520. Detect Capital (/problems/detect-capital/)

July 10, 2020 | 28.4K views

Average Rating: 4.26 (39 votes)

Given a word, you need to judge whether the usage of capitals in it is right or not.

We define the usage of capitals in a word to be right when one of the following cases holds:

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

Otherwise, we define that this word doesn't use capitals in a right way.

Example 1:

```
Input: "USA"
Output: True
```

Example 2:

```
Input: "FlaG"
Output: False
```

Note: The input will be a non-empty word consisting of uppercase and lowercase latin letters.

Solution

Overview

It's a fairly easy problem because it does not require you to use any special trick, and all you need to do is to implement the solution step by step.

However, it would take some time if you want to make your code easily readable, beautiful, and short. Below two approaches are introduced, they are "*Character by Character*" method and "*Regex*" method.

Approach 1: Character by Character

Intuition

Recall (part of) the description of the problem:

> We define the usage of capitals in a word to be right when one of the following cases holds:
>
> 1. All letters in this word are capitals, like "USA".
> 2. All letters in this word are not capitals, like "leetcode".
> 3. Only the first letter in this word is capital, like "Google".

The problem gives us three patterns, and ask if the given `word` matches any of them. It would be easy to think of checking the cases one by one. In each case, we can just use the most simple method to check if `word` matches the pattern -- **check the char one by one**.

Algorithm

We need three bool variables to store if the pattern matches or not. We set the variables to be true at the beginning, and when the pattern doesn't match, we turn the variables into false. You can also do it otherwise, but the code would be a little longer.

The code is a little long... **Don't be afraid!** It's fairly easy to understand, and we will shorten it later.

Java Python3 Copy

Articles › 520. Detect

```
class Solution:
    def detectCapitalUse(self, word: str) -> bool:
        n = len(word)

        match1, match2, match3 = True, True, True

        # case 1: All capital
        for i in range(n):
            if not word[i].isupper():
                match1 = False
                break
        if match1:
            return True

        # case 2: All not capital
        for i in range(n):
            if word[i].isupper():
                match2 = False
                break
        if match2:
            return True

        # case 3: All not capital except first
        if not word[0].isupper():
            match3 = False
        if match3:
            for i in range(1, n):
                if word[i].isupper():
                    match3 = False
        if match3:
            return True

        # if not matching
        return False
```

There are a few points you should notice from the code above:

1. We use the built-in function `isUpperCase` (in `Java`) and `isupper` (in `Python`) to check whether a char is upper case. You can also use the ASCII (https://en.wikipedia.org/wiki/ASCII) to do that. Just use something like `word.charAt(i) >= 'A' && word.charAt(i) <= 'Z'`.

2. We use `break` after we find matching failed because there is no need to check whether the further char is valid.

3. You can combine the three `match` variables into one by reusing it after each case, but I prefer to separate it into three for better readability.

OK! Now we have solved this problem. The time complexity is $O(n)$ (where n is word length) because we need to check each char at most three times. This time complexity is great, and there is no too much we can do to improve it.

However, we can make the code looks better and shorter, without reducing the readability.

Improvement

Where to start? The biggest problem of the code above is that there are too many cases. What if we can combine them? Notice that the biggest difference between case 2 and case 3 is the condition of the first char.

By combining case 2 and case 3, we get a new pattern: No matter what first char is, the rest should be lowercase.

Java Python3 Copy

```
class Solution:
    def detectCapitalUse(self, word: str) -> bool:
        n = len(word)

        if len(word) == 1:
            return True

        # case 1: All capital
        if word[0].isupper() and word[1].isupper():
            for i in range(2, n):
                if not word[i].isupper():
                    return False
        # case 2 and case 3
        else:
            for i in range(1, n):
                if word[i].isupper():
                    return False

        # if pass one of the cases
        return True
```

Still, there are a few points you should notice from the code above:

1. We check the length of the word firstly because we need to use the first two char to check if the word matches case1. Fortunately, a word with 1 length would always match either case2 or case3.

2. You can count the number of uppercase/lowercase letters in the word instead of checking it one by one and return immediately. That can also work.

3. Some programming languages have built-in methods to check if the word matches certain case, such as `istitle()` in `Python` and `word.toUpperCase().equals(word)` in `Java`. Those methods are doing the same things as our code above. It would be great if you can know both these APIs and how they implemented.

Complexity Analysis

- Time complexity: $O(n)$, where n is the length of the word. We only need to check each char at most constant times.
- Space complexity : $O(1)$. We only need constant spaces to store our variables.

Approach 2: Regex

Intuition

Hey, if we want to do pattern matching, why don't we use Regular Expression (https://en.wikipedia.org/wiki/Regular_expression) (Regex)? Regex is a great way to match a given pattern to a string.

Algorithm

The pattern of case 1 in regex is $[A-Z]*$, where $[A-Z]$ matches one char from 'A' to 'Z', $*$ represents repeat the pattern before it at least 0 times. Therefore, this pattern represents "All capital".

The pattern of case 2 in regex is $[a-z]*$, where similarly, $[a-z]$ matches one char from 'a' to 'z'. Therefore, this pattern represents "All not capital".

Similarly, the pattern of case 3 in regex is $[A-Z][a-z]*$.

Take these three pattern together, we have $[A-Z]*|[a-z]*|[A-Z][a-z]*$, where "|" represents "or".

Still, we can combine case 2 and case 3, and we get $.[a-z]*$, where "." can matches any char.

Therefore, the final pattern is $[A-Z]*|.[a-z]*$.

Java Python3 Copy

```
import re

class Solution:
    def detectCapitalUse(self, word: str) -> bool:
        return re.fullmatch(r"[A-Z]*|.[a-z]*", word)
```

However, it is worth pointing out that the speed of regex is highly dependent on its pattern and its implementation, and the time complexity can vary from $O(1)$ to $O(2^n)$. If you want to control the speed yourself, using Approach 1 would be better.

Complexity Analysis

- Time complexity: Basically $O(n)$, but depends on implementation.
- Space complexity : $O(1)$. We only need constant spaces to store our pattern.

Rate this article:

Previous (/articles/goat-latin/) Next (/articles/most-common-word/)

Comments: 66 Sort By

Type comment here... (Markdown is supported)

Preview Post

divinker (/divinker) 93 July 15, 2020 12:00 PM

(/divinker)

My Java Solution:

```
class Solution {
    public boolean detectCapitalUse(String word) {
        int caps = 0;
```

Read More

27 Share Reply

SHOW 3 REPLIES

azimjohn (/azimjohn) 19 July 19, 2020 4:51 AM

(/azimjohn)

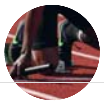
My Python Solution:

```
class Solution:
    def detectCapitalUse(self, word: str) -> bool:
        return word in (
```

Read More

18 Share Reply

SHOW 1 REPLY



(/kumar-jain)

Kumar-Jain (/kumar-jain) ★ 75 15 hours ago

I have one genuine question: Do interviewers care about your Regex knowledge?
It will be nice if some experienced guy might answer it.

5 Share Reply

SHOW 2 REPLIES

(/guptamuskan867)

guptamuskan867 (/guptamuskan867) ★ 1 6 hours ago

My C++ solution:

```
class Solution {
public:
    bool detectCapitalUse(string word) {
```

Read More

1 Share Reply

(/heenalsapovadia)

heenalsapovadia (/heenalsapovadia) ★ 5 13 hours ago

Java Solution :

- Will not parse the whole string if false
- Check the 1st and 2nd char of the string and decide if rest chars should be capitals (boolean allcaps) or smalls (boolean allsmall)

Read More

1 Share Reply

(/elulcao)

elulcao (/elulcao) ★ 10 July 15, 2020 1:04 AM

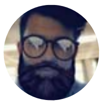
Nice article, if you ask me I prefer first approach. However regex looks always pretty good.

On the other hand, it seems there is a typo in

```
matches one char from 'a' to 'a'.
```

Read More

1 Share Reply



(/hacker_007_)

hacker_007_ (/hacker_007_) ★ 25 16 hours ago

my really simple C++ solution in O(n) time and O(1) space

Read More

1 Share Reply

SHOW 2 REPLIES

ayaankhan (/ayaankhan) ★ 0 9 minutes ago

A basic solution for basic question, prettry straight forward

(/ayaankhan)

```
class Solution {
public:
    bool detectCapitalUse(string word) {
```

Read More

0 Share Reply

guybrush2323 (/guybrush2323) ★ 1 22 minutes ago

My C++ Solution (it breaks out of the loop as soon as the string is invalid):

(/guybrush2323)

```
    static constexpr bool isCapital(char c) noexcept {
        return c >= 'A' && c <= 'Z';
    }
```

Read More

0 Share Reply

NirmalSilwal (/nirmalsilwal) ★ 0 37 minutes ago

My Java solution:

(/nirmalsilwal)

```
class Solution {
            public boolean detectCapitalUse(String word) {
            if (word.equals(word.toUpperCase())){
```

Read More

0 Share Reply

‹ 1 2 3 4 5 6 7 ›