DSA PRACTICE PROBLEMS- DAY 8

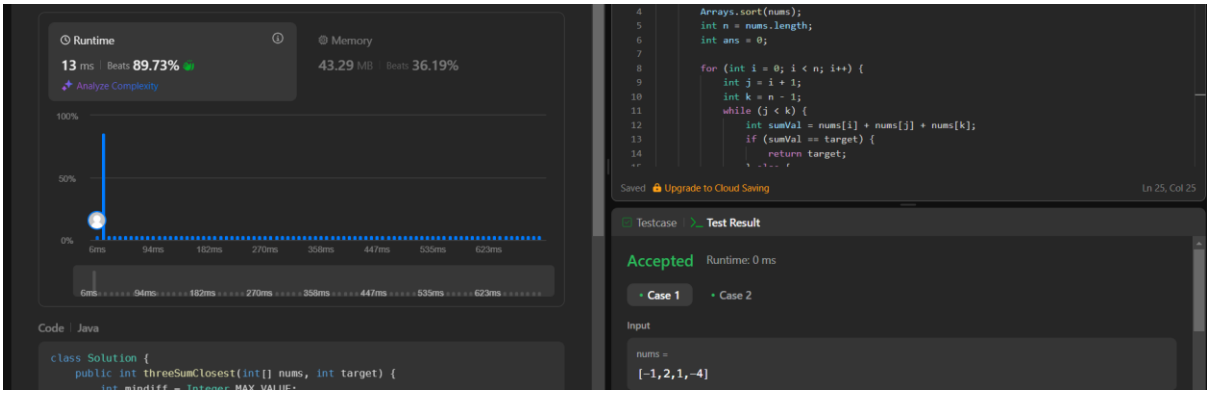
NAME: Dhejan R
REG NO: 22IT022
DATE: 20/11/2024

1. 3Sum Closest

Code Solution

```
class Solution {
    public int threeSumClosest(int[] nums, int target) {
        int mindiff = Integer.MAX_VALUE;
        Arrays.sort(nums);
        int n = nums.length;
        int ans = 0;

        for (int i = 0; i < n; i++) {
            int j = i + 1;
            int k = n - 1;
            while (j < k) {
                int sumVal = nums[i] + nums[j] + nums[k];
                if (sumVal == target) {
                    return target;
                } else {
                    int diff = Math.abs(target - sumVal);
                    if (diff < mindiff) {
                        mindiff = diff;
                        ans = sumVal;
                    }
                }
                if (sumVal < target) {
                    j++;
                } else if (sumVal > target) {
                    k--;
                }
            }
        }
        return ans;
    }
}
```

Output

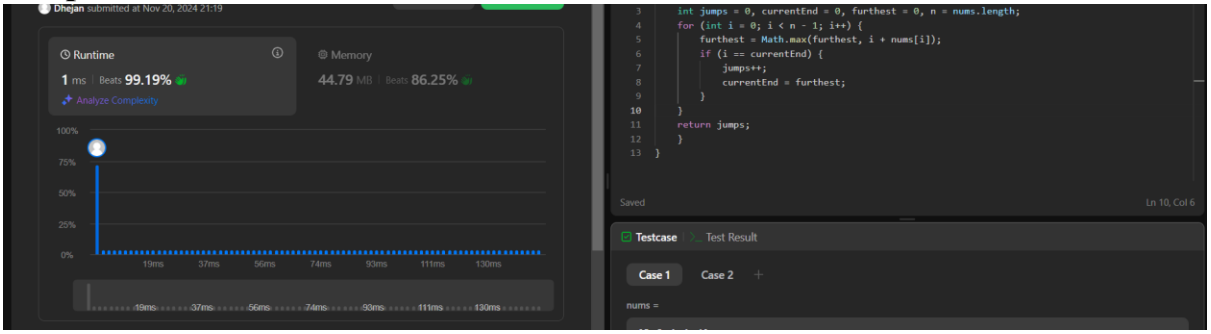


Time complexity: O(n^2)
Space complexity: O(1)

2. Jump Game II
Code Solution

```
class Solution {
    public int jump(int[] nums) {
    int jumps = 0, currentEnd = 0, furthest = 0, n = nums.length;
    for (int i = 0; i < n - 1; i++) {
        furthest = Math.max(furthest, i + nums[i]);
        if (i == currentEnd) {
            jumps++;
            currentEnd = furthest;
        }
    }
    return jumps;
    }
}
```

Output



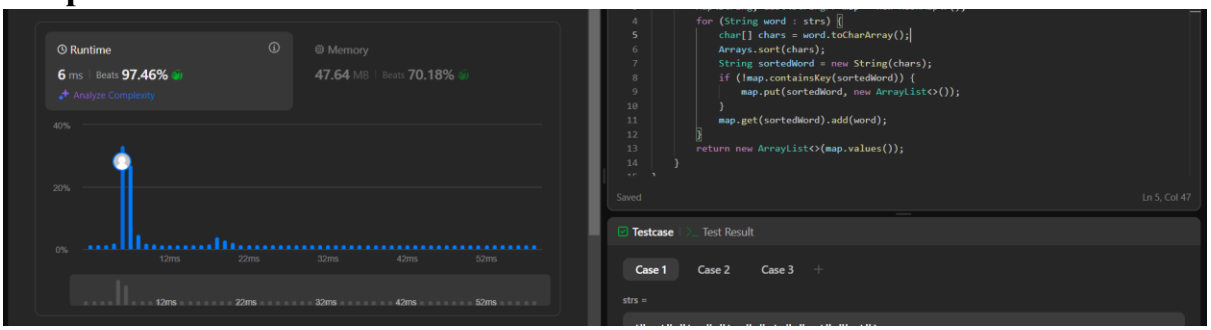
Time complexity: O(n)
Space complexity: O(1)

3. Group Anagrams
Code Solution

```
class Solution {
```

```
    public List<List<String>> groupAnagrams(String[] strs) {
        Map<String, List<String>> map = new HashMap<>();
        for (String word : strs) {
            char[] chars = word.toCharArray();
            Arrays.sort(chars);
            String sortedWord = new String(chars);
            if (!map.containsKey(sortedWord)) {
                map.put(sortedWord, new ArrayList<>());
            }
            map.get(sortedWord).add(word);
        }
        return new ArrayList<>(map.values());
    }
}
```

Output



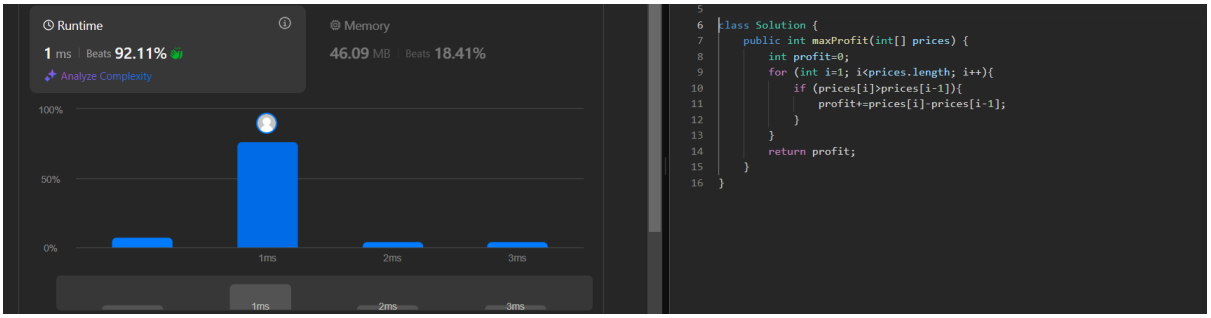
Time complexity: O(n·klogk)
Space complexity: O(n·k)

4. Best time to buy and sell stock II

Code Solution

```
class Solution {
    public int maxProfit(int[] prices) {
        int profit=0;
        for (int i=1; i<prices.length; i++){
            if (prices[i]>prices[i-1]){
                profit+=prices[i]-prices[i-1];
            }
        }
        return profit;
    }
}
```

Output



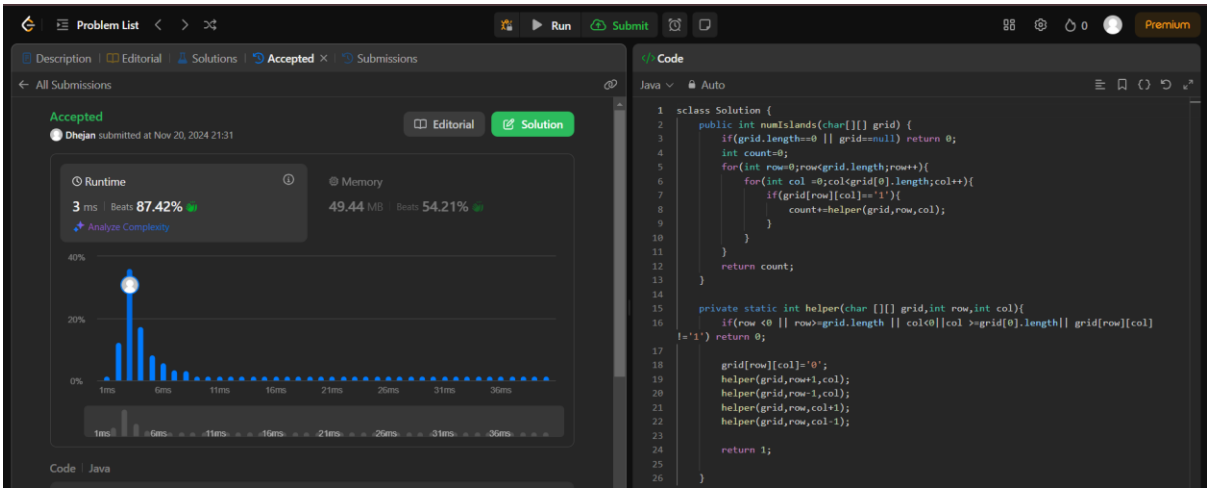
Time complexity: O(n)
Space complexity: O(1)

5.Number of islands

Code Solution

```
class Solution {
    public int numIslands(char[][] grid) {
        if(grid.length==0 || grid==null) return 0;
        int count=0;
        for(int row=0;row<grid.length;row++){
            for(int col =0;col<grid[0].length;col++){
                if(grid[row][col]=='1'){
                    count+=helper(grid,row,col);
                }
            }
        }
        return count;
    }

    private static int helper(char [][] grid,int row,int col){
        if(row <0 || row>=grid.length || col<0||col >=grid[0].length|| grid[row][col]!='1') return
0;

        grid[row][col]='0';
        helper(grid,row+1,col);
        helper(grid,row-1,col);
        helper(grid,row,col+1);
        helper(grid,row,col-1);

        return 1;

    }
}
```

Output



Time complexity: O(MXN)
Space complexity: O(MXN)

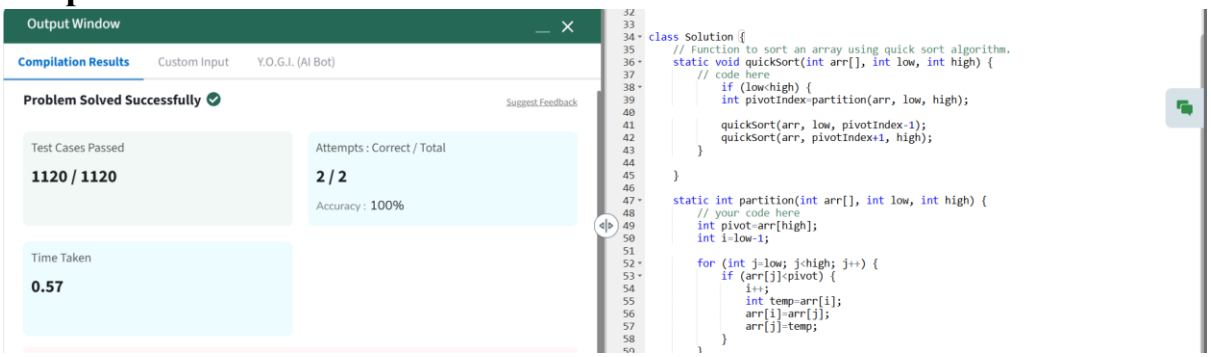
7. Quick Sort

Code Solution

```
class Solution {
    // Function to sort an array using quick sort algorithm.
    static void quickSort(int arr[], int low, int high) {
        // code here
            if (low<high) {
            int pivotIndex=partition(arr, low, high);

            quickSort(arr, low, pivotIndex-1);
            quickSort(arr, pivotIndex+1, high);
        }

    }

    static int partition(int arr[], int low, int high) {
        // your code here
        int pivot=arr[high];
        int i=low-1;

        for (int j=low; j<high; j++) {
            if (arr[j]<pivot) {
                i++;
                int temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
```

```
        }

        int temp=arr[i+1];
        arr[i+1]=arr[high];
        arr[high]=temp;

        return i+1;
    }
}
```

Output



Time complexity: O(n log n)
Space complexity: O(log n)

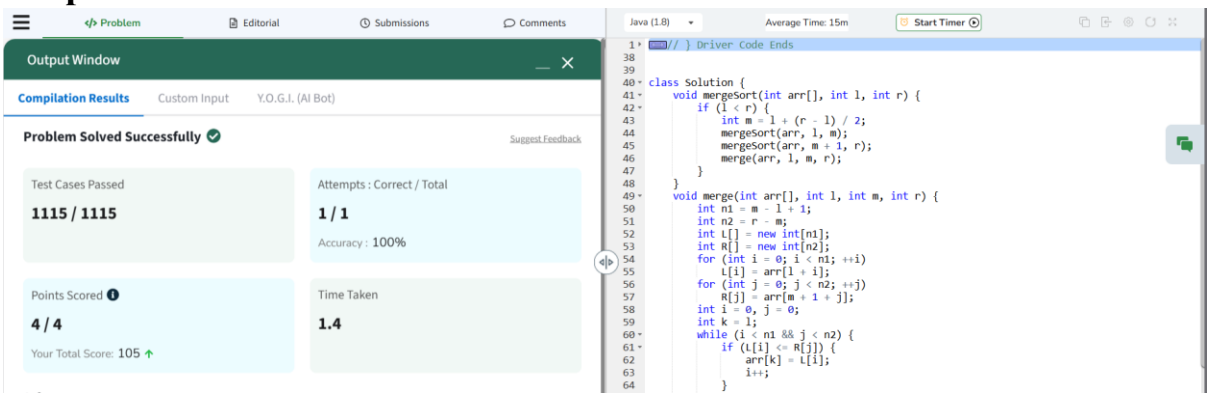
8. Merge Sort

Code Solution

```
class Solution {
    void mergeSort(int arr[], int l, int r) {
        if (l < r) {
            int m = l + (r - l) / 2;
            mergeSort(arr, l, m);
            mergeSort(arr, m + 1, r);
            merge(arr, l, m, r);
        }
    }
    void merge(int arr[], int l, int m, int r) {
        int n1 = m - l + 1;
        int n2 = r - m;
        int L[] = new int[n1];
        int R[] = new int[n2];
        for (int i = 0; i < n1; ++i)
            L[i] = arr[l + i];
        for (int j = 0; j < n2; ++j)
            R[j] = arr[m + 1 + j];
        int i = 0, j = 0;
        int k = l;
```

```
        while (i < n1 && j < n2) {
            if (L[i] <= R[j]) {
                arr[k] = L[i];
                i++;
            }
            else {
                arr[k] = R[j];
                j++;
            }
            k++;
        }
        while (i < n1) {
            arr[k] = L[i];
            i++;
            k++;
        }
        while (j < n2) {
            arr[k] = R[j];
            j++;
            k++;
        }
    }
    void printArray(int arr[]) {
        int n = arr.length;
        for (int i = 0; i < n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }
}
```

Output



Time complexity: O(n logn)

Space complexity: O(n)