

Ace your JavaScript Interview. [Get my ebook](#). 100 solved Javascript, 20 solved React, & 2 frontend system design questions (**1160+ copies sold**). Get a [Free preview](#).

Advertisements

# Count number of sub-string occurrence in a string

Posted on [December 5, 2018](#) | by [Prashant Yadav](#)

Posted in [Algorithms](#), [String](#) | Tagged [Easy](#)

Algorithm to count the number of sub string occurrence in a [string](#)

There are two types of occurrences in the string.

- Overlapping
- Non-Overlapping

## Example

**Input:**

**String:-** 'aaa'

**Sub-string:-** 'aa'

**Output:**

*//Overlapping (Will check at every next character)*

**2**

*//'aa' found at first and second position and 'aa' also found at second and third position*

*//Non-Overlapping (Will check at an interval of sub-strings length)*

**1**

*//'aa' found at first and second position then it will check for 3 and 4 the position, as 4 the elements does not exist so it returns 1 count only.*

Copy

Practically  
prepare for  
your  
JavaScript  
interview

[JavaScript  
Revision](#)

[JavaScript-  
Concept Based  
Problems](#)

[Data Structures](#)

[Algorithms](#)

[Machine  
Coding](#)

[Web  
Fundamentals](#)

Advertisements

## Solution

We can solve this using two approaches.

- Using regular expressions, but in this approach, Overlapping is not possible also it is slow.
- We can loop through each alphabet in the string and check for **Overlapping** as well as **Non-Overlapping** sub-strings.

With Regular Expressions (Non-Overlapping).

## Implementation

- We will use String `match()` method which uses regular expressions and returns the arrays of matching `objects`

```
//For case sensitive
var str = "Checking in regular expressions.";
var count = (str.match(/in/g) || []).length; // /g checks whole string till end
console.log(count);

//For case - insensitive
var str = "Checking IN regular expressions In case - iNsensitive.";
var count = (str.match(/in/gi) || []).length; // /gi check whole string and
ignores case sensitivity
console.log(count);
```

### Output:

```
2
4
```

Time Complexity:  $O(1)$ .

Space Complexity:  $O(n)$  where **n** is the no of matches found.

## Time and Space complexity

- We are using String `match()` method to solve this, Which uses regular expressions. Regular expressions are quite slow as there running time depends upon their implementation, but this operation is performed in one go so assume Time Complexity is  $O(1)$ , where  $O(1)$  can take any time.
- As String `match()` method returns array of matched objects so the Space Complexity is  $O(n)$ , where **n** is the no of matched objects.

---

## Naive approach using brute force method

### Implementation (Non-Overlapping)

- We will start at 0 position and check if sub-string exist's in the given string.
- As we are not checking overlapping sub-strings so we will increment the position with the length of sub-string.
- We continue this until no sub-string is found.
- To solve this we will be using String `indexOf()` method.

[Copy](#)

```
function countSubStrings(string, subString) {  
    //if string and sub-string is undefined then append it with blank strings as  
    we are going to use string method.  
    string += "";  
    subString += "";  
  
    if (subString.length <= 0) return (string.length + 1);  
  
    var count = 0, pos = 0, step = subString.length;  
  
    while (true){  
        pos = string.indexOf(subString, pos);  
        if (pos >= 0) {  
            ++count;  
            pos += step;  
        } else{  
            break;  
        }  
    }  
    return count;  
}
```

[Copy](#)**Input:**

```
console.log(countSubStrings('foofoobar', 'foo'));  
console.log(countSubStrings('aaaa', 'aaa'));
```

**Output:**

```
2
```

```
1
```

Time Complexity:  $O(n * m)$  where **n** is the length of the string and **m** is the length of the sub-string.

Space Complexity:  $O(1)$ .

## Time and Space Complexity

- We are using String [indexOf\(\)](#) method for checking the sub-string at interval of sub-strings length(m) and we are doing it on whole string(n), so Time Complexity is  $O(m * n)$ .
- We are using fixed space to store the values so the Space Complexity is  $O(1)$ .

---

## Implementation (Overlapping)

- We will start at 0 position and check if sub-string exist's in the given string.
- As we are checking overlapping sub-strings so we will increment the position by 1.
- We continue this until no sub-string is found.
- To solve this we will be using String [indexOf\(\)](#) method.

[Copy](#)

```
function countSubStrings(string, subString) {  
    //if string and sub-string is undefined then append it with blank strings as  
    we are going to use string method.  
    string += "";  
    subString += "";  
  
    if (subString.length <= 0) return (string.length + 1);  
  
    var count = 0, pos = 0, step = 1;  
  
    while (true){  
        pos = string.indexOf(subString, pos);  
        if (pos >= 0) {  
            ++count;  
            pos += step;  
        } else{  
            break;  
        }  
    }  
    return count;  
}
```

[Copy](#)**Input:**

```
console.log(countSubStrings('foofoobar', 'foo'));  
console.log(countSubStrings('aaaa', 'aaa'));
```

**Output:**

```
2  
2
```

Time Complexity:  $O(n * m)$  where **n** is the length of the string and **m** is the length of the sub-string.

Space Complexity:  $O(1)$ .

Advertisements



## Time and Space Complexity

- We are using String [indexOf\(\)](#) method for checking the sub-string at interval of sub-strings length(m) and we are doing it on whole string(n), so Time Complexity is  $O(m * n)$ .
- We are using fixed space to store the values so the Space Complexity is  $O(1)$ .

---

## Summing Overlapping and Non-Overlapping in single function

[Copy](#)

```
//We are using ES6 predefined function value
function countSubStrings(string, subString, overlapping = false) {
    //if string and sub-string is undefined then append it with blank strings as
    we are going to use string method.
    string += "";
    subString += "";

    if (subString.length <= 0) return (string.length + 1);

    var count = 0, pos = 0, step = overlapping ? 1 : subString.length;

    while (true){
        pos = string.indexOf(subString, pos);
        if (pos >= 0) {
            ++count;
            pos += step;
        } else{
            break;
        }
    }
    return count;
}
```

[Copy](#)**Input:**

```
console.log(countSubStrings('foofoobar', 'foo'));
console.log(countSubStrings('aaaa', 'aaa'));
console.log(countSubStrings('aaaa', 'aaa', true));
```

**Output:**

```
2
1
2
```

[Prepare for your JavaScript Interview](#)  
[practically on each Interview rounds and grab](#)  
[that job.](#)

[BEGIN LEARNING](#)

## Recommended Posts:

[Count all substrings having character k.](#)[Convert a string to lowercase in javascript](#)[Reverse last k elements in a queue](#)[Program to print all the permutation of string](#)[Find all the armstrong number between two numbers](#)[Implement stack with max and min function](#)[Check if binary tree is symmetric](#)[Recursive Insertion Sort Algorithm](#)[Sum and Product of all the nodes in the linked list which are less than k](#)[Find missing alphabets to make a string panagram](#)[Prev](#)[Next](#)

# Leave a Reply

Your email address will not be published. Required fields are marked \*

Comment

Start typing...

Name\*

Name

Email\*

Email

POST COMMENT

Advertisements



[About Us](#) [Contact Us](#) [Privacy Policy](#) [Advertise](#)



Handcrafted with somewhere in **Mumbai**

© 2023 [LearnersBucket](#) | [Prashant Yadav](#)

