Assignment - 1

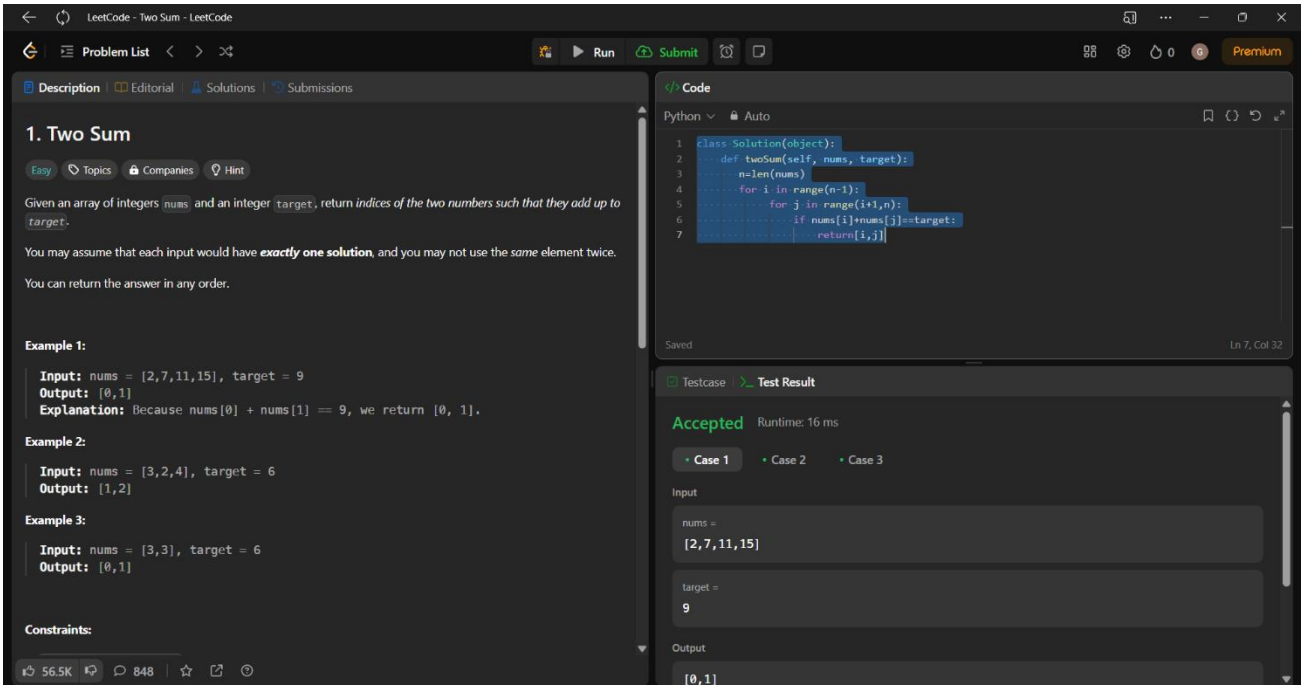
T Godwin Abilash – 192321012

1. Two Sum

Code:

```
class Solution(object):
    def twoSum(self, nums, target):
        n=len(nums)
        for i in range(n-1):
            for j in range(i+1,n):
                if nums[i]+nums[j]==target:
                    return[i,j]
```

Screenshot:



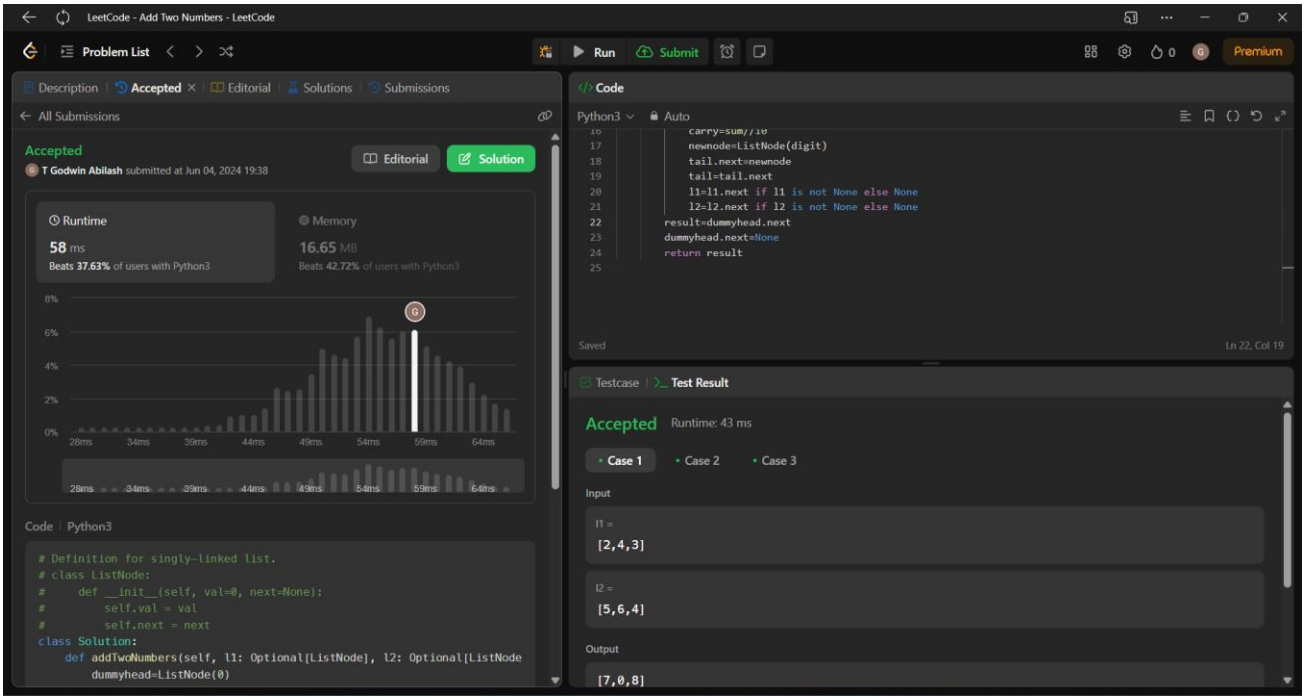
Time Complexity: O(n)

2. Two Sum

Code:

```
class Solution:
    def addTwoNumbers(self, l1: Optional[ListNode], l2:
Optional[ListNode]) -> Optional[ListNode]:
        dummyhead=ListNode(0)
        tail=dummyhead
        carry=0
        while l1 is not None or l2 is not None or carry !=0:
            d1=l1.val if l1 is not None else 0
            d2=l2.val if l2 is not None else 0
            sum=d1+d2+carry
            digit=sum%10
            carry=sum//10
            newnode=ListNode(digit)
            tail.next=newnode
            tail=tail.next
            l1=l1.next if l1 is not None else None
            l2=l2.next if l2 is not None else None
        result=dummyhead.next
        dummyhead.next=None
        return result
```

Screenshot for I/O:



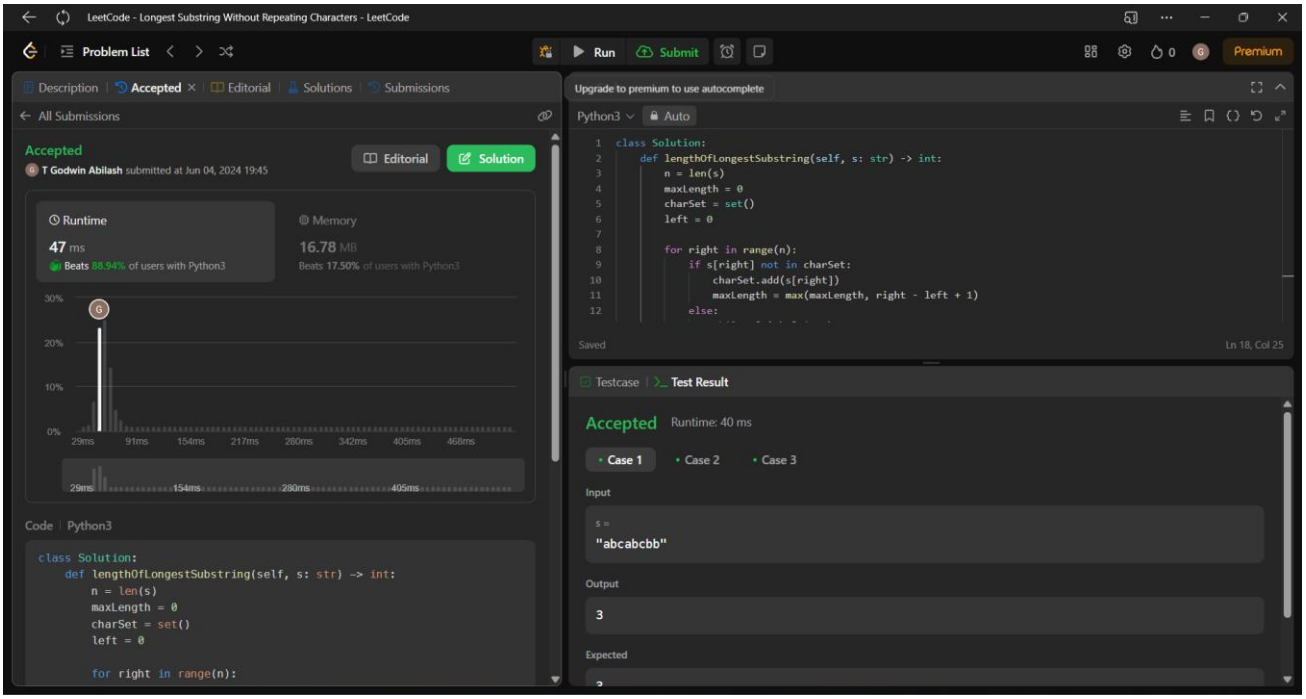
Time Complexity: O(max(m,n))

3. Longest Substring Without Repeating Characters:

Code:

```
class Solution:
    def lengthOfLongestSubstring(self, s: str) -> int:
        n = len(s)
        maxLength = 0
        charSet = set()
        left = 0
        for right in range(n):
            if s[right] not in charSet:
                charSet.add(s[right])
                maxLength = max(maxLength, right - left + 1)
            else:
                while s[right] in charSet:
                    charSet.remove(s[left])
                    left += 1
                charSet.add(s[right])
        return maxLength
```

Screenshot:



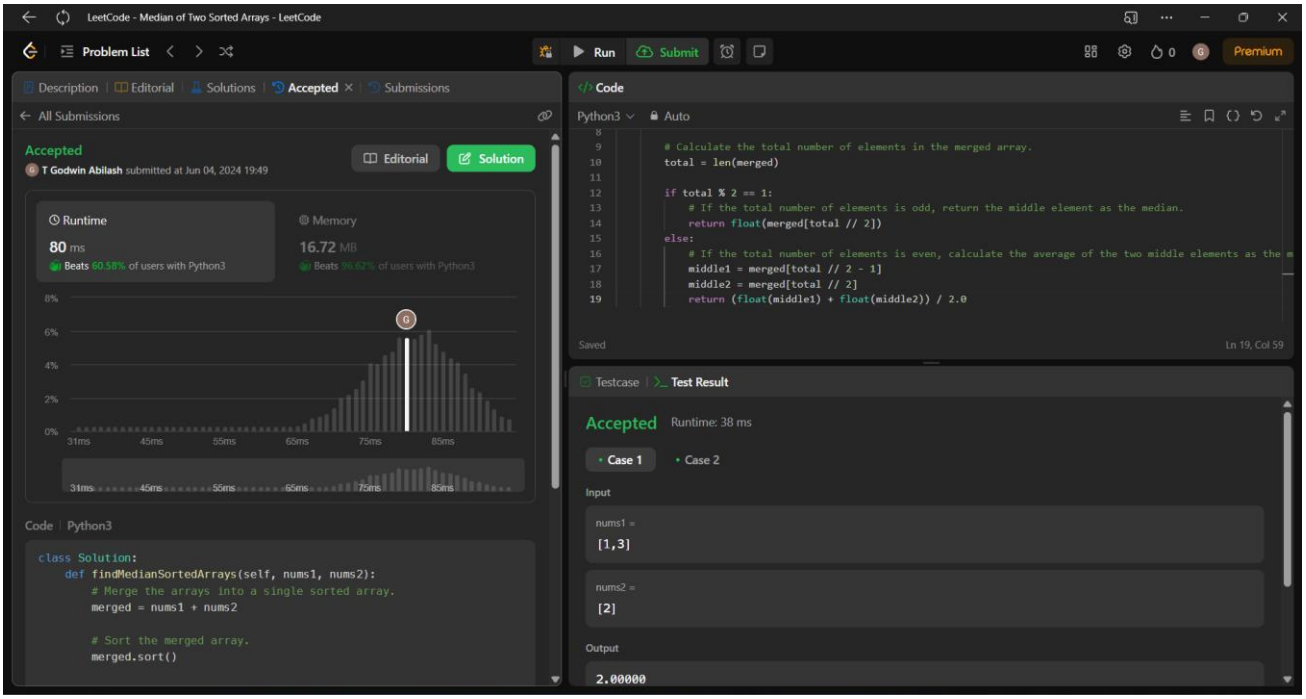
Time Complexity: O(n)

4. Median of Two Sorted Arrays

Code:

```
class Solution:
    def findMedianSortedArrays(self, nums1, nums2):
        merged = nums1 + nums2
        merged.sort()
        total = len(merged)
        if total % 2 == 1:
            return float(merged[total // 2])
        else:
            middle1 = merged[total // 2 - 1]
            middle2 = merged[total // 2]
            return (float(middle1) + float(middle2)) / 2.0
```

Screenshot:



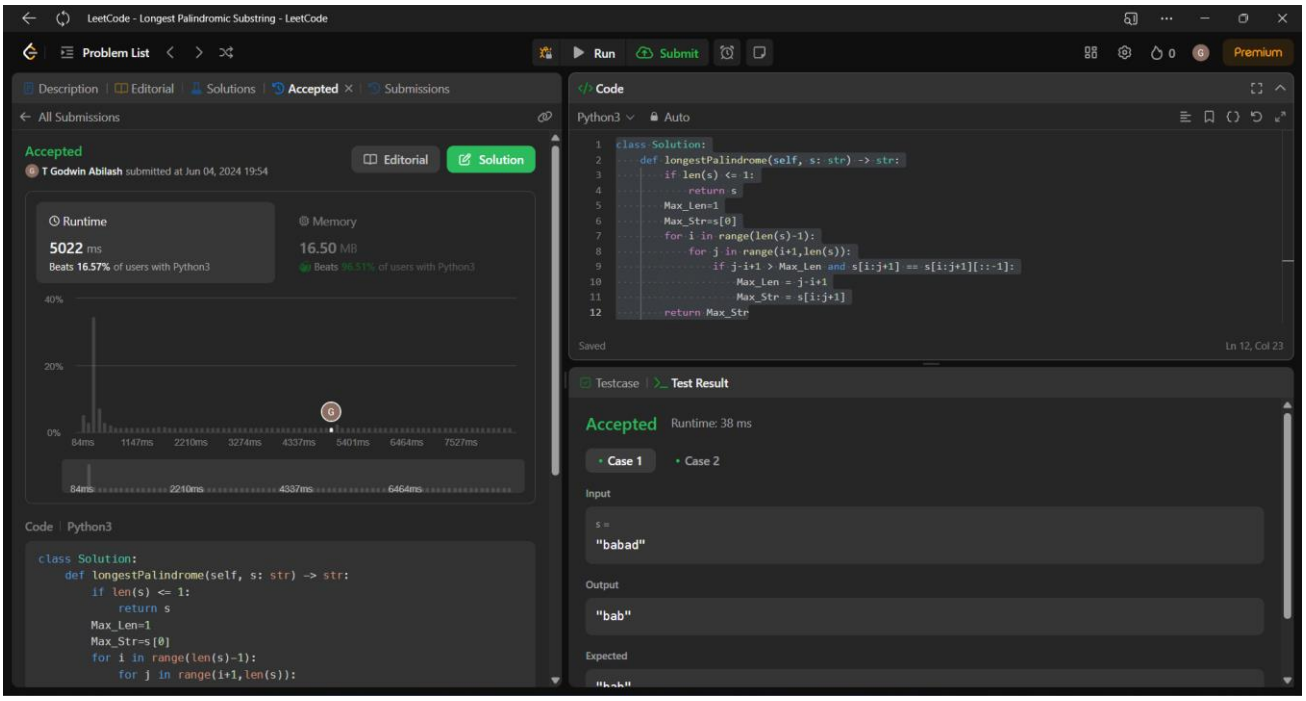
Time Complexity: O(n)

5. Longest Palindrome Substring

Code:

```
class Solution:
    def longestPalindrome(self, s: str) -> str:
        if len(s) <= 1:
            return s
        Max_Len=1
        Max_Str=s[0]
        for i in range(len(s)-1):
            for j in range(i+1,len(s)):
                if j-i+1 > Max_Len and s[i:j+1] == s[i:j+1][::-1]:
                    Max_Len = j-i+1
                    Max_Str = s[i:j+1]
        return Max_Str
```

Screenshot:



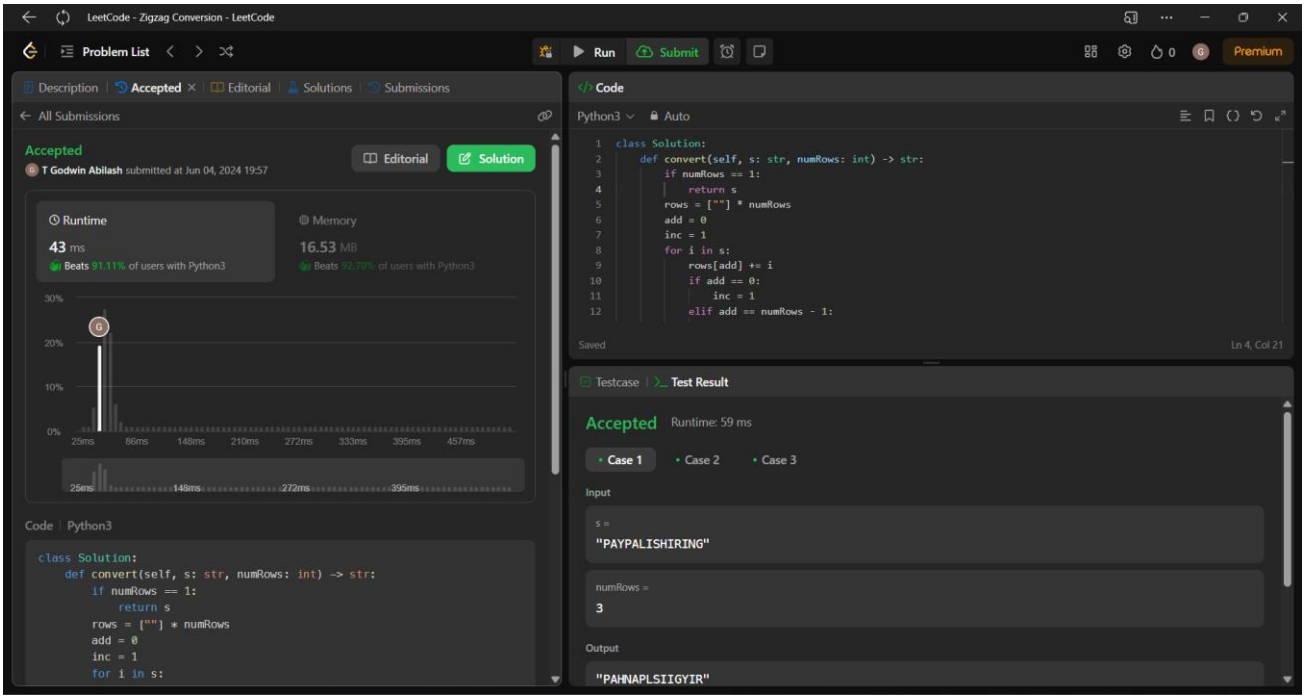
Time Complexity: O(n2)

6. Zigzag Conversion

Code:

```
class Solution:
    def convert(self, s: str, numRows: int) -> str:
        if numRows == 1:
            return s
        rows = [""] * numRows
        add = 0
        inc = 1
        for i in s:
            rows[add] += i
            if add == 0:
                inc = 1
            elif add == numRows - 1:
                inc = -1
            add += inc
        return "".join(rows)
```

Screenshot:



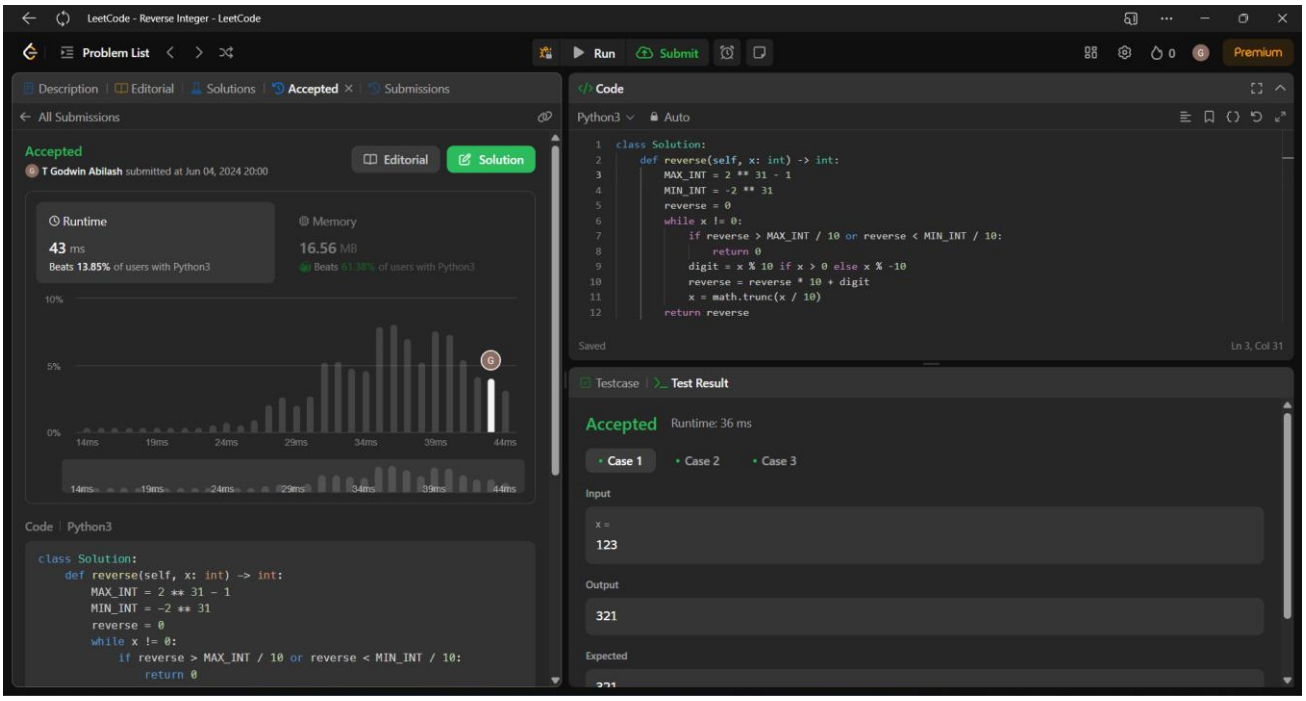
Time Complexity: O(n)

7. Reverse Integer

Code:

```
class Solution:
    def reverse(self, x: int) -> int:
        MAX_INT = 2 ** 31 - 1
        MIN_INT = -2 ** 31
        reverse = 0
        while x != 0:
            if reverse > MAX_INT / 10 or reverse < MIN_INT / 10:
                return 0
            digit = x % 10 if x > 0 else x % -10
            reverse = reverse * 10 + digit
            x = math.trunc(x / 10)
        return reverse
```

Screenshot:



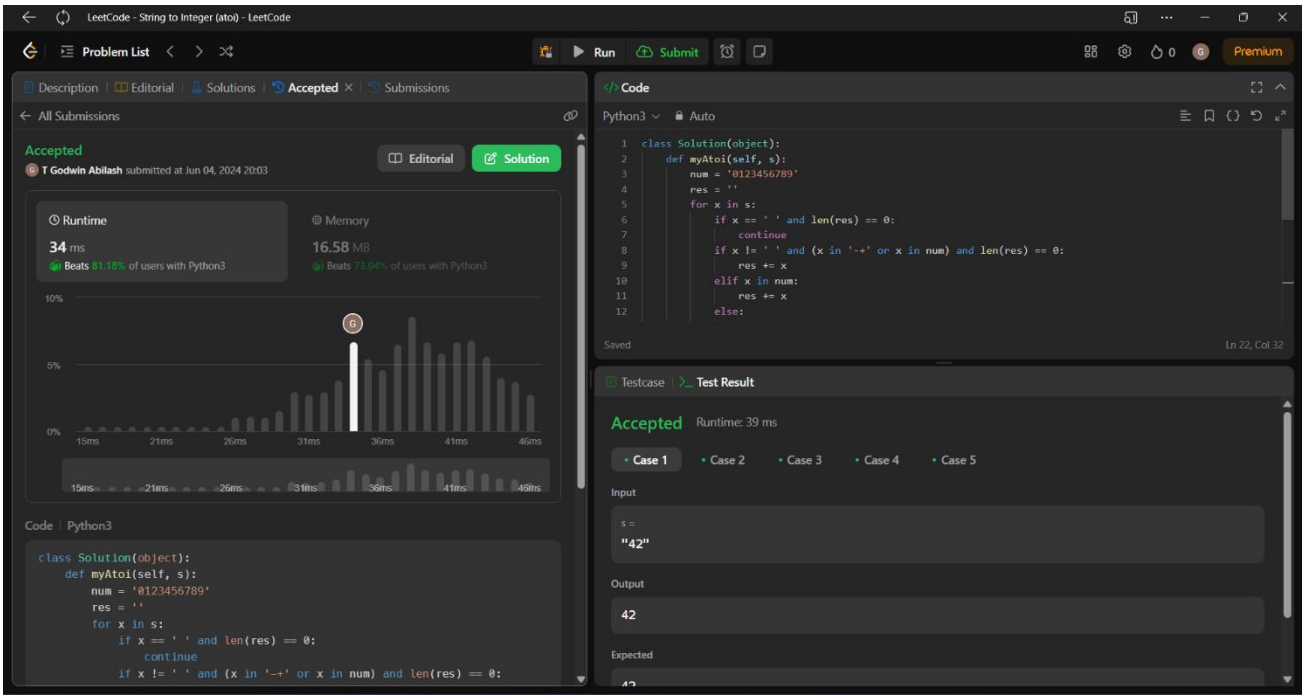
Time Complexity: O(n)

8. String to Integer (atoi)

Code:

```
class Solution(object):
    def myAtoi(self, s):
        num = '0123456789'
        res = ''
        for x in s:
            if x == ' ' and len(res) == 0:
                continue
            if x != ' ' and (x in '-+' or x in num) and len(res) == 0:
                res += x
            elif x in num:
                res += x
            else:
                break
        if res == '' or res in '-+':
            return 0
        else:
            if int(res) < -(2**31):
                return -(2**31)
            elif int(res) > (2**31 - 1):
                return (2**31 - 1)
            else:
                return int(res)
```

Screenshot:



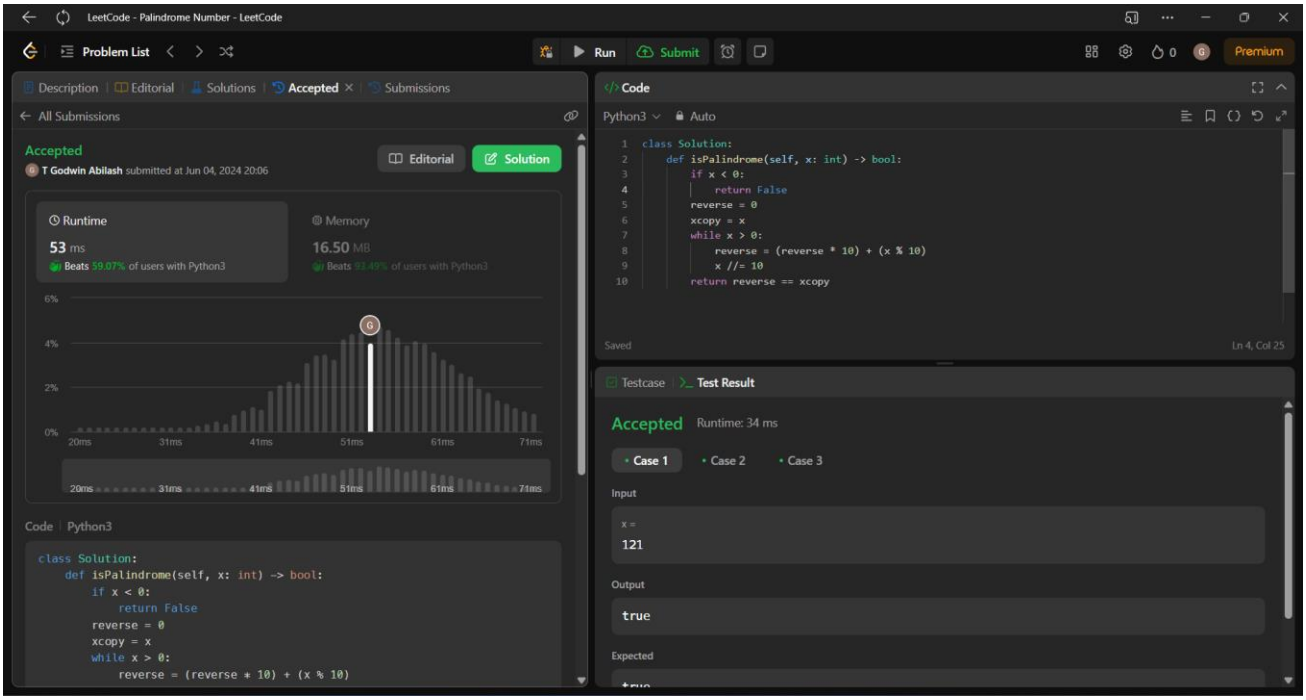
Time Complexity: O(n)

9. Palindrome Number

Code:

```
class Solution:
    def isPalindrome(self, x: int) -> bool:
        if x < 0:
            return False
        reverse = 0
        xcopy = x
        while x > 0:
            reverse = (reverse * 10) + (x % 10)
            x //= 10
        return reverse == xcopy
```

Screenshot:



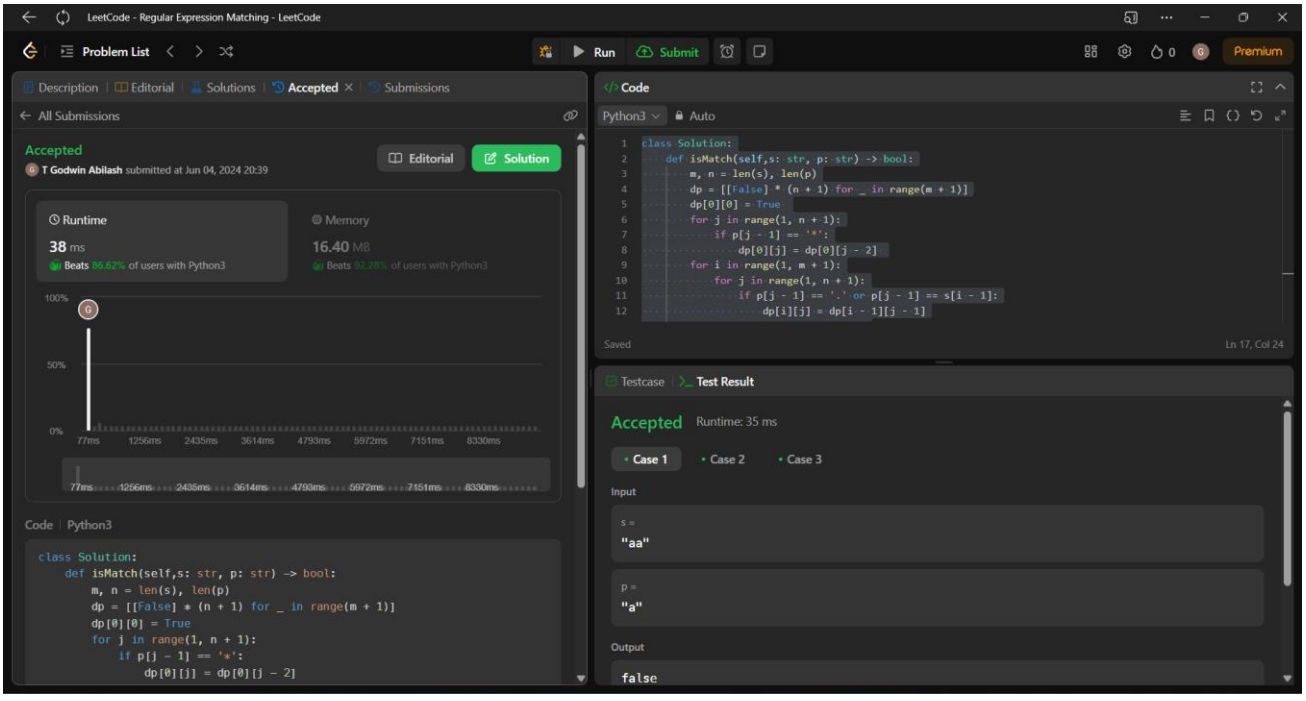
Time Complexity: O(logn)

10. Regular Expression Matching

Code:

```
class Solution:
    def isMatch(self,s: str, p: str) -> bool:
        m, n = len(s), len(p)
        dp = [[False] * (n + 1) for _ in range(m + 1)]
        dp[0][0] = True
        for j in range(1, n + 1):
            if p[j - 1] == '*':
                dp[0][j] = dp[0][j - 2]
        for i in range(1, m + 1):
            for j in range(1, n + 1):
                if p[j - 1] == '.' or p[j - 1] == s[i - 1]:
                    dp[i][j] = dp[i - 1][j - 1]
                elif p[j - 1] == '*':
                    dp[i][j] = dp[i][j - 2]
                    if p[j - 2] == '.' or p[j - 2] == s[i - 1]:
                        dp[i][j] = dp[i][j] or dp[i - 1][j]
        return dp[m][n]
```

Screenshot:



Time Complexity: O(m*n)