

■ Longest Subarray with Sum K

Given an array of integers and an integer K, find the length of the longest subarray whose sum is equal to K.

■ Brute Force Solution ($O(n^2)$)

- Iterate over all possible subarrays using two loops. - For each subarray, compute the sum. - If the sum equals K, update the longest length. - Time Complexity: $O(n^2)$ - Space Complexity: $O(1)$

Python Code:

```
class Solution: def longestSubarray(self, arr, k): n = len(arr) longest = 0 for i in range(n): total = 0 for j in range(i, n): total += arr[j] if total == k: longest = max(longest, j - i + 1) return longest
```

■ Optimized Solution using Hashing ($O(n)$)

- Use a dictionary (hash map) to store the first occurrence of prefix sums. - Keep track of the cumulative sum while traversing the array. - If (current_sum - k) exists in the map, we found a subarray with sum k. - Update the longest length accordingly. - Time Complexity: $O(n)$ - Space Complexity: $O(n)$

Python Code:

```
class Solution: def longestSubarray(self, arr, k): myDict = dict() total = 0 longest = 0 for i in range(len(arr)): total += arr[i] if total == k: longest = max(longest, i + 1) rem = total - k if rem in myDict: longest = max(longest, i - myDict[rem]) if total not in myDict: myDict[total] = i return longest
```