Number of subarrays with given sum k

Posted on April 20, 2019 | by admin

Posted in Algorithms, Arrays | Tagged Easy

An algorithm to find the number of subarrays with given sum k.

We are going to implement two different algorithms to find the number of subarrays with a given sum in javascript. Everything will be written in ES6.

Bruteforce approach O(n ^ 2)

Implementation

- We will use two loops to traverse all the elements of the given array and find the subarrays.
- If the sum of all the elements of the subarray will equal to k then we will increase the count.

```
function countSubArrays(arr, k){
   //get the size the of the array
   let length = arr.length;

   //Keep the count
   let count = 0;

   //traverse through the array
   for(let i = 0; i < length; i++){
      //temp variables to store the sum
      let sum = 0;

      //traverse through the every next element after i
      for(let j = i; j < length; j++){
         sum += arr[j];

         //if sum is equal to k then increase the count.
         if(sum === k){
           count++;
         }
      }
   }

   return count;
}
```

```
Input:
console.log(countSubArrays([3,4,-7,1,3,3,1,-4], 7));

Output:
4
```

Time complexity: O(N ^ 2).

Space complexity: O(1).

- We are using nested loops to find all the subarrays, so Time complexity is O(n ^ 2).
- We are using constant space to keep track of the count, so Space complexity is O(1).

Using hashtable to find the number of subarrays with given sum

The brute force solution works fine but it is slow. An effecient solution is to use map to find all the subarrays with given sum k and count them.

Implementation

- We will use a hashmap to keep track of sum of all the elements of the array.
- Then we will loop through the each element of array and add them to the current sum.
- If the current sum is equal to the given sum k then increase the count.
- If there is a element with the difference of current sum and given sum k in the hashmap then add its count to the total count.

- If the current sum is not in the hashmap then add it with count 1, else increment its count and repeat this for each element of the array.

```
function countSubArrays(arr, sum){
   //HashMap to keep track of the elements
   let prevSum = new Map();

   //To count the subarrays
   let count = 0;

   // Sum of elements so far.
   let currsum = 0;

   for (let i = 0; i < arr.length; i++) {

      // Add current element to sum so far.
      currsum += arr[i];

      // If currsum is equal to desired sum,
      // then a new subarray is found.
      // So increase count of subarrays.
      if (currsum == sum){
           count++;
      }

      // currsum has element
      // then add it to the count
      if (prevSum.has(currsum - sum)) {
          count += prevSum.get(currsum - sum);
      }


      // Add currsum value to count of
      // different values of sum.
      let total = prevSum.get(currsum);
      if (!total) {
          prevSum.set(currsum, 1);
      }
      else{
         prevSum.set(currsum, total+1);
      }
  }

  return count;
}
```

```
Input:
console.log(countSubArrays([3,4,-7,1,3,3,1,-4], 7));

Output:
4
```

Time complexity: O(n).

Space complexity: O(n).

- We are traversing each element of the array, so Time complexity is O(n).
- We are using hashmap to store the `prevsum`, so Space complexity is O(n).

Recommended Posts:

Alternatively merge two different arrays

Minimum characters to delete to make string anagram

Program to sort only positive numbers of the array

Factorial program in javascript

Print all subarrays with a given sum k in an array

Implement a Stack using Queue

Count number of sub string recursively

How to find elements with indexof in javascript

Program to print the chess board pattern in javascript

How to find length of an array in javascript

Prev

Next

Comments

Damir Črnila says:

September 13, 2019 At 3:24 Pm

This code is incomplete.

Try it with:

[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29, 30,31,32]

The result is much more than only 2.

Reply

Prashant Yadav says:

September 13, 2019 At 5:00 Pm

Here we are talking about contagious sub arrays, so sub array with sum 7 in your abo ve array is {3, 4} and {7}. {2, 5} and {1, 6} are sub arrays but they are not contagious.

Checkout this alogrithm

```
function findSum(arr, k){
  //use set to store unique elements
  var hashMap = new Set();

  //Initialize to false
  var count = 0;

  //Loop through each element
  for(var i = 0; i < arr.length; i++){
     //Check if difference pair already exists in hashMap
     if(hashMap.has(k - arr[i])){
       console.log(k-arr[i], arr[i]);
       count++;
     }

     hashMap.add(arr[i]);
  }
  return count;
}
```

This is what you are looking for.

Output for this is 3.

[3, 4]

[2, 5]

[1, 6]

Reply

Leave a Reply

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

Comment

Start typing...

Name*

Name

Email*

Email

POST COMMENT

Advertisements