Q2: 5: Longest palindromic substring
Given a string s, return the longest palindromic substring in s.
Example 1:
Input: $s =$ "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
Example 2:
Input: $s =$ "cbbd"
Output: "bb"
Constraints:
$1 \leq$ s.length $\leq$1000
s consist of only digits and English letters.

Solution

Method 1: Expand around Centers

1 Intuition

Since we are required to find a palindrome string but we are not given the center of each string, just assume that each character in the string may be the center of a possible palindrome string to simplify the problem.

2 Approach

Set a left and right pointer, respectively to help us find a palindrome string in two directions. Note that their position after moving each time cannot be out of range, indicating that `left` ≥ 0 and `right` $<$ `s.size`. To ensure a string is palindrome, the character that the left and right point to each time must be equal, indicating that $s[left] = s[right]$.

Finally, we need to be aware of that `left` < 0 when the loop ends. To assure that we can obtain a valid string, we need to adjust the starting position to *left*+1.

Hence, when finding a longest palindrome string, the strategy we can use is called "*challenge*". It means that we can randomly set any substring in the given string as the longest one. (Note that this may not be the one we would like to find.) Subsequently, we can update by finding a longer one through iteration.

3 Accepted Code

```cpp
class Solution {
public:
    string expand_center(string s, int left, int right){
        while(left >= 0 && right < s.size() && s[left] ==
        ↪  s[right]){
            left--;
            right++;
        }

        return s.substr(left + 1, right - left - 1);
    }

    string longestPalindrome(string s) {
        string longest = s.substr(0, 1);

        if(s.empty()) return "";

        for(int i = 0; i < s.size(); ++i){
            string odd = expand_center(s, i, i);
            if(odd.size() > longest.size())
                longest = odd;

            string even = expand_center(s, i, i + 1);
            if(even.size() > longest.size())
                longest = even;
        }

        return longest;
    }
};
```