Day 12

Assignment 8: Circular Queue Binary Search

Consider a circular queue (implemented using a fixed-size array) where the elements are sorted but have been rotated at an unknown index. Describe an approach to perform a binary search for a given element within this circular queue.

A)

Performing a binary search on a circularly sorted queue involves adapting the traditional binary search algorithm to handle the circular nature of the queue. Here's a step-by-step approach:

Find the Rotation Index:

Perform a modified binary search to find the rotation index, i.e., the index where the circular rotation begins.
The rotation index is the index of the smallest element in the circularly sorted queue.
Perform Binary Search:

Based on the rotation index, decide which half of the circular queue to search.
Use a modified binary search algorithm to search within the appropriate half.
Adjust indices to wrap around the circular queue if necessary.

Here's the code implementation in Java:
```
package circulequeue;
public class CircularQueue {
        public static int search(int[] nums, int target) {
        int rotationIndex = findRotationIndex(nums);
        int left = binarySearch(nums, target, 0, rotationIndex - 1);
        int right = binarySearch(nums, target, rotationIndex, nums.length - 1);
        return (left != -1) ? left : (right != -1) ? right : -1; }
        private static int findRotationIndex(int[] nums) {
        int left = 0;
        int right = nums.length - 1;
        while (left < right) {
        int mid = left + (right - left) / 2;
        if (nums[mid] > nums[right]) {
        left = mid + 1;
        } else {
        right = mid;
        }
        }
        return left;
        }
        private static int binarySearch(int[] nums, int target, int left, int right) {
        while (left <= right) {
        int mid = left + (right - left) / 2;
        if (nums[mid] == target) {
        return mid;
        } else if (nums[mid] < target) {
        left = mid + 1;
        } else {
        right = mid - 1;
        }
        }
        return -1; // Element not found
        }
        public static void main(String[] args) {
        int[] nums = {4, 5, 6, 7, 0, 1, 2};
        int target = 6;
        int index = search(nums, target);
        System.out.println("Index of " + target + " in the circular queue: " + index);
        }
        }
```

Explanation:

The search method is used to find the index of a target element in the circularly sorted array nums. It first finds the rotation index using the findRotationIndex method, then performs binary search on both sides of the rotation index using the binarySearch method.

The findRotationIndex method finds the rotation index, which is the index of the smallest element in the rotated array.

The binarySearch method performs a standard binary search to find the target element within a specified range of indices.

In the main method, an example array nums is defined, and the search method is called to find the index of the target element 6. The result is then printed.

Output:

```
Index of 6 in the circular queue: 2
```

In this output, the target element 6 is found at index 2 in the circularly sorted array nums.