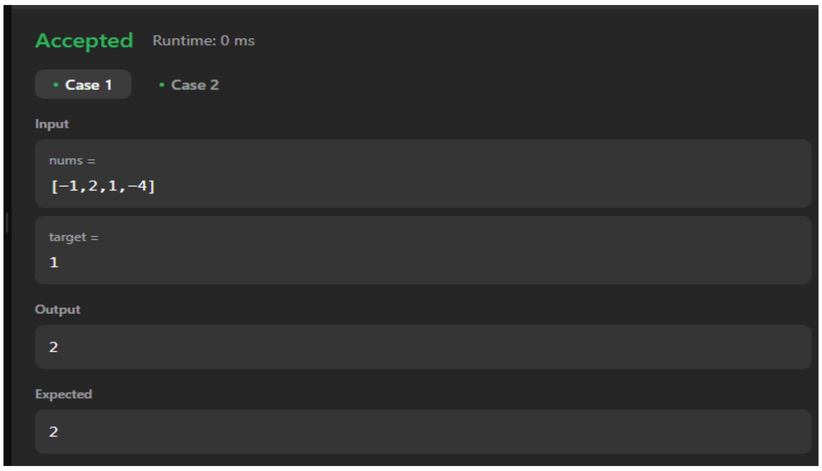
Coding practice Problems(20/11/2024)

Q 1) 3Sum Closest

```
class Solution {
    public int threeSumClosest(int[] nums, int target) {
        Arrays.sort(nums);
        int closest_sum=Integer.MAX_VALUE/2;
        for(int i=0;i<nums.length-2;++i){
            int left=i+1,right=nums.length-1;
            while(left<right){
                int current_sum=nums[i]+nums[left]+nums[right];
                if(Math.abs(current_sum-target)< Math.abs(closest_sum-target)){
                    closest_sum=current_sum;
                }
                if(current_sum<target){
                    ++left;
                }
                else if(current_sum>target){
                    --right;
                }
                else{
                    return current_sum;
                }
            }
        }
        return closest_sum;
    }
}
```

Time Complexity: O(n^2)

Output:



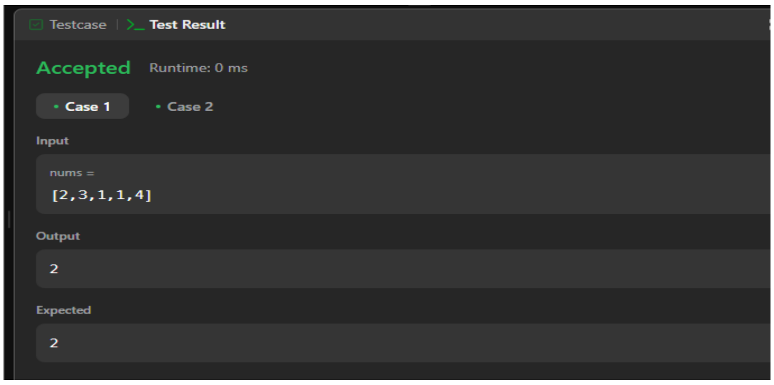
```
Accepted   Runtime: 0 ms
• Case 1   • Case 2
Input
nums =
[-1,2,1,-4]
target =
1
Output
2
Expected
2
```

Q 2) Jump Game II

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

Time Complexity: O(n)

Output:

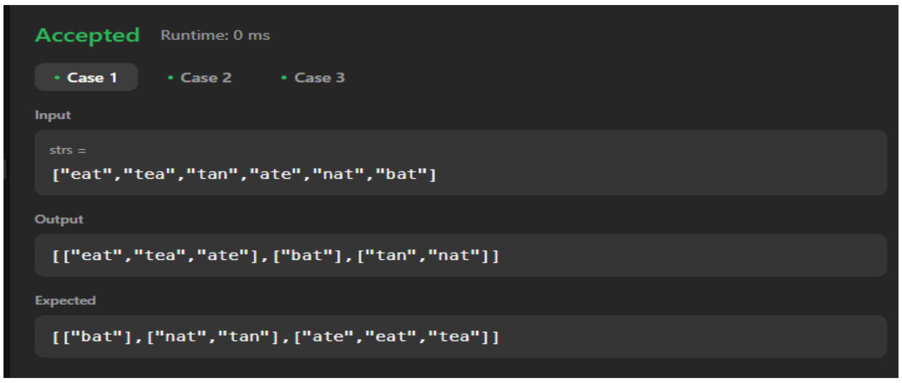


```
Testcase | >_ Test Result

Accepted   Runtime: 0 ms

• Case 1   • Case 2

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

Output
2

Expected
2
```

Q 3) Group Anagrams

```
class Solution {
    public List<List<String>> groupAnagrams(String[] strs) {
        Map<String,List<String>> ans=new HashMap<>();
        for(String s:strs){
            char[] chars=s.toCharArray();
            Arrays.sort(chars);
            String key=new String(chars);
            if(!ans.containsKey(key)){
                ans.put(key,new ArrayList<>());
            }
            ans.get(key).add(s);
        }
        return new ArrayList<>(ans.values());
    }
}
```

Output:



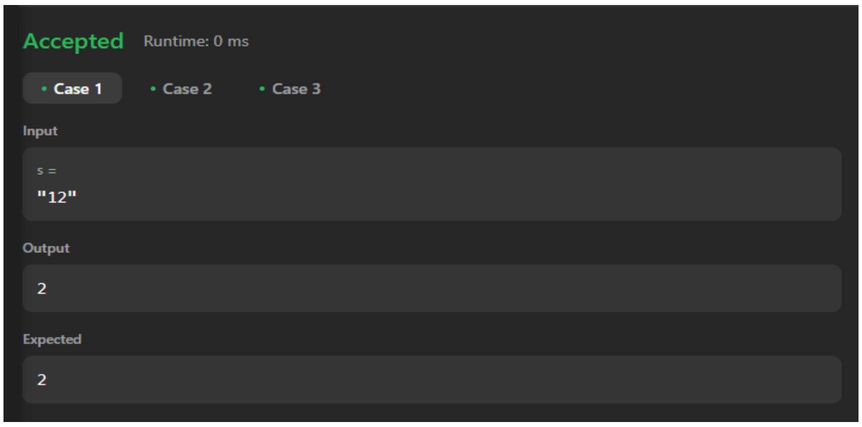
```
Accepted   Runtime: 0 ms
• Case 1   • Case 2   • Case 3
Input
strs =
["eat","tea","tan","ate","nat","bat"]
Output
[["eat","tea","ate"],["bat"],["tan","nat"]]
Expected
[["bat"],["nat","tan"],["ate","eat","tea"]]
```

Q 4) Decode Ways

```
class Solution {
    public int numDecodings(String s) {
        int strLen = s.length();
        int[] dp = new int[strLen + 1];
        dp[0] = 1;
        if (s.charAt(0) != '0') {
            dp[1] = 1;
        } else {
            return 0;
        }
        for (int i = 2; i <= strLen; ++i) {
            if (s.charAt(i - 1) != '0') {
                dp[i] += dp[i - 1];
            }
            if (s.charAt(i - 2) == '1' ||
                    (s.charAt(i - 2) == '2' && s.charAt(i - 1) <= '6')) {
                dp[i] += dp[i - 2];
            }
        }
        return dp[strLen];
    }
}
```

Time Complexity: O(n)

Output:



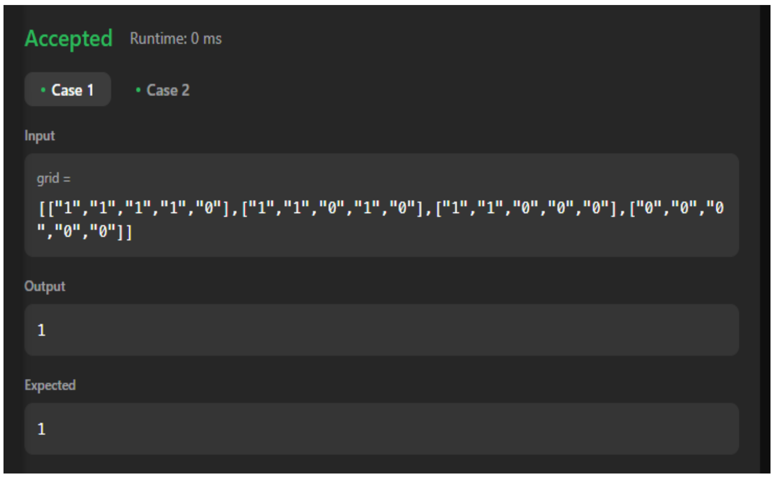
```
Accepted   Runtime: 0 ms
• Case 1   • Case 2   • Case 3
Input
s =
"12"
Output
2
Expected
2
```

Q 5) Number of Islands

```
class Solution {
    public void dfs(char[][] grid, int i, int j) {
        int m = grid.length, n = grid[0].length;
        if (i < 0 || j < 0 || i >= m || j >= n || grid[i][j] == '0') {
            return;
        }
        grid[i][j] = '0';
        dfs(grid, i + 1, j);
        dfs(grid, i - 1, j);
        dfs(grid, i, j + 1);
        dfs(grid, i, j - 1);
    }
    public int numIslands(char[][] grid) {
        int m = grid.length, n = grid[0].length, count = 0;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (grid[i][j] == '1') {
                    count++;
                    dfs(grid, i, j);
                }
            }
        }
        return count;
    }
}
```

Time Complexity: O(n*m)

Output:



Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

```
grid =
[["1","1","1","1","0"],["1","1","0","1","0"],["1","1","0","0","0"],["0","0","0","0","0"]]
```

Output

```
1
```

Expected

```
1
```