Daniel Gamboa

daniel-gamboa@lambdastudents.com

BASIC INFO

Test: Computer Science - Python III - Module Project

Solved: 6/6

Similarity: none

Score: 1400/1400

Finished On: 07 Apr 2021

Time Taken: 89m/168h

Labels: -

Task	Solve Time	Score	Similarity
csTimeComplexity	1min	100/100	-
csSpaceComplexity	1min	100/100	-
csLongestPossible	14min	300/300	none
csSortedTwoSum	8min	300/300	none
csFindAddedLetter	16min	300/300	none
csFirstUniqueChar	41min	300/300	none

Task details: csTimeComplexity

Description:

Using Big O notation, what is the correct classification of time complexity for the function below?

```
def do_lots_of_things(items):
    last = len(items) - 1
    print(items[last])

    middle = len(items) / 2
    i = 0
    while i < middle:
        print(items[i])
        i += 1

    for num in range(100):
        print(num)
```

[✓ O(n)] *(Correct)*

O(log n) *(Incorrect)*

O(n^2) *(Incorrect)*

O(1) *(Incorrect)*

Task details: csSpaceComplexity

Description:

Using Big O notation, what is the correct classification of space complexity for the function below?

```
def do_a_couple_things(n):
    my_list = []
    my_second_list = [0] * 26

    for _ in range(n):
        my_list.append("lambda")
        print(my_second_list[n % 25])

    return my_list
```

[✓ O(n)] *(Correct)*

O(1) *(Incorrect)*

O(2n) *(Incorrect)*

O(n^2) *(Incorrect)*

Task details: csLongestPossible

Description:

Given two strings that include only lowercase alpha characters, `str_1` and `str_2`, write a function that returns a new sorted string that contains any character (only once) that appeared in `str_1` or `str_2`.

Examples:

- csLongestPossible("aabbbcccdef", "xxyyzzz") -> "abcdefxyz"
- csLongestPossible("abc", "abc") -> "abc"

Solution (main.py3):

```
def csLongestPossible(str_1, str_2):
    combined_strings = list(str_1 + str_2)
    combined_strings.sort()
    combined_remove_dups = list(dict.fromkeys(combined_strings))
    return "".join(combined_remove_dups)

```

Task details: csSortedTwoSum

Description:

Given a sorted array (in ascending order) of integers and a target, write a function that finds the two integers that add up to the target.

Examples:

- csSortedTwoSum([3,8,12,16], 11) -> [0,1]
- csSortedTwoSum([3,4,5], 8) -> [0,2]
- csSortedTwoSum([0,1], 1) -> [0,1]

Notes:

- Each input will have exactly one solution.
- You may not use the same element twice.

Solution (main.py3):

```
def csSortedTwoSum(numbers, target):
    for i_a, n in enumerate(numbers):
        for i_b in range(i_a + 1, len(numbers)):
            if n + numbers[i_b] == target:
                return [i_a, i_b]
```

Task details: csFindAddedLetter

Description:

You are given two strings, `str_1` and `str_2`, where `str_2` is generated by randomly shuffling `str_1` and then adding one letter at a random position.

Write a function that returns the letter that was added to `str_2`.

Examples:

- `csFindAddedLetter(str_1 = "bcde", str_2 = "bcdef") -> "f"`
- `csFindAddedLetter(str_1 = "", str_2 = "z") -> "z"`
- `csFindAddedLetter(str_1 = "b", str_2 = "bb") -> "b"`
- `csFindAddedLetter(str_1 = "bf", str_2 = "bfb") -> "b"`

Notes:

- `str_1` and `str_2` both consist of only lowercase alpha characters.

Solution (main.py3):

```
def csFindAddedLetter(str_1, str_2):
    str_1_list = list(str_1)
    str_2_list = list(str_2)

    str_1_list.sort()
    str_2_list.sort()

    for i, c in enumerate(str_1_list):
        if c != str_2_list[i]:
            return str_2_list[i]

    return str_2_list[-1]

```

Task details: csFirstUniqueChar

Description:

Given a string, write a function that returns the index of the first unique (non-repeating) character. If there isn't a unique (non-repeating) character, return -1.

Examples:

- `csFirstUniqueChar(input_str = "lambdaschool") -> 2`
- `csFirstUniqueChar(input_str = "ilovelambdaschool") -> 0`
- `csFirstUniqueChar(input_str = "vvv") -> -1`

Notes:

- `input_str` will only contain lowercase alpha characters.

Solution (main.py3):

```
def csFirstUniqueChar(input_str):
    tracker = dict.fromkeys(input_str, 0)
    index_list = []

    for i, c in enumerate(input_str):
        tracker[c] += 1

    for c, v in tracker.items():
        if v == 1:
            index_list.append(input_str.index(c))

    if len(index_list) == 0:
        return -1

    return min(index_list)

```