520. Detect Capital

Easy String

Leetcode Link

Problem Description

The problem at hand defines a correct capital usage in a word according to three rules:

1. All letters in the word are uppercase (for example, "USA").
2. All letters in the word are lowercase (for example, "leetcode").
3. Only the first letter in the word is uppercase (for example, "Google").

We are given a word and need to determine if the capitalization of the word follows any of the above rules. The function should return `true` if the word's usage of capitals is right and `false` otherwise.

Intuition

The solution is based on the observation that we can only have three correct capitalization scenarios. The approach involves counting the number of uppercase characters in the word and then applying conditional logic based on the count obtained. The reasoning process is as follows:

1. Count the number of uppercase letters in the word.
2. Determine if the number of uppercase letters equals the length of the word, which means all characters are uppercase.
3. Check if there are no uppercase letters, which means all are lowercase.
4. The final condition is if there's only one uppercase letter which also has to be the first character.

If any of these conditions are met, the capital usage is correct, and thus we should return `true`. Otherwise, the usage doesn't match any of the correct forms, and we should return `false`.

Solution Approach

The provided Python code for the solution follows a straightforward approach using basic programming constructs. The steps of the implementation are:

1. Initialize a variable `cnt` to `0` to keep track of the number of uppercase characters in the word.
2. Iterate through each character `c` in the given word.
3. For each character, use the `isupper()` method which is a string method in Python that returns `True` if all cased characters in the string are uppercase and there is at least one cased character, otherwise, it returns `False`.
4. If `c.isupper()` is `True`, increment the `cnt` by `1`.
5. After the loop, we have three conditions to check, corresponding to the right usage of capitals:
   - If `cnt` is `0`, this means there are no uppercase letters, and all letters are lowercase.
   - If `cnt` equals the length of the word, this means all letters in the word are uppercase.
   - If `cnt` is `1` and the first character of the word `word[0].isupper()` is `True`, this means only the first letter is uppercase.

If any of these three conditions are `True`, the function returns `True`, indicating the capital usage in the word is right. Otherwise, it returns `False`.

The algorithm primarily relies on character inspection and condition checking, without the need for any special data structures or complex patterns. It is an example of a linear scan through the input, with a constant-time check performed at each step.

In terms of time complexity, this approach operates in O(n) time, where n is the length of the word since it requires a single pass through all the letters of the word.

Example Walkthrough

Let's consider the word "FlaG" to demonstrate how the solution approach works step by step.

1. We start with `cnt = 0` because we haven't counted any uppercase characters yet.
2. We iterate through each character in the word "FlaG".
   - 'F' is uppercase, `cnt` becomes 1.
   - 'l' is lowercase, `cnt` remains 1.
   - 'a' is lowercase, `cnt` remains 1.
   - 'G' is uppercase, `cnt` becomes 2.
3. Now the iteration is complete and we have `cnt = 2` for the word which has a length of 4 characters.
4. We go through the three conditions to check whether the capital usage is correct.
   - `cnt` is not `0`, so we cannot say all letters are lowercase.
   - `cnt` does not equal the length of the word (2 != 4), so we cannot say all letters are uppercase.
   - `cnt` is not `1`, so we cannot say only the first letter is uppercase. Additionally, even if `cnt` were `1`, the uppercase letter has to be the first character, which is not the only uppercase character in our case.

Since none of the conditions for correct capital usage are met, the function should return `False` for the word "FlaG". The capitalization of this word doesn't adhere to any of the three described rules.

Python Solution

```python
class Solution:
    def detectCapitalUse(self, word: str) -> bool:
        # Initialize a counter for uppercase letters
        uppercase_count = 0

        # Loop through each character in the word
        for char in word:
            # If the character is uppercase, increment the count
            if char.isupper():
                uppercase_count += 1

        # Check the conditions for correct capital usage:
        # 1. No uppercase letters in the word
        # 2. All characters in word are uppercase
        # 3. Only the first character in word is uppercase
        return (uppercase_count == 0 or
                uppercase_count == len(word) or
                (uppercase_count == 1 and word[0].isupper()))

```

Java Solution

```java
class Solution {
    public boolean detectCapitalUse(String word) {
        int capitalLetterCount = 0; // Initialize count of capital letters to 0.

        // Loop through each character in the string.
        for (char character : word.toCharArray()) {
            // Increase the count if the current character is uppercase.
            if (Character.isUpperCase(character)) {
                capitalLetterCount++;
            }
        }

        // Check if the word is all lowercase, all uppercase, or capitalized properly.
        // A word is capitalized properly if it either (1) has no capital letters,
        // (2) is entirely capital letters, or (3) has the first letter as a capital
        // letter followed by all lowercase letters.
        return capitalLetterCount == 0 // No capital letters.
            || capitalLetterCount == word.length() // All letters are uppercase.
            || (capitalLetterCount == 1 && Character.isUpperCase(word.charAt(0))); // Only the first letter is uppercase.
    }
}

```

C++ Solution

```cpp
class Solution {
public:
    bool detectCapitalUse(string word) {
        int capitalCount = 0; // Initialize a counter for capital letters

        // Iterate through each character in the word
        for (char c : word) {
            // If the character is an uppercase letter, increment the counter
            if (isupper(c)) {
                ++capitalCount;
            }
        }

        // Check the capital usage rules:
        // 1. No capital letters in the word
        // 2. All letters are capital letters
        // 3. Only the first letter is capital
        bool allCapitals = capitalCount == word.size(); // All letters are capitals
        bool noCapitals = capitalCount == 0;            // No letters are capitals
        bool firstCapitalOnly = capitalCount == 1 && isupper(word[0]); // Only the first letter is capital

        // Return true if any of the conditions above are met
        return allCapitals || noCapitals || firstCapitalOnly;
    }
};

```

Typescript Solution

```typescript
/**
 * Checks if a given word uses capital letters correctly.
 * Correct use of capitals is defined by either all letters being capitals,
 * no letter being capital, or only the first letter being capital.
 *
 * @param {string} word - The word to check for correct capital use.
 * @returns {boolean} - True if the word uses capitals correctly, false otherwise.
 */
function detectCapitalUse(word: string): boolean {
    // Initialize a counter for capital letters.
    let capitalCount = 0;

    // Iterate through each character in the word.
    for (let i = 0; i < word.length; i++) {
        // If the character is an uppercase letter, increment the counter.
        if (word[i] >= 'A' && word[i] <= 'Z') {
            capitalCount++;
        }
    }

    // Check the capital usage rules.
    const allCapitals = capitalCount === word.length; // All letters are capitals.
    const noCapitals = capitalCount === 0; // No letters are capitals.
    const firstCapitalOnly = capitalCount === 1 && word[0] >= 'A' && word[0] <= 'Z'; // Only the first letter is capital.

    // Return true if any of the conditions above are met.
    return allCapitals || noCapitals || firstCapitalOnly;
}

```

Time and Space Complexity

The given Python code is designed to determine whether a word uses capitals correctly according to the following rules:

1. All letters in the word are capitals, like "USA".
2. All letters in the word are not capitals, like "leetcode".
3. Only the first letter in the word is capital, like "Google".

The time complexity and space complexity of the code are as follows:

Time Complexity

The time complexity of the function is `O(n)` where `n` is the length of the input string word. This is because the function goes through each character in the string exactly once to count how many uppercase letters are present.

Space Complexity

The space complexity of the code is `O(1)` since a fixed amount of extra space is used regardless of the input size. The extra space comes from the variable `cnt`, which is used to count the number of uppercase characters in the string. No additional space that grows with the input size is used.