Total Number of Occurrences of an Element in a Sorted Array.

Given a sorted array of integers and a target element, you need to find the total number of occurrences of the target element in the array.

Approach 1: Find the first and last occurrence of the target element using Linear Search.

This approach uses linear search to find the first and last occurrence of the target element in the array.

It initializes a counter variable count to 0.

It iterates through each element in the array and checks if the element is equal to the target.

If the element matches the target, it increments the count variable.

Finally, it returns the value of the count.

The time complexity is O(n), where n is the size of the array, as it performs a linear search through the array.

The space complexity is O(1) since it only uses a constant amount of extra space for variables and does not require additional data structures.

Approach 2: Find the first and last occurrence of the target element using Binary Search.

This approach uses binary search to find the first and last occurrence of the target element in the array.

It initializes first and last variables to -1.

The first binary search finds the index of the first occurrence of the target element:

It uses two pointers, low and high, initialized to the start and end of the array, respectively.

It calculates the mid index using the formula low + (high - low) / 2 to avoid integer overflow.

If the middle element is equal to the target, it updates the first variable with the mid index and moves the high pointer to search for earlier occurrences.

If the middle element is greater than the target, it updates the high pointer to search in the lower half of the remaining array.

If the middle element is less than the target, it updates the low pointer to search in the upper half of the remaining array.

The binary search continues until low becomes greater than high.

The second binary search finds the index of the last occurrence of the target element:

It follows a similar approach to the first binary search but updates the last variable when the middle element matches the target and adjusts the low and high pointers accordingly.

Finally, it returns the total number of occurrences by subtracting first from last and adding 1.

The time complexity is O(log n), where n is the size of the array, as it uses binary search to find the first and last occurrence of the target element.

The space complexity is O(1) since it only uses a constant amount of extra space for variables and do not require additional data structures.