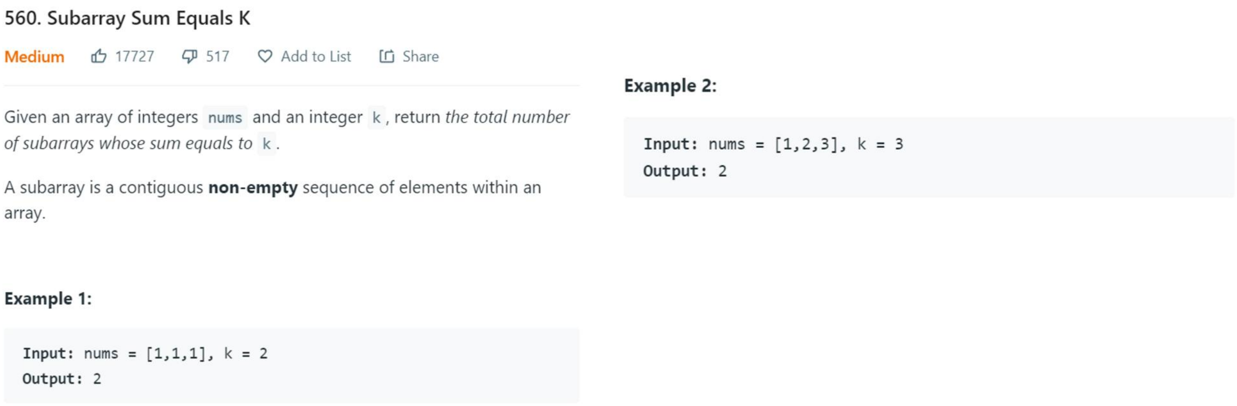
31. Count Subarray sum Equals K



560. Subarray Sum Equals K

Medium 17727 517 Add to List Share

Given an array of integers `nums` and an integer `k`, return *the total number of subarrays whose sum equals to* `k`.

A subarray is a contiguous **non-empty** sequence of elements within an array.

Example 1:

```
Input: nums = [1,1,1], k = 2
Output: 2
```

Example 2:

```
Input: nums = [1,2,3], k = 3
Output: 2
```

Number of subarray with sum k

arr[] = [1, 2, 3, -3, 1, 1, 1, 4, 2, -3]

k=3

Subarray: Contigous part of array

→ 1, 1, 1 ✓ → 1, 4, 2 ✗ (Not subarray, but subsequences)

→ it can be entire array, single element, ...

arr[] = [1, 2, 3, -3, 1, 1, 1, 4, 2, -3] k=3

[1, 2] [1, 2, 3, -3] [3] [2, 3, -3, -1]

[3, -3, 1, 1, 1] [1, 1, 1] [4, 2, -3] [-3, 1, 1, 1, 4, 2, -3]

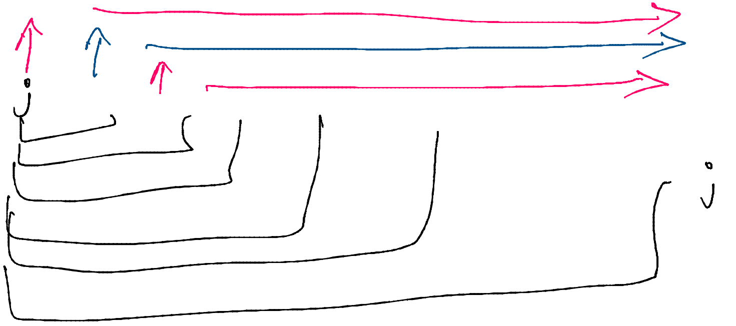
Total ⇒ 8 subarray

i) Brute Force :

→ Generate all subarray

- Start with one element and keep adding them

i
arr[] = [1, 2, 3, −3, 1, 1, 1, 4, 2, −3]



- j will start from 'i' till 'n-1'.
- Aagin move i and same 'j' will start from 'i' till 'n-1'.

Pseudo Code :

subarray is from [i..j]

```
cnt = 0
for(i=0; i<n; i++)
{
    for(j=i; j<n; j++)
    {
        sum = 0
        for(k = i → j) //as subarray is from i → j.
        {
            sum = sum + arr[k];
        }
        if(sum == k)
            cnt ++;
    }
}
```

T.C $\Rightarrow \simeq O(n^3)$ S.C $\Rightarrow O(1)$

2) Better Solution

$$arr[\,] = [\underbrace{\overset{i}{1}, 2, 3, -3, \overset{j}{1}}, 1, 1, 4, 2, -3]$$

→ when j move keep adding them to the sum.

→ Instead of accumlating the sum by running a loop from i to j, don't do everytime instead keep a sum = 0 and everytime there is an addition, add the number.

```
cnt=0
for(i=0; i<n; i++)
{   sum=0
  for (j = i; j<n; j++)
  {
      sum += arr [j];
    }
  }
if (sum == k)
    cnt ++;
```

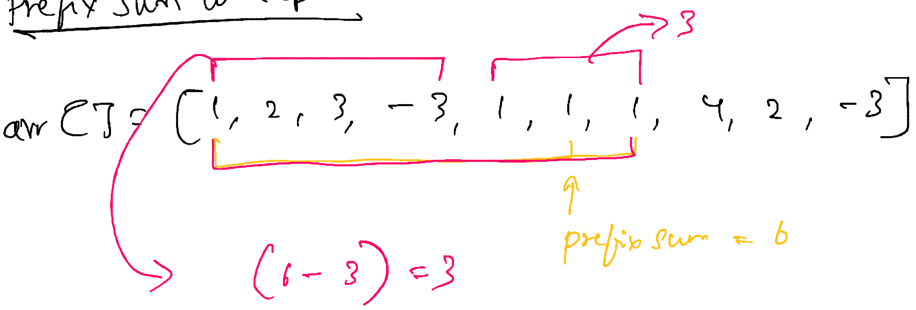
T.C $\Rightarrow \simeq O(n^2)$

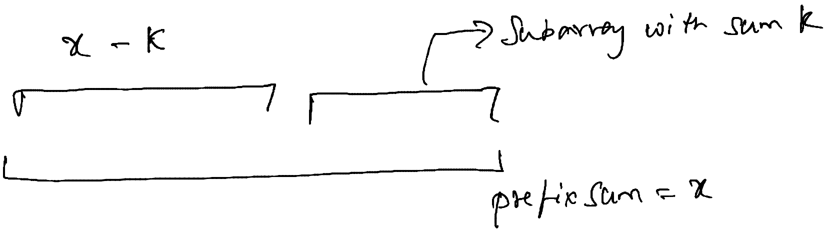
S.C $\Rightarrow O(1)$

3rd Approach :

Using the concept of prefix sum. (Refer L18)

Prefix Sum concept:





$x - k == $ subarray with sum k

↳ The value sum.

Dry Run:

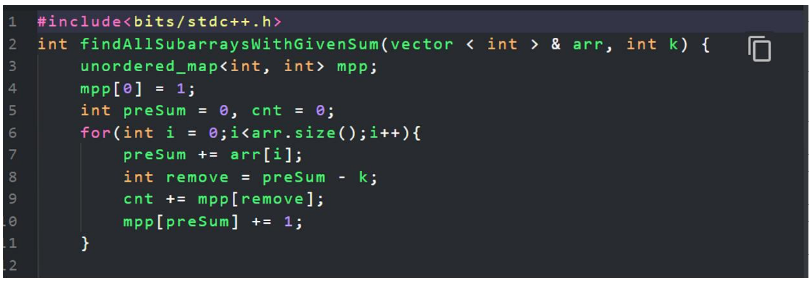
arr[] = [1, 2, 3, -3, 1, 1, 1, 4, 2, -3]   k=3

preSum = ~~0~~ ~~1~~ ~~3~~ ~~6~~ ~~3~~ ~~4~~ ~~5~~ ~~6~~ ~~10~~ ~~12~~ 9

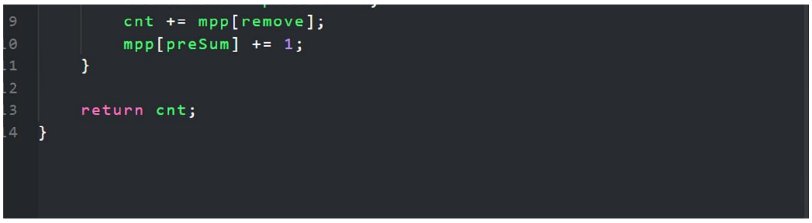
cnt = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~6~~ 8

(10, 1)
(5, 1)
(4, 1)
(6, ~~1~~) 2
(3, ~~2~~ ~~1~~) 2
(1, 1)
(0, 1)

(preSum, cnt)



```cpp
#include<bits/stdc++.h>
int findAllSubarraysWithGivenSum(vector < int > & arr, int k) {
    unordered_map<int, int> mpp;
    mpp[0] = 1;
    int preSum = 0, cnt = 0;
    for(int i = 0;i<arr.size();i++){
        preSum += arr[i];
        int remove = preSum - k;
        cnt += mpp[remove];
        mpp[preSum] += 1;
    }
```

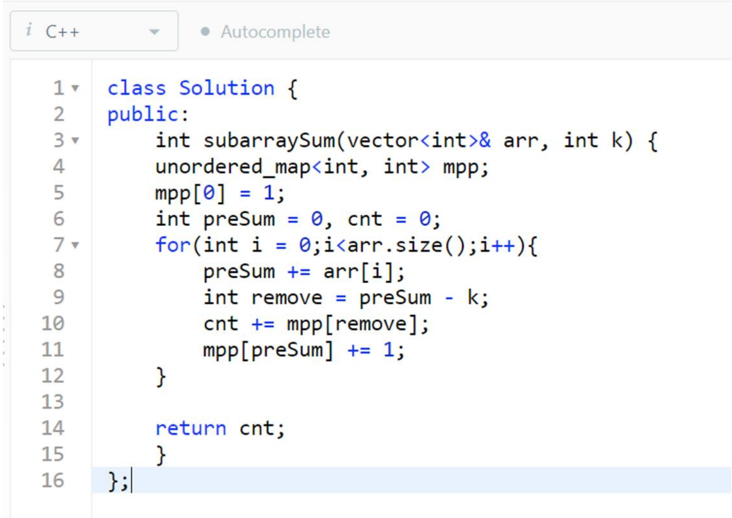


```
        cnt += mpp[remove];
        mpp[preSum] += 1;
    }

    return cnt;
}
```

(preSum, cnt)

key val



```
class Solution {
public:
    int subarraySum(vector<int>& arr, int k) {
    unordered_map<int, int> mpp;
    mpp[0] = 1;
    int preSum = 0, cnt = 0;
    for(int i = 0;i<arr.size();i++){
        preSum += arr[i];
        int remove = preSum - k;
        cnt += mpp[remove];
        mpp[preSum] += 1;
    }

    return cnt;
    }
};
```

T.C ⇒ O(N log N)

S.C ⇒ O(N)