2129. Capitalize the Title

Easy String

Leetcode Link

Problem Description

The problem presents us with a string `title` that includes one or more words, separated by a single space. Each word is made of English letters. Our task is to capitalize this string, but not in the conventional sense of making every first letter uppercase. Instead, we have to follow these specific rules:

1. If a word is only one or two letters long, we must convert all its letters to lowercase.
2. For words longer than two letters, we need to capitalize the first letter and make all the remaining letters lowercase.

The goal is to return the modified version of the `title` string adhering to these capitalization rules.

Intuition

The solution leverages Python's built-in `split`, `lower`, and `capitalize` string methods to transform the input `title` following the stated rules. Here's the step-by-step approach:

1. Split the input string `title` into individual words, which allows us to process each word separately.
2. Iterate over each word in the list of words.
3. For each word, check its length:
   - If it is less than 3 characters, convert the whole word to lowercase.
   - Otherwise, use the `capitalize` method, which converts the first letter to uppercase and the rest of the letters to lowercase.
4. After processing all words, join them back into a single string with a space separator.
5. Return the resultant capitalized title string.

By following this method, we can apply the required capitalization logic to each word individually and construct the desired output in an efficient, Pythonic way.

Solution Approach

The implementation of the given solution involves the use of the Python list comprehension, a concise and readable way to generate a new list by applying an expression to each item in an iterable. The steps of the algorithm use fundamental Python data structures such as lists and strings, and they take advantage of string manipulation methods provided by Python's standard library.

Here's a walkthrough of the code and the patterns used:

1. `title.split()`: This piece of code splits the input string `title` into a list of words. The `.split()` method divides the string at each instance of the space character, which is the default delimiter.
2. List comprehension: `[w.lower() if len(w) < 3 else w.capitalize() for w in title.split()]`: This is the core of the solution. It loops through each word `w` in the list of words obtained from `title.split()`. It applies the following logic to each word:
   - `w.lower()` is used when `len(w) < 3`, meaning if the word length is less than 3 characters, this part of the ternary expression converts the entire word to lowercase.
   - `w.capitalize()` is used otherwise, which capitalizes the first letter of the word and makes all other letters lowercase.
3. `" ".join(words)`: After the list comprehension generates the list of words adhering to the capitalization rules, this code joins the list back into a single string, separating each word with a space.

By combining the `.split()`, `.lower()`, `.capitalize()`, and `.join()` methods, the implementation elegantly navigates through the requirements without the need for complex algorithms or additional data structures. The solution is straightforward yet powerful, showing how well-chosen methods can make the code concise and effective.

Example Walkthrough

Let's take a simple example to illustrate the solution approach. Suppose the input `title` is "an example of a leetCode problem".

Following the steps of the solution:

1. We first use `title.split()` to split the input string into individual words. After this, our list of words will look like this: ["an", "example", "of", "a", "leetCode", "problem"].
2. Now the list comprehension comes into play. We process each word in the above list according to the length-based rules:
   - "an" has 2 characters, so it is converted to lowercase: "an".
   - "example" has 7 characters, so we capitalize the first letter: "Example".
   - "of" has 2 characters, so it is converted to lowercase: "of".
   - "a" has 1 character, so it is converted to lowercase: "a".
   - "leetCode" has 8 characters, but with capitalization rule, it becomes: "Leetcode".
   - "problem" has 7 characters, so we capitalize the first letter: "Problem".
3. Finally, we join the processed words back into one string with `" ".join(words)`, resulting in the final output: "an Example of a Leetcode Problem".

The list comprehension would look like this: `[w.lower() if len(w) < 3 else w.capitalize() for w in title.split()]`, and it simplifies the code by replacing a more elaborate for-loop with conditions into one compact line.

By the end of the process, we have successfully transformed the input string into a new string that follows the given capitalization rules.

Python Solution

```python
class Solution:
    def capitalizeTitle(self, title: str) -> str:
        # Split the title into words
        words = title.split()

        # Capitalize each word based on its length
        # Words with less than 3 characters are kept lowercase
        # Words with 3 or more characters are capitalized (first letter uppercase, remaining lowercase)
        capitalized_words = [word.lower() if len(word) < 3 else word.capitalize() for word in words]

        # Join the capitalized words back into a string with spaces
        # and return the modified title
        return " ".join(capitalized_words)

```

Java Solution

```java
class Solution {
    public String capitalizeTitle(String title) {
        // Create a list to store the processed words
        List<String> capitalizedWords = new ArrayList<>();

        // Split the title into individual words
        String[] words = title.split(" ");

        // Iterate through each word in the title
        for (String word : words) {
            // Check if the word length is less than 3
            if (word.length() < 3) {
                // Convert the entire word to lowercase and add to the list
                capitalizedWords.add(word.toLowerCase());
            } else {
                // Capitalize only the first letter and convert the rest to lowercase,
                // then add to the list
                String capitalizedWord = word.substring(0, 1).toUpperCase() + word.substring(1).toLowerCase();
                capitalizedWords.add(capitalizedWord);
            }
        }

        // Join the list elements into a string separated by spaces
        return String.join(" ", capitalizedWords);
    }
}

```

C++ Solution

```cpp
#include <sstream> // Include necessary header for string stream

class Solution {
public:
    // Function to capitalize the title according to the specific rules
    string capitalizeTitle(string title) {
        // Convert the whole input string to lowercase
        transform(title.begin(), title.end(), title.begin(), ::tolower);

        istringstream title_stream(title); // Use string stream to read words
        string result; // This will hold the final result
        string word; // This will hold each word after being extracted from the stream

        // Process each word in the input string
        while (title_stream >> word) {
            // Capitalize the first letter of words longer than 2 characters
            if (word.size() > 2) word[0] = toupper(word[0]);

            // Append the processed word to the result string
            result += word + " ";
        }

        // Remove the trailing space added after processing the last word
        if (!result.empty()) result.pop_back();

        // Return the result string with appropriate capitalization
        return result;
    }
};

```

Typescript Solution

```typescript
// Function to capitalize each word in a title string according to specific rules.
// Words with a length less than 3 are kept in lowercase,
// and words with a length of 3 or more have their first letter capitalized,
// with the rest of the letters in lowercase.
function capitalizeTitle(title: string): string {
    // Initialize an array to hold the processed words.
    const capitalizedWords: string[] = [];

    // Split the title into words by spaces and iterate over each word.
    for (const word of title.split(' ')) {
        // Check the length of the word.
        if (word.length < 3) {
            // If the word is less than 3 characters long, add it to the array in lowercase.
            capitalizedWords.push(word.toLowerCase());
        } else {
            // If the word is 3 characters or longer, capitalize the first letter
            // and add the rest in lowercase to the array.
            capitalizedWords.push(word.substring(0, 1).toUpperCase() + word.substring(1).toLowerCase());
        }
    }
    // Join the processed words back into a single string with spaces in between and return it.
    return capitalizedWords.join(' ');
}

```

Time and Space Complexity

Time Complexity

The time complexity of the given code primarily depends on a few operations:

1. The `split()` method, which runs in $O(n)$ where n is the length of the string `title`.
2. The list comprehension, which iterates over each word resulted from the `split()` operation. If there are m words, it checks the length and either converts to lower case or capitalizes the word, which also takes $O(n)$ in total since each character in the title will be processed exactly once.
3. The `join()` method, which again works in $O(n)$ because it combines the words back into a single string, where the total number of characters is still n.

Hence, the overall time complexity is $O(n)$ because all operations are linear with respect to the length of the input string.

Space Complexity

For space complexity:

1. A new list of words is created after the `split()` method, which requires $O(m)$ space where m is the number of words in the title.
2. Each word is converted to a lower or capitalized word which creates temporary strings. However, since strings are immutable in Python, each transformation may create a new string, so the space required is also $O(n)$.

Therefore, the total space complexity of the algorithm is $O(n)$, as it needs to store the transformed words equal to the total length of the input string.