Codility_

CodeCheck Report: training7GECDA-D4D

Test Name:

Check out Codility training tasks

Summary Timeline

Tasks summary

Task	Time spent	Score
StrSymmetryPoint ⚠ Python	15 min	100%

Total score

Tasks Details

		Task Score	Correctness	Performance
Easy	1. StrSymmetryPoint Find a symmetry point of a string, if any.	100%	100%	100%

Task description

Write a function:

```
def solution(S)
```

that, given a string S, returns the index (counting from 0) of a character such that the part of the string to the left of that character is a reversal of the part of the string to its right. The function should return −1 if no such index exists.

Note: reversing an empty string (i.e. a string whose length is zero) gives an empty string.

For example, given a string:

"racecar"

the function should return 3, because the substring to the left of the character "e" at index 3 is "rac", and the one to the right is "car".

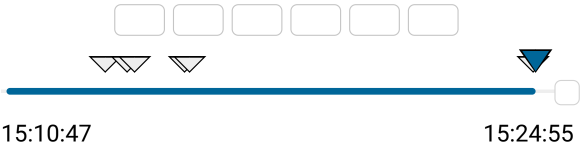
Given a string:

"x"

Solution

Programming language used:	Python	
Total time used:	15 minutes	?
Effective time used:	15 minutes	?
Notes:	*not defined yet*	

Task timeline ?



Code: 15:24:54 UTC, py, final, score: **100** show code in pop-up the function should return 0, because both substrings are empty.

Write an **efficient** algorithm for the following assumptions:

- the length of string S is within the range [0..2,000,000].

```python
# you can write to stdout for debugging pu
# print("this is a debug message")

def solution(S):
    # Implement your solution here
    left = 0
    right = len(S) - 1

    while left < right:
        if S[left] == S[right]:
            # Move the pointers inward
            left += 1
            right -= 1
        else:
            # Substrings are not reversals
            return -1

    # Check if the length is odd and retur
    if left == right:
        return left

    # String is a complete reversal, retur
    return -1
```

Analysis summary

The solution obtained perfect score.

Analysis

Detected time complexity: **O(length(S))**

expand all	Example tests
▶ example1 first example	✔ OK
▶ example2 second example	✔ OK
expand all	**Correctness tests**
▶ extreme_empty_or_one empty or one character strings	✔ OK
▶ symmetric short symmetric strings	✔ OK
▶ even even length or symmetric strings	✔ OK
▶ three_chars 3 characters (multiple runs)	✔ OK
▶ letters_a letters 'a' only	✔ OK
▶ alphabet_symmetric nontrivial symmetry, N = 51	✔ OK
▶ nonsymmetric_inside mismatch close to the middle, N = 43	✔ OK
▶ nonsymmetric_outside mismatch close to the ends, N = 43	✔ OK
expand all	**Performance tests**
▶ large_nonsymmetric	✔ OK

	nonsymmetric string, N = 100k+ + [aba]	
▶	large_symmetric1 symmetric string, N=100k	✔ OK
▶	large_symmetric2 symmetric string, N=200k	✔ OK
▶	big_symmetric3 symmetric string, N=1M+	✔ OK
▶	big_nonsymmetric nonsymmetric string, N = ~1M	✔ OK
▶	extreme_size N = ~2M	✔ OK