Subarray Sum Equals K

Question:

https://leetcode.com/problems/subarray-sum-equals-k/

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

Example 1:

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

Example 2:

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

Approach 1:

Using brute force to check each subarray sum.
Note: The sum() function takes O(n) time.

Solution 1:

```
def subarraySum(self,nums,k):
    count = 0
    for i in range(len(nums)):
        for j in range(i, len(nums)):
            if sum(nums[i:j+1]) == k:
                count += 1
    return count
```

Time Complexity: O(n^3)

Space Complexity: O(1)

Approach 2:

Optimised approach to reduce the time complexity to O(n^2) by applying space-time tradeoff using HashMap.
Storing the count of similar elements and returning total values of the equal sum.

Solution 2:

```
def subarraySum(self,nums,k):
        d = {}
        d[0] = 1
        s = 0
        count = 0
        for i in range(len(nums)):
                s += nums[i]
                if s-k in d: # --- I
                        count += d[s-k]
                if s in d:
                        d[s] += 1 # --- II
                else:
                        d[s] = 1

        return count
```

Time Complexity: O(n^3)

Space Complexity: O(1)