Data Structures and Algo in Java - Day 21

public class day21

{

// Search Element in Rotated Sorted Order

// We have to find which side is sorted to check it if the element is available in the sorted part

public static void main(String[] args)

{

// int arr [] = {7,8,9,1,2,3,4,5,6};

// int arr [] = {3,1,2,3,3,3,3,3};

// int arr [] = {7,8,9,0,1,2,3,4,5,6};

// minimumInRotatedArray(arr);

int arr [] = {7,8,9,0,1,2,3,4,5,6};

howManyTimesArrayRotated(arr);

// searchElementInRotated(arr,tar);

// if(searchElementInRotatedDubplicates(arr, tar))

// {

// System.out.println("Its there");

// }

// else

// {

// System.out.println("Its not there");

// }

}

public static void searchElementInRotated(int arr[] , int tar)

{

int low = 0;

int high = arr.length-1;

int ans = -1;

while(low<=high)

{

int mid = (low+high)/2;

if(arr[mid]==tar)

{

ans = mid;

}

// check left half is sorted

if(arr[low]<=arr[mid])

{

if(arr[low]<= tar && tar <= arr[mid])

{

high = mid - 1;

}

else

{

low = mid+1;

}

}

else

{

if(arr[mid]<=tar && tar <= arr[high])

{

low = mid +1;

}

else

{

high = mid -1;

}

}

}

System.out.println(ans);

}

public static boolean searchElementInRotatedDubplicates(int arr [] , int tar)

{

int low = 0;

int high = arr.length-1;

int ans = -1;

while(low<=high)

{

int mid = (low+high)/2;

if(arr[mid]==tar)

{

return true;

}

if(arr[low] == arr[mid] && arr[mid] == arr[high])

{

low = low + 1;

high = high -1;

continue;

}

// check left half is sorted

if(arr[low]<=arr[mid])

{

if(arr[low]<= tar && tar <= arr[mid])

{

high = mid - 1;

}

else

{

low = mid+1;

}

}

else

{

if(arr[mid]<=tar && tar <= arr[high])

{

low = mid +1;

}

else

{

high = mid -1;

}

}

}

return false;

}

public static void minimumInRotatedArray(int arr[])

{

int low = 0;

int high = arr.length-1;

int ans = Integer.MAX\_VALUE;

while(low<=high)

{

int mid = (low+high)/2;

if(arr[low]<=arr[high])

{

ans = Math.min(ans,arr[low]);

break;

}

if(arr[low]<=arr[mid])

{

ans = Math.min(ans,arr[low]);

low = mid + 1;

}

else if(arr[mid]<= arr[high])

{

ans = Math.min(ans,arr[mid]);

high = mid - 1;

}

}

System.out.println(ans);

}

public static void howManyTimesArrayRotated(int arr[])

{

int low = 0;

int high = arr.length-1;

int ans = Integer.MAX\_VALUE;

int index = -1;

while(low<=high)

{

int mid = (low+high)/2;

if(arr[low]<=arr[high])

{

if(arr[low]<ans)

{

index = low;

ans = arr[low];

}

break;

}

if(arr[low]<=arr[mid])

{

if(arr[low]<ans)

{

index = low;

ans = arr[low];

}

low = mid + 1;

}

else if(arr[mid]<= arr[high])

{

if(arr[mid]<ans)

{

index = mid;

ans = arr[mid];

}

high = mid - 1;

}

}

System.out.println(index);

}

}