LeetCode 560. Subarray Sum Equals K

1. Problem Title & Link

- **560. Subarray Sum Equals K**
- https://leetcode.com/problems/subarray-sum-equals-k/

2. Problem Statement (Short Summary)

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

3. Examples (Input → Output)

Input: nums = [1,1,1], k = 2
Output: 2
Explanation: Subarrays [1,1] (first and last) each sum to 2.

Input: nums = [1,2,3], k = 3
Output: 2
Explanation: Subarrays [1,2] and [3] both sum to 3.

4. Constraints

- 1 <= nums.length <= 2 * 10^4
- -1000 <= nums[i] <= 1000
- -10^7 <= k <= 10^7

5. Thought Process (Step by Step)

❌ Brute Force (O(n²))

Compute all subarrays' sums using two loops.
For each i..j, calculate sum(nums[i:j]) and check if equals k.
This is easy to write but **too slow** for large inputs.

✅ Optimal (Prefix Sum + HashMap) 💡

We use **Prefix Sum** and a **HashMap** to count subarrays efficiently.

Concept:
Let prefix[i] = sum(nums[0..i]).
Then, for any subarray (j..i):
sum(nums[j..i]) = prefix[i] - prefix[j-1]
We want:
prefix[i] - prefix[j-1] = k
→ prefix[j-1] = prefix[i] - k
So for each prefix[i],
we count **how many previous prefix sums = prefix[i] - k**.
We store prefix sums and their frequencies in a map (prefix_sum_count).

Algorithm Steps:

1. Initialize count = 0, prefix_sum = 0, and map = {0:1} (important — one empty prefix).
2. Traverse each number num in nums:
   - Update prefix_sum += num
   - If (prefix_sum - k) exists in map → found matching subarray(s)
   - Add its frequency to count
   - Update map with prefix_sum frequency

6. Pseudocode

```
count = 0
prefix_sum = 0
map = {0: 1}

for num in nums:
    prefix_sum += num
    if prefix_sum - k in map:
        count += map[prefix_sum - k]
    map[prefix_sum] = map.get(prefix_sum, 0) + 1

return count
```

7. Code Implementation

✅ Python

```
class Solution:
    def subarraySum(self, nums: List[int], k: int) -> int:
        count = 0
        prefix_sum = 0
        prefix_map = {0: 1}

        for num in nums:
            prefix_sum += num
            if prefix_sum - k in prefix_map:
                count += prefix_map[prefix_sum - k]
            prefix_map[prefix_sum] = prefix_map.get(prefix_sum, 0) + 1

        return count
```

✅ Java

```
class Solution {
    public int subarraySum(int[] nums, int k) {
        Map<Integer, Integer> map = new HashMap<>();
        map.put(0, 1);
        int count = 0, prefixSum = 0;

        for (int num : nums) {
            prefixSum += num;
            if (map.containsKey(prefixSum - k))
```

```
                count += map.get(prefixSum - k);
            map.put(prefixSum, map.getOrDefault(prefixSum, 0) + 1);
        }

        return count;
    }
}
```

8. Time & Space Complexity

- **Time:** O(n)
- **Space:** O(n) (for prefix map)

9. Dry Run (Step-by-Step Execution)

👉 Input: nums = [1, 2, 3], k = 3

Step	num	prefix_sum	prefix_sum - k	Exists?	count	prefix_map
1	1	1	-2	❌	0	{0:1, 1:1}
2	2	3	0	✅	1	{0:1, 1:1, 3:1}
3	3	6	3	✅	2	{0:1, 1:1, 3:1, 6:1}

✅ Output = 2

(Subarrays [1,2] and [3])

10. Concept Insight Table

Core Concept	Common Use Cases	Common Traps	Builds / Next Steps
Prefix Sum + HashMap Counting — use cumulative sums to identify subarray sums quickly.	- Count subarrays matching condition - Continuous sum problems - Range-sum queries	- Forgetting to add {0:1} initially - Misunderstanding prefix_sum - k logic - Using set instead of map (need frequencies)	🔷 Builds to **LeetCode 437 (Path Sum III)** 🔷 Related to **LeetCode 523 (Continuous Subarray Sum)** 🔷 Introduces **HashMap-based Prefix DP** pattern

11. Common Mistakes / Edge Cases

- Forgetting to handle subarrays starting at index 0 → need {0:1} in map.
- Overwriting map counts instead of incrementing.
- Confusing **subarray sum equals K** vs **subsequence sum equals K** (order and continuity matter).

12. Variations / Follow-Ups

- Find **longest subarray** with sum = K → similar prefix idea, store first occurrence.
- Find **subarray sum divisible by K** → (LC 523).
- Use **prefix XOR** for bitwise-sum version.