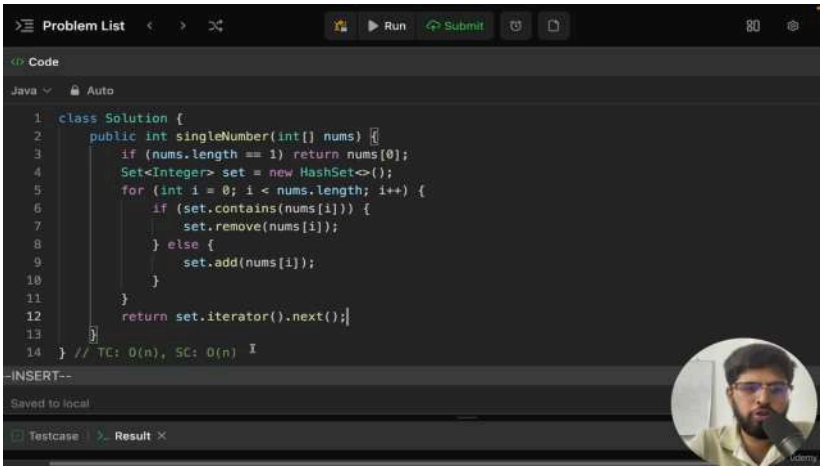
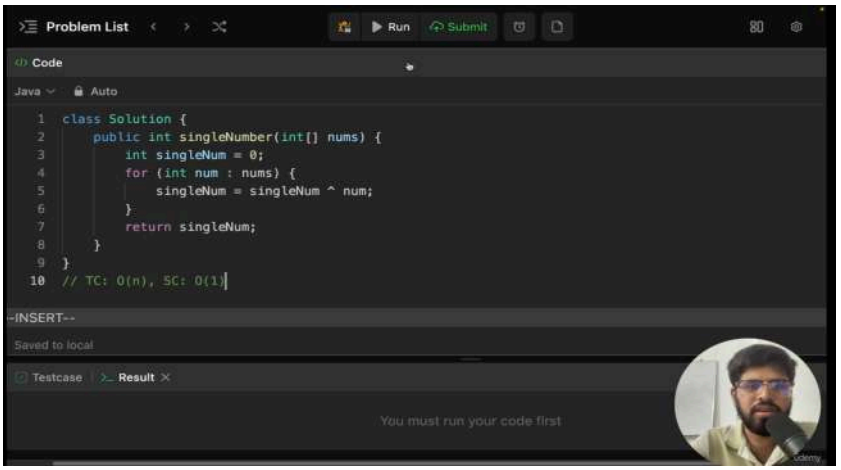
2. FizzBuzz (LeetCode 412) [Asked by LinkedIn]

It's through an error
Hint : *answer.add("fizzBuzz");*

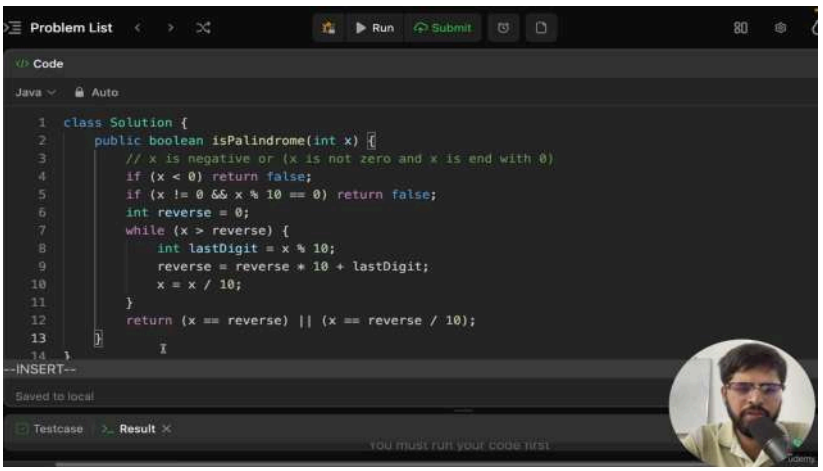
3. Single Number (LeetCode 136) [Asked by Amazon]
Brute force approach:



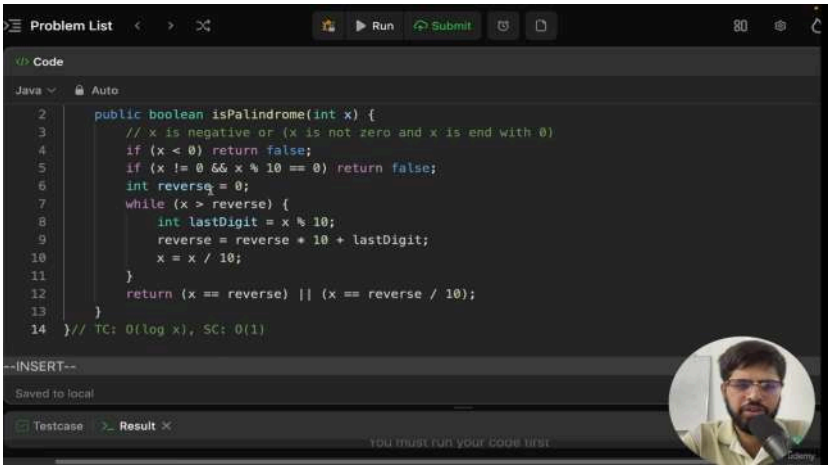
Optimal Solution:



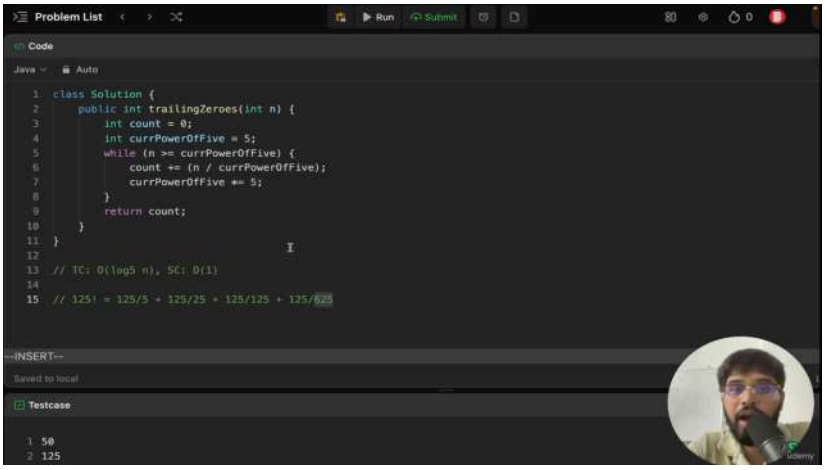
4. Palindrome Number (LeetCode 9) [Asked by Facebook/Bloomberg]



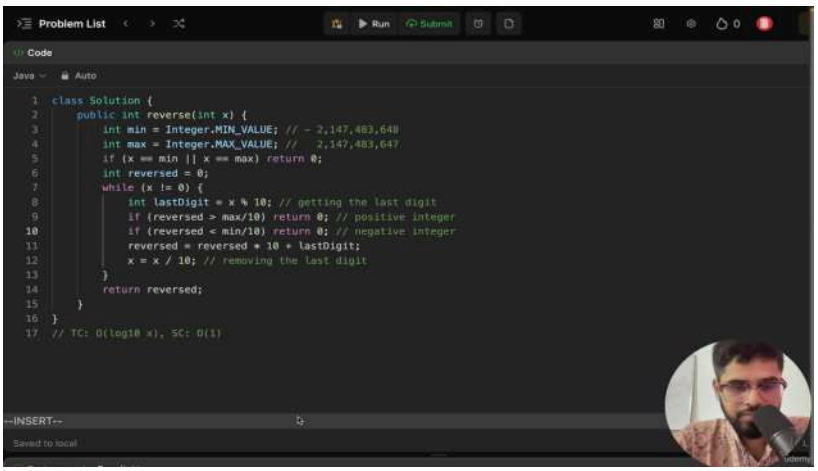
Time & Space Complexity:



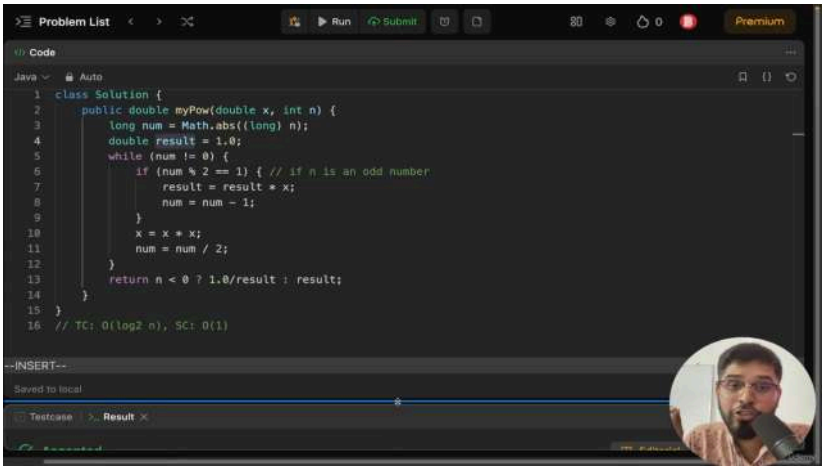
5. Factorial Trailing Zeroes (LeetCode 172) [Asked by Microsoft]



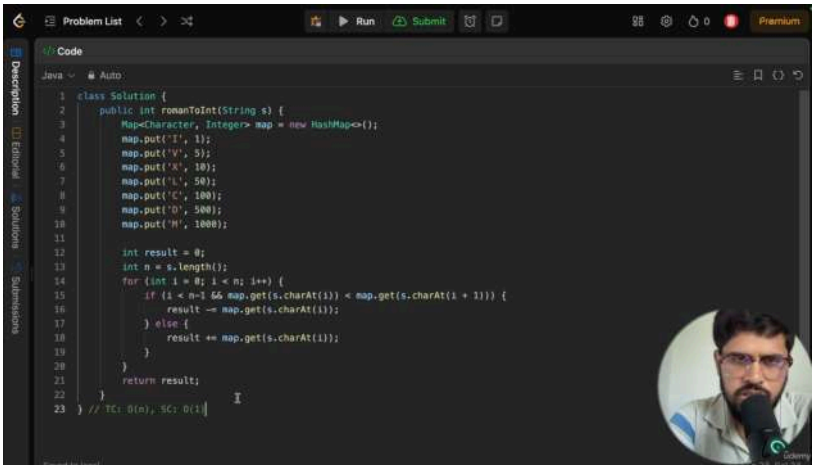
6. Reverse Integer (LeetCode 7) [Asked by Facebook]



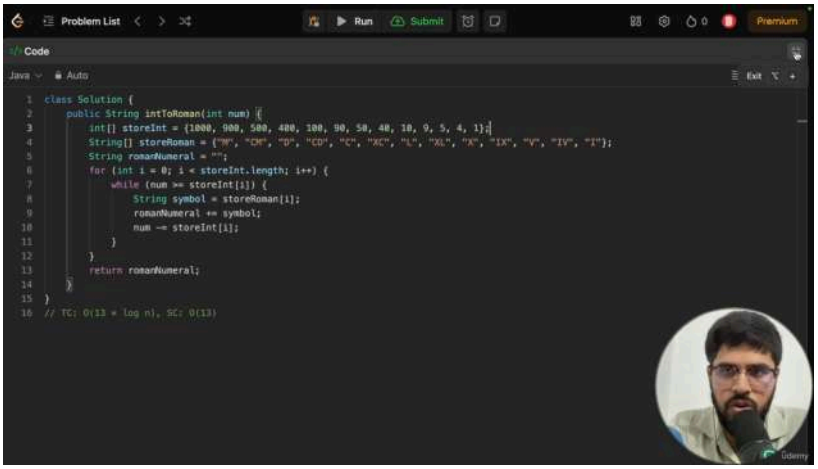
7. Pow(x, n) (LeetCode 50) [Asked by Facebook/Amazon/LinkedIn]



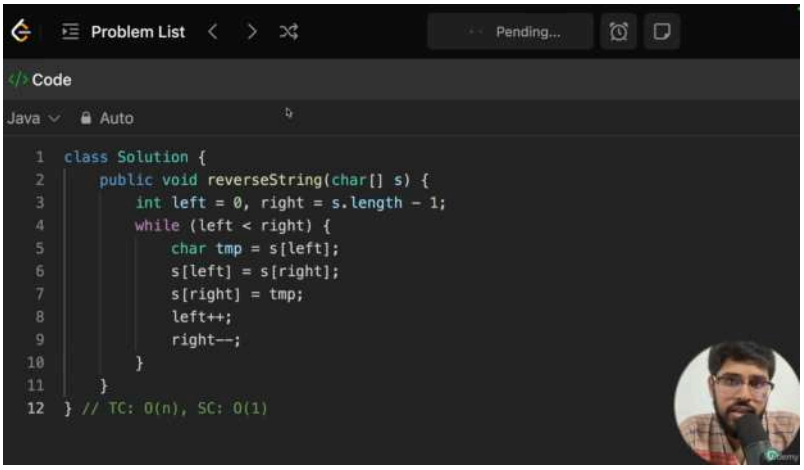
8. Roman to Integer (LeetCode 344) [Asked by MicroSoft/Uber]



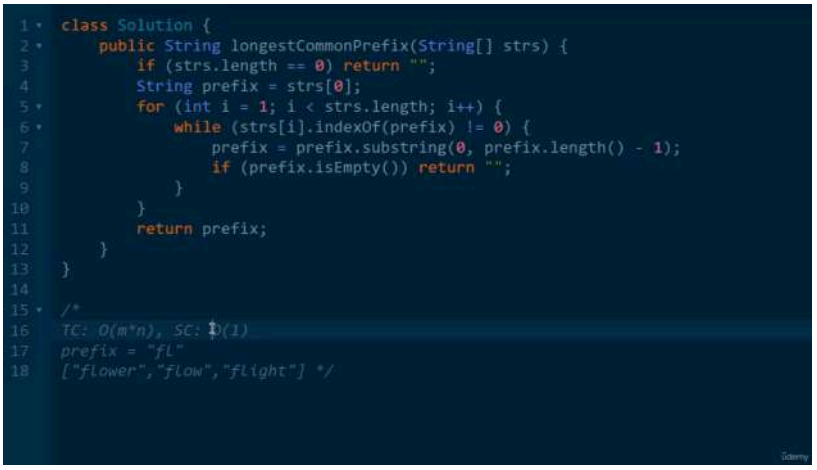
9.Integer to Roman



10.Reverse a String



9. Longest Common Prefix (LeetCode 14) [Asked by Facebook]

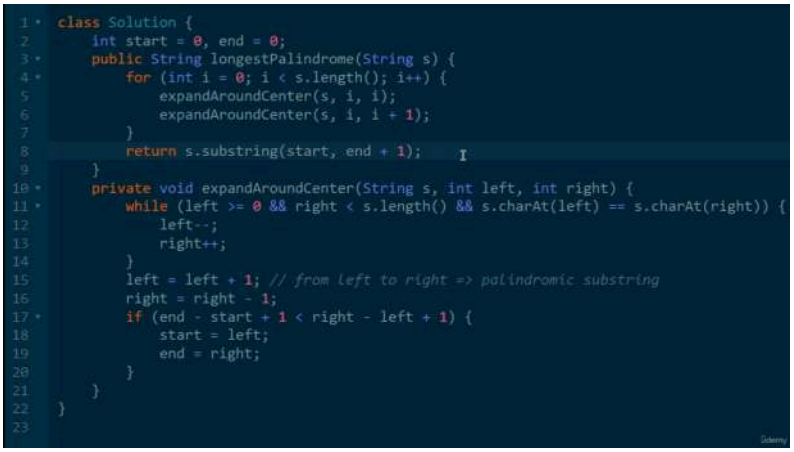


10. Repeated DNA Sequences (LeetCode 187) [Asked by Google]
Hashmap:

Set ds:

11. Valid Anagram (LeetCode 242) [Asked by Facebook]

12. Longest Palindromic Substring (LeetCode 5) [Asked by Facebook]



```
class Solution {
    int start = 0, end = 0;
    public String longestPalindrome(String s) {
        for (int i = 0; i < s.length(); i++) {
            expandAroundCenter(s, i, i);
            expandAroundCenter(s, i, i + 1);
        }
        return s.substring(start, end + 1);
    }
    private void expandAroundCenter(String s, int left, int right) {
        while (left >= 0 && right < s.length() && s.charAt(left) == s.charAt(right)) {
            left--;
            right++;
        }
        left = left + 1; // from left to right => palindromic substring
        right = right - 1;
        if (end - start + 1 < right - left + 1) {
            start = left;
            end = right;
        }
    }
}
```

13. Longest Substring without Repeating Characters (LeetCode 3) [Asked by Google]

```
class Solution {
    public int lengthOfLongestSubstring(String s) {
        int ans = 0;
        HashMap<Character, Integer> map = new HashMap<>(); // n
        int left = 0;
        for (int right = 0; right < s.length(); right++) { // n
            char ch = s.charAt(right);
            if (!map.containsKey(ch)) {
                map.put(ch, right);
            } else {
                left = Math.max(left, map.get(ch) + 1);
                map.put(ch, right);
            }
            ans = Math.max(ans, right - left + 1);// left to right
        }
        return ans;
    }
}

// TC: O(n), SC: O(n)
```

14. Reverse Words in a String (LeetCode 151) [Asked by Facebook/Amazon]

15. Integer to English Words (LeetCode 273) [Facebook, Microsoft]

16. Binary Search (LeetCode 704) [Asked by Infosys,Oracle]

```java
class Solution {
    public int search(int[] nums, int target) {
        int left = 0;
        int right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2; // (left + right) / 2
            if (nums[mid] == target) {
                return mid;
            }
            if (target > nums[mid]) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return -1;
    }
} // TC: O(log n), SC: O(1)
```

17. Search in Rotated Sorted Array (LeetCode 33) [Asked by Facebook]

```java
class Solution {
    public int search(int[] nums, int target) {
        int left = 0;
        int right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) return mid;
            if (nums[left] <= nums[mid]) { // left to right is sorted
                if (nums[left] <= target && target < nums[mid])
                    right = mid - 1;
                else
                    left = mid + 1;
            } else { // mid to right is sorted
                if (nums[mid] < target && target <= nums[right])
                    left = mid + 1;
                else
                    right = mid - 1;
            }
        }
        return -1;
    }
} // TC: O(log n), SC: O(1)
```

18. Find Minimum in Rotated Sorted Array (LeetCode 153) [Asked by Facebook]

```java
class Solution {
    public int findMin(int[] nums) {
        if (nums.length == 1) return nums[0];
        if (nums.length == 2) return Math.min(nums[0], nums[1]);
        if (nums[0] < nums[nums.length - 1]) return nums[0];
        int left = 0;
        int right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2; // (left + right) / 2
            // the array is decreasing at mid + 1
            if (nums[mid] > nums[mid+1]) return nums[mid+1];
            // the arrays is decreasing at mid
            if (nums[mid-1] > nums[mid]) return nums[mid];
            // discard the sorted part >> increasing part
            if (nums[left] < nums[mid])
                left = mid + 1;
            else
                right = mid - 1;
        }
        return 0;
    }
} // TC: O(log n), SC: O(1)
```

19. Two Sum (LeetCode 1) [Asked by Google]

```java
class Solution {
    public int[] twoSum(int[] nums, int target) {
        int[] ans = new int[2];
        HashMap<Integer, Integer> map = new HashMap<>();
        for (int i = 0; i < nums.length; i++) { // n
            int diff = target - nums[i];
            if (map.containsKey(diff)) {
                ans[0] = i;
                ans[1] = map.get(diff);
                break;
            } else {
                map.put(nums[i], i);
            }
        }
        return ans;
    }
} // TC: O(n), SC: O(n)
```

20. Move Zeroes (LeetCode 283) [Asked by Facebook]

```java
class Solution {
    public void moveZeroes(int[] nums) {
        if (nums.length == 1) return;
        int left = 0;
        for (int right = 0; right < nums.length; right++) {
            if (nums[right] != 0) {
                nums[left] = nums[right];
                left++;
            }
        }
        // from index: left to nums.length-1,, with 0's
        for (int i = left; i < nums.length; i++) {
            nums[i] = 0;
        }
    }
} // TC: O(n), SC: O(1)
```

21. Best Time to Buy And Sell Stock(LeetCode 121) [Asked by Facebook/Microsoft]

```java
class Solution {
    public int maxProfit(int[] prices) {
        if (prices.length == 1) return 0;

        int profit = 0;
        int left = 0;
        for (int right = 1; right < prices.length; right++) {
            if (prices[right] > prices[left]) {
                profit = Math.max(profit, prices[right] - prices[left]);
            } else {
                left = right;
            }
        }

        return profit;
    }
}

// TC: O(n), SC: O(1)
```

22. Ransom Note (LeetCode 383) [Asked by Amazon]

```java
class Solution {
    public boolean canConstruct(String ransomNote, String magazine) {
        int[] frequency = new int[26];
        for (int i = 0; i < magazine.length(); i++) {
            char c = magazine.charAt(i);
            frequency[c - 'a']++;
        }
        for (int i = 0; i < ransomNote.length(); i++) {
            char c = ransomNote.charAt(i);
            if (frequency[c - 'a'] == 0) return false;
            frequency[c - 'a']--;
        }
        return true;
    }
} // TC: O(m + n), SC: O(1)
```

23. Contains Duplicate (LeetCode 217) [Asked by Google]

```java
class Solution {
    public boolean containsDuplicate(int[] nums) {
        Set<Integer> set = new HashSet<>();
        for (int i = 0; i < nums.length; i++) {
            if (set.contains(nums[i])) {
                return true;
            } else {
                set.add(nums[i]);
            }
        }
        return false;
    }
} // TC: O(n), SC: O(n)
```

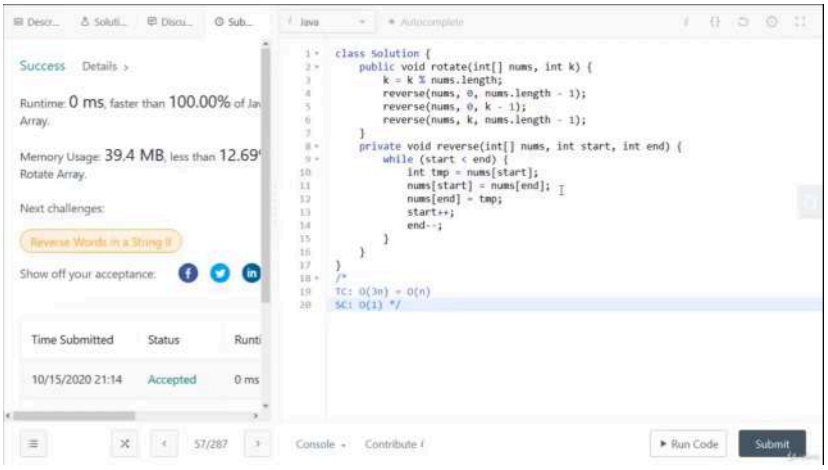
24. Length of Last Word (LeetCode 58) [Asked by Amazon]

```java
class Solution {
    public int lengthOfLastWord(String s) {
        int right = s.length() - 1;
        while (right >= 0 && s.charAt(right) == ' ') {
            right--;
        }
        int left = right;
        while (left >= 0 && s.charAt(left) != ' ') {
            left--;
        }
        return right - left;
    }
} // TC: O(n), SC: O(1)
```

25. Best Time to Buy and Sell Stock II (LeetCode 122) [Asked by Google]

```java
class Solution {
    public int maxProfit(int[] prices) {
        int profit = 0;
        for (int i = 1; i < prices.length; i++) {
            if (prices[i-1] < prices[i])
                profit += prices[i] - prices[i-1];
        }
        return profit;
    }
} // TC: O(n), SC: O(1)
```

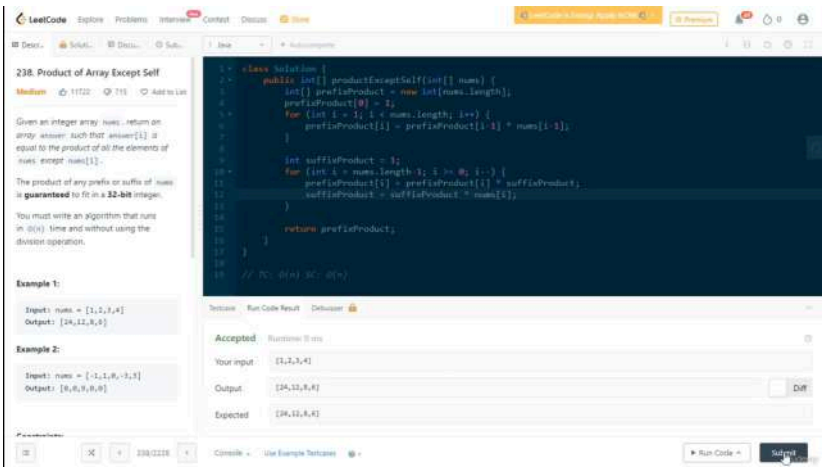
26. Rotate Array (LeetCode 189) [Asked by Microsoft]



27. Jump Game (LeetCode 55) [Asked by Google]

```java
class Solution {
    public boolean canJump(int[] nums) {
        int reachable = 0;
        for (int i = 0; i < nums.length && i <= reachable; i++) {
            reachable = Math.max(reachable, i + nums[i]);
            if (reachable >= nums.length - 1)
                return true;
        }
        return false;
    }
} // TC: O(n), SC: O(1)
```

28. Product of Array Except Self (LeetCode 238) [Asked by Facebook]



29. Sequential Digits (LeetCode 1291) [Asked by Facebook]

```
class Solution {
    public List<Integer> sequentialDigits(int low, int high) {
        String str = "123456789";
        List<Integer> list = new ArrayList<>();
        for (int i = 1; i <= 9; i++) {
            for (int j = 0; i + j <= 9; j++) {
                String tmp = str.substring(j, i + j);
                int val = Integer.valueOf(tmp);
                if (val >= low && val <= high) {
                    list.add(val);
                }
            }
        }
        return list;
    }
} // TC: O(1), SC: O(1)
```

30. Find All Disappeared Numbers in an Array (LeetCode 448) [Asked by Google]

```
class Solution {
    public List<Integer> findDisappearedNumbers(int[] nums) {
        for (int i = 0; i < nums.length; i++) {
            int index = Math.abs(nums[i]) - 1;
            if (nums[index] > 0) {
                nums[index] = -1 * nums[index];
            }
        }
        List<Integer> list = new ArrayList<>();
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] > 0) {
                list.add(i + 1);
            }
        }
        return list;
    }
} // TC: O(n) SC: O(1)
```

31. Find All Duplicates Numbers in an Array (Leetcode 442) [Asked by Facebook]

```java
class Solution {
    public List<Integer> findDuplicates(int[] nums) {
        List<Integer> list = new ArrayList<>();
        for (int i = 0; i < nums.length; i++) {
            int index = Math.abs(nums[i]) - 1;
            if (nums[index] < 0) {
                list.add(Math.abs(nums[i]));
            }
            if (nums[index] > 0) {
                nums[index] = - 1 * nums[index];
            }
        }
        return list;
    }
} // TC: O(n), SC: O(1)
```

32. First Missing Positive (LeetCode 41) [Asked by Microsoft]

```java
class Solution {
    public int firstMissingPositive(int[] nums) {
        // preprocess the array
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] <= 0) {
                nums[i] = nums.length + 1;
            }
        }
        // marking indices
        for (int i = 0; i < nums.length; i++) {
            int index = Math.abs(nums[i]) - 1;
            if (index < nums.length && nums[index] > 0) {
                nums[index] = -1 * nums[index];
            }
        }
        // scan the array
        for (int i = 0; i < nums.length; i++) {
            if (nums[i] > 0) {
                return i + 1;
            }
        }
        return nums.length + 1;
    }
} // TC: O(n), SC: O(1)
```

Testcase Run Code Result Debugger

33. Next Permutation (LeetCode 31) [Asked by Google]

```java
class Solution {
    public void nextPermutation(int[] nums) {
        int i = nums.length - 2;
        while (i >= 0 && nums[i] >= nums[i + 1]) {
            i--;
        }

        if (i >= 0) {
            int j = nums.length - 1;
            while (nums[i] >= nums[j]) {
                j--;
            }
            swap(nums, i, j);
        }
        reverse(nums, i + 1, nums.length - 1);
    }

    public void swap(int[] nums, int i, int j) {
        int tmp = nums[i];
        nums[i] = nums[j];
        nums[j] = tmp;
    }

    public void reverse(int[] nums, int i, int j) {
        while (i < j) {
            swap(nums, i, j);
            i++;
            j--;
        }
    }
}
// TC: O(n), SC: O(1)
```

34. Largest Subarray With 0 Sum [Asked by Microsoft]

```java
class GfG
{
    int maxLen(int arr[], int n)
    {
        int sum = 0, len = 0;
        HashMap<Integer, Integer> map = new HashMap<>();
        for (int i = 0; i < n; i++) {
            sum += arr[i];
            if (sum == 0) {
                len = i + 1;
            } else if (map.containsKey(sum)) {
                len = Math.max(len, i - map.get(sum));
            } else {
                map.put(sum, i);
            }
        }
        return len;
    }
} // TC: O(n), SC: O(n)
```

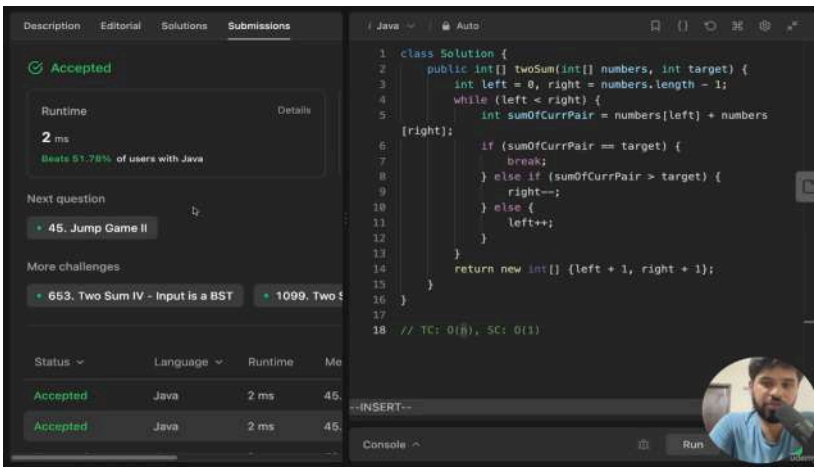
35. Subarray Product Less than K (LeetCode 713) [Asked by Goldman Sachs]

```java
class Solution {
    public int numSubarrayProductLessThanK(int[] nums, int k) {
        int prod = 1, count = 0, left = 0;
        for (int right = 0; right < nums.length; right++) {
            prod *= nums[right];
            while (left <= right && prod >= k) {
                prod /= nums[left];
                left++;
            }
            count += right - left + 1;
        }
        return count;
    }
} // O(n), O(1)
```

36. K-Diff Pairs in an Array (LeetCode 532) [Asked by Amazon]

```java
class Solution {
    public int findPairs(int[] nums, int k) {
        HashMap<Integer, Integer> map = new HashMap<>();
        for (int num : nums) {
            map.put(num, map.getOrDefault(num, 0) + 1);
        }
        int count = 0;
        for (Map.Entry<Integer, Integer> entry : map.entrySet()) {
            if (k == 0) {
                if (entry.getValue() >= 2) {
                    count++;
                }
            } else {
                if (map.containsKey(entry.getKey() + k)) {
                    count++;
                }
            }
        }
        return count;
    }
} // TC: O(n), SC: O(n)
```

37. Two Sum II - Input Array Is Sorted (LeetCode 167) [Asked by Amazon]



38. Is Subsequence (LeetCode 392) [Asked by Google]

```java
class Solution {
    public boolean isSubsequence(String s, String t) {
        int p1 = 0, p2 = 0;
        while (p1 < s.length() && p2 < t.length()) {
            if (s.charAt(p1) == t.charAt(p2)) {
                p1++;
                p2++;
            } else {
                p2++;
            }
        }
        return p1 == s.length();
    }
} // TC: O(n), SC: O(1)
```

39. Remove Duplicates from Sorted Array (LeetCode 26) [Asked by Facebook/Microsoft]

```java
class Solution {
    public int removeDuplicates(int[] nums) {
        int left = 0, right = 0; // 0 ~ left
        while (right < nums.length) {
            if (nums[left] != nums[right]) {
                left++;
                nums[left] = nums[right];
            }
            right++;
        }
        return left + 1;
    }
} // TC: O(n), SC: O(1)
```

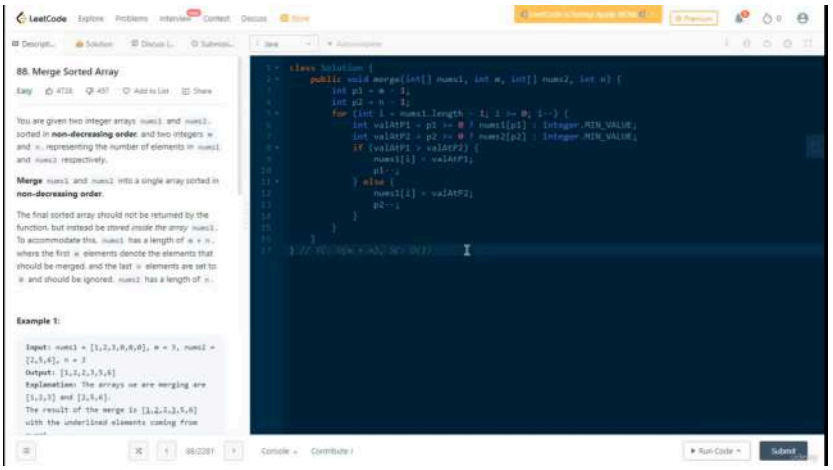
40. Sort Colors (LeetCode 75) [Asked by Facebook]

```java
class Solution {
    public void sortColors(int[] nums) {
        int left = 0, mid = 0, right = nums.length - 1;
        while (mid <= right) {
            if (nums[mid] == 0) {
                swap(left, mid, nums);
                left++;
                mid++;
            } else if (nums[mid] == 1) {
                mid++;
            } else {
                swap(mid, right, nums);
                right--;
            }
        }
    }
    private void swap(int i, int j, int[] nums) {
        int tmp = nums[i];
        nums[i] = nums[j];
        nums[j] = tmp;
    }
} // TC: O(n), SC: O(1)
```

41. Valid Palindrome (LeetCode 125) [Asked by Facebook]

```java
class Solution {
    public boolean isPalindrome(String s) {
        int p1 = 0, p2 = s.length() - 1;
        while (p1 <= p2) {
            char c1 = s.charAt(p1);
            char c2 = s.charAt(p2);
            if (Character.isLetterOrDigit(c1) == false) p1++;
            else if (Character.isLetterOrDigit(c2) == false) p2--;
            else {
                if (Character.toLowerCase(c1) != Character.toLowerCase(c2)) {
                    return false;
                }
                p1++;
                p2--;
            }
        }
        return true;
    }
} // TC: O(n), SC: O(1)
```

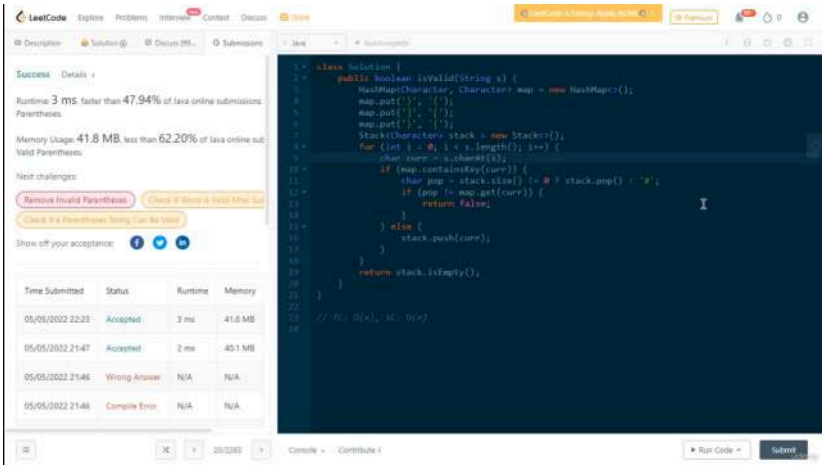
42. Merge Sorted Array (LeetCode 80) [Asked by Microsoft]



43. 3 Sum (LeetCode 15) [Asked by Facebook]

```java
class Solution {
    public List<List<Integer>> threeSum(int[] nums) {
        Arrays.sort(nums);
        List<List<Integer>> triplets = new ArrayList<>();
        for (int i = 0; i <= nums.length - 3; i++) {
            if (i == 0 || nums[i] != nums[i - 1]) {
                int left = i + 1, right = nums.length - 1;
                int target = 0 - nums[i];
                while (left < right) {
                    if (nums[left] + nums[right] == target) {
                        List<Integer> triplet = new ArrayList<>();
                        triplet.add(nums[i]);
                        triplet.add(nums[left]);
                        triplet.add(nums[right]);
                        triplets.add(triplet);

                        while (left < nums.length - 1 && nums[left] == nums[left + 1]) left++;
                        while (right > 0 && nums[right] == nums[right - 1]) right--;
                        left++;
                        right--;
                    } else if (nums[left] + nums[right] < target) {
                        left++;
                    } else {
                        right--;
                    }
                }
            }
        }
        return triplets;
    }
} // TC: O(n log n) + O(n^2), SC: O(1)
```

44. Valid Parentheses (LeetCode 20) [Facebook]



45. Asteroid Collisions (LeetCode 735) [Asked by Lyft]

46. Longest Valid Parentheses (LeetCode 32) [Asked by Google]

```java
class Solution {
    public int longestValidParentheses(String s) {
        int ans = 0;
        Stack<Integer> stack = new Stack<>();
        stack.push(-1);
        for (int i = 0; i < s.length(); i++) {
            char ch = s.charAt(i);
            if (ch == '(') {
                stack.push(i);
            } else {
                stack.pop();
                if (stack.size() == 0) {
                    stack.push(i);
                } else {
                    ans = Math.max(ans, i - stack.peek());
                }
            }
        }
        return ans;
    }
} // TC: O(n), SC: O(n)
```

47. Decode String (LeetCode 394) [Asked by Google]

```java
        Stack<Integer> numStack = new Stack<>();
        Stack<String> strStack = new Stack<>();
        StringBuilder sb = new StringBuilder();
        int len = s.length();
        for (int i = 0; i < len; i++) {
            char ch = s.charAt(i);
            if (Character.isDigit(ch)) {
                int num = ch - '0';
                while (i + 1 < len && Character.isDigit(s.charAt(i+1))) {
                    num = num * 10 + s.charAt(i+1) - '0';
                    i++;
                }
                numStack.push(num);
            } else if (ch == '[') {
                strStack.push(sb.toString());
                sb = new StringBuilder();
            } else if (ch == ']') {
                int k = numStack.pop();
                StringBuilder tmp = new StringBuilder(strStack.pop());
                for (int j = 0; j < k; k++) { // k times
                    tmp.append(sb);
                }
                sb = tmp;
            } else {
                sb.append(ch);
            }
        }
        return sb.toString();
    }
} // TC: O(n * k), SC: O(n)
```

48. Largest Rectangle in Histogram (LeetCode 84) [Asked by Google]

```java
class Solution {
    public int largestRectangleArea(int[] heights) {
        int maxArea = 0;
        Stack<Integer> stack = new Stack<>();
        stack.push(0);
        for (int i = 1; i <= heights.length; i++) {
            int curr = (i == heights.length) ? -1 : heights[i];
            while (!stack.isEmpty() && curr <= heights[stack.peek()]) {
                int height = heights[stack.pop()];
                int width = stack.isEmpty() ? i : i - stack.peek() - 1;
                int currArea = height * width;
                maxArea = Math.max(maxArea, height * width);
            }
            stack.push(i);
        }
        return maxArea;
    }
}

// TC: O(n), SC: O(n)
```

49. Kaden's Algorithm (Maximum Subarray :: LeetCode 53) [Asked by Google]

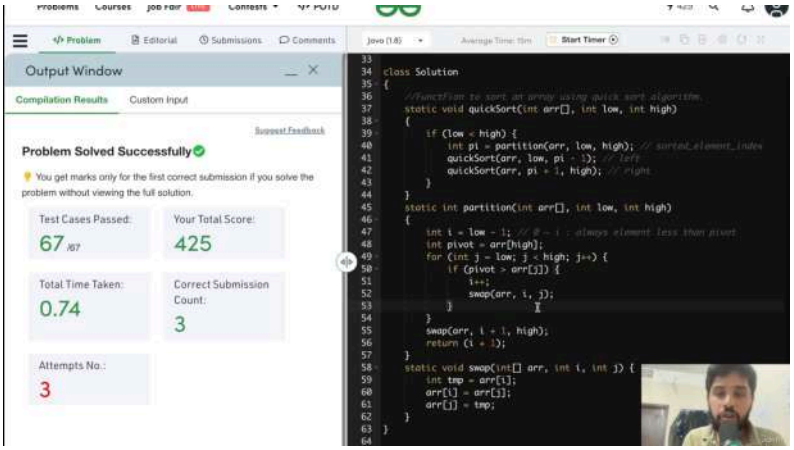
```java

class Solution {
    public int maxSubArray(int[] nums) {
        int curr = nums[0];
        int max = nums[0];
        for (int i = 1; i < nums.length; i++) {
            curr = Math.max(nums[i], nums[i] + curr);
            max = Math.max(max, curr);
        }
        return max;
    }
} // TC: O(n), SC: O(1)
```

50. Boyer-Moore Voting Algorithm (Majority Element- LeetCode 169) [Asked by Amazon]

51. Merge Sort Algorithm [Asked by Amazon, Microsoft]

52. Count Inversions -- Using Merge Sort [Asked by Amazon, Microsoft]

53. Quick Sort Algorithm [Asked by Amazon, Microsoft]



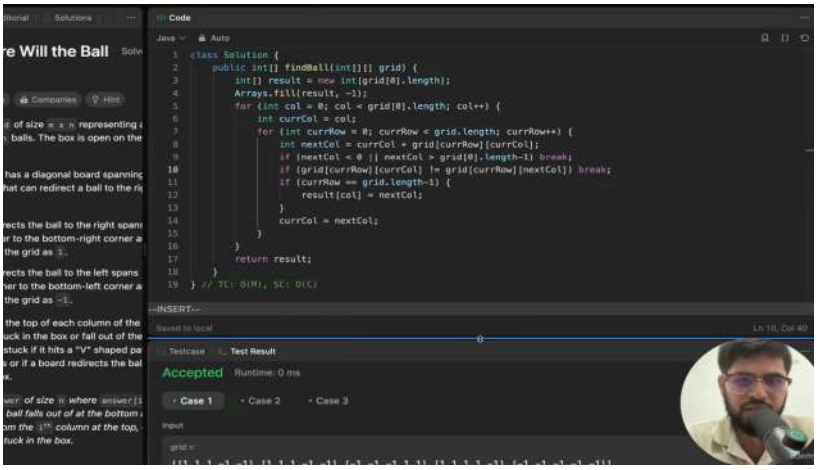
54. Kth Largest/Smallest -- Quick Select (LeetCode 215) [Asked by Google, Microsoft]

55. [OLD] Quick Sort ::Sort an Array (LeetCode 912)

56. Set Matrix Zeroes (LeetCode 73) [Asked by Facebook]

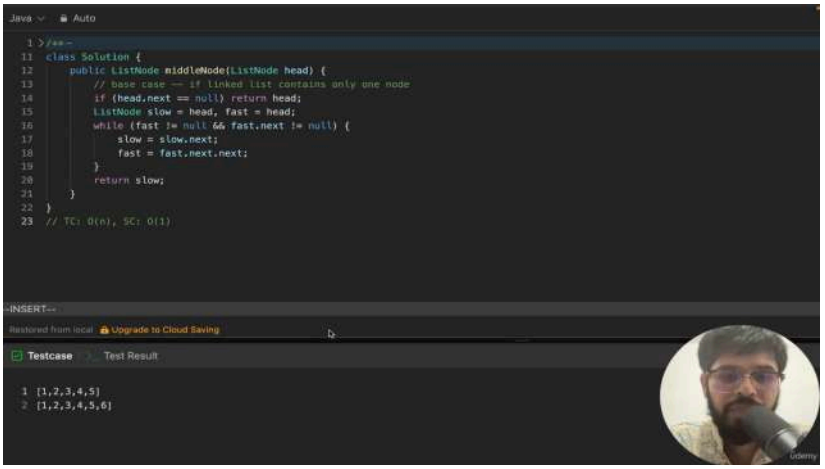
57. Range Sum Query 2D - Immutable (LeetCode 304) [Asked by Amazon]

58. Where Will the Ball Fall (LeetCode 1706) [Asked by Amazon]



59. Rotate Image (LeetCode 48) [Asked by Microsoft]

60.Middle of the Linked List



60. Delete Node in a Linked List (LeetCode 237) [Asked by Amazon,Microsoft]

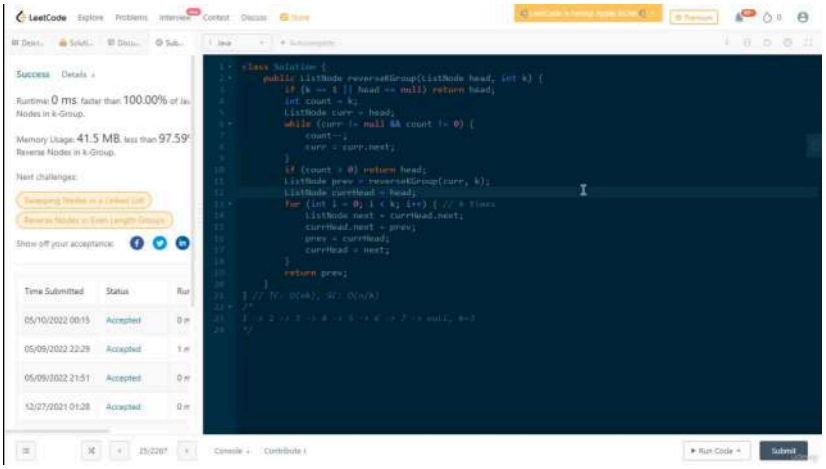
```java
 * }F
 */
class Solution {
    public void deleteNode(ListNode node) {
        node.val = node.next.val;
        node.next = node.next.next;
    }
}
```

61. Merge Two Sorted Lists (LeetCode 21) [Asked by Microsoft]

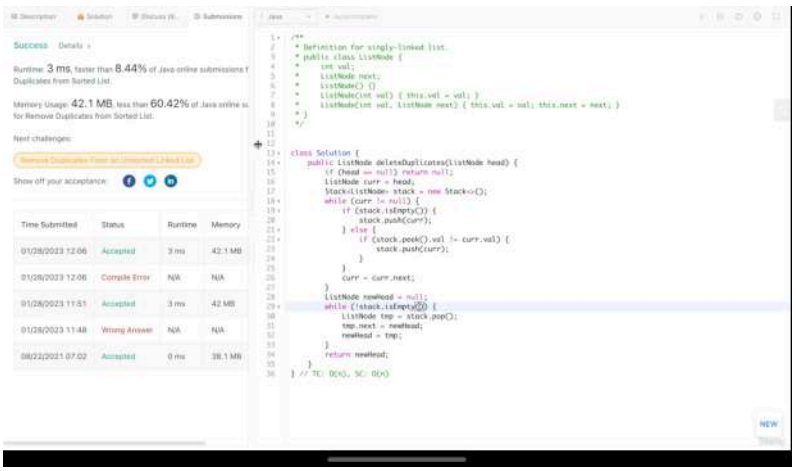
```java
class Solution {
    public ListNode mergeTwoLists(ListNode list1, ListNode list2) {
        ListNode newHead = new ListNode(-1);
        ListNode p1 = list1, p2 = list2, curr = newHead;
        while (p1 != null && p2 != null) {
            if (p1.val < p2.val) {
                curr.next = p1;
                curr = curr.next;
                p1 = p1.next;
            } else {
                curr.next = p2;
                curr = curr.next;
                p2 = p2.next;
            }
        }
        if (p1 == null) curr.next = p2;
        if (p2 == null) curr.next = p1;
        return newHead.next;
    }
} // TC: O(m + n), SC: O(1)
```

62. Reverse Linked List ::4 Ways:: (LeetCode 206) [Asked by Google, Facebook]

63. Reverse Nodes in k-group (LeetCode 25) [Asked by Microsoft]



64. Remove Duplicates from Sorted List ::O(N):: (LeetCode 83) [Asked by Microsoft]



65. Remove Duplicates from Sorted List ::O(1):: (LeetCode 83) [Asked by Microsoft]

```
 */
class Solution {
    public ListNode deleteDuplicates(ListNode head) {
        if (head == null) return null;
        ListNode dummyHead = new ListNode(-1);
        dummyHead.next = head;
        ListNode curr = head, prev = dummyHead;
        while (curr != null) {
            if (curr.next != null && curr.val == curr.next.val) {
                while (curr.next != null && curr.val == curr.next.val) {
                    curr = curr.next;
                }
                prev.next = curr;
                prev = curr;
            } else {
                prev = curr;
            }
            curr = curr.next;
        }
        return dummyHead.next;
    }
} // TC: O(n), SC: O(1)
```

66. Rotate List (LeetCode 61) [Asked by Microsoft]

```
public class Solution {
    public ListNode rotateRight(ListNode head, int k) {
        if (head == null) return null;
        int size = 0;
        ListNode curr = head;
        while (curr.next != null) {
            size++;
            curr = curr.next;
        }
        size += 1;
        curr.next = head; // tail -> head
        k = k % size;
        curr = head;
        for (int i = 1; i < size - k; i++) {
            curr = curr.next;
        }
        ListNode newHead = curr.next;
        curr.next = null;
        return newHead;
    }
} // TC: O(n), SC: O(1)
```

67. Reorder List (LeetCode 143) [Asked by Amazon,LinkedIn,Microsoft]

68. Palindrome Linked List (LeetCode 234) [Asked by Microsoft]

69. Code (Resource)

70. Tree Traversal Techniques (Level Order, Zigzag, Preorder, Inorder, Postorder)

71. Level Order Traversal Binary Tree (LeetCode 102) [Asked by Amazon, Microsoft]

```java
 * }
 */
class Solution {
    public List<List<Integer>> levelOrder(TreeNode root) {
        List<List<Integer>> traversal = new ArrayList<>();
        if (root == null) return traversal;
        Queue<TreeNode> queue = new LinkedList<>();
        queue.add(root);
        while (!queue.isEmpty()) {
            int size = queue.size();
            List<Integer> currLevel = new ArrayList<>();
            while (size != 0) {
                TreeNode tmp = queue.poll();
                currLevel.add(tmp.val);
                if (tmp.left != null) queue.add(tmp.left);
                if (tmp.right != null) queue.add(tmp.right);
                size--;
            }
            traversal.add(currLevel);
        }
        return traversal;
    }
} // TC: O(n), SC: O(n)
```

Testcase | Run Code Result | Debugger

Accepted Runtime: 0 ms

Your input [3,9,20,null,null,15,7]

72. Binary Tree Zigzag Level Order Traversal (LeetCode 103) [Asked by Microsoft]

```java
 */
class Solution {
    public List<List<Integer>> zigzagLevelOrder(TreeNode root) {
        List<List<Integer>> traversal = new ArrayList<>();
        if (root == null) return traversal;
        Queue<TreeNode> queue = new LinkedList<>();
        queue.add(root);
        int level = 0;
        while (!queue.isEmpty()) {
            int size = queue.size();
            List<Integer> currLevel = new ArrayList<>();
            while (size != 0) {
                TreeNode tmp = queue.poll();
                if (level % 2 == 0) currLevel.add(tmp.val);
                else if (level % 2 == 1) currLevel.add(0, tmp.val);
                if (tmp.left != null) queue.add(tmp.left);
                if (tmp.right != null) queue.add(tmp.right);
                size--;
            }
            level++;
            traversal.add(currLevel);
        }
        return traversal;
    }
} // TC: O(n), SC: O(n)
```

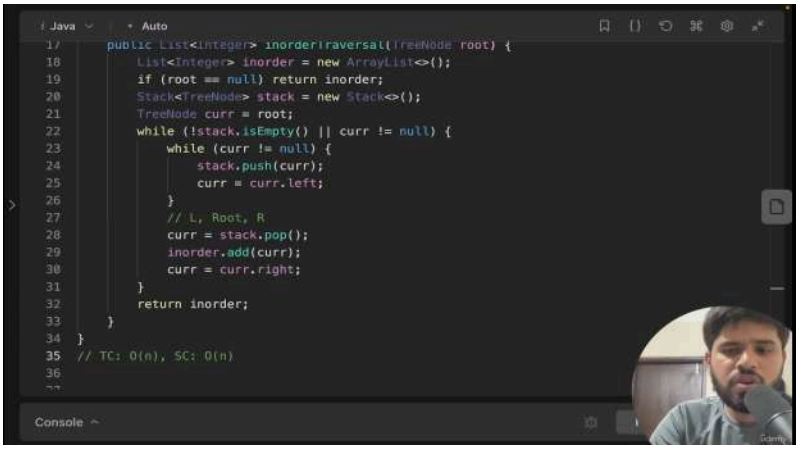
73. Binary Tree Preorder Traversal (LeetCode 144) [Asked by Microsoft]

Java | Auto

```java
 * }
 */
class Solution {
    public List<Integer> preorderTraversal(TreeNode root) {
        List<Integer> preorder = new ArrayList<>();
        if (root == null) return preorder;
        Stack<TreeNode> stack = new Stack<>();
        stack.push(root);
        TreeNode curr = root;
        while (!stack.isEmpty()) {
            curr = stack.pop();
            while (curr != null) {
                preorder.add(curr.val);
                if (curr.right != null)
                    stack.push(curr.right);
                curr = curr.left;
            }
        }
        return preorder;
    }
}
// TC: O(n), SC: O(n)
```

Console

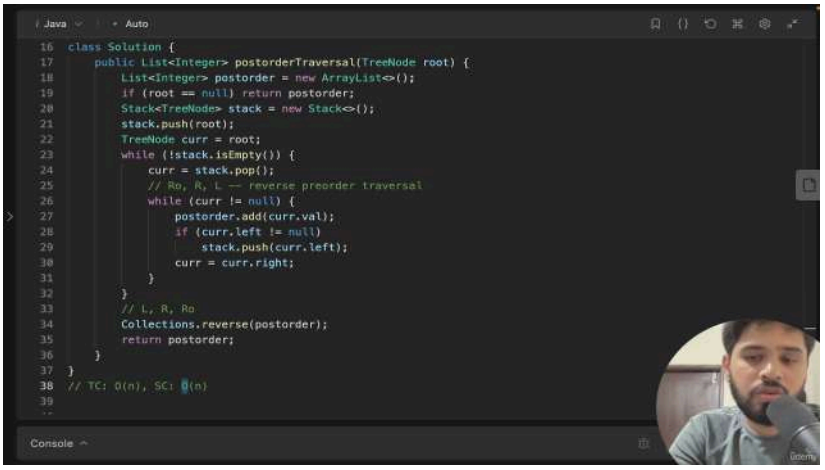
74. Binary Tree Inorder Traversal (LeetCode 94) [Asked by Microsoft]



```java
    public List<Integer> inorderTraversal(TreeNode root) {
        List<Integer> inorder = new ArrayList<>();
        if (root == null) return inorder;
        Stack<TreeNode> stack = new Stack<>();
        TreeNode curr = root;
        while (!stack.isEmpty() || curr != null) {
            while (curr != null) {
                stack.push(curr);
                curr = curr.left;
            }
            // L, Root, R
            curr = stack.pop();
            inorder.add(curr);
            curr = curr.right;
        }
        return inorder;
    }
}
// TC: O(n), SC: O(n)
```

It's through an error.
Hint: inorder.add(curr.val)

75. Binary Tree Post Order Traversal (LeetCode 145) [Asked by Microsoft]



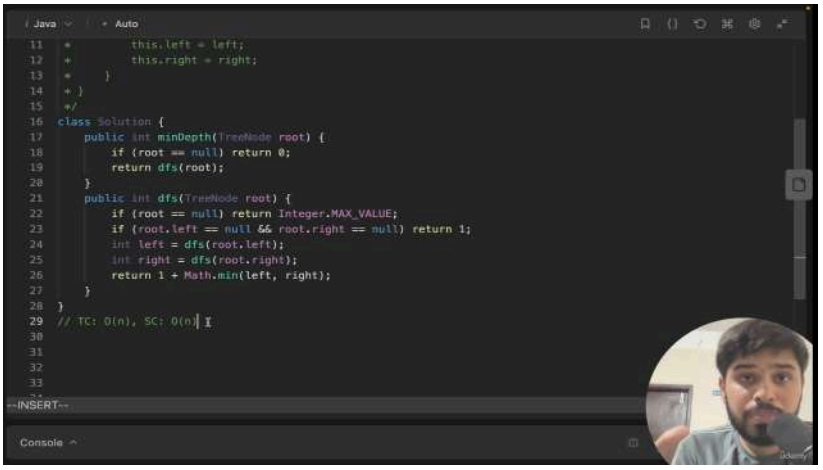
```java
class Solution {
    public List<Integer> postorderTraversal(TreeNode root) {
        List<Integer> postorder = new ArrayList<>();
        if (root == null) return postorder;
        Stack<TreeNode> stack = new Stack<>();
        stack.push(root);
        TreeNode curr = root;
        while (!stack.isEmpty()) {
            curr = stack.pop();
            // Ro, R, L -- reverse preorder traversal
            while (curr != null) {
                postorder.add(curr.val);
                if (curr.left != null)
                    stack.push(curr.left);
                curr = curr.right;
            }
        }
        // L, R, Ro
        Collections.reverse(postorder);
        return postorder;
    }
}
// TC: O(n), SC: O(n)
```

76. [Morris Traversal] Preorder Traversal (LeetCode 144) [Asked by Microsoft]

77. [Morris Traversal] Inorder Traversal (LeetCode 94) [Asked by Microsoft]

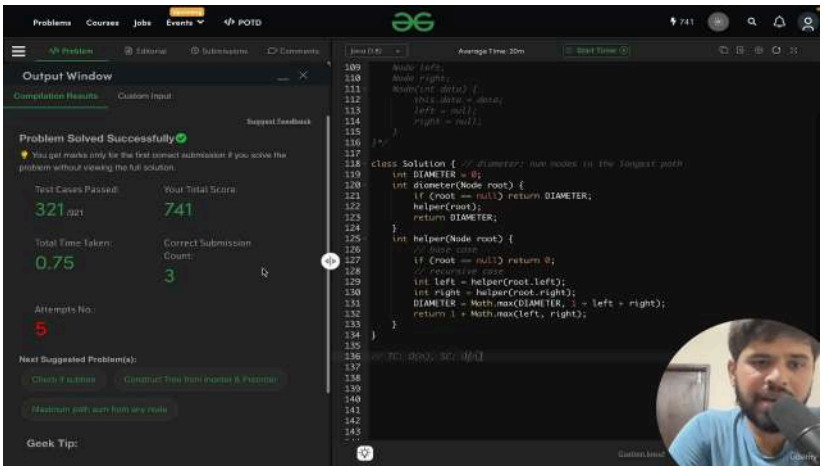
```java
    public List<Integer> inorderTraversal(TreeNode root) {
        List<Integer> inorder = new ArrayList<>();
        if (root == null) return inorder;
        TreeNode curr = root;
        while (curr != null) {
            if (curr.left == null) { // L, Ro, R
                inorder.add(curr.val);
                curr = curr.right;
            } else {
                TreeNode predecessor = curr.left; // rightMostNode
                while (predecessor.right != null && predecessor.right != curr) {
                    predecessor = predecessor.right;
                }
                if (predecessor.right == null) { // we have to create a thread
                    predecessor.right = curr;
                    curr = curr.left;
                } else { // we have a tread
                    predecessor.right = null;
                    inorder.add(curr.val);
                    curr = curr.right;
                }
            }
        }
        return inorder;
    }
}
// TC: O(n), SC: O(1)
```

78. Minimum Depth of Binary Tree (LeetCode 111) [Asked by Meta, Amazon, Microsoft]

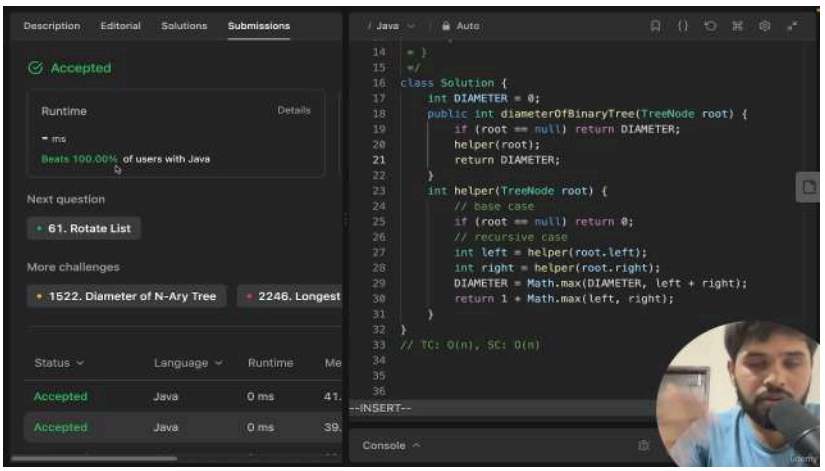


79. Maximum Depth of Binary Tree (LeetCode 104) [Asked by Meta, Microsoft, Amazon]

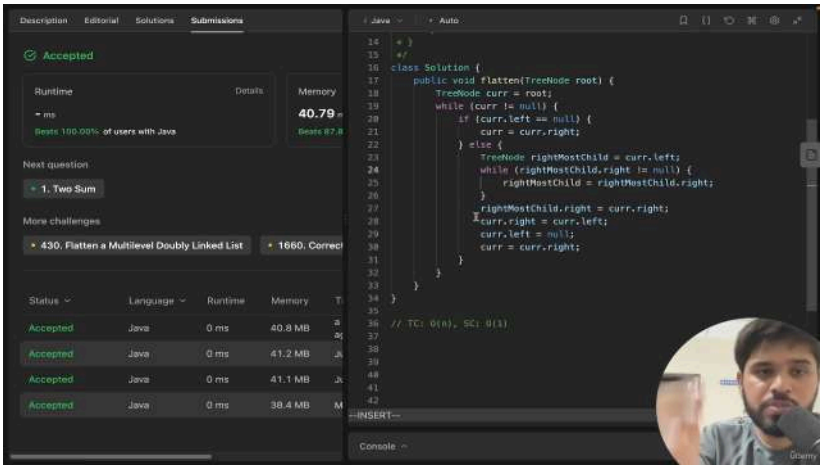
80. Diameter of Binary Tree (LeetCode 543) [Asked by Microsoft, Amazon]
Gfg:



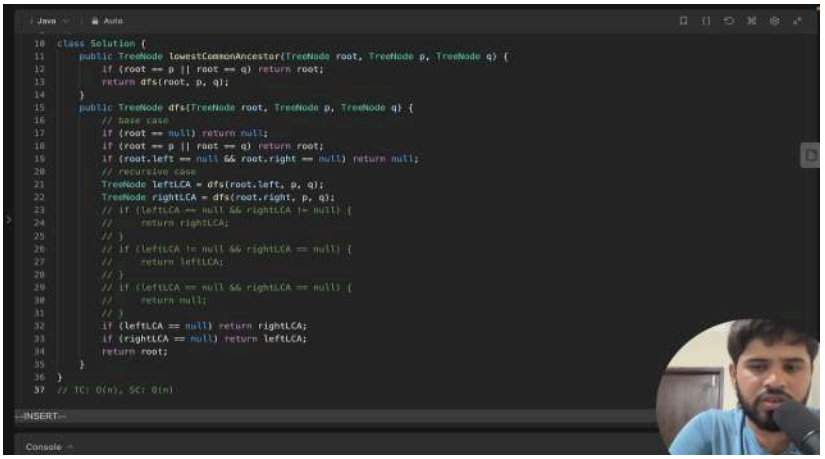
Leetcode:



81. Flatten Binary Tree to Linked List (LeetCode 114) [Asked by Microsoft]



82. Lowest Common Ancestor of Binary Tree (LeetCode 236) [Asked by Google,Microsoft]



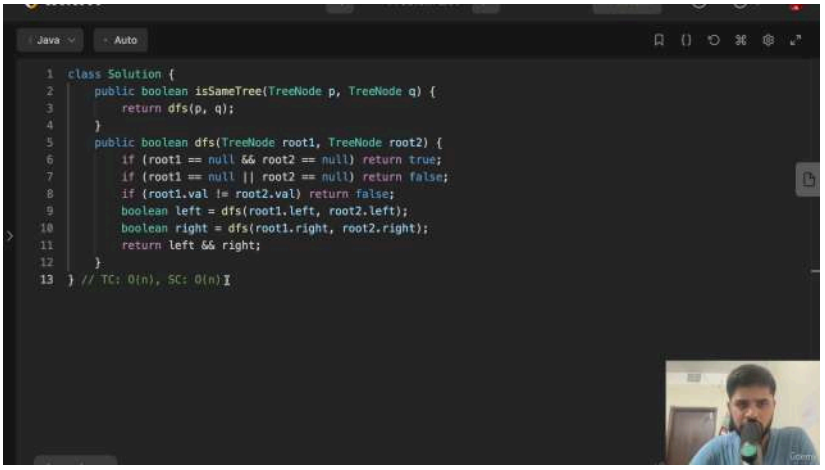
83. Populating Next Right Pointers in Each Node (LeetCode 116) [Asked by Meta]

```java
class Solution {
    public Node connect(Node root) {
        if (root == null) return null;
        Node levelStart = root;
        while (levelStart != null) {
            Node curr = levelStart;
            while (curr != null) {
                if (curr.left != null) {
                    curr.left.next = curr.right;
                    if (curr.next != null) {
                        curr.right.next = curr.next.left;
                    }
                }
                curr = curr.next;
            }
            levelStart = levelStart.left;
        }
        return root;
    }
}
// TC: O(n), SC: O(1)
```

84. Populating Next Right Pointers in Each Node II (LeetCode 117) [Asked by Meta]

```java
class Solution {
    public Node connect(Node root) {
        if (root == null) return null;
        Node levelStart = root;
        while (levelStart != null) {
            Node curr = levelStart; // travel through current level
            Node dummy = new Node(0); // to keep track the first node in the next level
            Node prev = dummy; // to connect node in the next level
            while (curr != null) {
                if (curr.left != null) {
                    prev.next = curr.left;
                    prev = prev.next;
                }
                if (curr.right != null) {
                    prev.next = curr.right;
                    prev = prev.next;
                }
                curr = curr.next;
            }
            levelStart = dummy.next;
            dummy.next = null;
        }
        return root;
    }
}
// TC: O(n), SC: O(1)
```

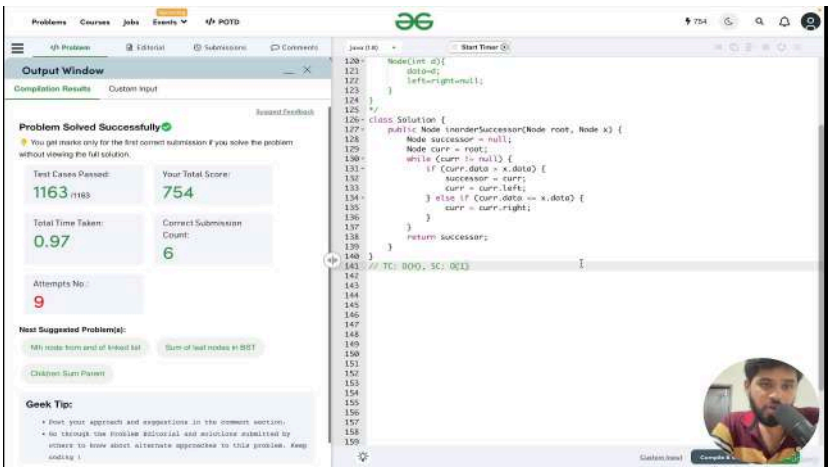
85. Same Tree (LeetCode 100) [Asked by Amazon,Microsoft]



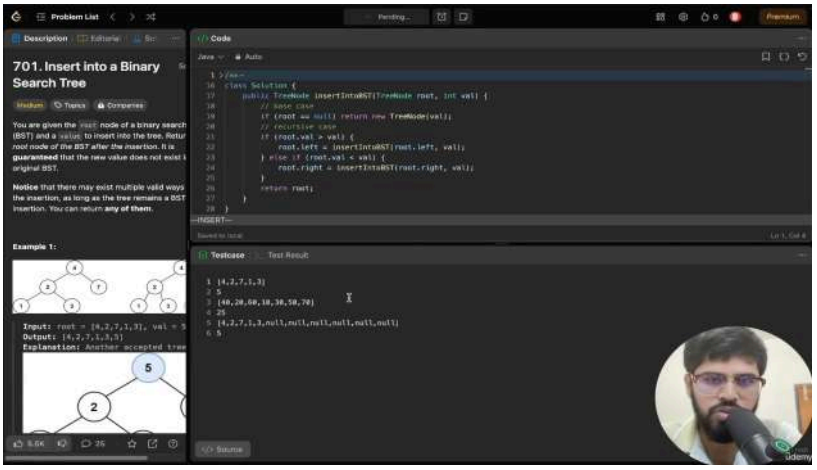
```java
class Solution {
    public boolean isSameTree(TreeNode p, TreeNode q) {
        return dfs(p, q);
    }
    public boolean dfs(TreeNode root1, TreeNode root2) {
        if (root1 == null && root2 == null) return true;
        if (root1 == null || root2 == null) return false;
        if (root1.val != root2.val) return false;
        boolean left = dfs(root1.left, root2.left);
        boolean right = dfs(root1.right, root2.right);
        return left && right;
    }
} // TC: O(n), SC: O(n)
```

86. Flip/Invert Binary Tree (LeetCode 226) [Asked Google, Amazon]

87. Inorder Successor of BST (LeetCode 285) [Asked by Microsoft/Amazon]



88. Insert into a Binary Search Tree (LeetCode 701) [Asked by Amazon, Microsoft]



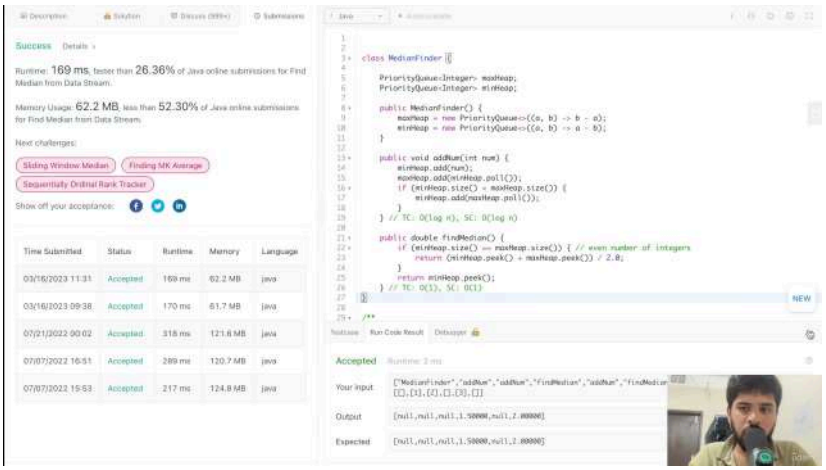
89. Sorted to Binary Search Tree (LeetCode 108) [Asked by Microsoft, Meta]

90. Construct BST from Preorder Traversal (LeetCode 1008) [Asked by Amazon]

91. Kth Largest Element in an Array (LeetCode 215) [Asked by Microsoft, Amazon]

```java
class Solution {
    public int findKthLargest(int[] nums, int k) {
        PriorityQueue<Integer> minHeap = new PriorityQueue<>();
        for (int i = 0; i < k; i++) {
            minHeap.add(nums[i]);
        }
        for (int i = k; i < nums.length; i++) {
            if (minHeap.peek() < nums[i]) {
                minHeap.poll();
                minHeap.add(nums[i]);
            }
        }
        return minHeap.peek();
    }
} // TC: O(k + (n-k)*log(k)), SC: O(k)
```

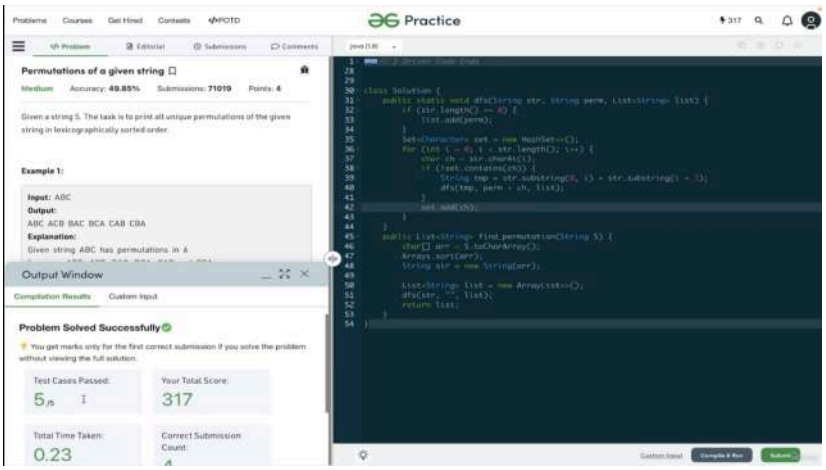
92. Find Median in a Data Stream



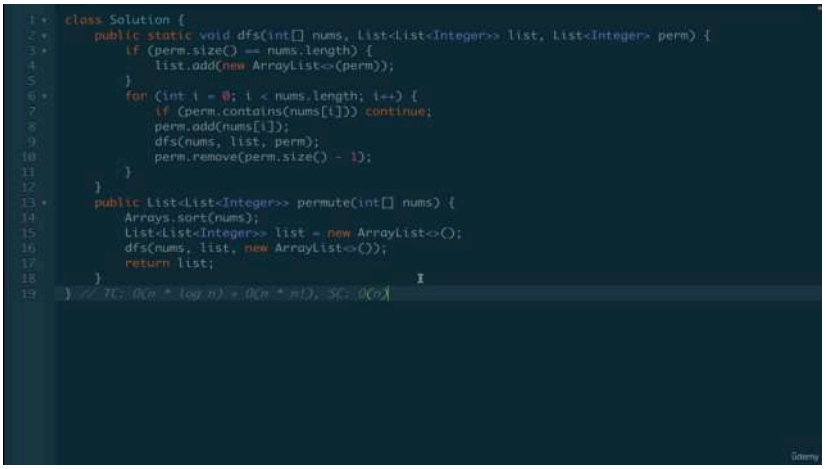
93. Implement Trie (LeetCode 208) :: HashMap :: [Asked by Amazon, Microsoft]

94. String Permutation - (Non Duplicates) [Asked by Google]

95. String Permutation - (Duplicates) [Asked by Google]



96. Permutations -- Array (LeetCode 46) [Asked by Google]



97. Combinations (LeetCode 77) [Asked by Google/Facebook/Amazon]

```java
class Solution {
    public static void dfs(List<List<Integer>> list, List<Integer> comb, int n, int k, int start) {
        if (comb.size() == k) {
            list.add(new ArrayList<>(comb));
            return;
        }
        for (int i = start; i <= n; i++) {
            comb.add(i);
            dfs(list, comb, n, k, i + 1);
            comb.remove(comb.size() - 1);
        }
    }
    public List<List<Integer>> combine(int n, int k) {
        List<List<Integer>> list = new ArrayList<>();
        dfs(list, new ArrayList<>(), n, k, 1);
        return list;
    }
}
```

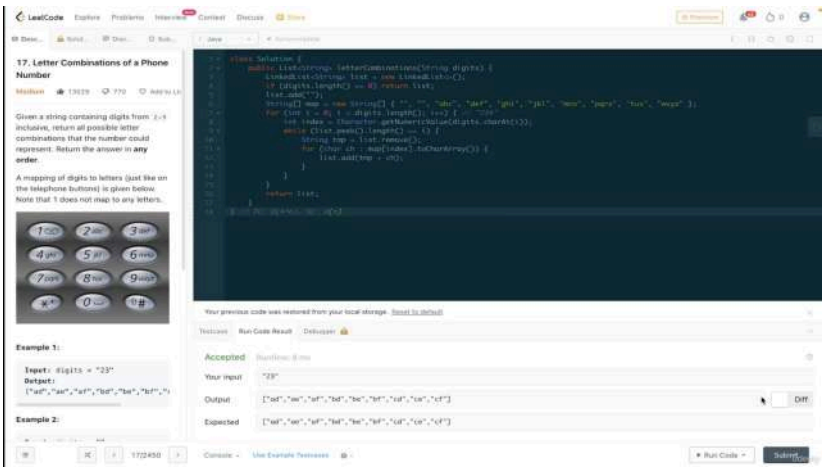
98. Combination Sum (LeetCode 39) [Asked by Google]

```java
class Solution {
    public static void dfs(int[] nums, List<List<Integer>> list, List<Integer> comb, int target, int sum, int start
        if (sum == target) {
            list.add(new ArrayList<>(comb));
            return;
        } else if (sum > target) return;

        for (int i = start; i < nums.length; i++) {
            comb.add(nums[i]);
            dfs(nums, list, comb, target, sum + nums[i], i);
            comb.remove(comb.size() - 1);
        }
    }
    public List<List<Integer>> combinationSum(int[] candidates, int target) {
        List<List<Integer>> list = new ArrayList<>();
        dfs(candidates, list, new ArrayList<>(), target, 0, 0);
        return list;
    }
}
```

Run Code Result Debugger

99. Generate Parentheses (LeetCode 22) [Asked Facebook,Microsoft]

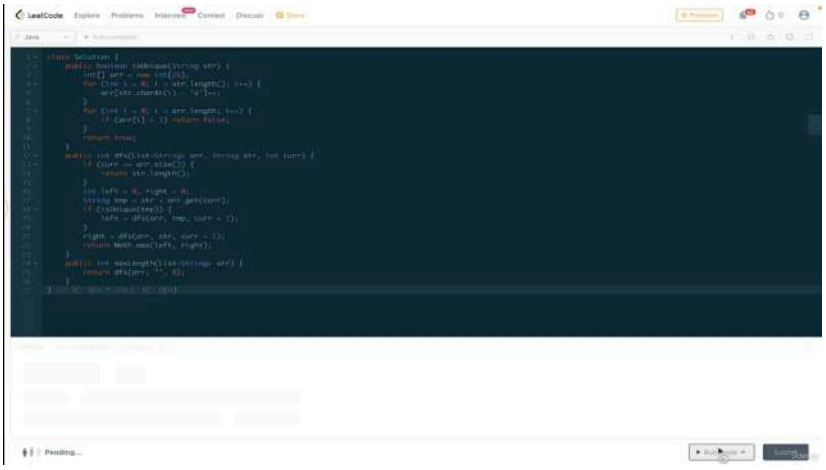
100. Letter Combinations of a Phone Number (LeetCode 17) [Asked by Google]



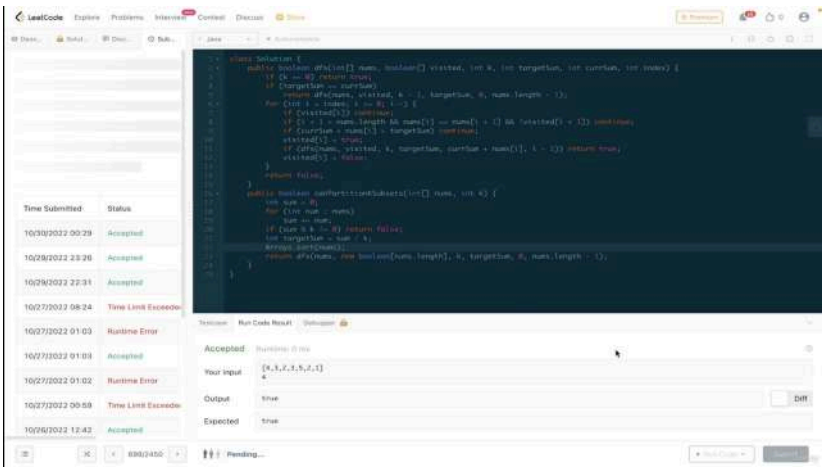
101. Splitting a String Into Descending Consecutive Values (1849) [Asked by Google]

```
class Solution {
    public boolean dfs(List<Long> list, String str, int start) {
        if (start == str.length()) {
            return list.size() >= 2;
        }
        long num = 0;
        for (int i = start; i < str.length(); i++) {
            num = num * 10 + str.charAt(i) - '0';
            if (list.size() == 0 || list.get(list.size() - 1) - num == 1) {
                list.add(num);
                if (dfs(list, str, i + 1)) return true;
                list.remove(list.size() - 1);
            }
        }
        return false;
    }
    public boolean splitString(String s) {
        return dfs(new ArrayList<>(), s, 0);
    }
} // TC: O(2^n), SC: O(n)
```

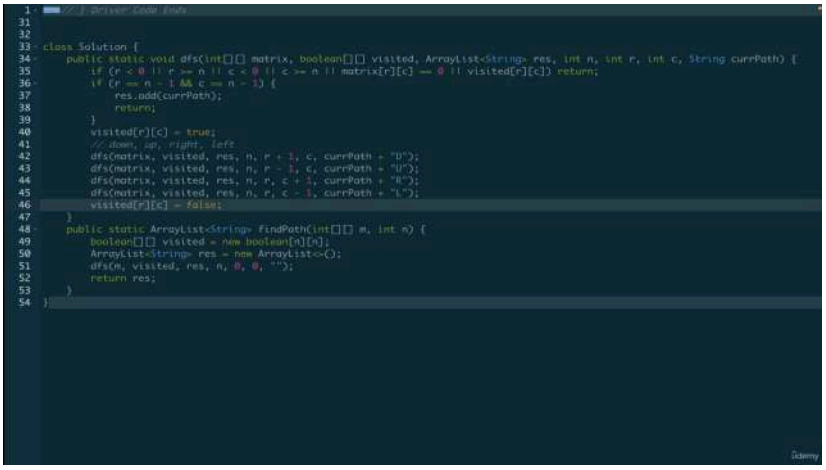
102. Max Len of a Concatenated Str with Unique Char (LeetCode 1239) [Asked by Google]



103. Partition to K Equal Sum Subsets (LeetCode 698) [Asked by Microsoft]

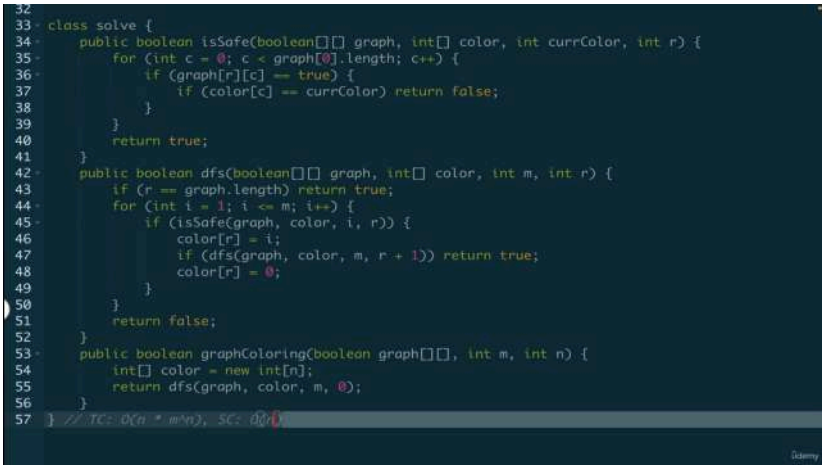


104. Rat in a Maze [Asked by Amazon/Microsoft]



```java
class Solution {
    public static void dfs(int[][] matrix, boolean[][] visited, ArrayList<String> res, int n, int r, int c, String currPath) {
        if (r < 0 || r >= n || c < 0 || c >= n || matrix[r][c] == 0 || visited[r][c]) return;
        if (r == n - 1 && c == n - 1) {
            res.add(currPath);
            return;
        }
        visited[r][c] = true;
        // down, up, right, left
        dfs(matrix, visited, res, n, r + 1, c, currPath + "D");
        dfs(matrix, visited, res, n, r - 1, c, currPath + "U");
        dfs(matrix, visited, res, n, r, c + 1, currPath + "R");
        dfs(matrix, visited, res, n, r, c - 1, currPath + "L");
        visited[r][c] = false;
    }
    public static ArrayList<String> findPath(int[][] m, int n) {
        boolean[][] visited = new boolean[n][n];
        ArrayList<String> res = new ArrayList<>();
        dfs(m, visited, res, n, 0, 0, "");
        return res;
    }
}
```

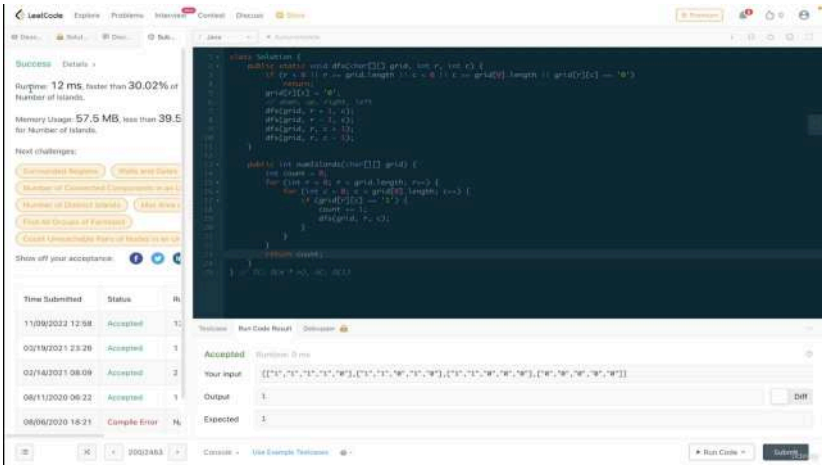
105. M-Coloring [Asked by Amazon]



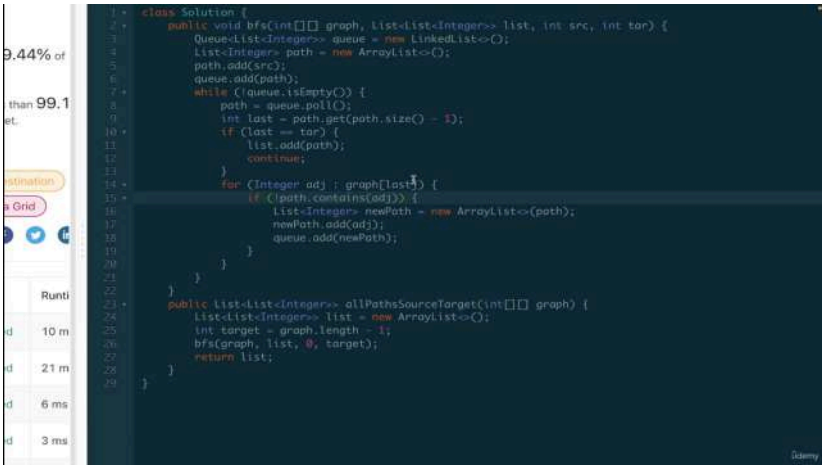
```java
class solve {
    public boolean isSafe(boolean[][] graph, int[] color, int currColor, int r) {
        for (int c = 0; c < graph[0].length; c++) {
            if (graph[r][c] == true) {
                if (color[c] == currColor) return false;
            }
        }
        return true;
    }
    public boolean dfs(boolean[][] graph, int[] color, int m, int r) {
        if (r == graph.length) return true;
        for (int i = 1; i <= m; i++) {
            if (isSafe(graph, color, i, r)) {
                color[r] = i;
                if (dfs(graph, color, m, r + 1)) return true;
                color[r] = 0;
            }
        }
        return false;
    }
    public boolean graphColoring(boolean graph[][], int m, int n) {
        int[] color = new int[n];
        return dfs(graph, color, m, 0);
    }
} // TC: O(n * m^n), SC: O(n)
```

106. BFS & DFS Traversal of Graph [Asked by Samsung]

107. Number of Islands [Asked by Microsoft, Apple] (LeetCode 200)

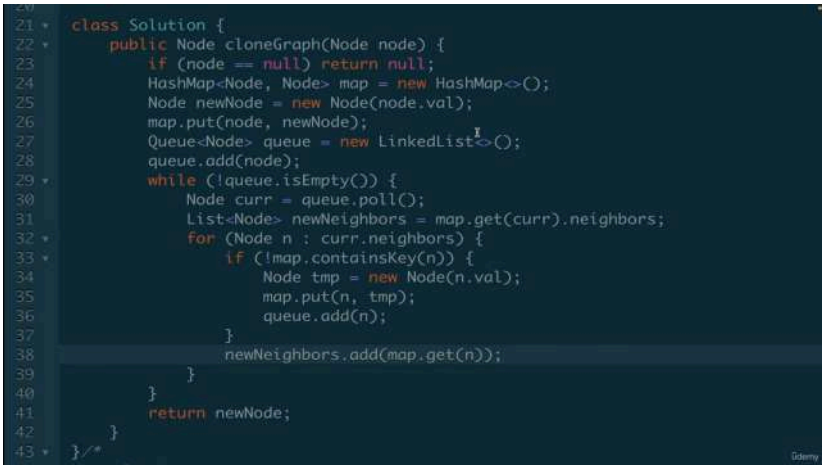


108. All Paths From Source to Target (LeetCode 797) [Asked by Google, Bloomberg]



```
class Solution {
    public void bfs(int[][] graph, List<List<Integer>> list, int src, int tar) {
        Queue<List<Integer>> queue = new LinkedList<>();
        List<Integer> path = new ArrayList<>();
        path.add(src);
        queue.add(path);
        while (!queue.isEmpty()) {
            path = queue.poll();
            int last = path.get(path.size() - 1);
            if (last == tar) {
                list.add(path);
                continue;
            }
            for (Integer adj : graph[last]) {
                if (!path.contains(adj)) {
                    List<Integer> newPath = new ArrayList<>(path);
                    newPath.add(adj);
                    queue.add(newPath);
                }
            }
        }
    }
    public List<List<Integer>> allPathsSourceTarget(int[][] graph) {
        List<List<Integer>> list = new ArrayList<>();
        int target = graph.length - 1;
        bfs(graph, list, 0, target);
        return list;
    }
}
```

109. Clone a Graph [Asked by Facebook] (LeetCode 133)

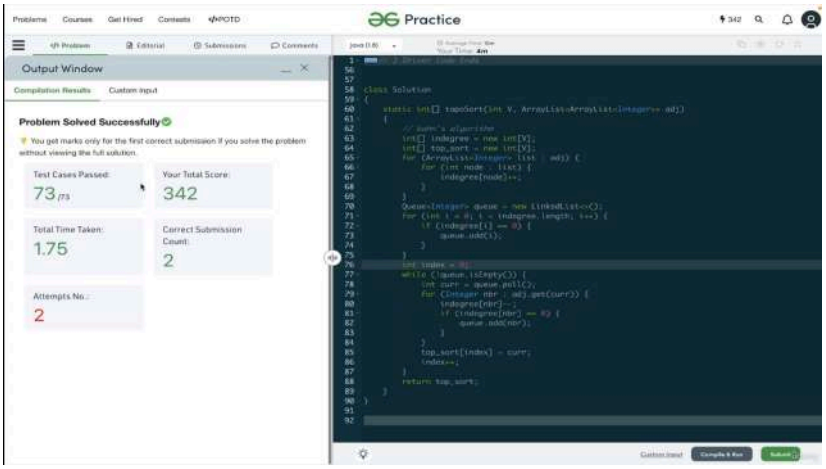


```
class Solution {
    public Node cloneGraph(Node node) {
        if (node == null) return null;
        HashMap<Node, Node> map = new HashMap<>();
        Node newNode = new Node(node.val);
        map.put(node, newNode);
        Queue<Node> queue = new LinkedList<>();
        queue.add(node);
        while (!queue.isEmpty()) {
            Node curr = queue.poll();
            List<Node> newNeighbors = map.get(curr).neighbors;
            for (Node n : curr.neighbors) {
                if (!map.containsKey(n)) {
                    Node tmp = new Node(n.val);
                    map.put(n, tmp);
                    queue.add(n);
                }
                newNeighbors.add(map.get(n));
            }
        }
        return newNode;
    }
}/*
```

110. Barpartite Graph [Asked by Facebook, Microsoft] (LeetCode 785)

```java
class Solution {
    public boolean BFS(int[][] graph, int[] colors, int node) {
        Queue<Integer> queue = new LinkedList<>();
        queue.add(node);
        colors[node] = 1; // 1, -1
        while (!queue.isEmpty()) {
            int curr = queue.poll();
            for (int adj : graph[curr]) {
                if (colors[adj] == 0) {
                    colors[adj] = -1 * colors[curr];
                    queue.add(adj);
                } else if (colors[curr] == colors[adj]) {
                    return false;
                }
            }
        }
        return true;
    }
    public boolean isBipartite(int[][] graph) {
        int v = graph.length;
        int[] colors = new int[v];
        for (int node = 0; node < v; node++) {
            if (colors[node] == 0) {
                if (BFS(graph, colors, node) == false) {
                    return false;
                }
            }
        }
        return true;
    }
} // TC: O(V + E), SC: O(V)
```

111. Topological Sort -- Kahn's Algorithm -- (Asked by Amazon,Microsoft)



112. Fibonacci Number (LeetCode 509) [Asked by Amazon, Microsoft]

```java
class Solution {
    public int fib(int n) {
        if (n <= 1) return n;
        int first = 0, second = 1;
        for (int i = 2; i <= n; i++) {
            int tmp = first + second;
            first = second;
            second = tmp;
        }
        return second;
    }
} // TC: O(n), SC: O(1)
```

113. Climbing Stairs (LeetCode 70) [Asked by Google, Microsoft, Amazon]

```java
class Solution {
    public int climbStairs(int n) {
        if (n <= 2) return n;
        int[] dp = new int[n + 1];
        dp[1] = 1;
        dp[2] = 2;
        for (int i = 3; i <= n; i++) {
            dp[i] = dp[i - 1] + dp[i - 2];
        }
        return dp[n];
    }
} // TC: O(n), SC: O(n)
```

Constant space complexity:

```java
class Solution {
    public int climbStairs(int n) {
        if (n <= 2) return n;
        int first = 1, second = 2;
        for (int i = 1; i <= n-2; i++) {
            int tmp = first + second;
            first = second;
            second = tmp;
        }
        return second;
    }
} // TC: O(n), SC: O(1)
```

114. Unique Path (LeetCode 62) [Asked by Amazon, Microsoft]

115. Partition Equal Subset Sum (LeetCode 416) [Asked by Google, Microsoft]

```java
class Solution{
    static int equalPartition(int N, int arr[]) {
        int total = Arrays.stream(arr).sum();
        if (total % 2 == 1) return 0;

        int target = total / 2;
        boolean[] dp = new boolean[target + 1];
        dp[0] = true;

        for (int i = 0; i < arr.length; i++) {
            for (int curr = dp.length - 1; curr >= 1; curr--) {
                if (curr >= arr[i]) {
                    dp[curr] = dp[curr] || dp[curr - arr[i]];
                }
            }
        }
        return dp[target] == true ? 1 : 0;
    }
} // TC: O(n * target), SC: O(target)
```

116. House Robber (LeetCode 198) [Asked by Google, Amazon]
Brute force solution:

```java
class Solution {
    public int rob(int[] nums) {
        if (nums.length == 1) return nums[0];
        int[] dp = new int[nums.length + 1];
        dp[0] = nums[0];
        dp[1] = Math.max(nums[0], nums[1]);
        for (int i = 2; i < nums.length; i++) {
            int rob = nums[i] + dp[i - 2];
            int notRob = dp[i - 1];
            dp[i] = Math.max(rob, notRob);
        }
        return dp[nums.length - 1];
    }
} // TC: O(n), SC: O(n) - O(1)
```

Optimal Solution:

```java
class Solution {
    public int rob(int[] nums) {
        if (nums.length == 1) return nums[0];
        int prev1 = nums[0];
        int prev2 = Math.max(nums[0], nums[1]);
        for (int i = 2; i < nums.length; i++) {
            int curr = Math.max(nums[i] + prev1, prev2);
            prev1 = prev2;
            prev2 = curr;
        }
        return prev2;
    }
} // TC: O(n), SC: O(1)
```

117. Coin Change (LeetCode 322) [Asked by Amazon,Microsoft]

```java
class Solution {
    public int coinChange(int[] coins, int amount) {
        int[] dp = new int[amount + 1];
        for (int i = 1; i <= amount; i++) {
            int min = Integer.MAX_VALUE;
            for (int j = 0; j < coins.length; j++) {
                if (i >= coins[j] && dp[i - coins[j]] != -1) {
                    min = Math.min(min, dp[i - coins[j]]);
                }
            }
            dp[i] = (min == Integer.MAX_VALUE) ? -1 : 1 + min;
        }
        return dp[amount];
    }
} // TC: (n * amount), SC: O(amount)
```

118. Decode Ways (LeetCode 91) [Asked by Meta,Microsoft,Amazon]

```java
class Solution {
    public int numDecodings(String s) {
        if (s.charAt(0) == '0') return 0;
        int n = s.length();

        int p = 1, pp = 1;
        for (int i = n - 1; i >= 0; i--) {
            int curr = s.charAt(i) == '0' ? 0 : p;
            if (i < n - 1 && (s.charAt(i) == '1' || s.charAt(i) == '2' && s.charAt(i + 1) <= '6'))
                curr += pp;
            }
            pp = p;
            p = curr;
        }

        return p;
    }
} // TC: O(n), SC: O(1)
```

Hint : if(i<n-1 && (s.charAt(i) == '1' ||.....'6')){

119. Word Break (LeetCode 139) [Asked by Microsoft]

```java
class Solution {
    public boolean wordBreak(String s, List<String> wordDict) {
        boolean[] dp = new boolean[s.length() + 1];
        dp[0] = true;
        Set<String> set = new HashSet<>(wordDict);
        for (int i = 1; i <= s.length(); i++) { // n
            for (int j = 0; j < i; j++) { // n
                String suffix = s.substring(j, i); // n
                if (set.contains(suffix) && dp[j] == true) {
                    dp[i] = true;
                    break;
                }
            }
        }
        return dp[s.length()];
    }
}

// TC: O(n^3 + m), SC: O(n + m)
```

Testcase | Run Code Result | Debugger

120. Edit Distance (LeetCode 72) [Asked by Google, Amazon, Microsoft]

```java
class Solution {
    public int minDistance(String word1, String word2) {
        int m = word1.length(), n = word2.length();
        if (m == 0) return n;
        if (n == 0) return m;
        int[][] dp = new int[m + 1][n + 1];
        for (int i = 0; i <= m; i++) dp[i][0] = i; // row
        for (int i = 0; i <= n; i++) dp[0][i] = i; // column
        for (int i = 1; i <= m; i++) {
            for (int j = 1; j <= n; j++) {
                char c1 = word1.charAt(i - 1);
                char c2 = word2.charAt(j - 1);
                if (c1 == c2) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    // insert, delete, replace
                    int insert = dp[i][j - 1];
                    int delete = dp[i - 1][j];
                    int replace = dp[i - 1][j - 1];
                    dp[i][j] = 1 + Math.min(insert, Math.min(delete, replace));
                }
            }
        }
        return dp[m][n];
    }
} // TC: O(m * n), SC: O(m * n)
```

121. Interleaving String (LeetCode 97) [Asked by Google, Microsoft, Amazon]

```java
class Solution {
    public boolean isInterleave(String s1, String s2, String s3) {
        int n1 = s1.length(), n2 = s2.length(), n3 = s3.length();
        if (n1 + n2 != n3) return false;
        boolean[][] dp = new boolean[n1 + 1][n2 + 1];
        dp[n1][n2] = true;
        for (int i = n1; i >= 0; i--) {
            for (int j = n2; j >= 0; j--) {
                if (i < n1 && s1.charAt(i) == s3.charAt(i + j) && dp[i + 1][j] == true) {
                    dp[i][j] = true;
                }
                if (j < n2 && s2.charAt(j) == s3.charAt(i + j) && dp[i][j + 1] == true) {
                    dp[i][j] = true;
                }
            }
        }
        return dp[0][0];
    }
}

// TC: O(S*T), SC: O(S*T)
```

122. Longest Increasing Subsequence(LeetCode 300) [Asked by Google, Microsoft]

```java
class Solution {
    public int lengthOfLIS(int[] nums) {
        int n = nums.length;
        if (n == 1) return 1;
        int[] dp = new int[n];
        dp[0] = 1;
        int ans = 0;
        for (int i = 1; i < n; i++) {
            int len = 0;
            for (int j = 0; j < i; j++) {
                if (nums[j] < nums[i]) {
                    len = Math.max(len, dp[j]);
                }
            }
            dp[i] = 1 + len;
            ans = Math.max(ans, dp[i]);
        }
        return ans;
    }
} // TC: O(n^2), SC: O(n)
```

123. Longest Common Subsequence (LeetCode 1143)

```java
class Solution {
    public int longestCommonSubsequence(String text1, String text2) {
        int[][] dp = new int[text1.length() + 1][text2.length() + 1];
        for (int i = 1; i < dp.length; i++) {
            for (int j = 1; j < dp[0].length; j++) {
                if (text1.charAt(i - 1) == text2.charAt(j - 1)) {
                    dp[i][j] = 1 + dp[i-1][j-1];
                } else {
                    dp[i][j] = Math.max(dp[i][j-1], dp[i-1][j]);
                }
            }
        }
        return dp[text1.length()][text2.length()];
    }
} // TC: O(m * n), SC: O(m * n)
```

124. Longest Palindromic Subsequence (LeetCode 516) [Asked by Google, Amazon]

usercode > main.java

```java
import java.util.*;

public class Main{
    public static int dfs(String s, int start, int end, int[][] dp) {
        // base case
        if (start == end) return 1;
        if (start > end) return 0;
        if (dp[start][end] != -1) return dp[start][end];
        // recursive case
        if (s.charAt(start) == s.charAt(end)) {
            return 2 + dfs(s, start + 1, end - 1, dp);
        }
        int left = dfs(s, start + 1, end, dp);
        int right = dfs(s, start, end - 1, dp);
        return dp[start][end] = Math.max(left, right);
    }
    public static int longestPalindromicSubsequence(String s) {
        int n = s.length();
        int[][] dp = new int[n][n];
        for (int[] row : dp) {
            Arrays.fill(row, -1);
        }
        return dfs(s, 0, n - 1, dp);
    }
}
// TC: O(n^2), SC: O(n^2)
```

125. Power Set (Asked by Snapdeal)

Problem | Editorial | Submissions | Comments

Output Window

Compilation Results | Custom Input

Suggest Feedback

Problem Solved Successfully

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

Test Cases Passed:	Your Total Score:
90 /90	425
Total Time Taken: 1.95	Correct Submission Count: 6
Attempts No.: 11	

Java (1.8) | Start Timer

```java
//User function Template for Java

class Solution
{
    public List<String> AllPossibleStrings(String s)
    {
        int n = s.length();
        List<String> powerset = new ArrayList<>();
        int max = (1 << n) - 1;
        for (int i = 1; i <= max; i++) {
            String str = "";
            for (int bit = 0; bit < n; bit++) {
                if ((i & (1 << bit)) != 0) {
                    str = s.charAt(n-1-bit) + str;
                }
            }
            powerset.add(str);
        }
        Collections.sort(powerset);
        return powerset;
    }
}

// TC: O(2^n * n) + O(k log k)
// SC: O(2^n * n)
```

126. Complement of Base 10 Integer (LeetCode 1009)

127. Sum of Two Integers (LeetCode 371)

128. Divide Two Integers (LeetCode 29)