

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:

The screenshot displays the LeetCode interface for the problem "Words Within Two Edits of Dictionary". The top navigation bar includes links for Explore, Problems, Contest, Discuss, Interview, and Store. The problem page shows a "Success" message with details: Runtime: 158 ms, faster than 48.04% of Python3 online submissions; Memory Usage: 16.7 MB, less than 41.90% of Python3 online submissions. A table lists the submission details, showing it was accepted on 06/06/2024 at 19:35. The code editor shows the Python solution, and the test case results show it passed.

Success Details >

Runtime: 158 ms, faster than 48.04% of Python3 online submissions for Words Within Two Edits of Dictionary.

Memory Usage: 16.7 MB, less than 41.90% of Python3 online submissions for Words Within Two Edits of Dictionary.

Next challenges:

Word Ladder

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
06/06/2024 19:35	Accepted	158 ms	16.7 MB	python3

Python3 Autocomplete

```
1 class Solution:
2     def twoEditWords(self, queries: List[str], dictionary: List[str]) -> List[str]:
3         result = []
4         for queries_word in queries:
5             for dictionary_word in dictionary:
6                 count_difference = 0
7                 for i in range(len(queries_word)):
8                     if queries_word[i] != dictionary_word[i]:
9                         count_difference += 1
10                if count_difference <= 2:
11                    result.append(queries_word)
12                    break
13        return result
```

Testcase Run Code Result Debugger

Accepted Runtime: 44 ms

Your input

```
["word","note","ants","wood"]
["wood","jake","moat"]
```

Output

```
["word","note","wood"]
```

Expected

```
["word","note","wood"]
```

Run Code Submit

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:

The screenshot displays the LeetCode interface for the 'Odd String Difference' problem. On the left, the 'Success' message is shown, indicating that the solution was accepted with a runtime of 40 ms and memory usage of 16.4 MB. Below this, there is a table of submissions with columns for Time Submitted, Status, Runtime, Memory, and Language. The right-hand side of the interface shows the Python3 code for the solution, which is a class Solution with a method oddString. The code calculates the difference between the indices of consecutive characters in each word and returns the word(s) that have a unique difference. The bottom panel shows the test case results, indicating that the solution is accepted for the input ["adc","wzy","abc"] with the expected output "abc".

Time Submitted	Status	Runtime	Memory	Language
06/06/2024 19:30	Accepted	40 ms	16.4 MB	python3

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:

Success Details >

Runtime: 2237 ms, faster than 5.16% of Python3 online submissions for Next Greater Element IV.

Memory Usage: 31 MB, less than 76.13% of Python3 online submissions for Next Greater Element IV.

Next challenges:

- Next Greater Element I
- Replace Elements with Greatest Element on Right Side
- Apply Operations to Maximize Score

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
06/06/2024 19:43	Accepted	2237 ms	31 MB	python3

```
17 n2 = len( second_stack )
18 else:
19     break
20
21 while( n1 > 0):
22     top = first_stack[ n1 -1 ]
23     if nums[i] > nums[top]:
24         second_stack.insert(n2, top)
25         first_stack.pop( n1 -1 )
26         n1 = len( first_stack )
27     else:
28         break
29     first_stack.append(i)
30
31
32 return result
```

Your previous code was restored from your local storage. [Reset to default](#)

Testcase Run Code Result Debugger

Accepted Runtime: 35 ms

Your input [2,4,0,9,6]

Output [9,6,6,-1,-1] Diff

Expected [9,6,6,-1,-1]

Problems Pick One < Prev 2454/3203 Next > Console Use Example Testcases Run Code Submit

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

🔔

🔥 0

👤

Premium

Description

Solution

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:

f

t

in

Time Submitted	Status	Runtime	Memory	Language
06/06/2024 19:46	Accepted	31 ms	16.5 MB	python3

Python3

Autocomplete

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

```
1 class Solution:
2     def makeIntegerBeautiful(self, n: int, target: int) -> int:
3         i=n
4         l=1
5         while i<=10**12:
6             s=0
7             for j in str(i):
8                 s+=int(j)
9             if s<=target:
10                 return i-n
11             i//=10**1
12             i+=1
13             i*=10**1
14             l+=1
15
```

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