Assignment - 2

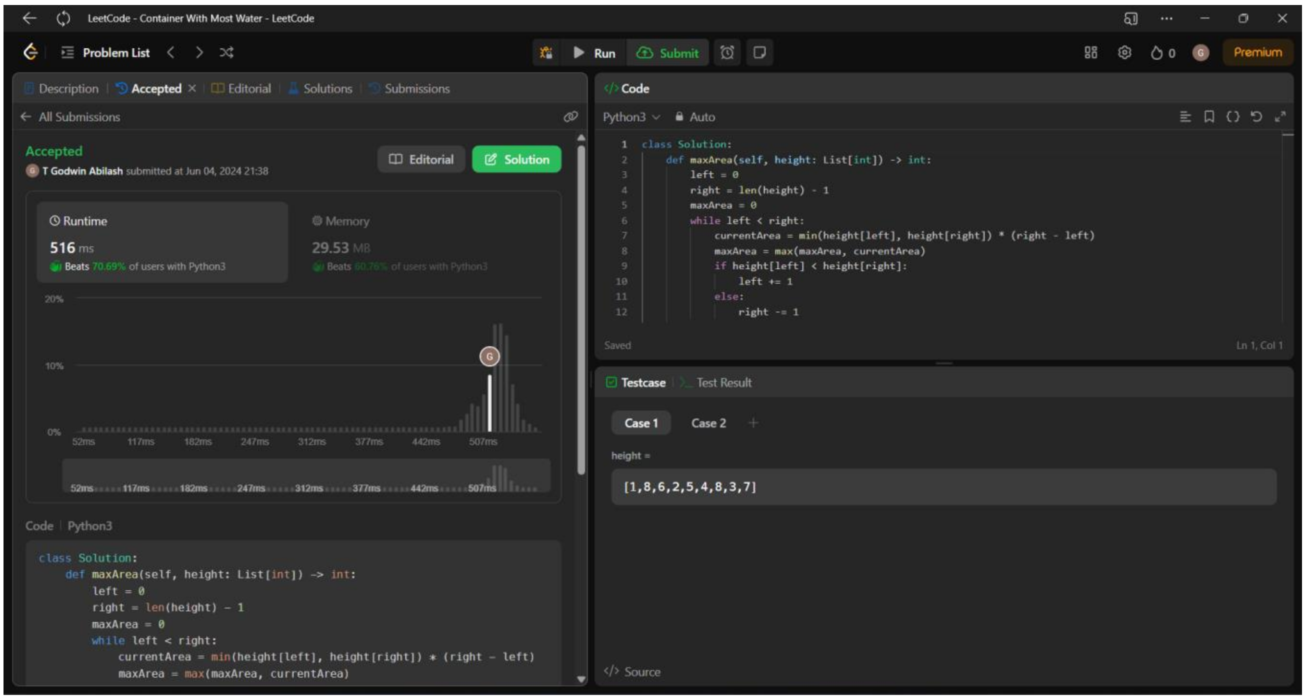
T Godwin Abilash – 192321012

11. Container With Most Water

Code:

```
class Solution:
    def maxArea(self, height: List[int]) -> int:
        left = 0
        right = len(height) - 1
        maxArea = 0
        while left < right:
            currentArea = min(height[left], height[right]) * (right
- left)
            maxArea = max(maxArea, currentArea)
            if height[left] < height[right]:
                left += 1
            else:
                right -= 1
        return maxArea
```

Screenshot:



LeetCode - Container With Most Water - LeetCode

Accepted

T Godwin Abilash submitted at Jun 04, 2024 21:38

Runtime 516 ms Beats 70.69% of users with Python3

Memory 29.53 MB Beats 60.76% of users with Python3

Code | Python3

```
class Solution:
    def maxArea(self, height: List[int]) -> int:
        left = 0
        right = len(height) - 1
        maxArea = 0
        while left < right:
            currentArea = min(height[left], height[right]) * (right - left)
            maxArea = max(maxArea, currentArea)
```

```
class Solution:
    def maxArea(self, height: List[int]) -> int:
        left = 0
        right = len(height) - 1
        maxArea = 0
        while left < right:
            currentArea = min(height[left], height[right]) * (right - left)
            maxArea = max(maxArea, currentArea)
            if height[left] < height[right]:
                left += 1
            else:
                right -= 1
```

height =

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

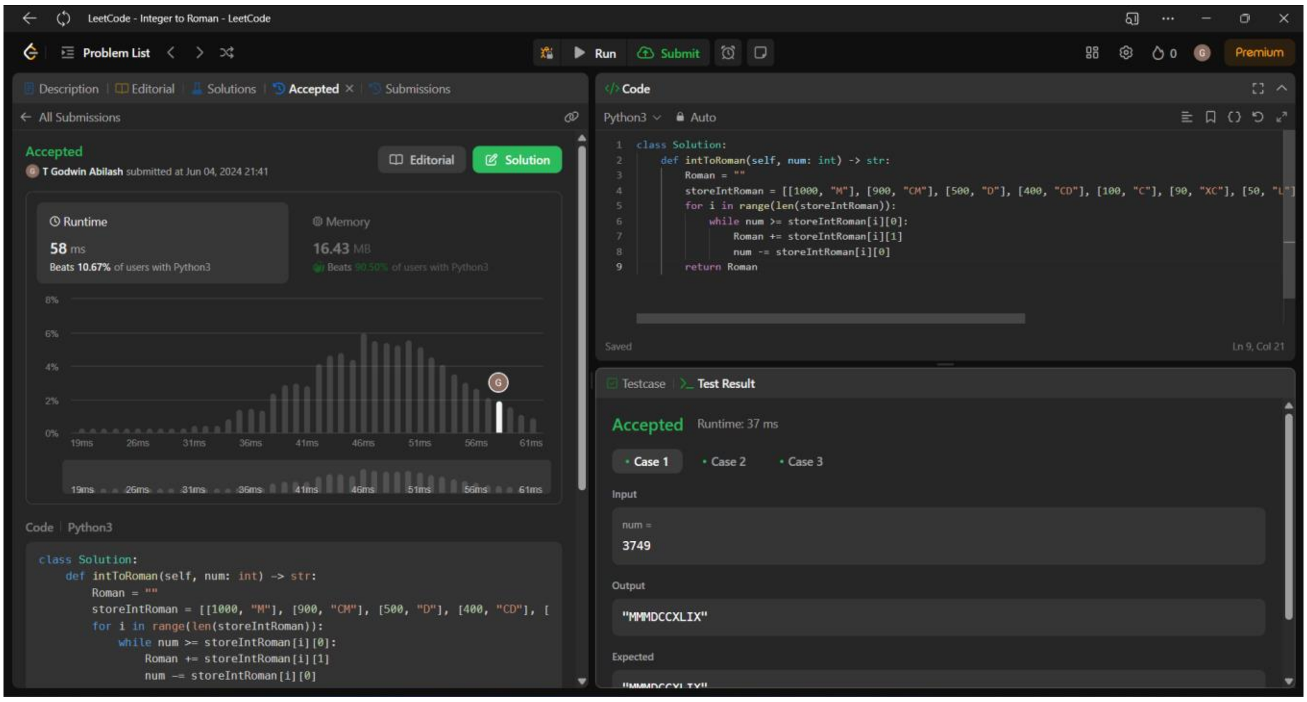
Time Complexity: O(n)

12. Integer to Roman

Code:

```
class Solution:
    def intToRoman(self, num: int) -> str:
        Roman = ""
        storeIntRoman = [[1000, "M"], [900, "CM"], [500, "D"], [400,
"CD"], [100, "C"], [90, "XC"], [50, "L"], [40, "XL"], [10, "X"], [9,
"IX"], [5, "V"], [4, "IV"], [1, "I"]]
        for i in range(len(storeIntRoman)):
            while num >= storeIntRoman[i][0]:
                Roman += storeIntRoman[i][1]
                num -= storeIntRoman[i][0]
        return Roman
```

Screenshot for I/O:


LeetCode - Integer to Roman - LeetCode
Problem List
Run
Submit
Premium
Description | Editorial | Solutions | Accepted | Submissions
All Submissions
Editorial
Solution
Code
Python3
Auto

Accepted
T Godwin Abilash submitted at Jun 04, 2024 21:41

Runtime
58 ms
Beats 10.67% of users with Python3

Memory
16.43 MB
Beats 90.50% of users with Python3

Code | Python3

```
class Solution:
    def intToRoman(self, num: int) -> str:
        Roman = ""
        storeIntRoman = [[1000, "M"], [900, "CM"], [500, "D"], [400, "CD"], [
        for i in range(len(storeIntRoman)):
            while num >= storeIntRoman[i][0]:
                Roman += storeIntRoman[i][1]
                num -= storeIntRoman[i][0]
```

```
class Solution:
    def intToRoman(self, num: int) -> str:
        Roman = ""
        storeIntRoman = [[1000, "M"], [900, "CM"], [500, "D"], [400, "CD"], [100, "C"], [90, "XC"], [50, "L"]
        for i in range(len(storeIntRoman)):
            while num >= storeIntRoman[i][0]:
                Roman += storeIntRoman[i][1]
                num -= storeIntRoman[i][0]
        return Roman
```

Saved
Ln 9, Col 21

Testcase | Test Result

Accepted Runtime: 37 ms

Case 1 · Case 2 · Case 3

Input

num =
3749

Output

"MMMDCCXLIX"

Expected

"MMMDCCXLIX"

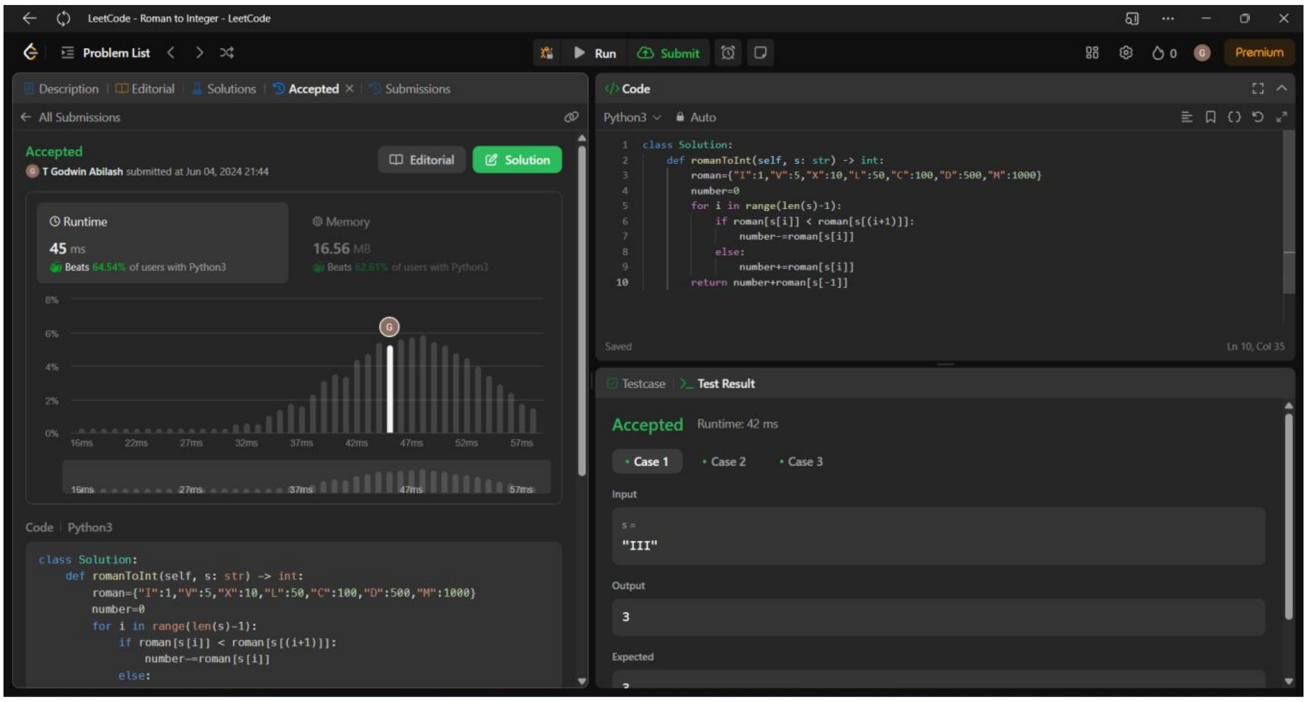
Time Complexity: O(max(m,n))

13. Roman to Integer

Code:

```
class Solution:
    def romanToInt(self, s: str) -> int:
        roman={"I":1,"V":5,"X":10,"L":50,"C":100,"D":500,"M":1000}
        number=0
        for i in range(len(s)-1):
            if roman[s[i]] < roman[s[(i+1)]]:
                number-=roman[s[i]]
            else:
                number+=roman[s[i]]
        return number+roman[s[-1]]
```

Screenshot:


LeetCode - Roman to Integer - LeetCode
Problem List
Run
Submit
Premium
Description | Editorial | Solutions | Accepted | Submissions
All Submissions
Editorial
Solution
Code
Python3
Auto
Testcase
Test Result

Accepted
T Godwin Abilash submitted at Jun 04, 2024 21:44

Runtime
45 ms
Beats 64.54% of users with Python3

Memory
16.56 MB
Beats 62.61% of users with Python3

```
class Solution:
    def romanToInt(self, s: str) -> int:
        roman={"I":1,"V":5,"X":10,"L":50,"C":100,"D":500,"M":1000}
        number=0
        for i in range(len(s)-1):
            if roman[s[i]] < roman[s[(i+1)]]:
                number-=roman[s[i]]
            else:
                number+=roman[s[i]]
        return number+roman[s[-1]]
```

Saved   Ln 10, Col 35

Accepted   Runtime: 42 ms

Case 1 · Case 2 · Case 3

Input

```
s =
"III"
```

Output

```
3
```

Expected

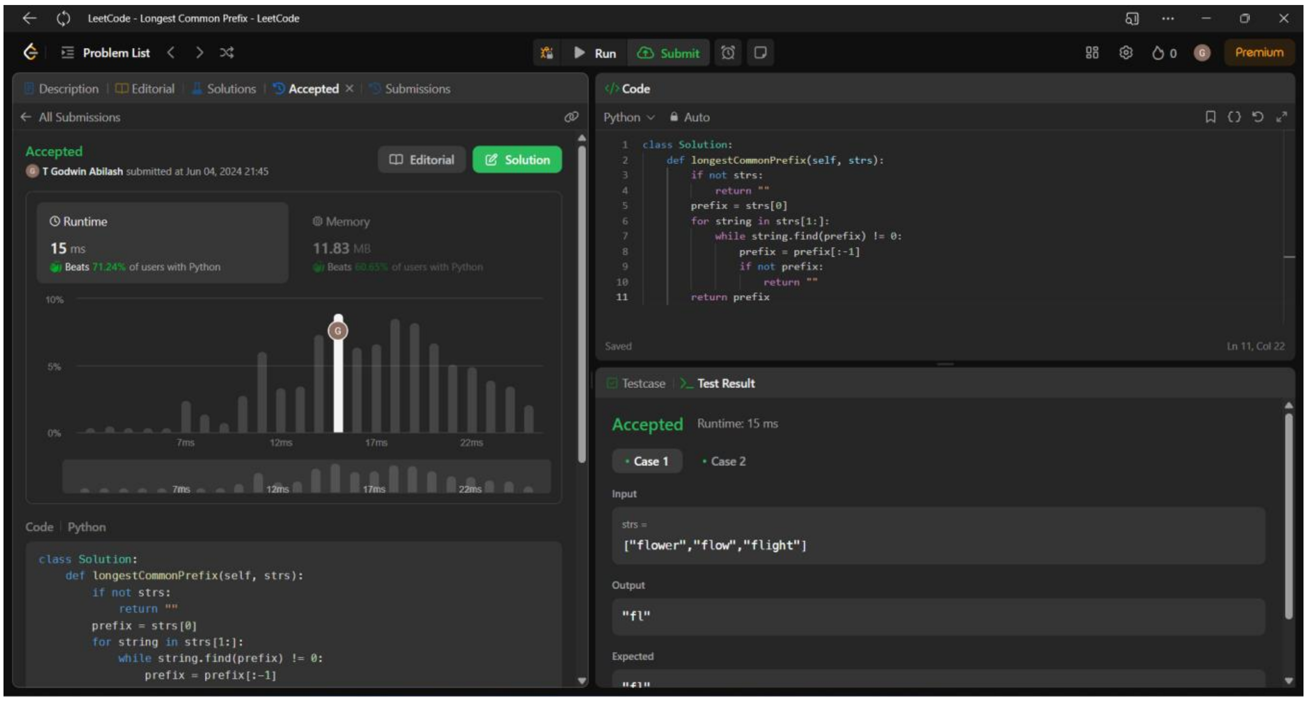
Time Complexity: O(n)

14. Longest Common Prefix

Code:

```
class Solution:
    def longestCommonPrefix(self, strs):
        if not strs:
            return ""
        prefix = strs[0]
        for string in strs[1:]:
            while string.find(prefix) != 0:
                prefix = prefix[:-1]
                if not prefix:
                    return ""
        return prefix
```

Screenshot:



Accepted
T Godwin Abilash submitted at Jun 04, 2024 21:45

Runtime
15 ms
Beats 71.24% of users with Python

Memory
11.83 MB
Beats 60.65% of users with Python

Code | Python

```
class Solution:
    def longestCommonPrefix(self, strs):
        if not strs:
            return ""
        prefix = strs[0]
        for string in strs[1:]:
            while string.find(prefix) != 0:
                prefix = prefix[:-1]
```

```
class Solution:
    def longestCommonPrefix(self, strs):
        if not strs:
            return ""
        prefix = strs[0]
        for string in strs[1:]:
            while string.find(prefix) != 0:
                prefix = prefix[:-1]
                if not prefix:
                    return ""
        return prefix
```

Saved
Ln 11, Col 22

Accepted Runtime: 15 ms

Input

```
strs =
["flower","flow","flight"]
```

Output

```
"fl"
```

Expected

```
"fl"
```

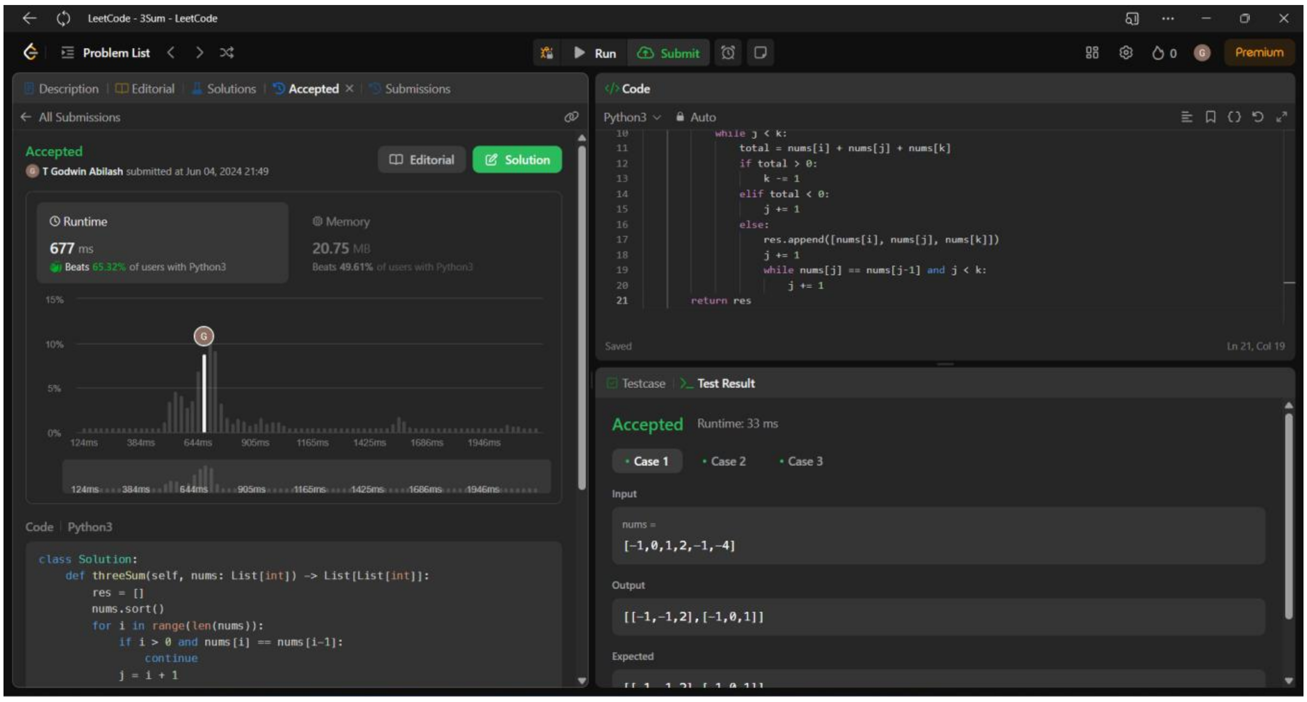
Time Complexity: O(n)

15. 3Sum

Code:

```
class Solution:
    def threeSum(self, nums: List[int]) -> List[List[int]]:
        res = []
        nums.sort()
        for i in range(len(nums)):
            if i > 0 and nums[i] == nums[i-1]:
                continue
            j = i + 1
            k = len(nums) - 1
            while j < k:
                total = nums[i] + nums[j] + nums[k]
                if total > 0:
                    k -= 1
                elif total < 0:
                    j += 1
                else:
                    res.append([nums[i], nums[j], nums[k]])
                    j += 1
                    while nums[j] == nums[j-1] and j < k:
                        j += 1
        return res
```

Screenshot:



Accepted
T Godwin Abilash submitted at Jun 04, 2024 21:49

Runtime
677 ms
Beats 65.32% of users with Python3

Memory
20.75 MB
Beats 49.61% of users with Python3

Code | Python3

```
class Solution:
    def threeSum(self, nums: List[int]) -> List[List[int]]:
        res = []
        nums.sort()
        for i in range(len(nums)):
            if i > 0 and nums[i] == nums[i-1]:
                continue
            j = i + 1
```

```
            while j < k:
                total = nums[i] + nums[j] + nums[k]
                if total > 0:
                    k -= 1
                elif total < 0:
                    j += 1
                else:
                    res.append([nums[i], nums[j], nums[k]])
                    j += 1
                    while nums[j] == nums[j-1] and j < k:
                        j += 1
        return res
```

Saved   Ln 21, Col 19

Accepted   Runtime: 33 ms

Case 1   Case 2   Case 3

Input

nums =
```
[-1,0,1,2,-1,-4]
```

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

Expected

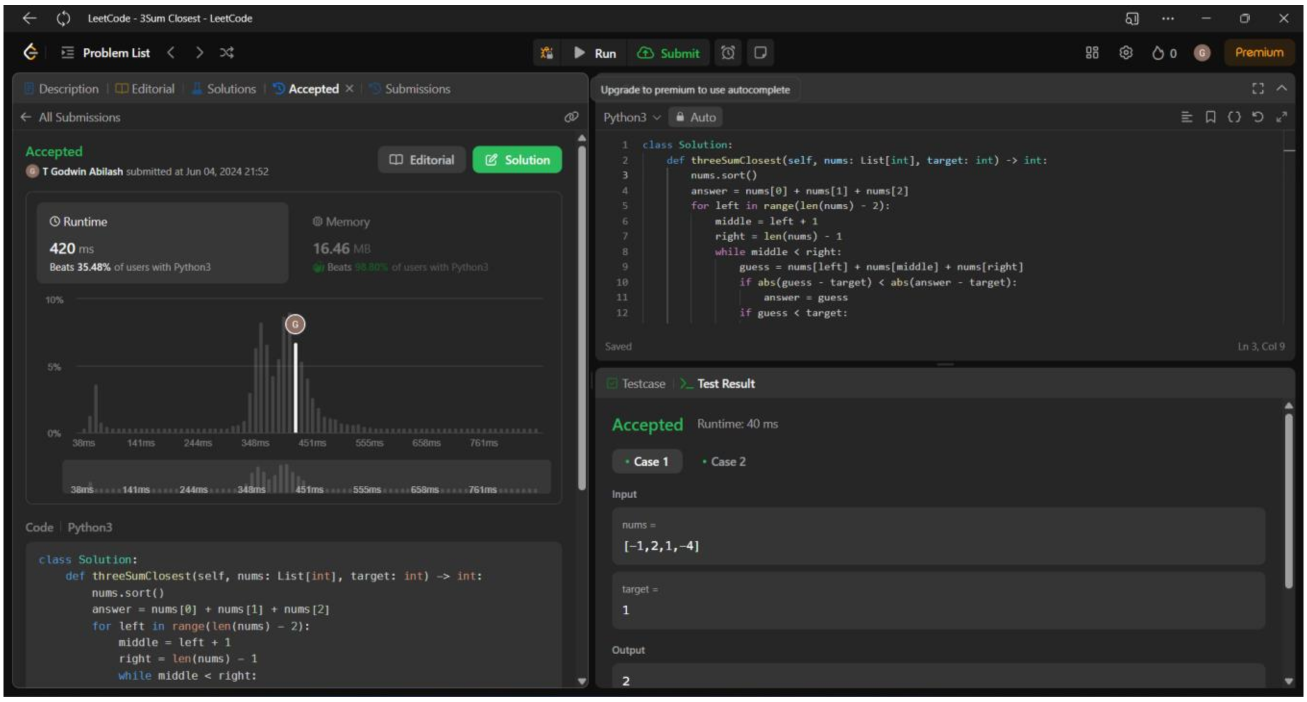
Time Complexity: O(n2)

16. 3Sum Closet

Code:

```python
class Solution:
    def threeSumClosest(self, nums: List[int], target: int) -> int:
        nums.sort()
        answer = nums[0] + nums[1] + nums[2]
        for left in range(len(nums) - 2):
            middle = left + 1
            right = len(nums) - 1
            while middle < right:
                guess = nums[left] + nums[middle] + nums[right]
                if abs(guess - target) < abs(answer - target):
                    answer = guess
                if guess < target:
                    middle += 1
                elif guess > target:
                    right -= 1
                else:
                    return target
        return answer
```

Screenshot:


LeetCode - 3Sum Closest - LeetCode

Accepted
T Godwin Abilash submitted at Jun 04, 2024 21:52

Runtime
420 ms
Beats 35.48% of users with Python3

Memory
16.46 MB
Beats 98.80% of users with Python3

Code | Python3

```python
class Solution:
    def threeSumClosest(self, nums: List[int], target: int) -> int:
        nums.sort()
        answer = nums[0] + nums[1] + nums[2]
        for left in range(len(nums) - 2):
            middle = left + 1
            right = len(nums) - 1
            while middle < right:
```

```python
class Solution:
    def threeSumClosest(self, nums: List[int], target: int) -> int:
        nums.sort()
        answer = nums[0] + nums[1] + nums[2]
        for left in range(len(nums) - 2):
            middle = left + 1
            right = len(nums) - 1
            while middle < right:
                guess = nums[left] + nums[middle] + nums[right]
                if abs(guess - target) < abs(answer - target):
                    answer = guess
                if guess < target:
```

Saved
Ln 3, Col 9

Testcase | Test Result

Accepted Runtime: 40 ms

Case 1 · Case 2

Input

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

target =
1

Output

2

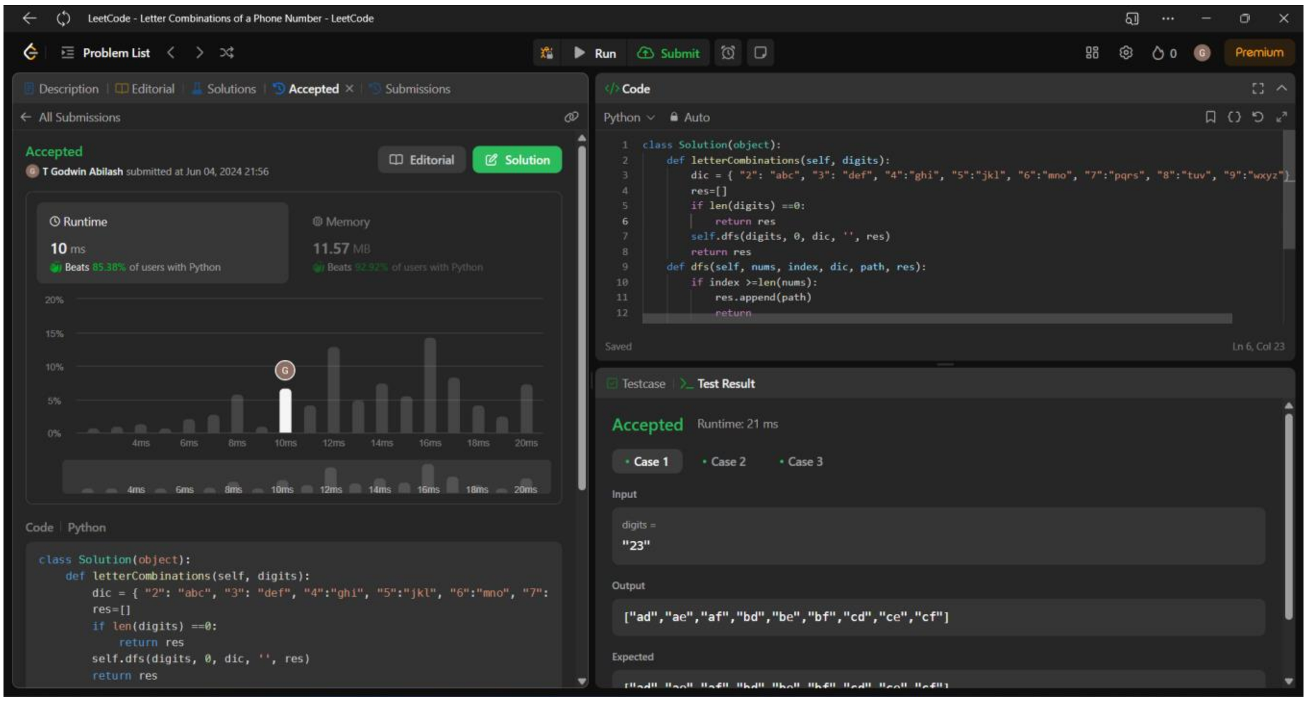
Time Complexity: O(n)

17. Letter Combinations of the phone number

Code:

```
class Solution(object):
    def letterCombinations(self, digits):
        dic = { "2": "abc", "3": "def", "4":"ghi", "5":"jkl", "6":"mno",
"7":"pqrs", "8":"tuv", "9":"wxyz"}
        res=[]
        if len(digits) ==0:
            return res
        self.dfs(digits, 0, dic, '', res)
        return res
    def dfs(self, nums, index, dic, path, res):
        if index >=len(nums):
            res.append(path)
            return
        string1 =dic[nums[index]]
        for i in string1:
            self.dfs(nums, index+1, dic, path + i, res)
```

Screenshot:



LeetCode - Letter Combinations of a Phone Number - LeetCode

Accepted
T Godwin Abilash submitted at Jun 04, 2024 21:56

Runtime
10 ms
Beats 85.38% of users with Python

Memory
11.57 MB
Beats 92.92% of users with Python

Code | Python

```
class Solution(object):
    def letterCombinations(self, digits):
        dic = { "2": "abc", "3": "def", "4":"ghi", "5":"jkl", "6":"mno", "7":
        res=[]
        if len(digits) ==0:
            return res
        self.dfs(digits, 0, dic, '', res)
        return res
```

Code

Python Auto

```
class Solution(object):
    def letterCombinations(self, digits):
        dic = { "2": "abc", "3": "def", "4":"ghi", "5":"jkl", "6":"mno", "7":"pqrs", "8":"tuv", "9":"wxyz"}
        res=[]
        if len(digits) ==0:
            return res
        self.dfs(digits, 0, dic, '', res)
        return res
    def dfs(self, nums, index, dic, path, res):
        if index >=len(nums):
            res.append(path)
            return
```

Saved   Ln 6, Col 23

Testcase | Test Result

Accepted Runtime: 21 ms

Case 1 · Case 2 · Case 3

Input

digits =
"23"

Output

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

Expected

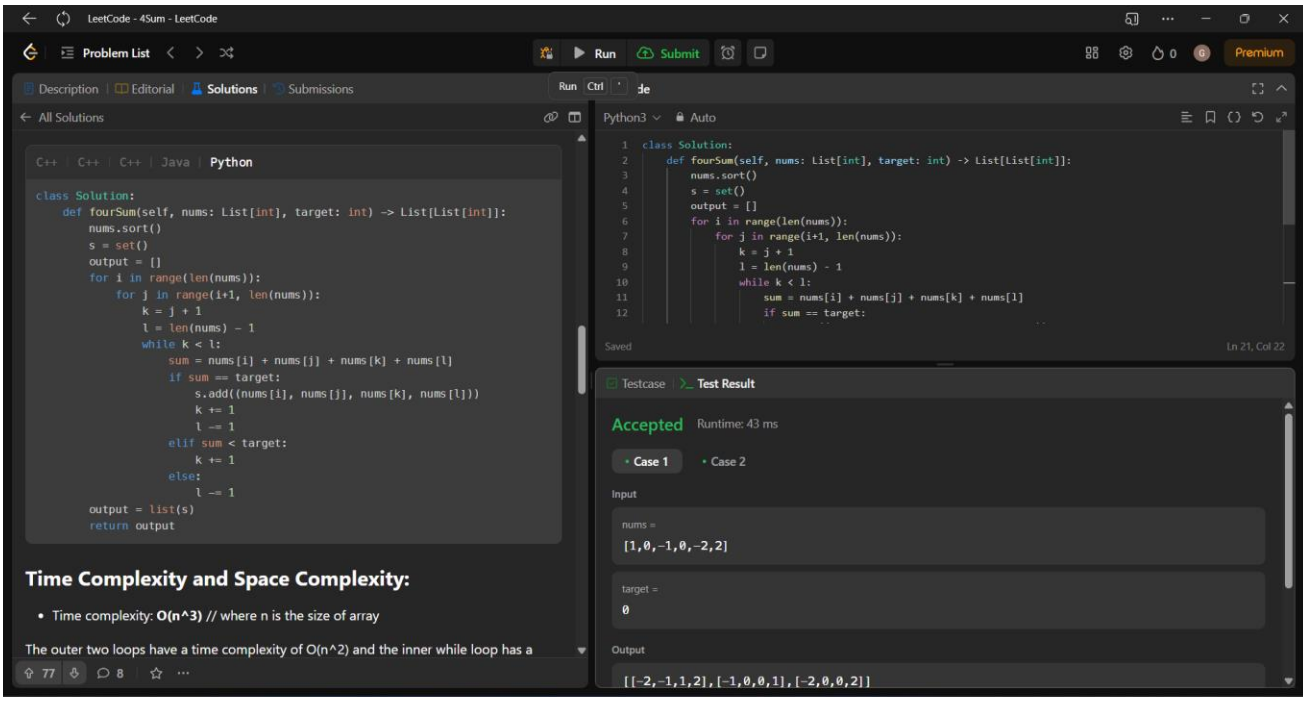
Time Complexity: O(n)

18. 4Sum

Code:

```
class Solution:
    def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
        nums.sort()
        s = set()
        output = []
        for i in range(len(nums)):
            for j in range(i+1, len(nums)):
                k = j + 1
                l = len(nums) - 1
                while k < l:
                    sum = nums[i] + nums[j] + nums[k] + nums[l]
                    if sum == target:
                        s.add((nums[i], nums[j], nums[k], nums[l]))
                        k += 1
                        l -= 1
                    elif sum < target:
                        k += 1
                    else:
                        l -= 1
        output = list(s)
        return output
```

Screenshot:



```
class Solution:
    def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
        nums.sort()
        s = set()
        output = []
        for i in range(len(nums)):
            for j in range(i+1, len(nums)):
                k = j + 1
                l = len(nums) - 1
                while k < l:
                    sum = nums[i] + nums[j] + nums[k] + nums[l]
                    if sum == target:
                        s.add((nums[i], nums[j], nums[k], nums[l]))
                        k += 1
                        l -= 1
                    elif sum < target:
                        k += 1
                    else:
                        l -= 1
        output = list(s)
        return output
```

Time Complexity and Space Complexity:

- Time complexity: **O(n^3)** // where n is the size of array

The outer two loops have a time complexity of O(n^2) and the inner while loop has a

```
class Solution:
    def fourSum(self, nums: List[int], target: int) -> List[List[int]]:
        nums.sort()
        s = set()
        output = []
        for i in range(len(nums)):
            for j in range(i+1, len(nums)):
                k = j + 1
                l = len(nums) - 1
                while k < l:
                    sum = nums[i] + nums[j] + nums[k] + nums[l]
                    if sum == target:
```

Accepted Runtime: 43 ms

Case 1 Case 2

Input

```
nums =
[1,0,-1,0,-2,2]
target =
0
```

Output

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

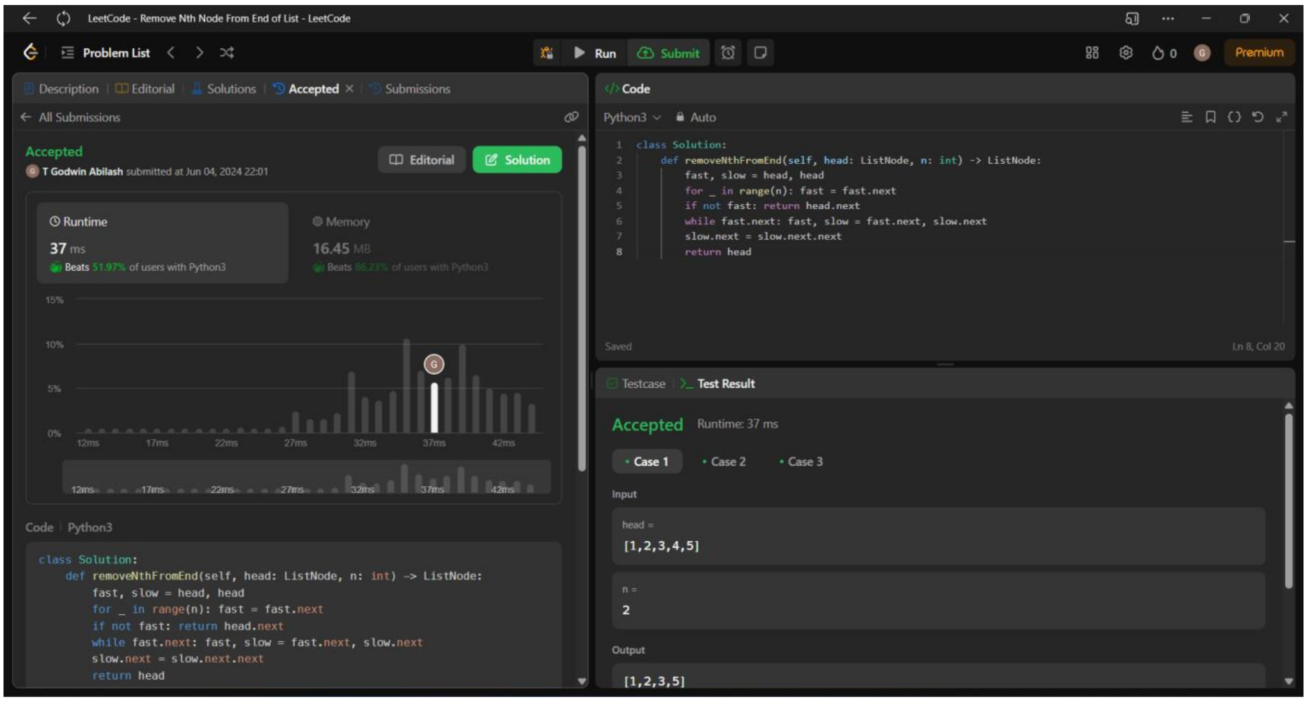
Time Complexity: O(n2)

19. Remove Nth Node from end of the list

Code:

```
class Solution:
    def removeNthFromEnd(self, head: ListNode, n: int) -> ListNode:
        fast, slow = head, head
        for _ in range(n): fast = fast.next
        if not fast: return head.next
        while fast.next: fast, slow = fast.next, slow.next
        slow.next = slow.next.next
        return head
```

Screenshot:



Accepted
T Godwin Abilash submitted at Jun 04, 2024 22:01

Runtime
37 ms
Beats 51.97% of users with Python3

Memory
16.45 MB
Beats 86.23% of users with Python3

Code | Python3

```
class Solution:
    def removeNthFromEnd(self, head: ListNode, n: int) -> ListNode:
        fast, slow = head, head
        for _ in range(n): fast = fast.next
        if not fast: return head.next
        while fast.next: fast, slow = fast.next, slow.next
        slow.next = slow.next.next
        return head
```

Saved
Ln 8, Col 20

Accepted Runtime: 37 ms

Case 1 · Case 2 · Case 3

Input

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

n =
```
2
```

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

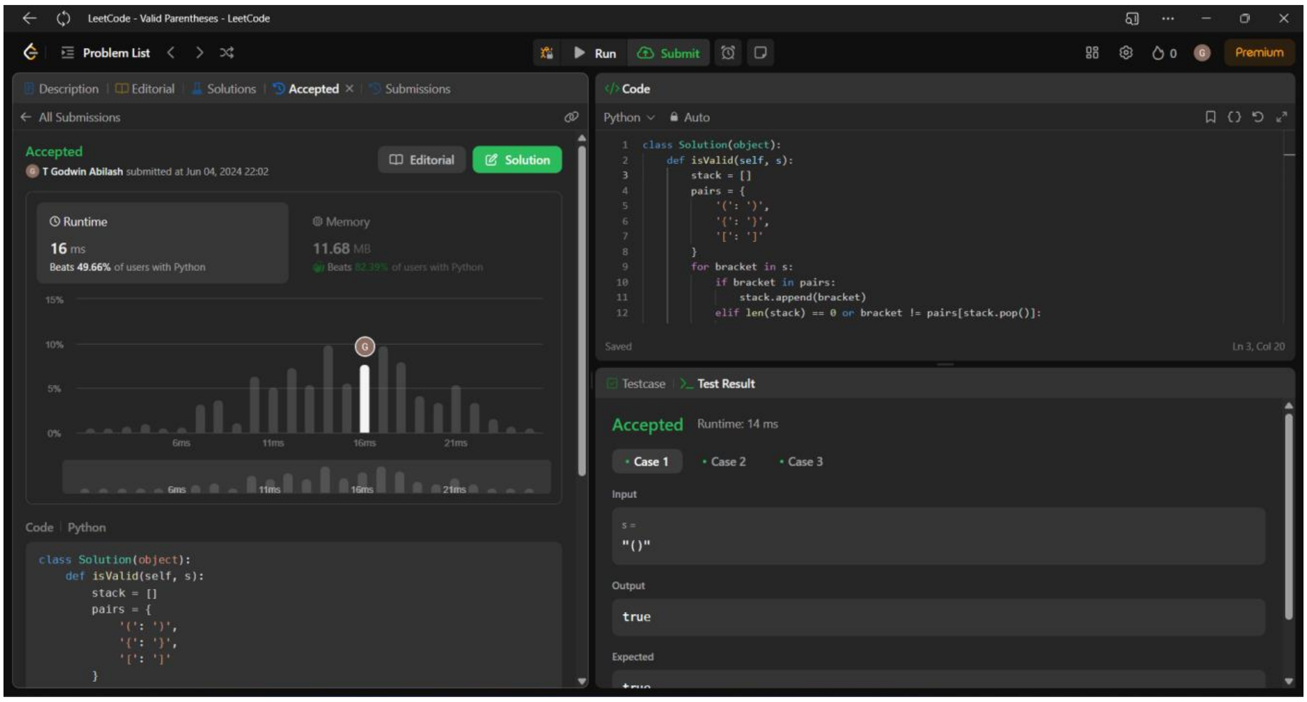
Time Complexity: O(n)

10. Regular Expression Matching

Code:

```
class Solution(object):
    def isValid(self, s):
        stack = []
        pairs = {
            '(': ')',
            '{': '}',
            '[': ']'
        }
        for bracket in s:
            if bracket in pairs:
                stack.append(bracket)
            elif len(stack) == 0 or bracket != pairs[stack.pop()]:
                return False
        return len(stack) == 0
```

Screenshot:


LeetCode - Valid Parentheses - LeetCode

Accepted
T Godwin Abilash submitted at Jun 04, 2024 22:02

Runtime
16 ms
Beats 49.66% of users with Python

Memory
11.68 MB
Beats 82.39% of users with Python

Code | Python

```
class Solution(object):
    def isValid(self, s):
        stack = []
        pairs = {
            '(': ')',
            '{': '}',
            '[': ']'
        }
```

Code

Python Auto

```
class Solution(object):
    def isValid(self, s):
        stack = []
        pairs = {
            '(': ')',
            '{': '}',
            '[': ']'
        }
        for bracket in s:
            if bracket in pairs:
                stack.append(bracket)
            elif len(stack) == 0 or bracket != pairs[stack.pop()]:
```

Saved Ln 3, Col 20

Test Result

Accepted Runtime: 14 ms

Case 1 Case 2 Case 3

Input

```
s =
"()"
```

Output

```
true
```

Expected

Time Complexity: O(n)