Date : 14/11/2024

DSA Practice Problems

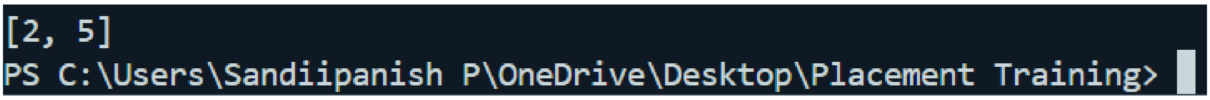
1. First and Last Occurences

Code:

```
import java.util.Arrays;


public class FirstAndLastOccurrences {


    public static int[] findFirstAndLast(int[] arr, int x) {
        int[] result = {-1, -1};
        int first = findFirst(arr, x);
        if (first == -1) {
            return result;
        }
        int last = findLast(arr, x);
        result[0] = first;
        result[1] = last;
        return result;
    }


    private static int findFirst(int[] arr, int x) {
        int low = 0, high = arr.length - 1;
        int 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 static int findLast(int[] arr, int x) {

    int low = 0, high = arr.length - 1;

    int 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;

}
```

```
    public static void main(String[] args) {
        int[] arr1 = {1, 3, 5, 5, 5, 5, 67, 123, 125};
        int x1 = 5;
        System.out.println(Arrays.toString(findFirstAndLast(arr1, x1)));
    }
}
```

Output:



```
[2, 5]
PS C:\Users\Sandiipanish P\OneDrive\Desktop\Placement Training>
```

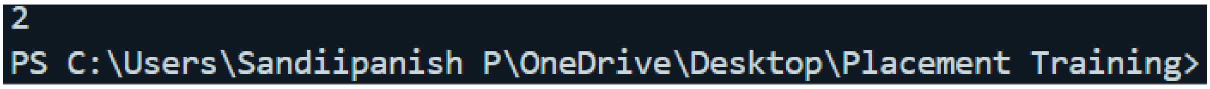
Time Complexity: O(log n)

2. Remove Duplicates in a Sorted Array

Code:

```
public class RemoveDuplicates {

    public static int removeDuplicates(int[] arr) {
        if (arr.length == 0) return 0;

        int uniqueCount = 1;

        for (int i = 1; i < arr.length; i++) {
            if (arr[i] != arr[uniqueCount - 1]) {
                arr[uniqueCount] = arr[i];
                uniqueCount++;
            }
        }

        return uniqueCount;
    }
    public static void main(String[] args) {
        int[] arr1 = {2,2,2,2,2};
        int uniqueSize1 = removeDuplicates(arr1);
        for (int i = 0; i < uniqueSize1; i++) {
            System.out.print(arr1[i] + " ");
        }
        System.out.println();

    }
}
```

Output:



```
2
PS C:\Users\Sandiipanish P\OneDrive\Desktop\Placement Training>
```

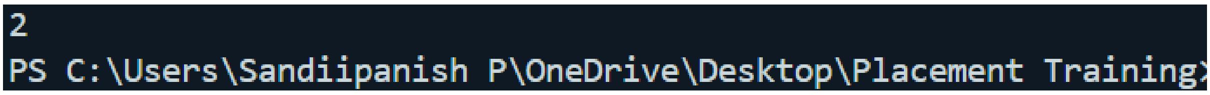
Time Complexity: O(n)

3. First Repeating Element

Code:

```
import java.util.HashMap;

public class FirstRepeatingElement {

    public static int firstRepeatingElement(int[] arr) {
        HashMap<Integer, Integer> map = new HashMap<>();
        int minIndex = Integer.MAX_VALUE;

        for (int i = 0; i < arr.length; i++) {
            if (map.containsKey(arr[i])) {
                minIndex = Math.min(minIndex, map.get(arr[i]));
            } else {

                map.put(arr[i], i);

            }

        }


        return (minIndex == Integer.MAX_VALUE) ? -1 : minIndex + 1;

    }


    public static void main(String[] args) {

        int[] arr1 = {1, 5, 3, 4, 3, 5, 6};

        System.out.println(firstRepeatingElement(arr1));

    }

}
```

Output:



```
2
PS C:\Users\Sandiipanish P\OneDrive\Desktop\Placement Training>
```

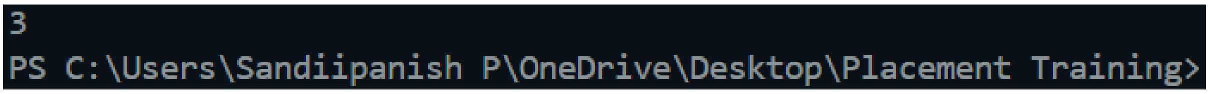
Time Complexity: O(n)

4. Find transition point

Code:

```
public class TransitionPointFinder {

    public static int findTransitionPoint(int[] arr) {

        int l = 0, r = arr.length - 1;


        if (arr[r] == 0) return -1;

        if (arr[l] == 1) return 0;


        while (l <= r) {

            int m = l + (r - l) / 2;
            if (arr[m] == 1 && (m == 0 || arr[m - 1] == 0)) return m;

            else if (arr[m] == 0) l = m + 1;

            else r = m - 1;

        }

        return -1;

    }


    public static void main(String[] args) {

        int[] arr = {0, 0, 0, 1, 1};

        System.out.println(findTransitionPoint(arr));

    }

}
```

Output:



```
3
PS C:\Users\Sandiipanish P\OneDrive\Desktop\Placement Training>
```

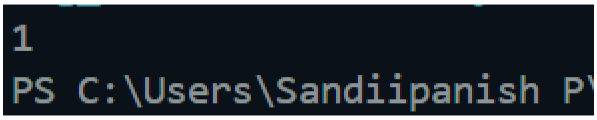
Time Complexity: O(log n)

5. Stock Buy and Sell

Code:

```
import java.util.*;


public class BuyAndSellStocks {

    public static int stockBuySell(int[] prices, int n) {

        List<int[]> trades = new ArrayList<>();

        int i = 0;


        while (i < n - 1) {

            while (i < n - 1 && prices[i + 1] <= prices[i]) i++;

            if (i == n - 1) break;
            int buy = i++;

            while (i < n && prices[i] >= prices[i - 1]) i++;
            int sell = i - 1;

            trades.add(new int[]{buy, sell});
        }

        return trades.isEmpty() ? 0 : 1;
    }

    public static void main(String[] args) {
        int[] prices = {100, 180, 260, 310, 40, 535, 695};
        int result = stockBuySell(prices, prices.length);

        if (result == 0) System.out.println("No Profit");
        else System.out.println(result);
    }
}
```

Output:



```
1
PS C:\Users\Sandiipanish P
```

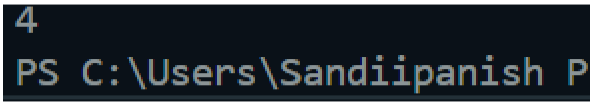
Time Complexity: O(n)

6. Coin Change(count ways)

Code:

```
public class CoinChange {
    public static int countWays(int[] coins, int sum) {
        int[] dp = new int[sum + 1];
        dp[0] = 1;

        for (int coin : coins) {
            for (int j = coin; j <= sum; j++) {
                dp[j] += dp[j - coin];
            }
        }

        return dp[sum];
    }

    public static void main(String[] args) {
        int[] coins1 = {1, 2, 3};
        int sum1 = 4;
        System.out.println(countWays(coins1, sum1));
    }
}
```

Output:



```
4
PS C:\Users\Sandiipanish P
```

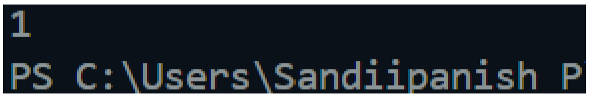
Time Complexity: O(N * sum)

7. Maximum Index

Code:

```
public class MaxIndexDifference {
    public static int maxIndexDiff(int[] arr) {
        int n = arr.length;
        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 (i < n && j < n) {
            if (leftMin[i] < rightMax[j]) {
                maxDiff = Math.max(maxDiff, j - i);
                j++;
            } else {
                i++;
            }
        }
        return maxDiff;
    }

    public static void main(String[] args) {
        int[] arr1 = {1, 10};
        System.out.println(maxIndexDiff(arr1));
    }
}
```

Output:



```
1
PS C:\Users\Sandiipanish P
```

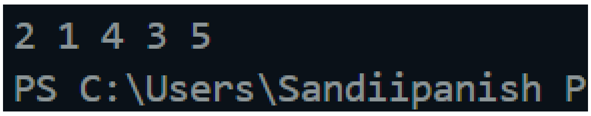
Time Complexity: O(n)

8. Wave Array

Code:

```
public class WaveArray {
    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;
        }
    }

    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3, 4, 5};
        convertToWave(arr1);
        for (int num : arr1) System.out.print(num + " ");
    }
}
```

Output:



```
2 1 4 3 5
PS C:\Users\Sandiipanish P
```

Time Complexity: O(n)