[CS 11] Prac 10e – Pál II

Problem Statement

Erdős Pál now has a binary string. The binary string consists only of the characters `0` or `1`.

Pál wants the binary string to have a very long palindromic substring. To do this, he can perform the following operation *exactly once*:

- Select a bit, and flip it (that is, replace `0` with `1` or `1` with `0`)

If Pál chooses the operation optimally, what is the length of the longest palindromic substring that can be obtained?

Luckily, Erdős Pál will help you out—he knows how to find the length of the longest palindromic substring of any string you give him *quickly*.

Task Details

Your task is to implement a function called `longest_palindrome_after_flip`. This function has a single parameter, a `str` consisting of the characters `0` or `1`.

The function must return an `int` denoting the the length of the longest palindromic substring that can be obtained after performing the operation exactly once.

Note: For this task, a new module will be available, specifically for this problem, called `oj`. You may import the function `longest_palindrome_substr` from it, e.g.,

```
from oj import longest_palindrome_substr
```

Copy

The function receives a `str` and returns an `int` denoting the length of the longest palindromic substring of the `str` you passed. The function is moderately efficient—it calculates the result in $O(n)$ time.

The function raises a `ValueError` if it receives anything other than a `str`.

For example:

```
assert longest_palindrome_substr('madame') == 5
assert longest_palindrome_substr('Madame') == 3
```

Copy

Important: If your solution requires something similar to this function, then you should use this function instead of implementing your own, otherwise you may receive the Time Limit Exceeded (TLE) verdict.

Restrictions

(See 10a for more restrictions)

For this problem in particular:

- The following import is allowed: `longest_palindrome_substr` from `oj`
- The source code limit is 2000.

Example Calls

Example 1 Function Call

```
longest_palindrome_after_flip('0100110111')
```

Copy

Example 1 Return Value

```
8
```

Copy

Example 1 Explanation

If the third bit is flipped, the string becomes `0110110111`, and its longest palindromic substring is `11011011`, of length 8. It can be shown that we cannot do better than 8.

Testing

To test your solution locally, you should create a file named `oj.py` and implement the `longest_palindrome_substr` function there. This `oj.py` file is not to be submitted! The judge will have its own `oj.py` with its own (efficient) implementation of `longest_palindrome_substr`. The `oj.py` file you create is only for testing purposes.

Constraints

- The function `longest_palindrome_after_flip` will be called at most $1{,}000$ times.
- Each input is a nonempty string of at most $2{,}000$ characters.
- The total length of the bit strings across all calls will be $\leq 2{,}000$.

Scoring

- You get 60 ❤ points if you solve all test cases where:
  - Each input string has length at most 40.
  - The total length of the bit strings across all calls will be 320.
- You get 60 ❤ points if you solve all test cases where:
  - Each input string has length at most 300.
  - The total length of the bit strings across all calls will be 900.
- You get 65 ❤ points if you solve all test cases.

Submit solution

[CS 11] Practice 10

Points: 185 (partial)

Time limit: 11.0s

Memory limit: 1G

Author: kvatienza (Kevin Atienza)

Problem type

Allowed languages ~~NONE~~, py3

Clarifications

Report an issue

No clarifications have been made at this time.