Assignment - 3

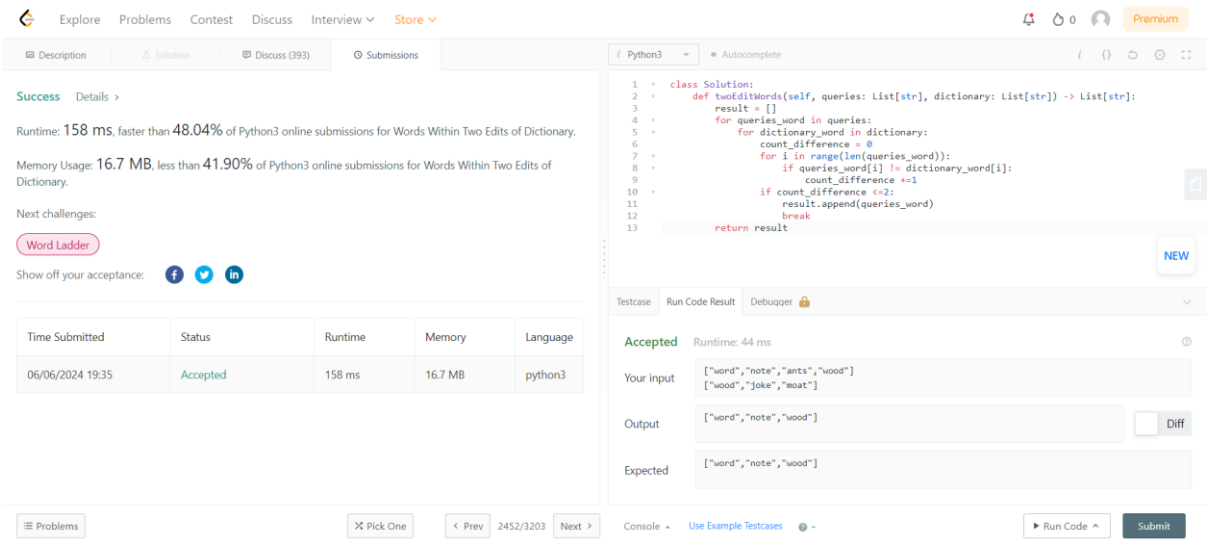
V Pranav - 192311029

2. Words within two edits of dictionary

Code:

```
class Solution:
    def twoEditWords(self, queries: List[str], dictionary: List[str]) -> List[str]:
        result = []
        for queries_word in queries:
            for dictionary_word in dictionary:
                count_difference = 0
                for i in range(len(queries_word)):
                    if queries_word[i] != dictionary_word[i]:
                        count_difference +=1
                if count_difference <=2:
                    result.append(queries_word)
                    break
        return result
```

Screenshot:

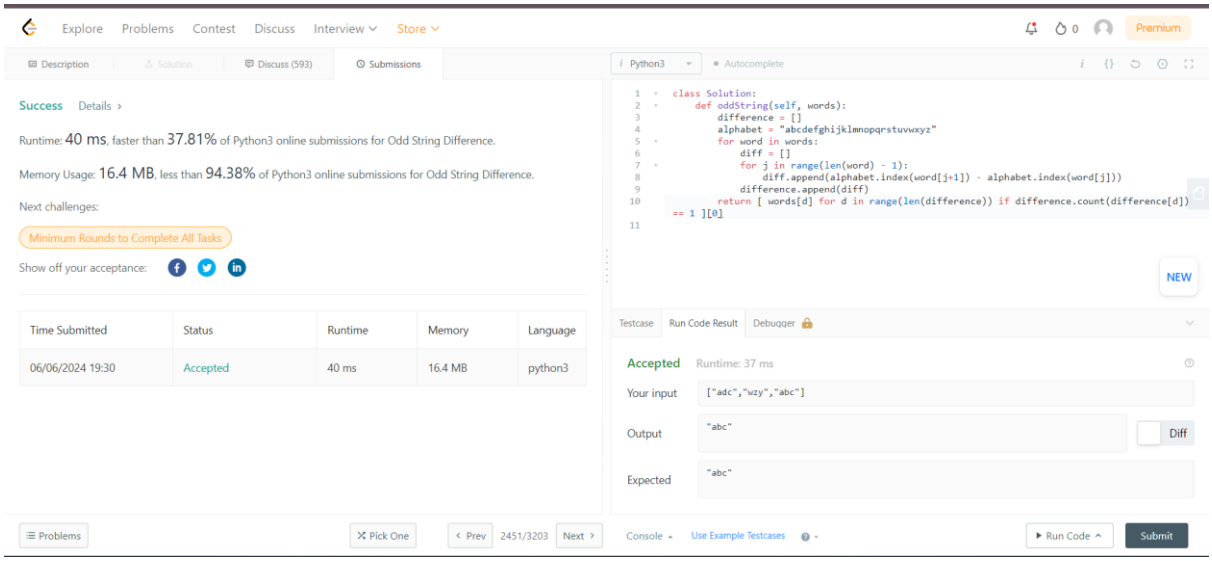


1. Odd string difference

Code:

```
class Solution:
    def oddString(self, words):
        difference = []
        alphabet = "abcdefghijklmnopqrstuvwxyz"
        for word in words:
            diff = []
            for j in range(len(word) - 1):
                diff.append(alphabet.index(word[j+1]) - alphabet.index(word[j]))
            difference.append(diff)
        return [ words[d] for d in range(len(difference)) if difference.count(difference[d]) == 1 ][0]
```

Screenshot:

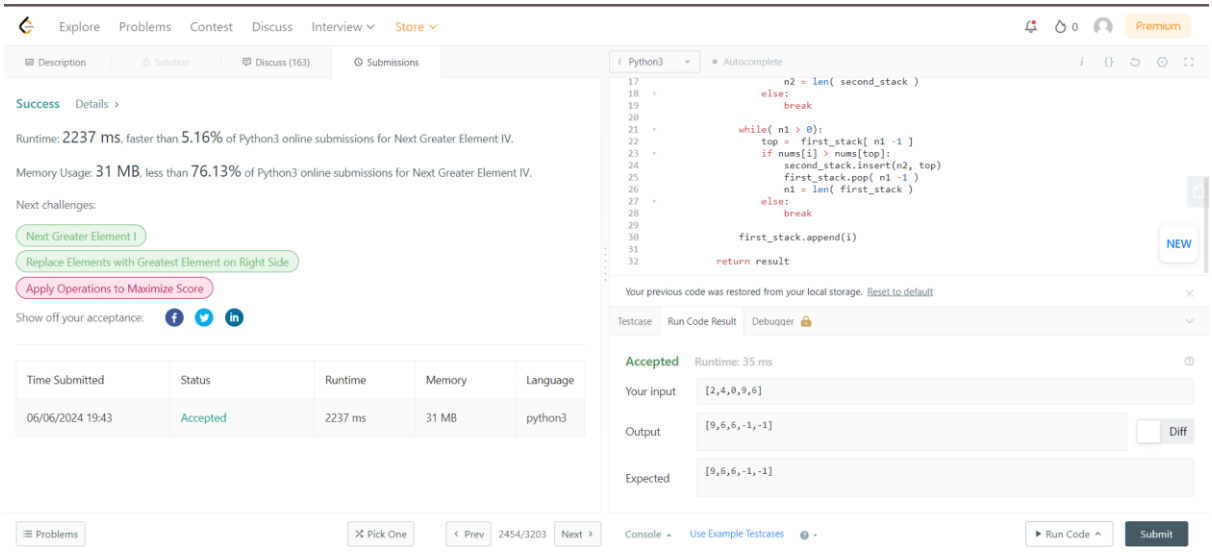


3 . Next greater element IV

Code:

```
class Solution(object):
    def secondGreaterElement(self, nums):
        first_stack = []
        second_stack = []
        n = len(nums)
        result = [-1]*n
        for i in range(0, n, 1):
            n1 = len( first_stack )
            n2 = len( second_stack )
            while( n2 > 0 ):
                top = second_stack[ n2 - 1 ]
                if nums[i] > nums[top]:
                    result[ top ] = nums[i]
                    second_stack.pop( n2 -1 )
                    n2 = len( second_stack )
                else:
                    break
            while( n1 > 0):
                top =  first_stack[ n1 -1 ]
                if nums[i] > nums[top]:
                    second_stack.insert(n2, top)
                    first_stack.pop( n1 -1 )
                    n1 = len( first_stack )
                else:
                    break
            first_stack.append(i)
        return result
```

Screenshot:

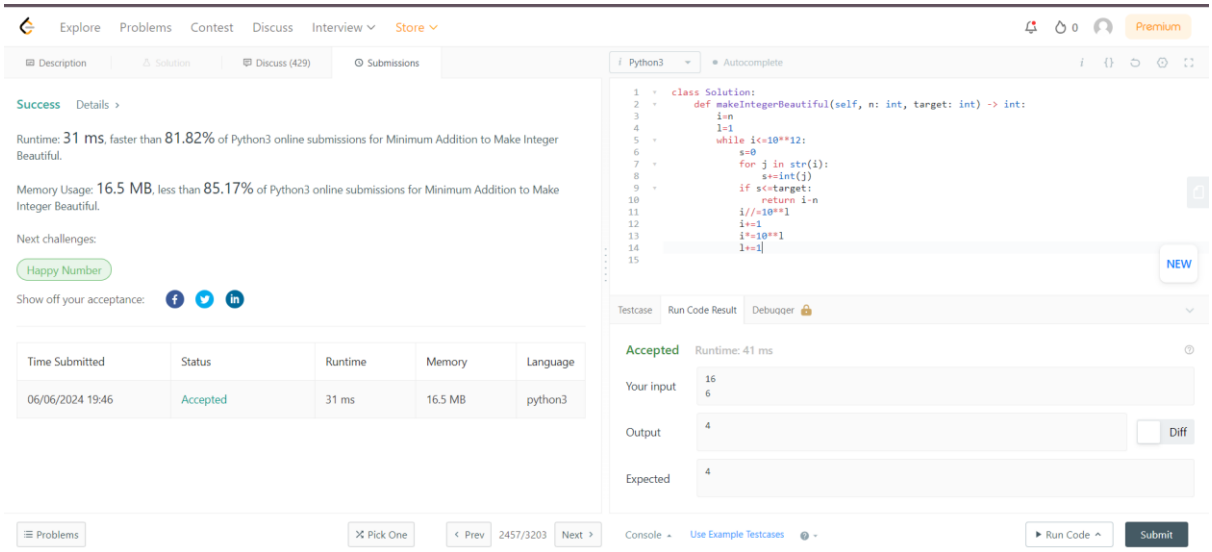


4. Minimum addition to make integer beautiful

Code:

```
class Solution:
    def makeIntegerBeautiful(self, n: int, target: int) -> int:
        i=n
        l=1
        while i<=10**12:
            s=0
            for j in str(i):
                s+=int(j)
            if s<=target:
                return i-n
            i//=10**l
            i+=1
            i*=10**l
            l+=1
```

Screenshot:


Explore
Problems
Contest
Discuss
Interview
Store
Premium
Description
Discuss (429)
Submissions
Success
Details
Runtime: 31 ms, faster than 81.82% of Python3 online submissions for Minimum Addition to Make Integer Beautiful.
Memory Usage: 16.5 MB, less than 85.17% of Python3 online submissions for Minimum Addition to Make Integer Beautiful.
Next challenges:
Happy Number
Show off your acceptance:
Time Submitted
Status
Runtime
Memory
Language
06/06/2024 19:46
Accepted
31 ms
16.5 MB
python3
Python3
Autocomplete
class Solution:
def makeIntegerBeautiful(self, n: int, target: int) -> int:
i=n
l=1
while i<=10**12:
s=0
for j in str(i):
s+=int(j)
if s<=target:
return i-n
i//=10**l
i+=1
i*=10**l
l+=1
NEW
Testcase
Run Code Result
Debugger
Accepted
Runtime: 41 ms
Your input
16
6
Output
4
Diff
Expected
4
Problems
Pick One
Prev
2457/3203
Next
Console
Use Example Testcases
Run Code
Submit