Longest Subarray with Sum 'k'

Longest Subarray With Sum K.

Moderate • 80/80 • Average time to solve is 30m

Contributed by Amit | 388 upvotes | Asked in companies

Problem statement Send feedback

Ninja and his friend are playing a game of subarrays. They have an array 'NUMS' of length 'N'. Ninja's friend gives him an arbitrary integer 'K' and asks him to find the length of the longest subarray in which the sum of elements is equal to 'K'.
Ninjas asks for your help to win this game. Find the length of the longest subarray in which the sum of elements is equal to 'K'.
If there is no subarray whose sum is 'K' then you should return 0.

Example:

Input: 'N' = 5, 'K' = 4, 'NUMS' = [1, 2, 1, 0, 1]

Output: 4

There are two subarrays with sum = 4, [1, 2, 1] and [2, 1, 0, 1]. Hence the length of the longest subarray with sum = 4 is 4.

Detailed explanation (Input/output format, Notes, Images)

Constraints :

1 <= T <= 10
1 <= N <= 10^5
-10^6 <= NUMS[i] <= 10^6
-10^6 <= K <= 10^6

Sum of N Over all the Test cases <= 10^5

In Subarray sum equals 'k', we counted no of subarrays that can have sum equals 'k'

But now, we look at lengths of such equal sum of subarrays and extract max length out of them

So, in Subarray sum equals k, we store Occurances of each elements for counting purpose,
Here we store indices to calculate length

Example : arr[] => { 1, 2, 1, 0, 1 }

[1, 2, 1, 0, 1], k = 4

Iterations	prefixSum (x)	remainingSum (x-k)	map	length
0th idx				
1	1	1 - 4 = -3	1 - 0	-
2	3	3 - 4 = -1	1 - 0 3 - 1	-
3	4	4 - 4 = 0 (x == k)	1 - 0 3 - 1 4 - 2	length = index + 1 = 3
4	4	4 - 4 = 0		length = index + 1 = 4
5	5	5 - 4 = 1	1 - 0 3 - 1 5 - 4 0 - 3	length = max(length, i - map.get(remainingSum))

Except this portion is Our answer

```
map<Integer, Integer> Map = new HashMap<>();

for(int i=0; i<n; i++)
{
    prefixSum += arr[i];

    if(prefixSum == k)
        length = i+1;

    if(!map.containsKey(prefixSum))
        map.put(prefixSum, i)

    int remainingSum = prefixSum - k;

    if(map.containsKey(remainingSum))
    {
        length = max(length, i - map.get(remainingSum));
    }
```

→ if sum == k → then our subarray is starting from 0th idx

put prefixSum into map with index

take (x-k) → remainingSum

if you find remainingSum in map update length