11/11/2024

1.Floor in sorted array

```
import java.util.*;

public class Solution {
    public int searchInsert(int[] nums, int target) {
        int left = 0, right = nums.length - 1;

        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 left;
    }
```

Time complexity:O(log n)

Space complexity: O(1)

output:

Checkequalarrays

```
class Checkequalarrays {

    public static boolean areEqual(int[] a, int[] b) {
        int n = a.length;
        int m =b.length;

        if (n != m)
            return false;

        Map<Integer,Integer> map=new HashMap<>();
        for (int i = 0; i < n; i++) {
            map.put(a[i], map.getOrDefault(a[i], 0) + 1);
        }

        for (int i = 0; i < n; i++) {
            if (!map.containsKey(b[i]))
                return false;
            int count= map.get(b[i]);
            if (count == 0)
                return false;
            map.put(b[i],count-1);
        }

        return true;
    }
```

time and space complexity :

O(n), O(n)

Output: Yes

3.Palindrome linked list:

```
public class PalindromeLL {
    public boolean isPalindrome(ListNode H) {
        if (H == null || H.next == null) return true;

        ListNode S = H, F = H, P = null;

        while (F != null && F.next != null) {
            F = F.next.next;
            ListNode T = S.next;
            S.next = P;
            P = S;
            S = T;
        }

        if (F != null) S = S.next;

        while (P != null && S != null) {
            if (P.val != S.val) return false;
            P = P.next;
            S = S.next;
        }

        return true;
    }
```

Output: True

Time complexity:O(n)

Space complexity:O(1)

4 Triplet sum in array:

```
public class TripletSumInArray {
    public int[][] threeSum(int[] nums) {
        Arrays.sort(nums);
        List<int[]> result = new ArrayList<>();

        for (int i = 0; i < nums.length; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) {
                continue;
            }

            int j = i + 1;
            int k = nums.length - 1;

            while (j < k) {
                int sum = nums[i] + nums[j] + nums[k];

                if (sum > 0) {
                    k--;
                } else if (sum < 0) {
                    j++;
                } else {
                    result.add(new int[]{nums[i], nums[j], nums[k]});

                    while (j < k && nums[j] == nums[j + 1]) {
                        j++;
                    }

                    while (j < k && nums[k] == nums[k - 1]) {
                        k--;
                    }
```

```
                j++;
                k--;
            }
        }
    }

    return result.toArray(new int[result.size()][]);
  }
```

output: [-1, -1, 2]

[-1, 0, 1]

Time complexity : O(N^2)

Space complexity: O(N)