HARDIK KHARE | 70765344

HW 7.1 - Roman Numeral Converter (50 pts)

Write a program that converts integers into Roman numerals and Roman numerals into integers. We will be using the rule that no Roman numeral is to be repeated in succession more than 3 times in a roman number input, thus there will be a bound on the size of the numbers your program can process.

Your program should be efficient, both in terms of space and memory.

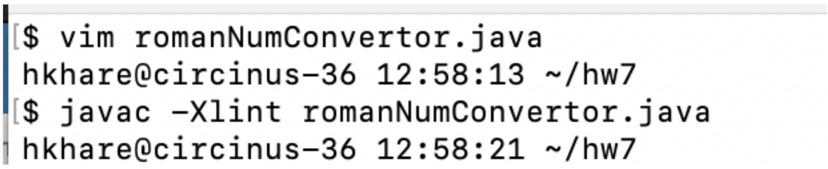
Input:

- a Roman numeral or Integer (via stdin)

Output:

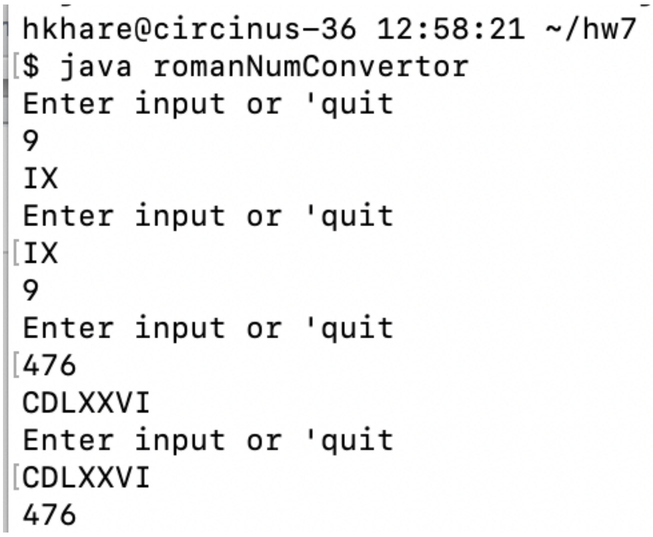
- the corresponding converted form if possible (via stdout)

Section 1: Successful compilation of program



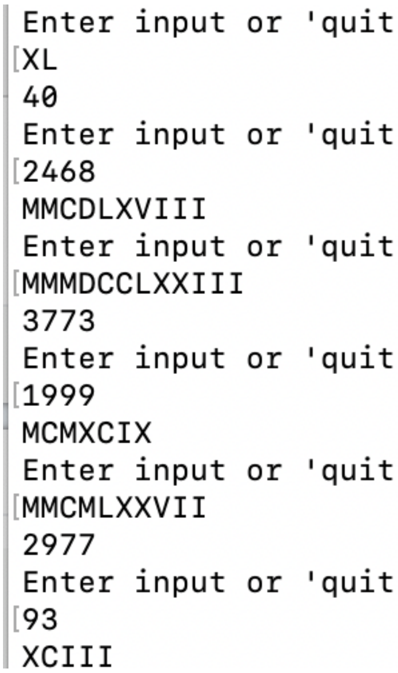
```
[$ vim romanNumConvertor.java
hkhare@circinus-36 12:58:13 ~/hw7
[$ javac -Xlint romanNumConvertor.java
hkhare@circinus-36 12:58:21 ~/hw7
```

Section 2: program running on the provided example from the assignment



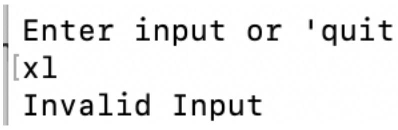
```
hkhare@circinus-36 12:58:21 ~/hw7
[$ java romanNumConvertor
Enter input or 'quit
9
IX
Enter input or 'quit
[IX
9
Enter input or 'quit
[476
CDLXXVI
Enter input or 'quit
[CDLXXVI
476
```

Section 3: Provided test input



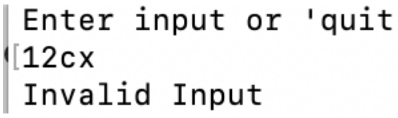
```
Enter input or 'quit
XL
40
Enter input or 'quit
2468
MMCDLXVIII
Enter input or 'quit
MMMDCCLXXIII
3773
Enter input or 'quit
1999
MCMXCIX
Enter input or 'quit
MMCMLXXVII
2977
Enter input or 'quit
93
XCIII
```

Section 4: Edge Case #1
Description: Lowercase characters numerals
Input: ‘xl’
Expected Output: ‘Invalid Input’



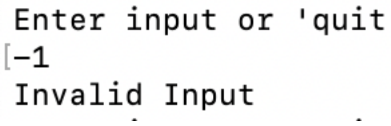
```
Enter input or 'quit
xl
Invalid Input
```

Section 5: Edge Case #2
Description: Alphanumeric string
Expected Output: ‘Invalid input’



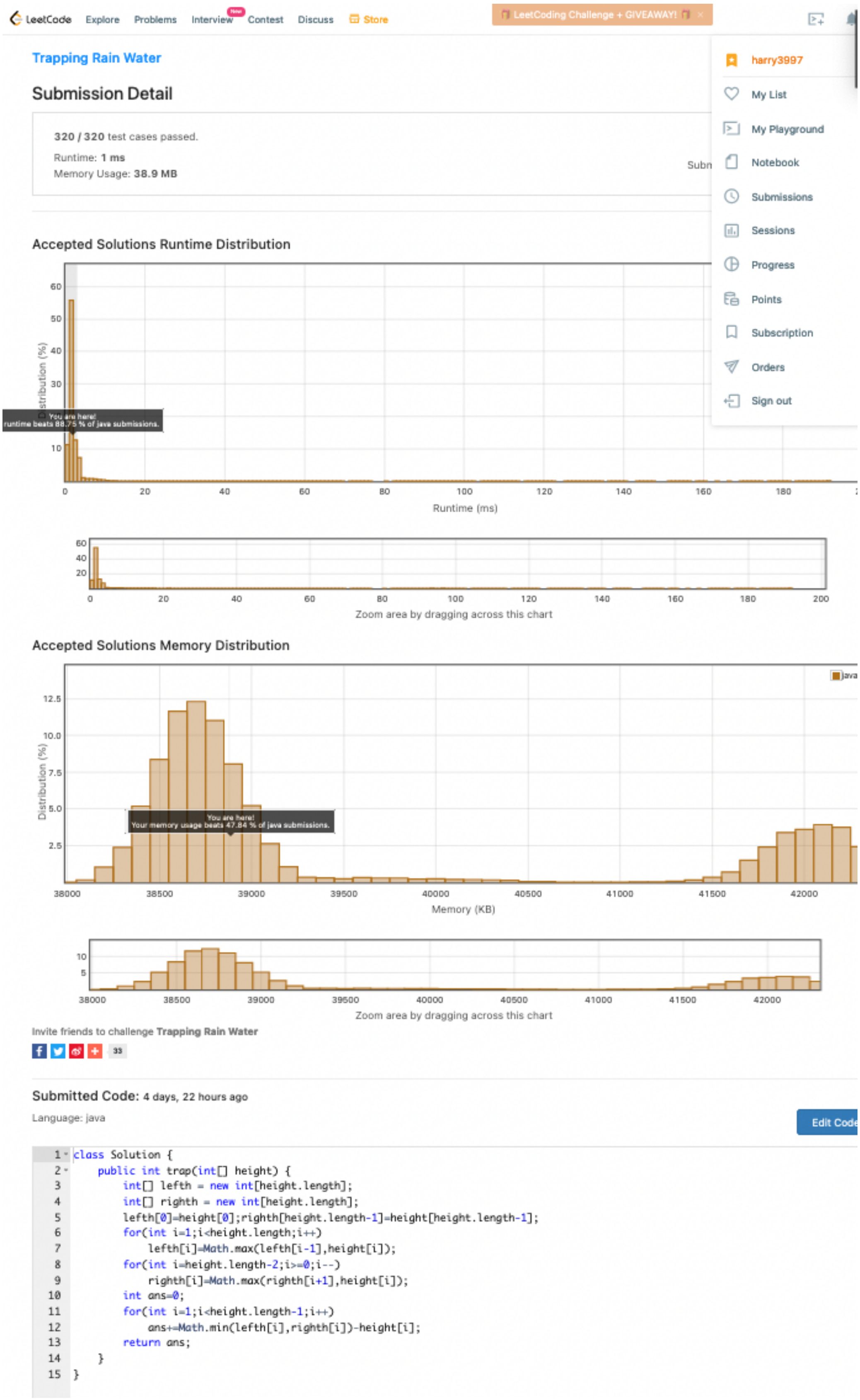
```
Enter input or 'quit
12cx
Invalid Input
```

Section 6: Edge Case #3
Description: Negative integer
Expected Output: ‘Invalid input’



```
Enter input or 'quit
-1
Invalid Input
```

https://leetcode.com/problems/trapping-rain-water/



Trapping Rain Water

Submission Detail

320 / 320 test cases passed.

Runtime: **1 ms**

Memory Usage: **38.9 MB**

Accepted Solutions Runtime Distribution

You are here!
runtime beats 88.75 % of java submissions.

Accepted Solutions Memory Distribution

You are here!
Your memory usage beats 47.84 % of java submissions.

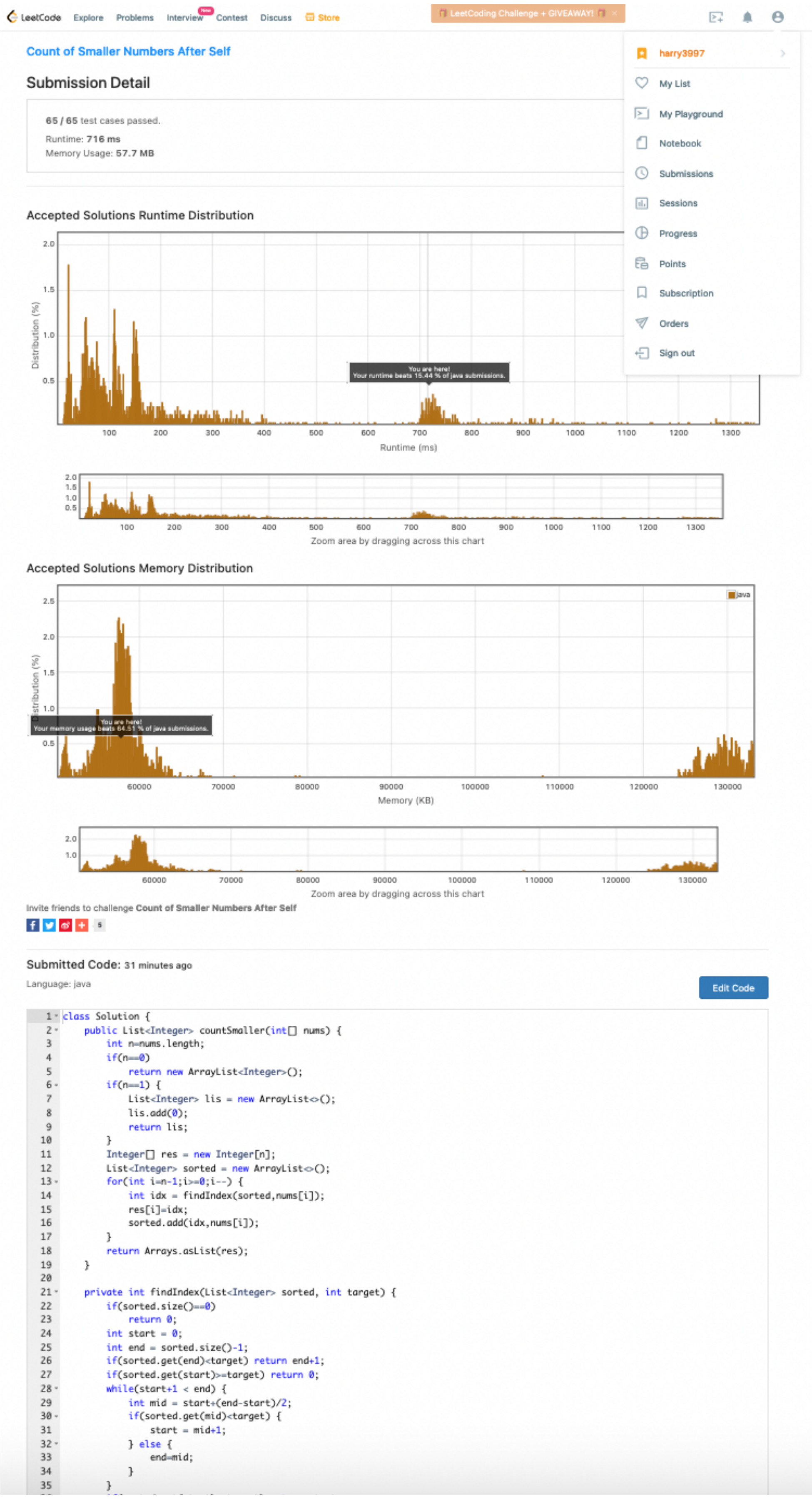
Invite friends to challenge **Trapping Rain Water**

Submitted Code: 4 days, 22 hours ago

Language: java

```java
class Solution {
    public int trap(int[] height) {
        int[] lefth = new int[height.length];
        int[] righth = new int[height.length];
        lefth[0]=height[0];righth[height.length-1]=height[height.length-1];
        for(int i=1;i<height.length;i++)
            lefth[i]=Math.max(lefth[i-1],height[i]);
        for(int i=height.length-2;i>=0;i--)
            righth[i]=Math.max(righth[i+1],height[i]);
        int ans=0;
        for(int i=1;i<height.length-1;i++)
            ans+=Math.min(lefth[i],righth[i])-height[i];
        return ans;
    }
}
```

https://leetcode.com/problems/count-of-smaller-numbers-after-self/



Count of Smaller Numbers After Self

Submission Detail

65 / 65 test cases passed.

Runtime: 716 ms

Memory Usage: 57.7 MB

Accepted Solutions Runtime Distribution

You are here!
Your runtime beats 15.44 % of java submissions.

Zoom area by dragging across this chart

Accepted Solutions Memory Distribution

You are here!
Your memory usage beats 64.51 % of java submissions.

Zoom area by dragging across this chart

Invite friends to challenge **Count of Smaller Numbers After Self**

Submitted Code: 31 minutes ago

Language: java

```java
class Solution {
    public List<Integer> countSmaller(int[] nums) {
        int n=nums.length;
        if(n==0)
            return new ArrayList<Integer>();
        if(n==1) {
            List<Integer> lis = new ArrayList<>();
            lis.add(0);
            return lis;
        }
        Integer[] res = new Integer[n];
        List<Integer> sorted = new ArrayList<>();
        for(int i=n-1;i>=0;i--) {
            int idx = findIndex(sorted,nums[i]);
            res[i]=idx;
            sorted.add(idx,nums[i]);
        }
        return Arrays.asList(res);
    }

    private int findIndex(List<Integer> sorted, int target) {
        if(sorted.size()==0)
            return 0;
        int start = 0;
        int end = sorted.size()-1;
        if(sorted.get(end)<target) return end+1;
        if(sorted.get(start)>=target) return 0;
        while(start+1 < end) {
            int mid = start+(end-start)/2;
            if(sorted.get(mid)<target) {
                start = mid+1;
            } else {
                end=mid;
            }
        }
```