

DATE: 14-11-2024

CODING PRACTICE PROBLEMS – DAY 4

1. Stock buy and sell

```
class Solution {
    ArrayList<ArrayList<Integer>> stockBuySell(int A[], int n) {
        ArrayList<ArrayList<Integer>> result = new ArrayList<>();

        int i = 0;
        while (i < n - 1) {
            while (i < n - 1 && A[i + 1] <= A[i]) {
                i++;
            }
            if (i == n - 1) break;
            int buy = i++;
            while (i < n && A[i] >= A[i - 1]) {
                i++;
            }
            int sell = i - 1;
            ArrayList<Integer> pair = new ArrayList<>();
            pair.add(buy);
            pair.add(sell);
            result.add(pair);
        }
        return result;
    }
}
```

Time complexity: $O(n)$

OUTPUT:

Output Window

Compilation Results

Custom Input

Compilation Completed

For Input:  

7

100 180 260 310 40 535 695

Your Output:

1

Expected Output:

1

2. Coin Change (Count Ways)

```
class Solution {  
    public int count(int coins[], int sum) {  
        int[] dp = new int[sum + 1];  
        dp[0] = 1;  
  
        for (int coin : coins) {  
            for (int i = coin; i <= sum; i++) {  
                dp[i] += dp[i - coin];  
            }  
        }  
  
        return dp[sum];  
    }  
}
```

Time complexity: $O(n*m)$



OUTPUT:

Output Window

Compilation Results

Custom Input

Compilation Completed

For Input:  

1 2 3

4

Your Output:

4

Expected Output:

4

3. First and Last Occurrences

```
class GFG {
    ArrayList<Integer> find(int arr[], int x) {
        ArrayList<Integer> result = new ArrayList<>();
        result.add(findFirst(arr, x));
        result.add(findLast(arr, x));
        return result;
    }
    private int findFirst(int[] arr, int x) {
        int low = 0, high = arr.length - 1, result = -1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == x) {
                result = mid;
                high = mid - 1;
            } else if (arr[mid] < x) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return result;
    }
    private int findLast(int[] arr, int x) {
        int low = 0, high = arr.length - 1, result = -1;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == x) {
                result = mid;
                low = mid + 1;
            } else if (arr[mid] < x) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return result;
    }
}
```



Time complexity: $O(\log n)$

OUTPUT:

Output Window

Compilation Results Custom Input Y.O.G.I. (AI Bot)

Compilation Completed

For Input:  
1 3 5 5 5 5 6 7 12 3 12 5
5
Your Output:
2 5
Expected Output:
2 5

4. Find Transition Point

```
class Solution {
    int transitionPoint(int arr[]) {
        int low = 0, high = arr.length - 1;
        if (arr[high] == 0) return -1;
        if (arr[0] == 1) return 0;
        while (low <= high) {
            int mid = low + (high - low) / 2;
            if (arr[mid] == 1 && (mid == 0 || arr[mid - 1] == 0)) {
                return mid;
            }
            else if (arr[mid] == 1) {
                high = mid - 1;
            }
            else {
                low = mid + 1;
            }
        }
        return -1;
    }
}
```

Time complexity: $O(\log n)$



OUTPUT:

Output Window

Compilation Results

Custom Input

Compilation Completed

For Input:  

0 0 0 1 1

Your Output:

3

Expected Output:

3

5. First Repeating Element

```
class Solution {
    public static int firstRepeated(int[] arr) {
        Map<Integer, Integer> elementIndexMap = new HashMap<>();
        int minIndex = Integer.MAX_VALUE;
        for (int i = 0; i < arr.length; i++) {
            if (elementIndexMap.containsKey(arr[i])) {
                minIndex = Math.min(minIndex, elementIndexMap.get(arr[i]));
            } else {
                elementIndexMap.put(arr[i], i + 1);
            }
        }
        return minIndex == Integer.MAX_VALUE ? -1 : minIndex;
    }
}
```

Time complexity: $O(n)$


OUTPUT:

Output Window

Compilation Results

Custom Input

Compilation Completed

For Input:  

1 5 3 4 3 5 6

Your Output:

2

Expected Output:

2

6. Remove Duplicates Sorted Array

```
class Solution {
    public int remove_duplicate(List<Integer> arr) {
        if (arr.size() == 0) return 0;

        int uniqueIndex = 1;

        for (int i = 1; i < arr.size(); i++) {
            if (!arr.get(i).equals(arr.get(uniqueIndex - 1))) {
                arr.set(uniqueIndex, arr.get(i));
                uniqueIndex++;
            }
        }
        return uniqueIndex;
    }
}
```


Time complexity: $O(n)$

Output Window

Compilation Results

Custom Input

Compilation Completed

For Input:  

2 2 2 2 2

Your Output:

2

Expected Output:

2

7. Maximum Index

```
class Solution {
    int maxIndexDiff(int[] arr) {
        int n = arr.length;
        if (n <= 1) return 0;
        int[] leftMin = new int[n];
        int[] rightMax = new int[n];

        leftMin[0] = arr[0];
        for (int i = 1; i < n; i++) {
            leftMin[i] = Math.min(arr[i], leftMin[i - 1]);
        }

        rightMax[n - 1] = arr[n - 1];
        for (int j = n - 2; j >= 0; j--) {
            rightMax[j] = Math.max(arr[j], rightMax[j + 1]);
        }

        int i = 0, j = 0, maxDiff = -1;
        while (j < n && i < n) {
            if (leftMin[i] < rightMax[j]) {
                maxDiff = Math.max(maxDiff, j - i);
                j++;
            }
        }
    }
}
```

```

        } else {
            i++;
        }
    }
    return maxDiff;
}
}

```

Time complexity:



OUTPUT:

Output Window

Compilation Results

Custom Input

Compilation Completed

For Input:  

1 10

Your Output:

1

Expected Output:

1

8. Wave Array

```

class Solution {
    public static void convertToWave(int[] arr) {
        for (int i = 0; i < arr.length - 1; i += 2) {
            int temp = arr[i];
            arr[i] = arr[i + 1];
            arr[i + 1] = temp;
        }
    }
}

```


Time complexity: $O(n)$

OUTPUT:

Output Window

Compilation Results

Custom Input

Compilation Completed

For Input:  

1 2 3 4 5

Your Output:

2 1 4 3 5

Expected Output:

2 1 4 3 5