

56m left

BETA Can't read the text? Switch theme

4. Stable Segments

ALL

In an organization, there are n servers, each with a capacity of $capacity[i]$. A contiguous subsegment $[l, r]$ of servers is said to be stable if $capacity[l] = capacity[r] = sum[l+1, r-1]$. In other words, the capacity of the servers at the endpoints of the segment should be equal to the sum of the capacities of all the interior servers.

Find the number of stable sub-segments of length 3 or more.

Example

For example, $n=5$ and $capacity=[9, 3, 3, 3, 9]$.

Segment	First & Last Capacity	Interior Capacity Sum	Balanced
[9, 3, 3]	9	6	No
[3, 3, 3]	3	3	Yes
[3, 3, 9]	3	3	No
[9, 3, 3, 3]	9	6	No
[3, 3, 3, 9]	3	9	No
[9, 3, 3, 3, 9]	9	9	Yes

There are 2 stable subsegments: [3, 3, 3] and [9, 3, 3, 3, 9].

Language C++20

```

1 > #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5  * Complete the countStableSegments function.
6  * The function is expected to return an integer.
7  * The function accepts INTEGER_ARRAY capacity as parameter.
8  */
9
10
11
12
13
14
15
16
17 int countStableSegments(vector<int> capacity) {
18     int n = capacity.size();
19     int cnt = 0;
20     unordered_map<int, int> prefixSumMap;
21     int prefixSum = 0;
22
23     for(int i = 0; i < n; i++) {
24         prefixSum += capacity[i];
25
26         if(i >= 2) {
27             int prevPrefixSum = prefixSum - capacity[i];
28             cnt += prefixSumMap[prevPrefixSum];
29             prefixSumMap[prevPrefixSum]++;
30         }
31
32         if(prefixSum == 0) {
33             cnt++;
34         }
35     }
36
37     return cnt;
38 }
39
40 > int main() ...

```

$capacity[l] = sum[l+1, r-1]$. In other words, the capacity of the servers at the endpoints of the segment should be equal to the sum of the capacities of

segments of length 3 or

[9, 3, 3, 3, 9].

Interior Capacity Sum	Balanced
	No
	Yes
	No
	No
	No
	Yes

and [9, 3, 3, 3, 9].

```

15 // The function accepts INTEGER_ARRAY capacity as parameter.
16
17 int countStableSegments(vector<int> capacity) {
18     int n = capacity.size();
19     int cnt = 0;
20     unordered_map<int, int> prefixSumMap;
21     int prefixSum = 0;
22
23     for(int i = 0; i < n; i++) {
24         prefixSum += capacity[i];
25
26         if(i >= 2) {
27             int prevPrefixSum = prefixSum - capacity[i];
28             cnt += prefixSumMap[prevPrefixSum];
29             prefixSumMap[prevPrefixSum]++;
30         }
31
32         if(prefixSum == 0) {
33             cnt++;
34         }
35     }
36
37     return cnt;
38 }
39
40 > int main() ...

```

Test Results

Custom Input