Count all substrings having character k.

Posted on April 9, 2019 | by Prashant Yadav

Posted in Algorithms, String | Tagged medium

An algorithm to count all the substrings of the given string which has the character k at least once.

Example

```
Input:
'abb'
'b'

Output:
5
```

For example all the possible substring of the given string `'abb'` are `'a', 'ab', 'abb', 'b', 'bb', 'b'`. So there are `5` substrings which has character `'b'`.

We are going to implement two different algorithms in ES6.

Simple approach

A simple algorithm will be to find all the substrings having character k of the given string and return the count;

Implementation

- We are going to find all the substrings of the given string.
- For each substring we will check if it contains the character k or not using includes() method.
- Return the count of substrings with character k.

Practically prepare for your JavaScript interview

JavaScript Revision
JavaScript-Concept Based Problems
Data Structures
Algorithms
Machine Coding
Web Fundamentals

```
function countSubStrings(str, char){
  let count = 0;

  //Loop twice two find all the substrings
  for(let i = 0; i < str.length; i++){
    for(let j = i; j < str.length + 1; j++){
      //Get the substring
      let sub = str.slice(i, j);

      //Check if the substring has letter k
      //then increase the count
      if(sub.includes(char)){
        count++;
      }
    }
  }

  //return the count
  return count;
}
```

```
Input:
console.log(countSubStrings('abb', 'b'));
console.log(countSubStrings('abcabc', 'c'));

Output:
5
15
```

Time complexity: O(n ^ 3).

Space complexity: O(1).

Time and Space complexity

- We are using nested loops to find all the substrings which will take O(n ^ 2) time and then we are checking if the substring has character k which will take O(n) time. As we are checking this for all the substrings the Time complexity is O(n ^ 2) * O(n) = O(n ^ 3).
- We are using constant space, so Space complexity is O(1).

Optimized way to count all the substrings

The above approach works fine but it is very slow as it uses nested loops. We can optimize this algorithm and count all the substrings with character k in O(n) or linear time.

If we read the problem clearly we will understand that we don't have to find all the substrings with character k, we just need to count them.

All the possible substrings for a given string with length `n` can be calculated using this formula `(n * n + 1) / 2`.

Implementation

- We will first count all the possible substrings of the string using the formula we have `(n * n + 1) / 2`.
- Then we are going to loop through the string and find the all the characters before our given character k.
- We can consider these characters as a string and count their possible substrings.
- As these substrings will not have character k we can reduce their count from the total count.

```
Copy

//Calculate the total combination of the substrings possible
let calc = (str) => {
  let n = str.length;
  return Math.floor((n * (n + 1)) / 2);
}

let countSubstring = (str, char) => {
  //Get the count of possible substrings
  let total = calc(str);

  let temp = '';
  for(let i = 0; i < str.length; i++){
    //Check for the substrings without given character
    if(str[i] !== char){
      temp += str[i];
    }else{
      //Reduce the count of substrings without character from the total
      let tempTotal = calc(temp);
      total -= tempTotal;
      temp = '';
    }
  }

  //Check if there is still substring
  //Then reduce their count from total as well
  if(temp){
    let tempTotal = calc(temp);
    total -= tempTotal;
  }

  //Return the total count;
  return total;
}
```

```
Input:
console.log(countSubstring('abb', 'b'));
console.log(countSubstring('abcabc', 'c'));

Output:
5
15
```

How it works

All the possible substrings for the string `'abcabc'` is

```
'a'
'ab'
'abc'
'abca'
'abcab'
'abcab'
'b'
'bc'
'bca'
'bcab'
'bcab'
'c'
'ca'
'cab'
'cabc'
'a'
'ab'
'abc'
'b'
'bc'
'c'
```

1. The total no of possible substring for `'abcabc'` is `n * n + 1 / 2` = `6 * 7 / 2` = `42 / 2 = 21`.
2. Now we find the substring before our given character `'abcabc'` = `'ab'`.
3. The total count of substring for `'ab'` is `(2 * 3 / 2) = 3`. Reduce this from the total count `21 - 3 = 18` as it is not having our given character in it.
4. Repeat the step 2 and 3 and remove the count for substring `'abcabc'` and then total will be `18 - 3 = 15`.

If the given character is not there at the end of the string then this below code will handle it.

```
//Check if there is still substring
//Then reduce their count from total as well
if(temp){
    let tempTotal = calc(temp);
    total -= tempTotal;
}
```

Time complexity: O(n).

Space complexity: O(1).

Time and Space complexity

- We are looping through all the characters of the string, so Time complexity is O(n).
- We are using constant space, so Space complexity is O(1).

Recommended Posts:

Median of two sorted arrays

Maximum consecutive one's in a binary array

Find the LCM of two numbers in javascript

Program to print the pascal triangle patterns

Longest repeated subsequence

Find largest subarray with equal numbers of 0's and 1's

Binary search in javascript

knapsack problem in Javascript (Bounded & Unbounded)

Difference between square of sum of numbers and sum of square of numbers.

Find inorder predecessor of a given key in a BST.

Prev Next