.DAY 12 :

Task 8: Circular Queue Binary Search

Q) 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

EXPLANATION:

Class Overview

The CircularQueue class is designed to search for an element in a circular (rotated) sorted array. It consists of three main methods: search, findRotationIndex, and binarySearch. Each of these methods plays a specific role in achieving the search functionality.

Method Descriptions

search(int[] nums, int target)

Purpose : This is the main method used to find the index of a target element within a rotated sorted array.

Process:

1. Find Rotation Index : It first determines the rotation index, which is the position of the smallest element in the array. This helps in identifying where the rotation occurred.

2. Binary Search in Two Halves : Once the rotation index is identified, the array is split into two subarrays: one from the start of the array to just before the rotation index, and the other from the rotation index to the end of the array.

3. Search in Subarrays : It performs a binary search in both subarrays to find the target element. If the target is found in either of the subarrays, its index is return

2. findRotationIndex(int[] nums)

Purpose: This method finds the index of the smallest element in the rotated sorted array, which is the rotation point.

Process:

1. Initialization: It initializes two pointers, left and right, to the start and end of the array respectively.

2. Binary Search for Pivot: Using a while loop and binary search technique, it calculates the mid-point. If the mid-point element is greater than the right-most element, it means the smallest element is to the right of the mid-point. Otherwise, it's to the left or at the mid-point itself.

3. Find the Pivot: This process continues until the smallest element is found, which indicates the rotation index

3. binarySearch(int[] nums, int target, int left, int right)

Purpose : This is a standard binary search method used to find the target element within a specified range of indices in the array.

Process :

1. Initialization : It initializes two pointers, left and right, to the start and end of the given range respectively.

2. Binary Search: Using a while loop, it calculates the mid-point of the range. If the mid-point element equals the target, it returns the mid-point index.

3. Adjust Search Range: If the target is greater than the mid-point element, the search continues in the right half of the current range. If the target is less, the search continues in the left half.

4. Element Not Found: If the target is not found within the specified range, it returns -1.

Execution Flow in main Method

1. Array Initialization : An example rotated sorted array nums and a target element target are defined.

2. Search Method Call: The search method is called with the array and target element as arguments.

3. Output Result : The index of the target element is printed. If the element is found, its index is shown; otherwise, -1 is printed indicating the element is not present in the array.

Summary

Rotation Index : The rotation index is found to determine the point at which the array is rotated.

Binary Search : The array is divided into two parts based on the rotation index, and binary search is performed on each part to find the target element.

Efficiency : This approach leverages the efficiency of binary search, ensuring a time complexity of O(log n), making it a fast and efficient method to search within a rotated sorted array.