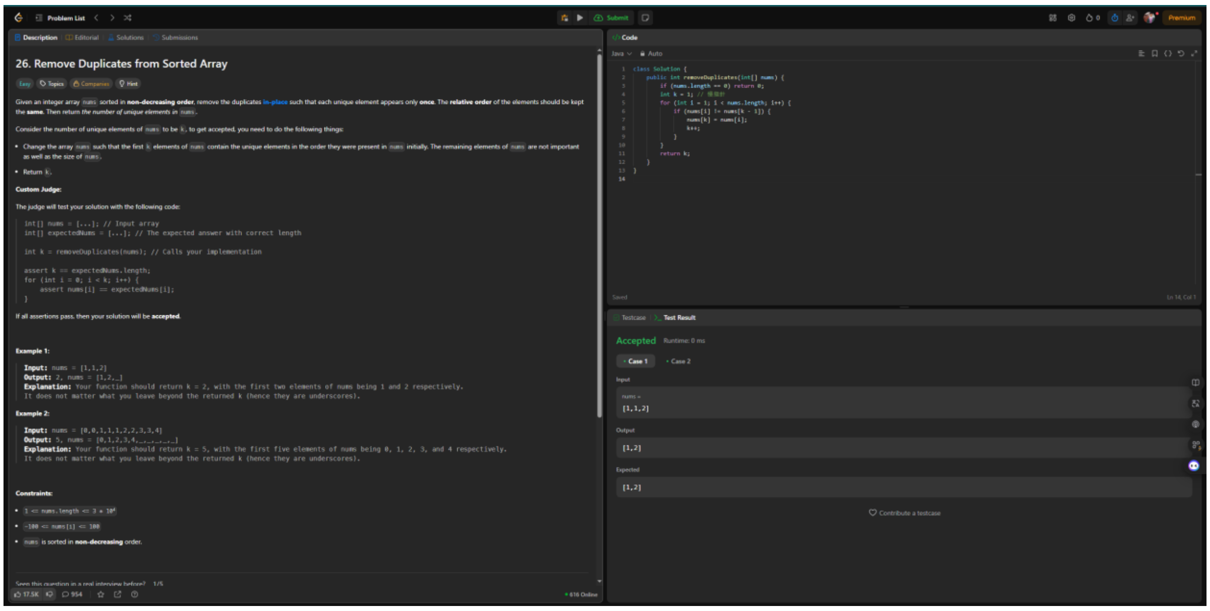
k =
```
2
```

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

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

Contribute a testcase

26. Remove Duplicates from Sorted Array.JAVA



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 element appears only **once**. The **relative order** of the elements should be kept the **same**. Then return *the number of unique elements in* `nums`.

Consider the number of unique elements of `nums` to be `k`, to get accepted, you need to do the following things:

- Change the array `nums` such that the first `k` elements of `nums` contain the unique elements in the order they were present in `nums` initially. The remaining elements of `nums` are not important as well as the size of `nums`.
- Return `k`.

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:

- `1 <= nums.length <= 3 * 10^4`
- `-100 <= nums[i] <= 100`
- `nums` is sorted in **non-decreasing** order.

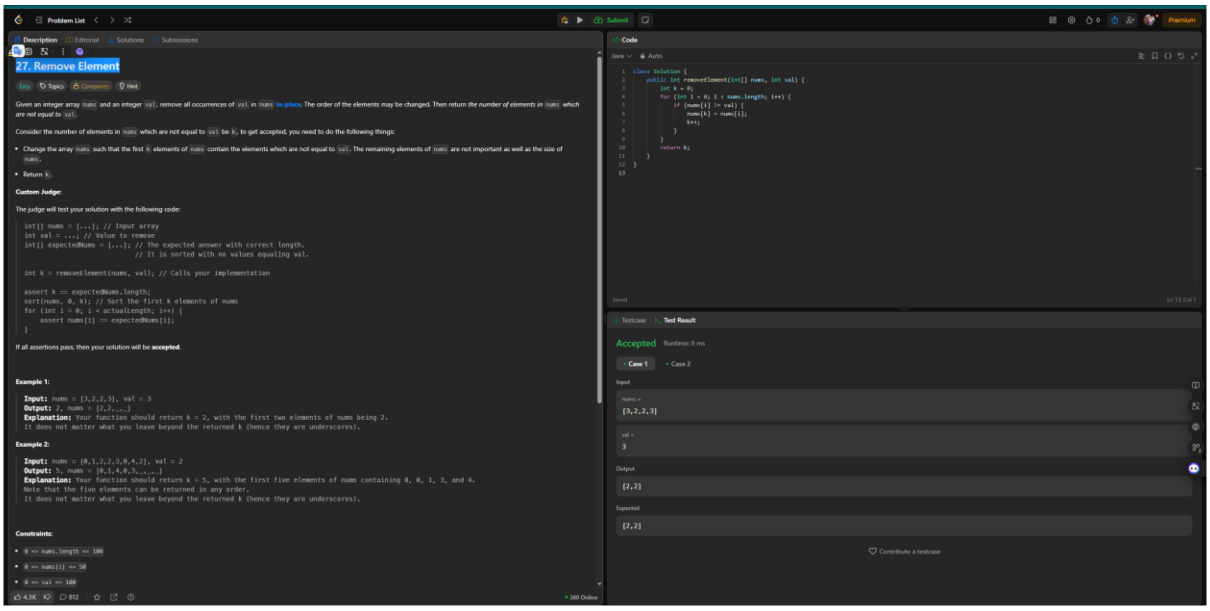
Seen this question in a real interview before? 1/5

```
class Solution {
    public int removeDuplicates(int[] nums) {
        if (nums.length == 0) return 0;
        int k = 1; // 慢指針
        for (int i = 1; i < nums.length; i++) {
            if (nums[i] != nums[k - 1]) {
                nums[k] = nums[i];
                k++;
            }
        }
        return k;
    }
}
```

Accepted Runtime: 0 ms

```
Input
nums =
[1,1,2]
Output
[1,2]
Expected
[1,2]
```

27. Remove Element.JAVA



27. Remove Element

Given an integer array `nums` and an integer `val`, remove all occurrences of `val` in `nums` in-place. The order of the elements may be changed. Then return *the number of elements in* `nums` *which are not equal to* `val`.

Consider the number of elements in `nums` which are not equal to `val` be `k`, to get accepted, you need to do the following things:

- Change the array `nums` such that the first `k` elements of `nums` contain the elements which are not equal to `val`. The remaining elements of `nums` are not important as well as the size of `nums`.
- Return `k`.

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:

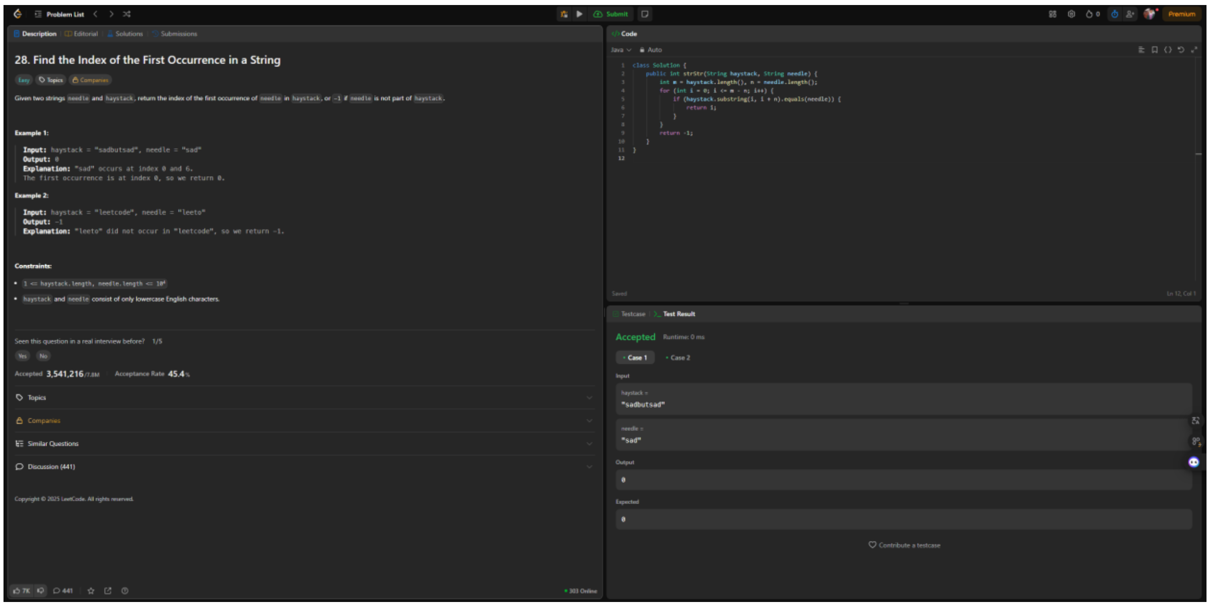
- `0 <= nums.length <= 100`
- `0 <= nums[i] <= 50`
- `0 <= val <= 100`

```
class Solution {
    public int removeElement(int[] nums, int val) {
        int k = 0;
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] != val) {
                nums[k] = nums[i];
                k++;
            }
        }
        return k;
    }
}
```

Accepted Runtime: 0 ms

```
Input
nums =
[3,2,2,3]
val =
3
Output
[2,2]
Expected
[2,2]
```

28. Find the Index of the First Occurrence in a String.JAVA



28. Find the Index of the First Occurrence in a String

Given two strings `needle` and `haystack`, return the index of the first occurrence of `needle` in `haystack`, or `-1` if `needle` is not part of `haystack`.

Example 1:

```
Input: haystack = "sadbutsad", needle = "sad"
Output: 0
Explanation: "sad" occurs at index 0 and 6.
The first occurrence is at index 0, so we return 0.
```

Example 2:

```
Input: haystack = "leetcode", needle = "leeto"
Output: -1
Explanation: "leeto" did not occur in "leetcode", so we return -1.
```

Constraints:

- `1 <= haystack.length, needle.length <= 10^4`
- `haystack` and `needle` consist of only lowercase English characters.

Seen this question in a real interview before? 1/5

Accepted 3,541,216/7.8M Acceptance Rate 45.4%

Copyright © 2025 LeetCode. All rights reserved.

```java
class Solution {
    public int strStr(String haystack, String needle) {
        int m = haystack.length(), n = needle.length();
        for (int i = 0; i <= m - n; i++) {
            if (haystack.substring(i, i + n).equals(needle)) {
                return i;
            }
        }
        return -1;
    }
}
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

haystack =
"sadbutsad"

needle =
"sad"

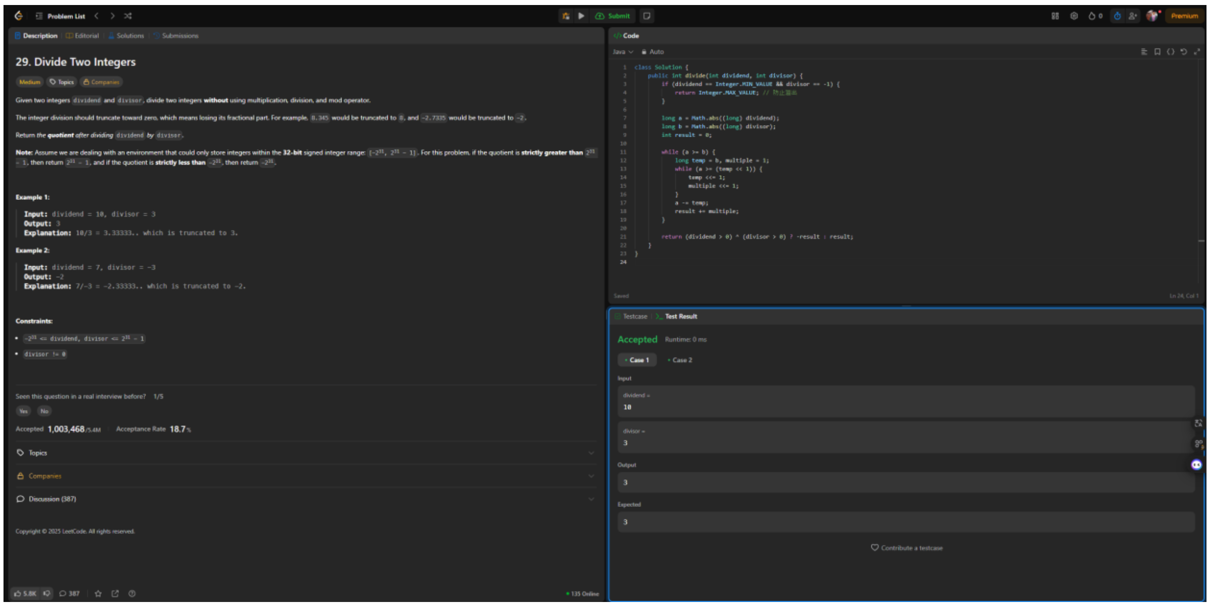
Output

0

Expected

0

29. Divide Two Integers.JAVA



29. Divide Two Integers

Given two integers `dividend` and `divisor`, divide two integers **without** using multiplication, division, and mod operator.

The integer division should truncate toward zero, which means losing its fractional part. For example, `8.345` would be 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 range: $[-2^{31}, 2^{31} - 1]$. For this problem, if the quotient is **strictly greater than** $2^{31} - 1$, then return $2^{31} - 1$, and if the quotient is **strictly less than** -2^{31}, then return -2^{31}.

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.
```

Constraints:

- -2^{31} `<= dividend, divisor <=` $2^{31} - 1$
- `divisor != 0`

Seen this question in a real interview before? 1/5

Accepted 1,003,468/5.4M Acceptance Rate 18.7%

Copyright © 2025 LeetCode. All rights reserved.

```java
class Solution {
    public int divide(int dividend, int divisor) {
        if (dividend == Integer.MIN_VALUE && divisor == -1) {
            return Integer.MAX_VALUE; // 防止溢出
        }

        long a = Math.abs((long) dividend);
        long b = Math.abs((long) divisor);
        int result = 0;

        while (a >= b) {
            long temp = b, multiple = 1;
            while (a >= (temp << 1)) {
                temp <<= 1;
                multiple <<= 1;
            }
            a -= temp;
            result += multiple;
        }

        return (dividend > 0) ^ (divisor > 0) ? -result : result;
    }
}
```

Accepted Runtime: 0 ms

Case 1 Case 2

Input

dividend =
10

divisor =
3

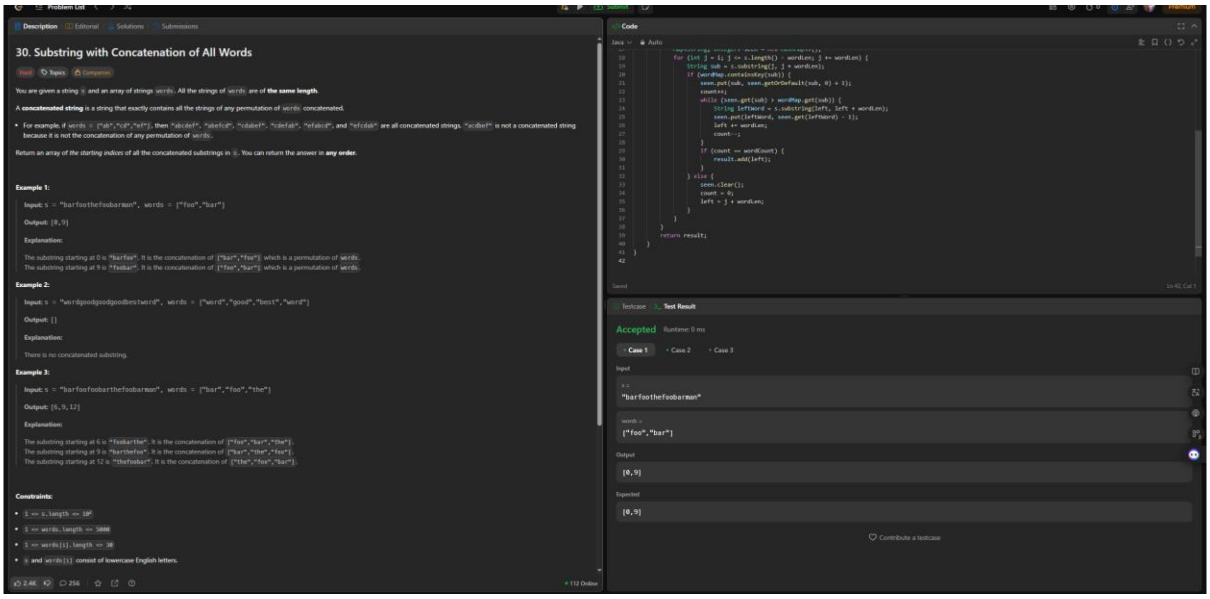
Output

3

Expected

3

30. Substring with Concatenation of All Words.JAVA



30. Substring with Concatenation of All Words

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

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

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

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

Example 1:

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

Output: [0,9]

Explanation:

The substring starting at 0 is `"barfoo"`. It is the concatenation of `["bar","foo"]` which is a permutation of `words`.
The substring starting at 9 is `"foobar"`. It is the concatenation of `["foo","bar"]` which is a permutation of `words`.

Example 2:

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

Output: []

Explanation:

There is no concatenated substring.

Example 3:

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

Output: [6,9,12]

Explanation:

The substring starting at 6 is `"foobarthe"`. It is the concatenation of `["foo","bar","the"]`.
The substring starting at 9 is `"barthefoo"`. It is the concatenation of `["bar","the","foo"]`.
The substring starting at 12 is `"thefoobar"`. It is the concatenation of `["the","foo","bar"]`.

Constraints:

- `1 <= s.length <= 10^4`
- `1 <= words.length <= 5000`
- `1 <= words[i].length <= 30`
- `s` and `words[i]` consist of lowercase English letters.

```java
            for (int j = i; j <= s.length() - wordLen; j += wordLen) {
                String sub = s.substring(j, j + wordLen);
                if (wordMap.containsKey(sub)) {
                    seen.put(sub, seen.getOrDefault(sub, 0) + 1);
                    count++;
                    while (seen.get(sub) > wordMap.get(sub)) {
                        String leftWord = s.substring(left, left + wordLen);
                        seen.put(leftWord, seen.get(leftWord) - 1);
                        left += wordLen;
                        count--;
                    }
                    if (count == wordCount) {
                        result.add(left);
                    }
                } else {
                    seen.clear();
                    count = 0;
                    left = j + wordLen;
                }
            }
        }
        return result;
    }
}
```

Accepted Runtime: 0 ms

Input

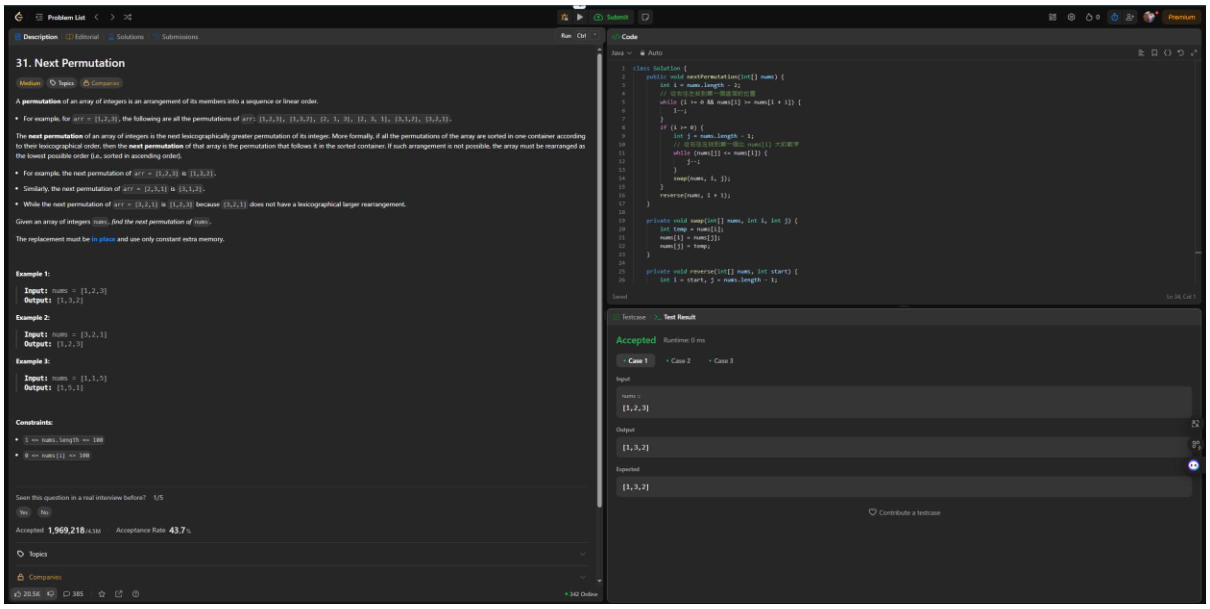
s =
```
"barfoothefoobarman"
```

words =
```
["foo","bar"]
```

Output
```
[0,9]
```

Expected
```
[0,9]
```

31. Next Permutation.JAVA



31. Next Permutation

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

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

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

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

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

The replacement must be in place and use only constant extra memory.

Example 1:

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

Example 2:

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

Example 3:

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

Constraints:

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

Seen this question in a real interview before? 1/5

Accepted 1,969,218/4.5M Acceptance Rate 43.7%

Topics

Companies

```java
class Solution {
    public void nextPermutation(int[] nums) {
        int i = nums.length - 2;
        // 從右往左找到第一個遞減的位置
        while (i >= 0 && nums[i] >= nums[i + 1]) {
            i--;
        }
        if (i >= 0) {
            int j = nums.length - 1;
            // 從右往左找到第一個比 nums[i] 大的數字
            while (nums[j] <= nums[i]) {
                j--;
            }
            swap(nums, i, j);
        }
        reverse(nums, i + 1);
    }

    private void swap(int[] nums, int i, int j) {
        int temp = nums[i];
        nums[i] = nums[j];
        nums[j] = temp;
    }

    private void reverse(int[] nums, int start) {
        int i = start, j = nums.length - 1;
```

Accepted Runtime: 0 ms

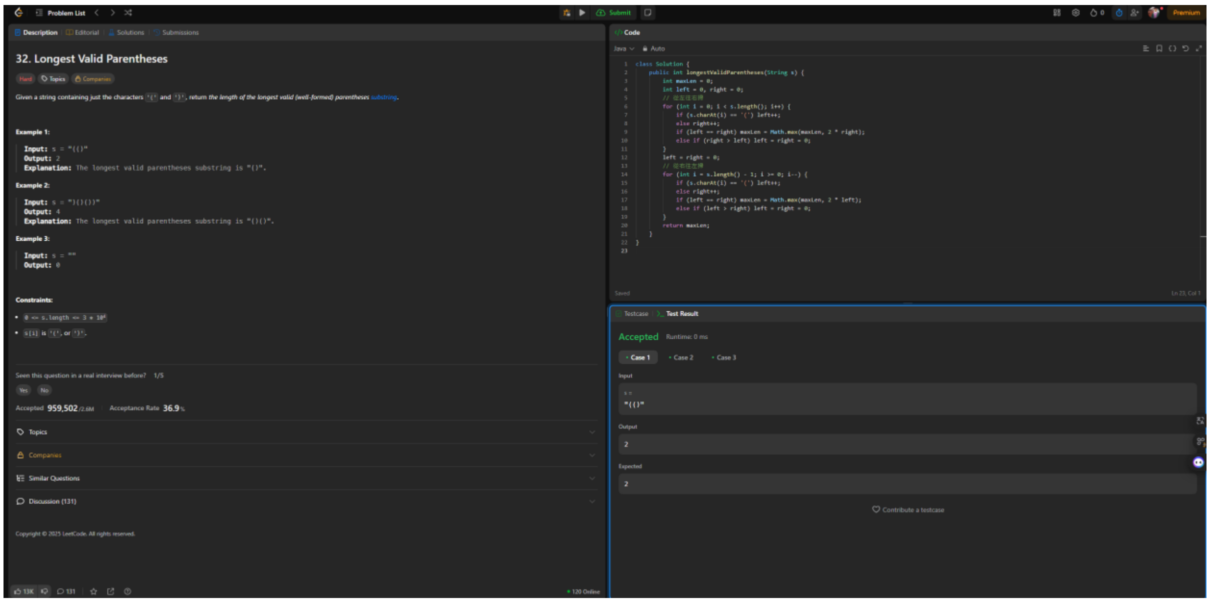
Input

nums =
```
[1,2,3]
```

Output
```
[1,3,2]
```

Expected
```
[1,3,2]
```

32. Longest Valid Parentheses



32. Longest Valid Parentheses

Given a string containing just the characters `'('` and `')'`, *return the length of the longest valid (well-formed) parentheses substring*.

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 * 10`^{4}
- `s[i]` is `'('`, or `')'`.

Seen this question in a real interview before? 1/5

Yes No

Accepted 959,502 /2.6M | Acceptance Rate 36.9 %

Topics

Companies

Similar Questions

Discussion (131)

Copyright © 2025 LeetCode. All rights reserved.

13K 131 • 120 Online

```java
class Solution {
    public int longestValidParentheses(String s) {
        int maxLen = 0;
        int left = 0, right = 0;
        // 從左往右掃
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == '(') left++;
            else right++;
            if (left == right) maxLen = Math.max(maxLen, 2 * right);
            else if (right > left) left = right = 0;
        }
        left = right = 0;
        // 從右往左掃
        for (int i = s.length() - 1; i >= 0; i--) {
            if (s.charAt(i) == '(') left++;
            else right++;
            if (left == right) maxLen = Math.max(maxLen, 2 * left);
            else if (left > right) left = right = 0;
        }
        return maxLen;
    }
}
```

Saved Ln 23, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

s =
"(()"

Output

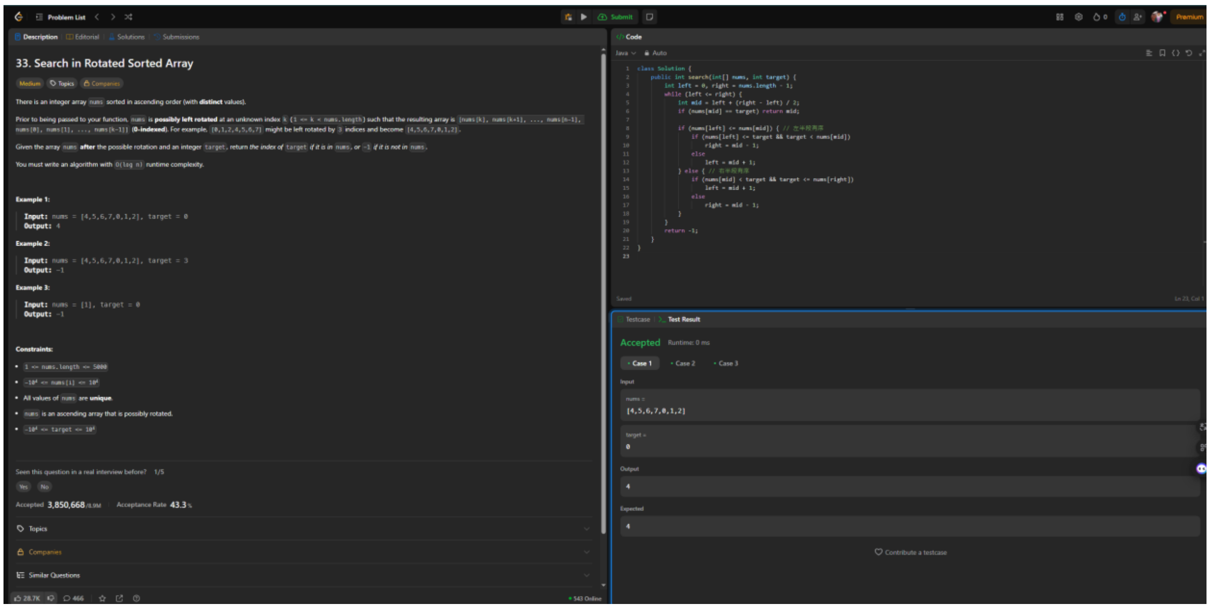
2

Expected

2

Contribute a testcase

33. Search in Rotated Sorted Array.JAVA



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 left rotated** at an unknown 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**). For example, `[0,1,2,4,5,6,7]` might be left rotated by `3` indices 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`
- `-10`^{4} `<= nums[i] <= 10`^{4}
- All values of `nums` are **unique**.
- `nums` is an ascending array that is possibly rotated.
- `-10`^{4} `<= target <= 10`^{4}

Seen this question in a real interview before? 1/5

Yes No

Accepted 3,850,668 /8.9M | Acceptance Rate 43.3 %

Topics

Companies

Similar Questions

28.7K 466 • 543 Online

```java
class Solution {
    public int search(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) return mid;

            if (nums[left] <= nums[mid]) { // 左半段有序
                if (nums[left] <= target && target < nums[mid])
                    right = mid - 1;
                else
                    left = mid + 1;
            } else { // 右半段有序
                if (nums[mid] < target && target <= nums[right])
                    left = mid + 1;
                else
                    right = mid - 1;
            }
        }
        return -1;
    }
}
```

Saved Ln 23, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

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

target =
0

Output

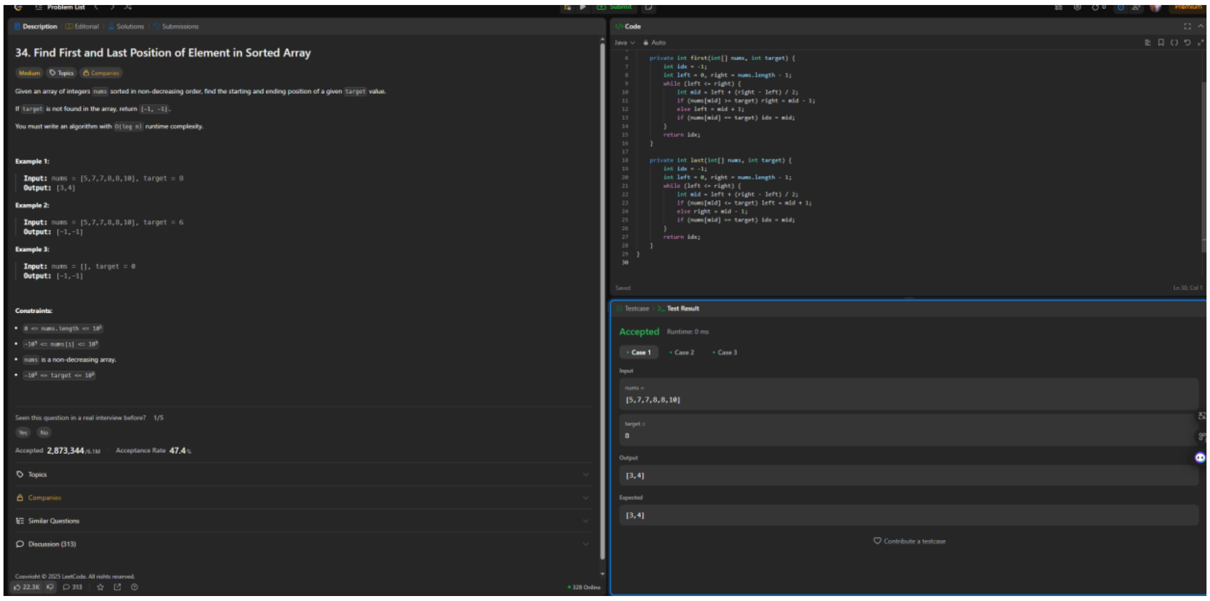
4

Expected

4

Contribute a testcase

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


Problem List

Description | Editorial | Solutions | Submissions

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

Medium · Topics · Companies

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given `target` 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`

Seen this question in a real interview before? 1/5

Yes No

Accepted 2,873,344/6.1M | Acceptance Rate 47.4%

Topics

Companies

Similar Questions

Discussion (313)

Copyright © 2025 LeetCode. All rights reserved.

22.3K | 313 · 328 Online

Code — Java · Auto

```java
    private int first(int[] nums, int target) {
        int idx = -1;
        int left = 0, right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] >= target) right = mid - 1;
            else left = mid + 1;
            if (nums[mid] == target) idx = mid;
        }
        return idx;
    }

    private int last(int[] nums, int target) {
        int idx = -1;
        int left = 0, right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] <= target) left = mid + 1;
            else right = mid - 1;
            if (nums[mid] == target) idx = mid;
        }
        return idx;
    }
}
```

Saved · Ln 30, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3

Input

nums =
```
[5,7,7,8,8,10]
```

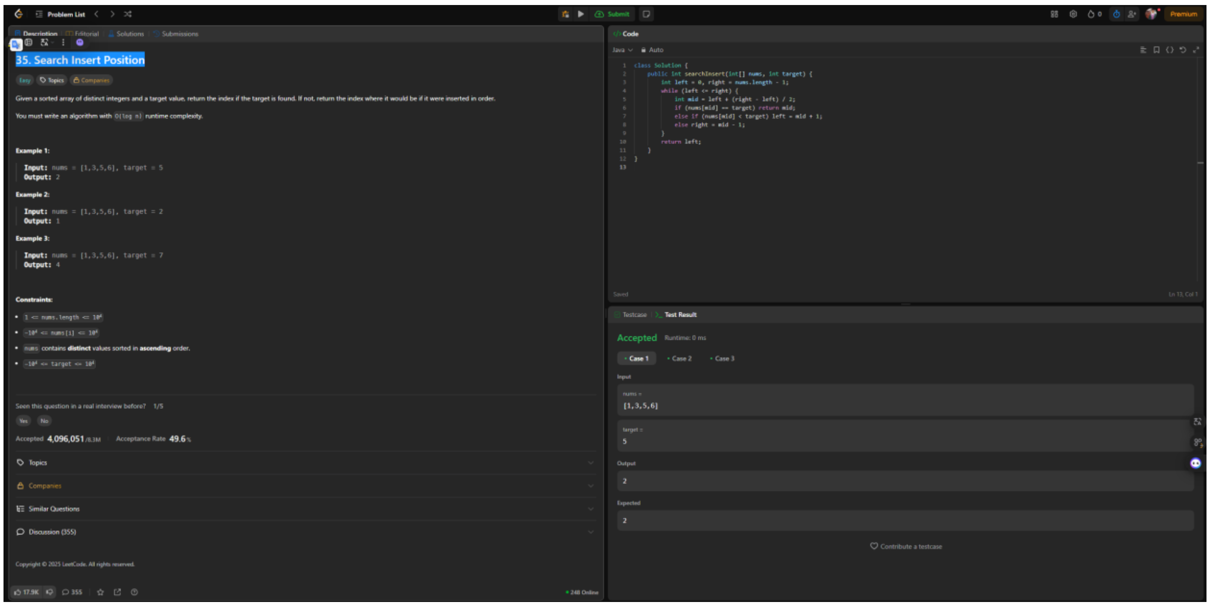
target =
```
8
```

Output
```
[3,4]
```

Expected
```
[3,4]
```

Contribute a testcase

35. Search Insert Position.JAVA


Problem List

Description | Editorial | Solutions | Submissions

35. Search Insert Position

Easy · Topics · Companies

Given a sorted array of distinct integers and a target value, return the index if the target is found. If not, return the index 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
```

Constraints:

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

Seen this question in a real interview before? 1/5

Yes No

Accepted 4,096,051/8.3M | Acceptance Rate 49.6%

Topics

Companies

Similar Questions

Discussion (355)

Copyright © 2025 LeetCode. All rights reserved.

17.9K | 355 · 248 Online

Code — Java · Auto

```java
class Solution {
    public int searchInsert(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) return mid;
            else if (nums[mid] < target) left = mid + 1;
            else right = mid - 1;
        }
        return left;
    }
}
```

Saved · Ln 13, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3

Input

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

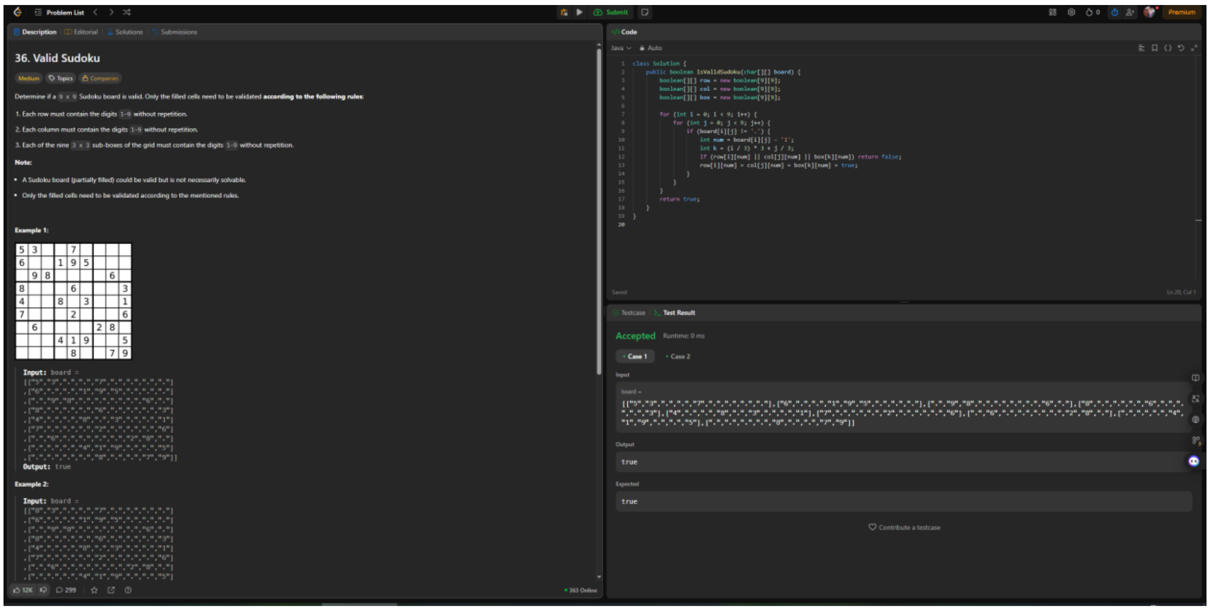
target =
```
5
```

Output
```
2
```

Expected
```
2
```

Contribute a testcase

36. Valid Sudoku.JAVA



Problem List

Description | Editorial | Solutions | Submissions

36. Valid Sudoku

Medium | Topics | Companies

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

1. Each row must contain the digits `1-9` without repetition.
2. Each column must contain the digits `1-9` without repetition.
3. 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 =

Output: true

Example 2:

Input: board =

Code — Java — Auto

```java
class Solution {
    public boolean isValidSudoku(char[][] board) {
        boolean[][] row = new boolean[9][9];
        boolean[][] col = new boolean[9][9];
        boolean[][] box = new boolean[9][9];

        for (int i = 0; i < 9; i++) {
            for (int j = 0; j < 9; j++) {
                if (board[i][j] != '.') {
                    int num = board[i][j] - '1';
                    int k = (i / 3) * 3 + j / 3;
                    if (row[i][num] || col[j][num] || box[k][num]) return false;
                    row[i][num] = col[j][num] = box[k][num] = true;
                }
            }
        }
        return true;
    }
}
```

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2

Input

board =

Output

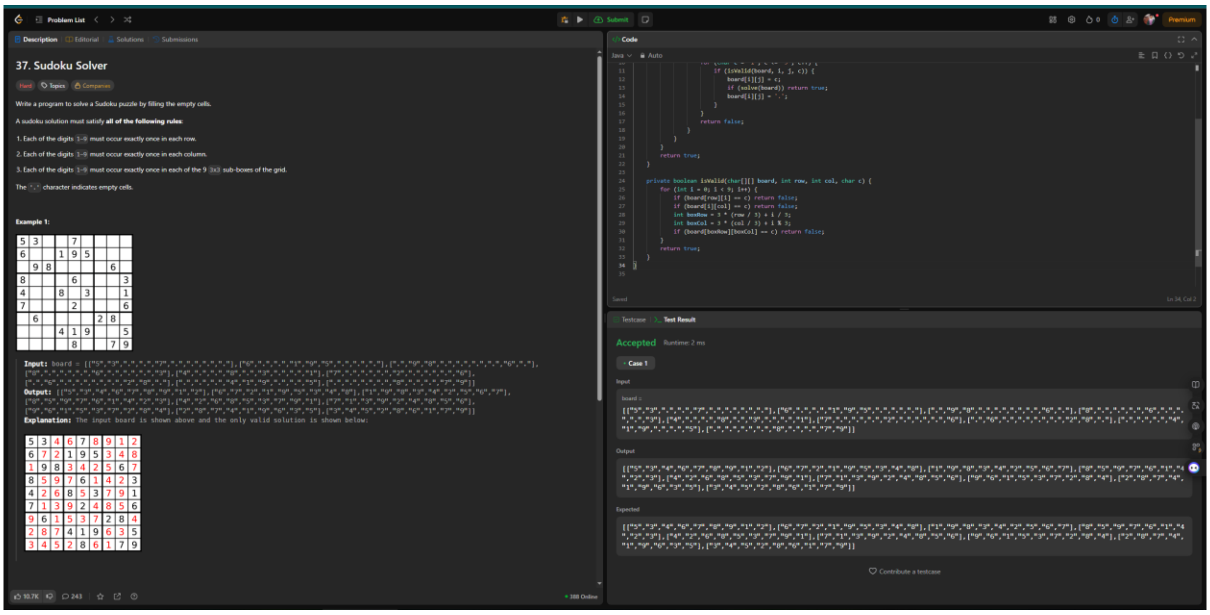
```
true
```

Expected

```
true
```

Contribute a testcase

37. Sudoku Solver.JAVA



Problem List

Description | Editorial | Solutions | Submissions

37. Sudoku Solver

Hard | Topics | Companies

Write a program to solve a Sudoku puzzle by filling the empty cells.

A sudoku solution must satisfy **all of the following rules**:

1. Each of the digits `1-9` must occur exactly once in each row.
2. Each of the digits `1-9` must occur exactly once in each column.
3. 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 =

Output:

Explanation: The input board is shown above and the only valid solution is shown below:

Code — Java — Auto

```java
                    if (isValid(board, i, j, c)) {
                        board[i][j] = c;
                        if (solve(board)) return true;
                        board[i][j] = '.';
                    }
                }
                return false;
            }
        }
    }
    return true;
}

private boolean isValid(char[][] board, int row, int col, char c) {
    for (int i = 0; i < 9; i++) {
        if (board[row][i] == c) return false;
        if (board[i][col] == c) return false;
        int boxRow = 3 * (row / 3) + i / 3;
        int boxCol = 3 * (col / 3) + i % 3;
        if (board[boxRow][boxCol] == c) return false;
    }
    return true;
}
}
```

Testcase | Test Result

Accepted Runtime: 2 ms

Case 1

Input

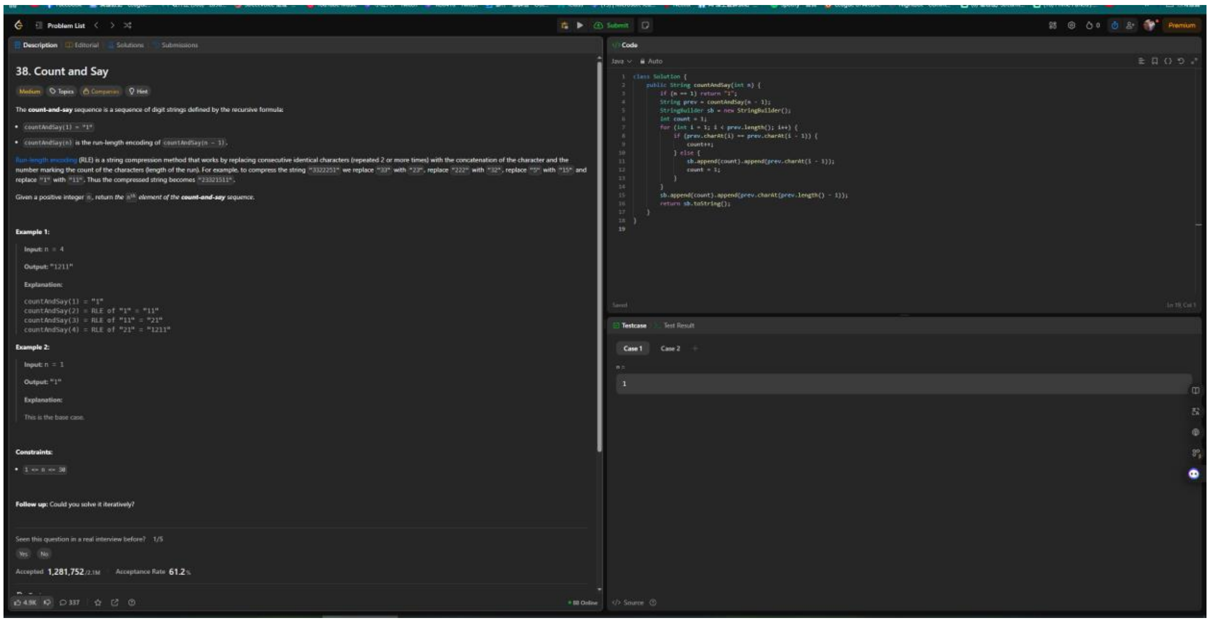
board =

Output

Expected

Contribute a testcase

38. Count and Say.JAVA



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 run-length encoding of `countAndSay(n - 1)`.

Run-length encoding (RLE) 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 `"3322251"` we replace `"33"` with `"23"`, replace `"222"` with `"32"`, replace `"5"` with `"15"` and replace `"1"` with `"11"`. Thus the compressed string becomes `"23321511"`.

Given a positive integer `n`, return *the* `n^th` *element of the* ***count-and-say*** *sequence.*

Example 1:

Input: n = 4

Output: "1211"

Explanation:

```
countAndSay(1) = "1"
countAndSay(2) = RLE of "1" = "11"
countAndSay(3) = RLE of "11" = "21"
countAndSay(4) = RLE of "21" = "1211"
```

Example 2:

Input: n = 1

Output: "1"

Explanation:

This is the base case.

Constraints:

- `1 <= n <= 30`

Follow up: Could you solve it iteratively?

Seen this question in a real interview before? 1/5

Yes No

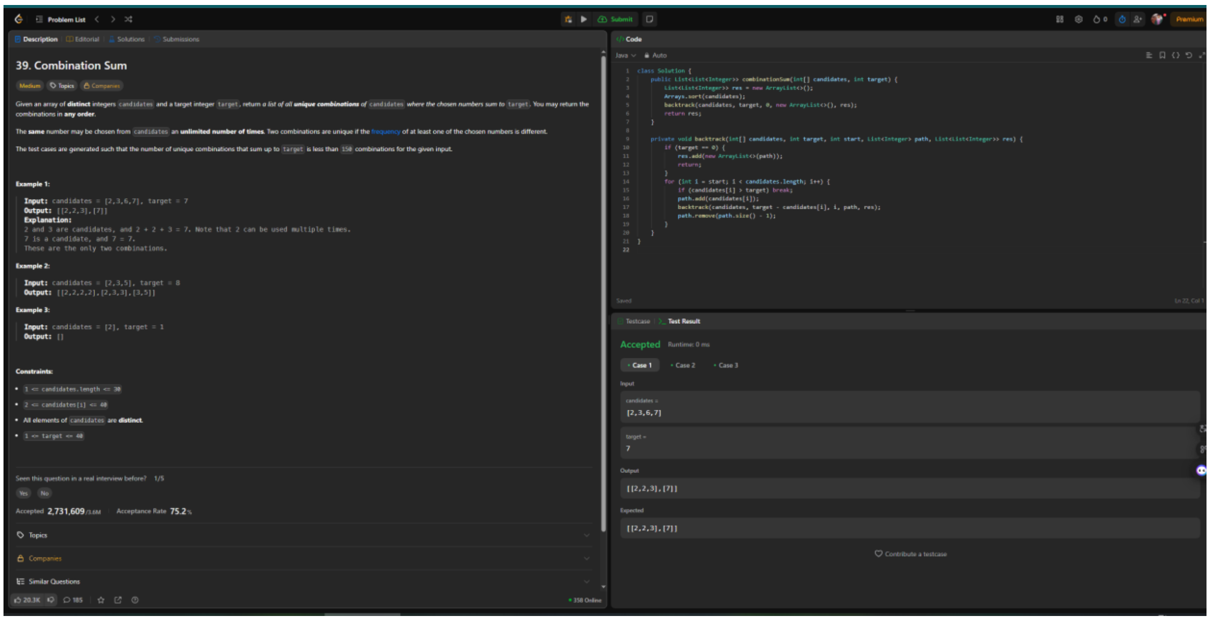
Accepted 1,281,752 /2.1M | Acceptance Rate 61.2 %

```java
class Solution {
    public String countAndSay(int n) {
        if (n == 1) return "1";
        String prev = countAndSay(n - 1);
        StringBuilder sb = new StringBuilder();
        int count = 1;
        for (int i = 1; i < prev.length(); i++) {
            if (prev.charAt(i) == prev.charAt(i - 1)) {
                count++;
            } else {
                sb.append(count).append(prev.charAt(i - 1));
                count = 1;
            }
        }
        sb.append(count).append(prev.charAt(prev.length() - 1));
        return sb.toString();
    }
}
```

n =

1

39. Combination Sum.JAVA



39. Combination Sum

Given an array of **distinct** integers `candidates` and a target integer `target`, return *a list of all* ***unique combinations*** *of* `candidates` *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.

The test cases are generated such that the number of unique combinations that sum up to `target` is less than `150` combinations for the given 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: []
```

Constraints:

- `1 <= candidates.length <= 30`
- `2 <= candidates[i] <= 40`
- All elements of `candidates` are **distinct.**
- `1 <= target <= 40`

Seen this question in a real interview before? 1/5

Yes No

Accepted 2,731,609 /3.6M | Acceptance Rate 75.2 %

```java
class Solution {
    public List<List<Integer>> combinationSum(int[] candidates, int target) {
        List<List<Integer>> res = new ArrayList<>();
        Arrays.sort(candidates);
        backtrack(candidates, target, 0, new ArrayList<>(), res);
        return res;
    }

    private void backtrack(int[] candidates, int target, int start, List<Integer> path, List<List<Integer>> res) {
        if (target == 0) {
            res.add(new ArrayList<>(path));
            return;
        }
        for (int i = start; i < candidates.length; i++) {
            if (candidates[i] > target) break;
            path.add(candidates[i]);
            backtrack(candidates, target - candidates[i], i, path, res);
            path.remove(path.size() - 1);
        }
    }
}
```

Accepted Runtime: 0 ms

Input

candidates =

[2,3,6,7]

target =

7

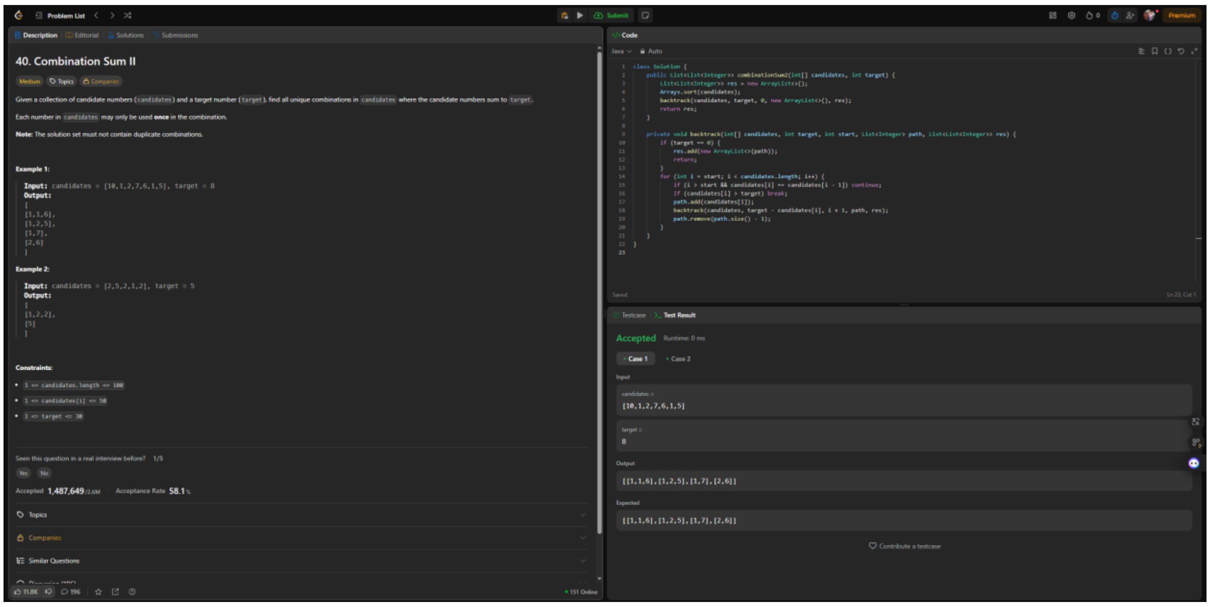
Output

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

Expected

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

40. Combination Sum II.JAVA



40. Combination Sum II

Given a collection of candidate numbers (`candidates`) and a target number (`target`), find all unique combinations in `candidates` 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`

Seen this question in a real interview before? 1/5

Accepted 1,487,649 /2.6M | Acceptance Rate 58.1%

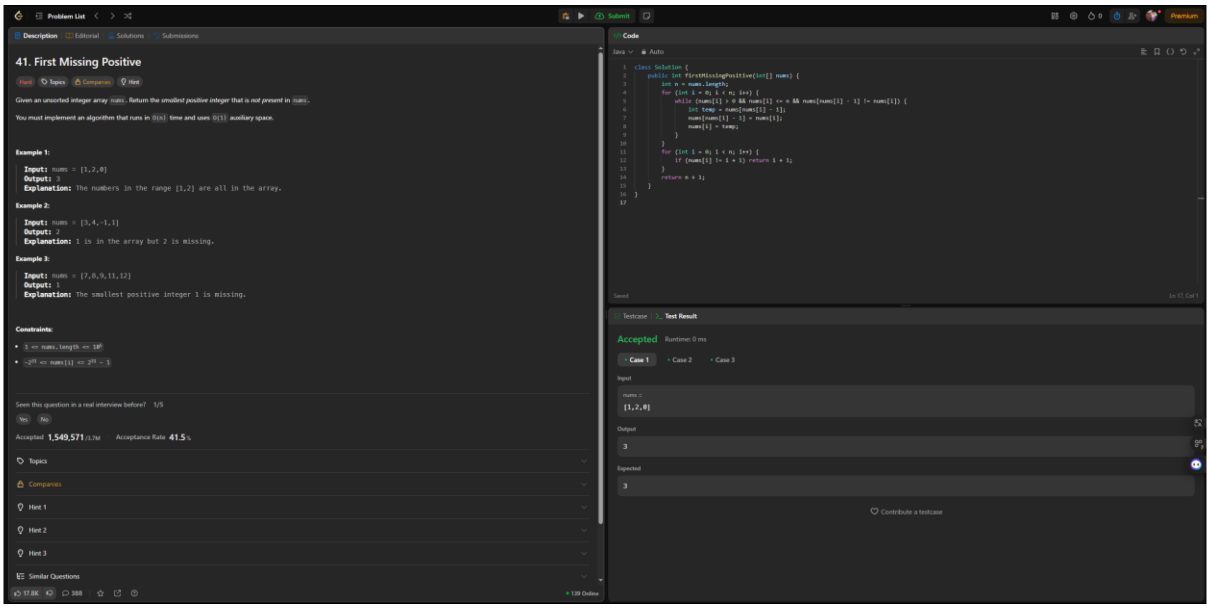
```java
class Solution {
    public List<List<Integer>> combinationSum2(int[] candidates, int target) {
        List<List<Integer>> res = new ArrayList<>();
        Arrays.sort(candidates);
        backtrack(candidates, target, 0, new ArrayList<>(), res);
        return res;
    }

    private void backtrack(int[] candidates, int target, int start, List<Integer> path, List<List<Integer>> res) {
        if (target == 0) {
            res.add(new ArrayList<>(path));
            return;
        }
        for (int i = start; i < candidates.length; i++) {
            if (i > start && candidates[i] == candidates[i - 1]) continue;
            if (candidates[i] > target) break;
            path.add(candidates[i]);
            backtrack(candidates, target - candidates[i], i + 1, path, res);
            path.remove(path.size() - 1);
        }
    }
}
```

Accepted Runtime: 0 ms

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

41. First Missing Positive.JAVA



41. First Missing Positive

Given an unsorted integer array `nums`. Return the *smallest positive integer* that is *not present* in `nums`.

You must implement an algorithm that runs in `O(n)` time and uses `O(1)` auxiliary space.

Example 1:

```
Input: nums = [1,2,0]
Output: 3
Explanation: The numbers in the range [1,2] are all in the array.
```

Example 2:

```
Input: nums = [3,4,-1,1]
Output: 2
Explanation: 1 is in the array but 2 is missing.
```

Example 3:

```
Input: nums = [7,8,9,11,12]
Output: 1
Explanation: The smallest positive integer 1 is missing.
```

Constraints:

- `1 <= nums.length <= 10`^{5}
- `-2`^{31} `<= nums[i] <= 2`^{31} `- 1`

Seen this question in a real interview before? 1/5

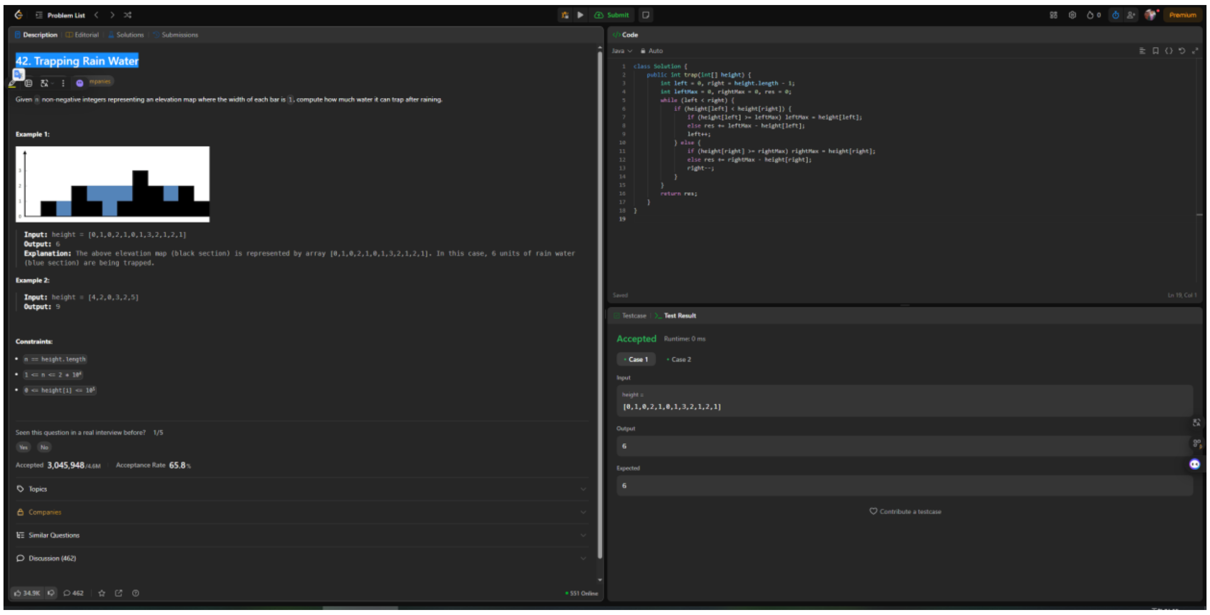
Accepted 1,549,571 /3.7M | Acceptance Rate 41.5%

```java
class Solution {
    public int firstMissingPositive(int[] nums) {
        int n = nums.length;
        for (int i = 0; i < n; i++) {
            while (nums[i] > 0 && nums[i] <= n && nums[nums[i] - 1] != nums[i]) {
                int temp = nums[nums[i] - 1];
                nums[nums[i] - 1] = nums[i];
                nums[i] = temp;
            }
        }
        for (int i = 0; i < n; i++) {
            if (nums[i] != i + 1) return i + 1;
        }
        return n + 1;
    }
}
```

Accepted Runtime: 0 ms

```
Input
nums =
[1,2,0]
Output
3
Expected
3
```

42. Trapping Rain Water.JAVA



Problem List

Description | Editorial | Solutions | Submissions

42. Trapping Rain Water

Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it 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 * 10^4`
- `0 <= height[i] <= 10^5`

Seen this question in a real interview before? 1/5

Yes No

Accepted 3,045,948/4.6M | Acceptance Rate 65.8%

Topics

Companies

Similar Questions

Discussion (462)

34.9K 462 • 551 Online

Code

Java Auto

```java
class Solution {
    public int trap(int[] height) {
        int left = 0, right = height.length - 1;
        int leftMax = 0, rightMax = 0, res = 0;
        while (left < right) {
            if (height[left] < height[right]) {
                if (height[left] >= leftMax) leftMax = height[left];
                else res += leftMax - height[left];
                left++;
            } else {
                if (height[right] >= rightMax) rightMax = height[right];
                else res += rightMax - height[right];
                right--;
            }
        }
        return res;
    }
}
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

height =
[0,1,0,2,1,0,1,3,2,1,2,1]

Output

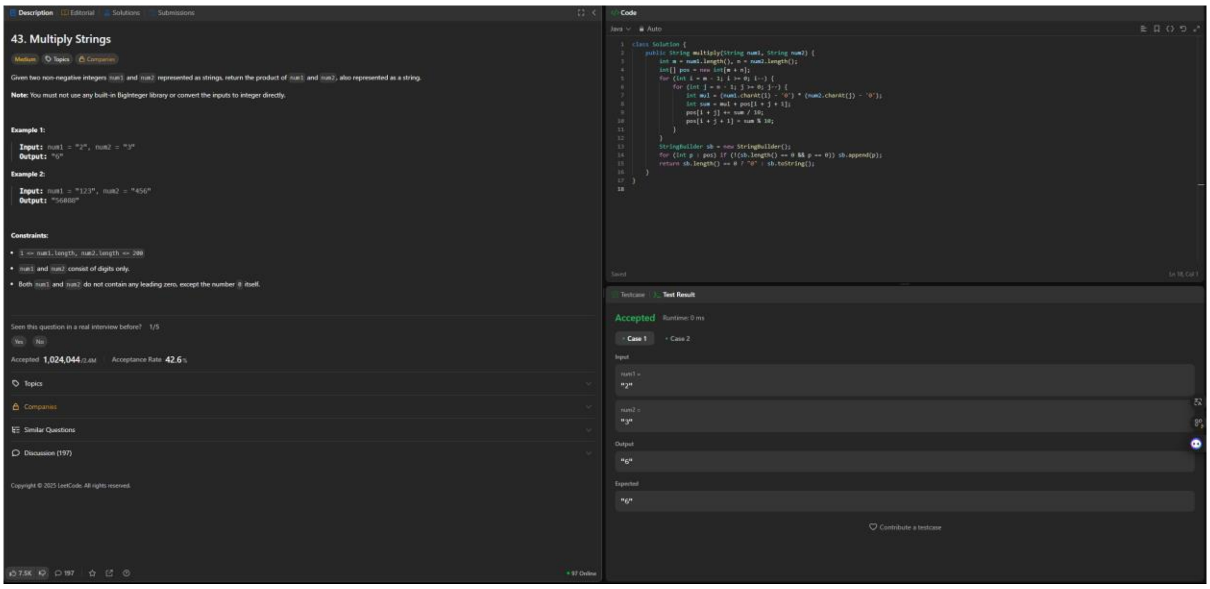
6

Expected

6

Contribute a testcase

43. Multiply Strings.JAVA



Description | Editorial | Solutions | Submissions

43. Multiply Strings

Medium Topics Companies

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.

Seen this question in a real interview before? 1/5

Yes No

Accepted 1,024,044/2.4M | Acceptance Rate 42.6%

Topics

Companies

Similar Questions

Discussion (197)

Copyright © 2025 LeetCode. All rights reserved.

7.5K 197 • 97 Online

Code

Java Auto

```java
class Solution {
    public String multiply(String num1, String num2) {
        int m = num1.length(), n = num2.length();
        int[] pos = new int[m + n];
        for (int i = m - 1; i >= 0; i--) {
            for (int j = n - 1; j >= 0; j--) {
                int mul = (num1.charAt(i) - '0') * (num2.charAt(j) - '0');
                int sum = mul + pos[i + j + 1];
                pos[i + j] += sum / 10;
                pos[i + j + 1] = sum % 10;
            }
        }
        StringBuilder sb = new StringBuilder();
        for (int p : pos) if (!(sb.length() == 0 && p == 0)) sb.append(p);
        return sb.length() == 0 ? "0" : sb.toString();
    }
}
```

Saved

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

num1 =
"2"

num2 =
"3"

Output

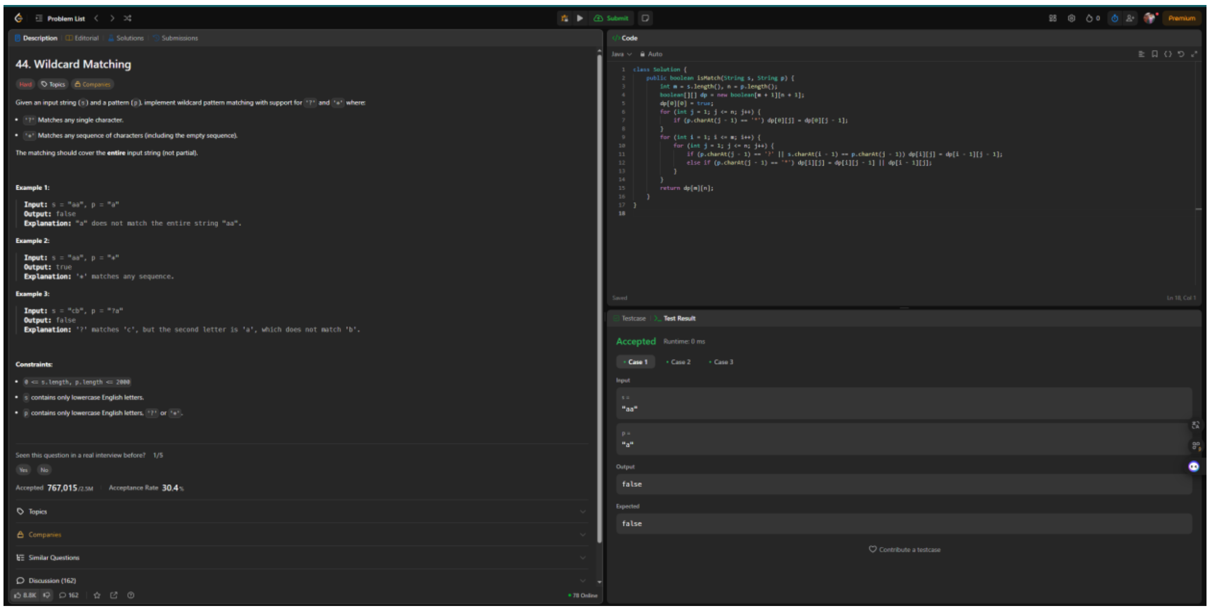
"6"

Expected

"6"

Contribute a testcase

44. Wildcard Matching.JAVA


Problem List

Description | Editorial | Solutions | Submissions

44. Wildcard Matching

Hard · Topics · Companies

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'.
```

Constraints:

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

Seen this question in a real interview before? 1/5

Yes No

Accepted 767,015 /2.5M | Acceptance Rate 30.4%

Topics

Companies

Similar Questions

Discussion (162)

8.8K · 162 · 78 Online

Code

Java · Auto

```java
class Solution {
    public boolean isMatch(String s, String p) {
        int m = s.length(), n = p.length();
        boolean[][] dp = new boolean[m + 1][n + 1];
        dp[0][0] = true;
        for (int j = 1; j <= n; j++) {
            if (p.charAt(j - 1) == '*') dp[0][j] = dp[0][j - 1];
        }
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                if (p.charAt(j - 1) == '?' || s.charAt(i - 1) == p.charAt(j - 1)) dp[i][j] = dp[i - 1][j - 1];
                else if (p.charAt(j - 1) == '*') dp[i][j] = dp[i][j - 1] || dp[i - 1][j];
            }
        }
        return dp[m][n];
    }
}
```

Saved · Ln 18, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3

Input

s =
"aa"

p =
"a"

Output

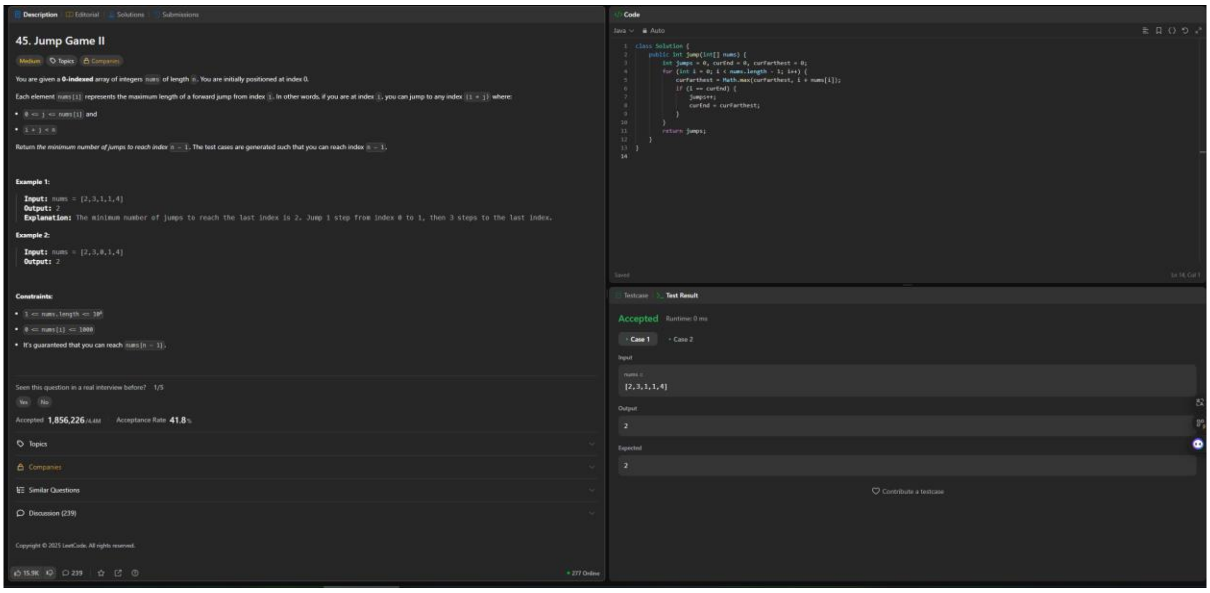
false

Expected

false

Contribute a testcase

45. Jump Game II.JAVA


Description | Editorial | Solutions | Submissions

45. Jump Game II

Medium · Topics · Companies

You are given a **0-indexed** array of integers `nums` of length `n`. You are initially positioned at index 0.

Each element `nums[i]` represents the maximum length of a forward jump from index `i`. In other words, if you are at index `i`, you can jump to any index `(i + j)` where:

- `0 <= j <= nums[i]` and
- `i + j < n`

Return *the minimum number of jumps to reach index* `n - 1`. The test cases are generated such that you can reach index `n - 1`.

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 <=` 10^4
- `0 <= nums[i] <= 1000`
- It's guaranteed that you can reach `nums[n - 1]`.

Seen this question in a real interview before? 1/5

Yes No

Accepted 1,856,226 /4.4M | Acceptance Rate 41.8%

Topics

Companies

Similar Questions

Discussion (239)

Copyright © 2025 LeetCode. All rights reserved.

15.9K · 239 · 277 Online

Code

Java · Auto

```java
class Solution {
    public int jump(int[] nums) {
        int jumps = 0, curEnd = 0, curFarthest = 0;
        for (int i = 0; i < nums.length - 1; i++) {
            curFarthest = Math.max(curFarthest, i + nums[i]);
            if (i == curEnd) {
                jumps++;
                curEnd = curFarthest;
            }
        }
        return jumps;
    }
}
```

Saved · Ln 14, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

nums =
[2,3,1,1,4]

Output

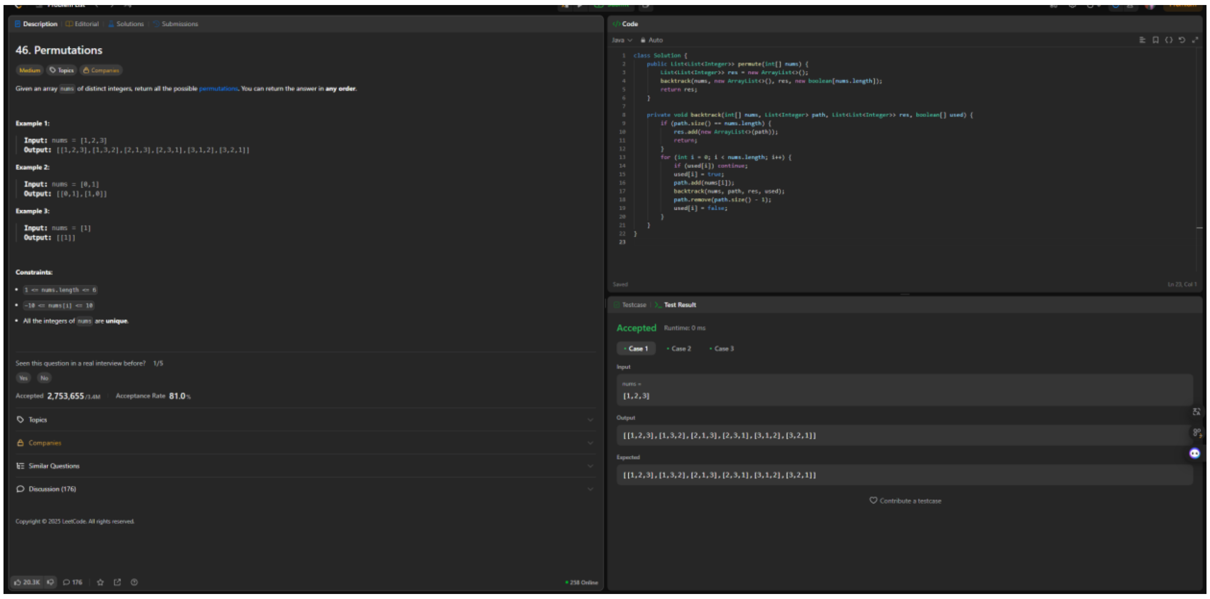
2

Expected

2

Contribute a testcase

46. Permutations.JAVA



46. Permutations

Medium | Topics | Companies

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**.

Seen this question in a real interview before? 1/5

Yes No

Accepted 2,753,655 /3.4M | Acceptance Rate 81.0 %

Topics

Companies

Similar Questions

Discussion (176)

Copyright © 2025 LeetCode. All rights reserved.

20.3K | 176 | 258 Online

Code

Java | Auto

```java
class Solution {
    public List<List<Integer>> permute(int[] nums) {
        List<List<Integer>> res = new ArrayList<>();
        backtrack(nums, new ArrayList<>(), res, new boolean[nums.length]);
        return res;
    }

    private void backtrack(int[] nums, List<Integer> path, List<List<Integer>> res, boolean[] used) {
        if (path.size() == nums.length) {
            res.add(new ArrayList<>(path));
            return;
        }
        for (int i = 0; i < nums.length; i++) {
            if (used[i]) continue;
            used[i] = true;
            path.add(nums[i]);
            backtrack(nums, path, res, used);
            path.remove(path.size() - 1);
            used[i] = false;
        }
    }
}
```

Saved | Ln 23, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 | Case 2 | Case 3

Input

```
nums =
[1,2,3]
```

Output

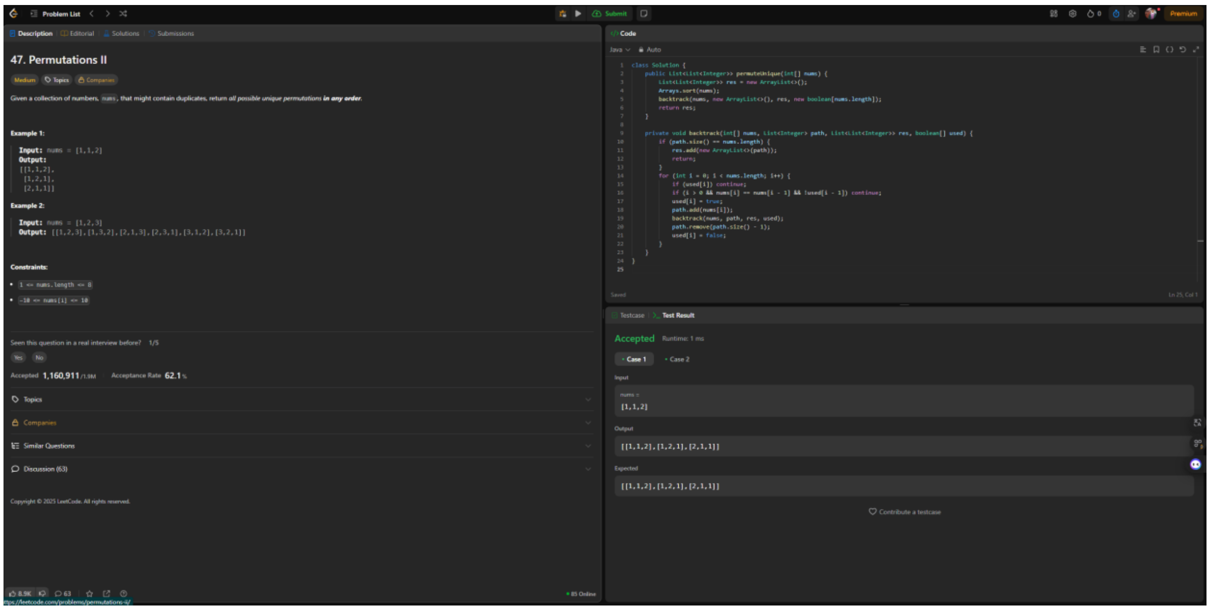
```
[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Expected

```
[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Contribute a testcase

47. Permutations II.JAVA



Problem List | Submit | Premium

47. Permutations II

Medium | Topics | Companies

Given a collection of numbers, `nums`, that might contain duplicates, return *all possible unique permutations* ***in any order***.

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`

Seen this question in a real interview before? 1/5

Yes No

Accepted 1,160,911 /1.9M | Acceptance Rate 62.1 %

Topics

Companies

Similar Questions

Discussion (63)

Copyright © 2025 LeetCode. All rights reserved.

8.9K | 63 | 85 Online

ttps://leetcode.com/problems/permutations-ii/

Code

Java | Auto

```java
class Solution {
    public List<List<Integer>> permuteUnique(int[] nums) {
        List<List<Integer>> res = new ArrayList<>();
        Arrays.sort(nums);
        backtrack(nums, new ArrayList<>(), res, new boolean[nums.length]);
        return res;
    }

    private void backtrack(int[] nums, List<Integer> path, List<List<Integer>> res, boolean[] used) {
        if (path.size() == nums.length) {
            res.add(new ArrayList<>(path));
            return;
        }
        for (int i = 0; i < nums.length; i++) {
            if (used[i]) continue;
            if (i > 0 && nums[i] == nums[i - 1] && !used[i - 1]) continue;
            used[i] = true;
            path.add(nums[i]);
            backtrack(nums, path, res, used);
            path.remove(path.size() - 1);
            used[i] = false;
        }
    }
}
```

Saved | Ln 25, Col 1

Testcase | Test Result

Accepted Runtime: 1 ms

Case 1 | Case 2

Input

```
nums =
[1,1,2]
```

Output

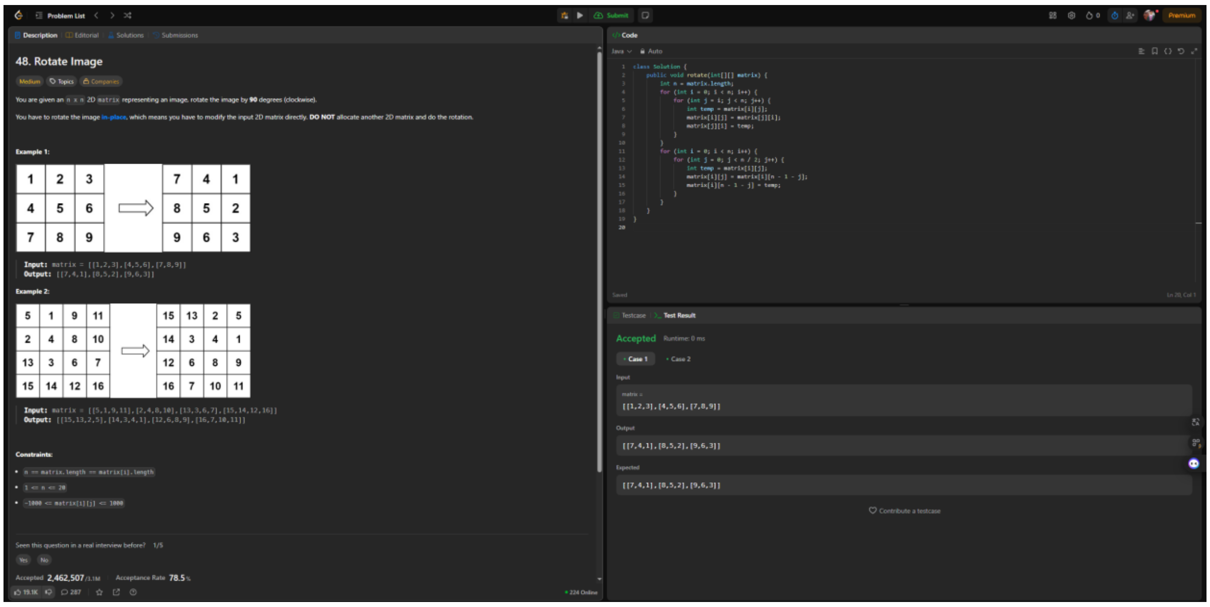
```
[[1,1,2],[1,2,1],[2,1,1]]
```

Expected

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

Contribute a testcase

48. Rotate Image.JAVA



48. Rotate Image

Medium · Topics · Companies

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** allocate another 2D matrix and do the rotation.

Example 1:

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

Example 2:

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

Constraints:

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

Seen this question in a real interview before? 1/5

Yes No

Accepted 2,462,507/3.1M | Acceptance Rate 78.5%

19.1K · 287 · 224 Online

```java
class Solution {
    public void rotate(int[][] matrix) {
        int n = matrix.length;
        for (int i = 0; i < n; i++) {
            for (int j = i; j < n; j++) {
                int temp = matrix[i][j];
                matrix[i][j] = matrix[j][i];
                matrix[j][i] = temp;
            }
        }
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n / 2; j++) {
                int temp = matrix[i][j];
                matrix[i][j] = matrix[i][n - 1 - j];
                matrix[i][n - 1 - j] = temp;
            }
        }
    }
}
```

Saved · Ln 20, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

matrix =

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

Output

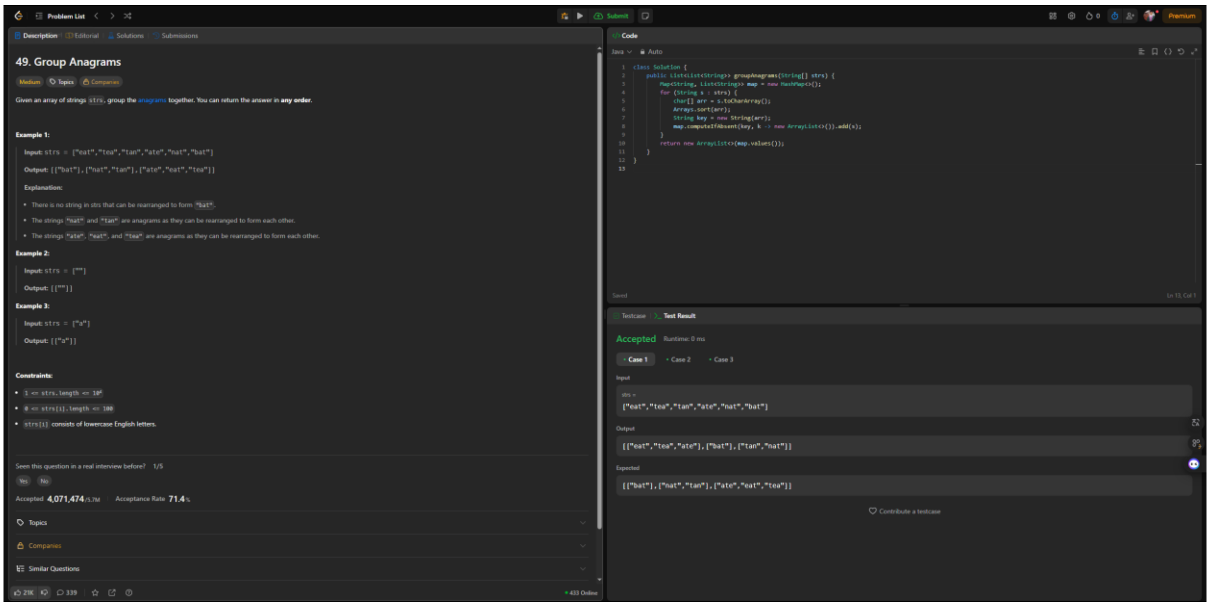
```
[[7,4,1],[8,5,2],[9,6,3]]
```

Expected

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

Contribute a testcase

49. Group Anagrams.JAVA



49. Group Anagrams

Medium · Topics · Companies

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

Example 1:

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

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

Explanation:

- There is no string in strs that can be rearranged to form `"bat"`.
- The strings `"nat"` and `"tan"` are anagrams as they can be rearranged to form each other.
- The strings `"ate"`, `"eat"`, and `"tea"` are anagrams as they can be rearranged to form each other.

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.

Seen this question in a real interview before? 1/5

Yes No

Accepted 4,071,474/5.7M | Acceptance Rate 71.4%

Topics

Companies

Similar Questions

21K · 339 · 433 Online

```java
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        Map<String, List<String>> map = new HashMap<>();
        for (String s : strs) {
            char[] arr = s.toCharArray();
            Arrays.sort(arr);
            String key = new String(arr);
            map.computeIfAbsent(key, k -> new ArrayList<>()).add(s);
        }
        return new ArrayList<>(map.values());
    }
}
```

Saved · Ln 13, Col 1

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3

Input

strs =

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

Output

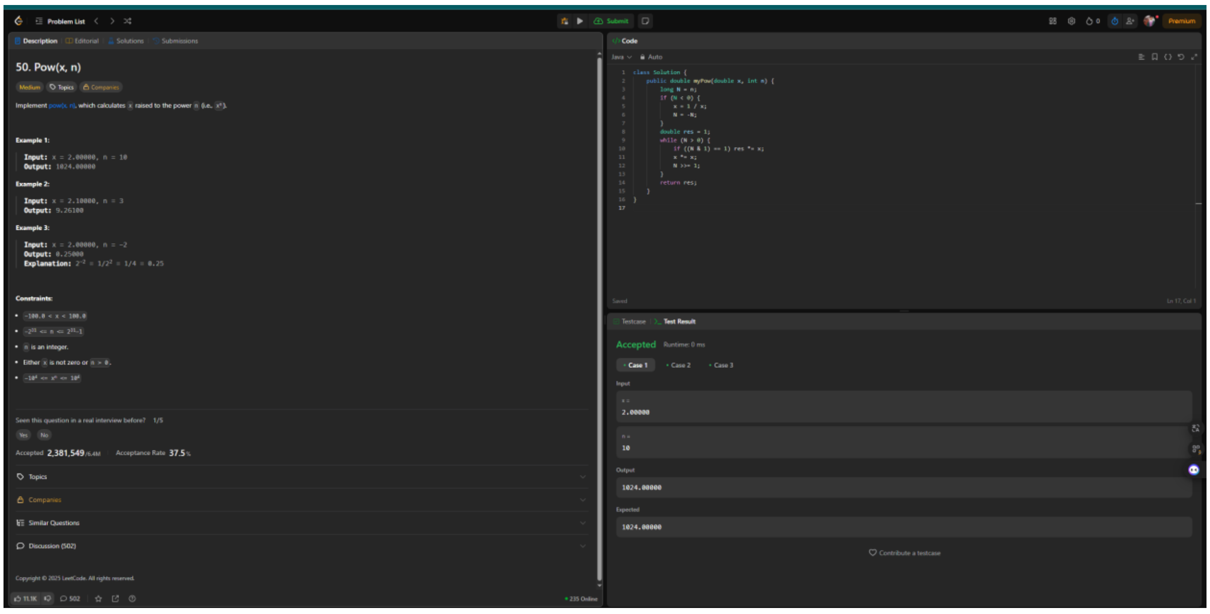
```
[["eat","tea","ate"],["bat"],["tan","nat"]]
```

Expected

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

Contribute a testcase

50. Pow(x, n).JAVA



50. Pow(x, n)

Implement pow(x, n), which calculates `x` raised to the power `n` (i.e., x^n).

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 = 1/2^2 = 1/4 = 0.25
```

Constraints:

- `-100.0 < x < 100.0`
- -2^{31} `<= n <=` $2^{31}-1$
- `n` is an integer.
- Either `x` is not zero or `n > 0`.
- -10^4 `<=` x^n `<=` 10^4

Seen this question in a real interview before? 1/5

Accepted 2,381,549/6.4M | Acceptance Rate 37.5%

```java
class Solution {
    public double myPow(double x, int n) {
        long N = n;
        if (N < 0) {
            x = 1 / x;
            N = -N;
        }
        double res = 1;
        while (N > 0) {
            if ((N & 1) == 1) res *= x;
            x *= x;
            N >>= 1;
        }
        return res;
    }
}

```

Accepted Runtime: 0 ms

Input

x =
```
2.00000
```

n =
```
10
```

Output
```
1024.00000
```

Expected
```
1024.00000
```