Find First and Last Occurrence of an element in a Sorted Array CodeStudio

The problem is to find the first and last occurrence of a target element in a given sorted array.

Input: 0 5 5 6 6 6, target: 3

Output: -1, -1

Input: 0 0 1 1 2 2 2 2, target: 2

Output: 4, 7

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 iterates through the array and updates the first and last indices accordingly.

This approach has a **time complexity of O(n), where n is the size of the array.**

This approach has a **space complexity of O(1) because they only use a constant amount of extra space to store the indices first and last.**

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

Initialization:

Initialize first and last variables to -1, which will store the indices of the first and last occurrences of the target element, respectively.

Set low as 0, representing the leftmost index of the array, and high as n-1, representing the rightmost index of the array.

Define a mid variable to store the middle index during the binary search.

Finding the First Occurrence:

Use a while loop to perform binary search until low is less than or equal to high.

Calculate mid as low + (high - low) / 2 to avoid integer overflow.

Compare the element at the mid index with the target element.

If they are equal, update first to mid and move the high pointer to the left (mid - 1) to search for a potential earlier occurrence of the target element.

If the element at mid is greater than the target, update high to (mid - 1) to search in the left half of the array.

If the element at mid is smaller than the target, update low to (mid + 1) to search in the right half of the array.

Finding the Last Occurrence:

Reset low and high to the initial values (0 and n-1).

Use another while loop to perform binary search until low is less than or equal to high.

Calculate mid as low + (high - low) / 2.

Compare the element at the mid index with the target element.

If they are equal, update last to mid and move the low pointer to the right (mid + 1) to search for a potential later occurrence of the target element.

If the element at mid is greater than the target, update high to (mid - 1) to search in the left half of the array.

If the element at mid is smaller than the target, update low to (mid + 1) to search in the right half of the array.

Return the Result:

Return a pair of values (first, last) representing the indices of the first and last occurrences of the target element, respectively.

This approach has a **time complexity of O(log n), where n is the size of the array.**

This approach has a **space complexity of O(1) because they only use a constant amount of extra space to store the indices first and last.**

Which approach to use:

In case of a sorted array using the Binary search approach is an efficient solution. In case of an unsorted array the linear search approach will be efficient and the binary search approach will not work.