



Experiment1.3

Student Name: Sameer Anand

UID:20BCS9834

Branch: BE CSE

Section/Group: 20BCS_DM_714

Semester: 6TH

Subject Name: CC LAB

Subject Code: 20CSP-351

1. Aim: To implement the concept of Heap model

Leetcode code and output :

a. Kth Largest Element in an Array

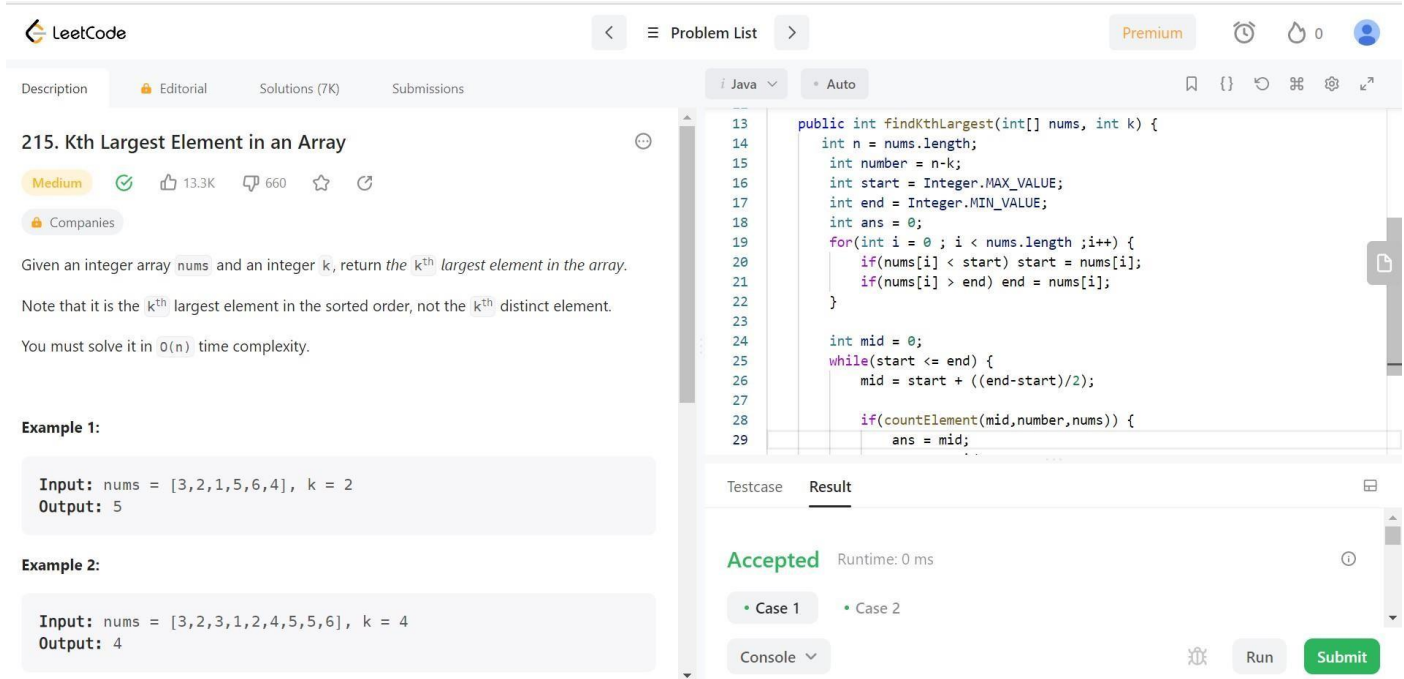
```
class Solution {    public static boolean countElement(int mid , int
number , int nums[]) {        int count = 0;        for(int i = 0; i <
nums.length ; i++) {            if(mid > nums[i]) {
count++;
                }        }        if(count
<= number) return true;        else
return false;
    }    public int findKthLargest(int[]
nums, int k) {        int n = nums.length;
int number = n-k;
        int start = Integer.MAX_VALUE;
int end = Integer.MIN_VALUE;        int
ans = 0;
        for(int i = 0 ; i < nums.length ;i++) {
if(nums[i] < start) start = nums[i];
if(nums[i] > end) end = nums[i];
        }
int mid = 0;
while(start <= end) {
    mid = start + ((end-start)/2);

if(countElement(mid,number,nums)) {
ans = mid;        start = mid+1;

}else {
    end = mid - 1;
}
}
```

```
        return ans;
    }
}
```

OUTPUT:



LeetCode

Problem List

Premium

215. Kth Largest Element in an Array

Medium

13.3K 660

Companies

Given an integer array `nums` and an integer `k`, return the k^{th} largest element in the array.

Note that it is the k^{th} largest element in the sorted order, not the k^{th} distinct element.

You must solve it in $O(n)$ time complexity.

Example 1:

Input: `nums = [3,2,1,5,6,4]`, `k = 2`
Output: 5

Example 2:

Input: `nums = [3,2,3,1,2,4,5,5,6]`, `k = 4`
Output: 4

```
13 public int findKthLargest(int[] nums, int k) {
14     int n = nums.length;
15     int number = n-k;
16     int start = Integer.MAX_VALUE;
17     int end = Integer.MIN_VALUE;
18     int ans = 0;
19     for(int i = 0 ; i < nums.length ; i++) {
20         if(nums[i] < start) start = nums[i];
21         if(nums[i] > end) end = nums[i];
22     }
23
24     int mid = 0;
25     while(start <= end) {
26         mid = start + ((end-start)/2);
27
28         if(countElement(mid,number,nums)) {
29             ans = mid;
```

Testcase Result

Accepted Runtime: 0 ms

Case 1 Case 2

Console Run Submit

b. Last Stone Weight

Code:

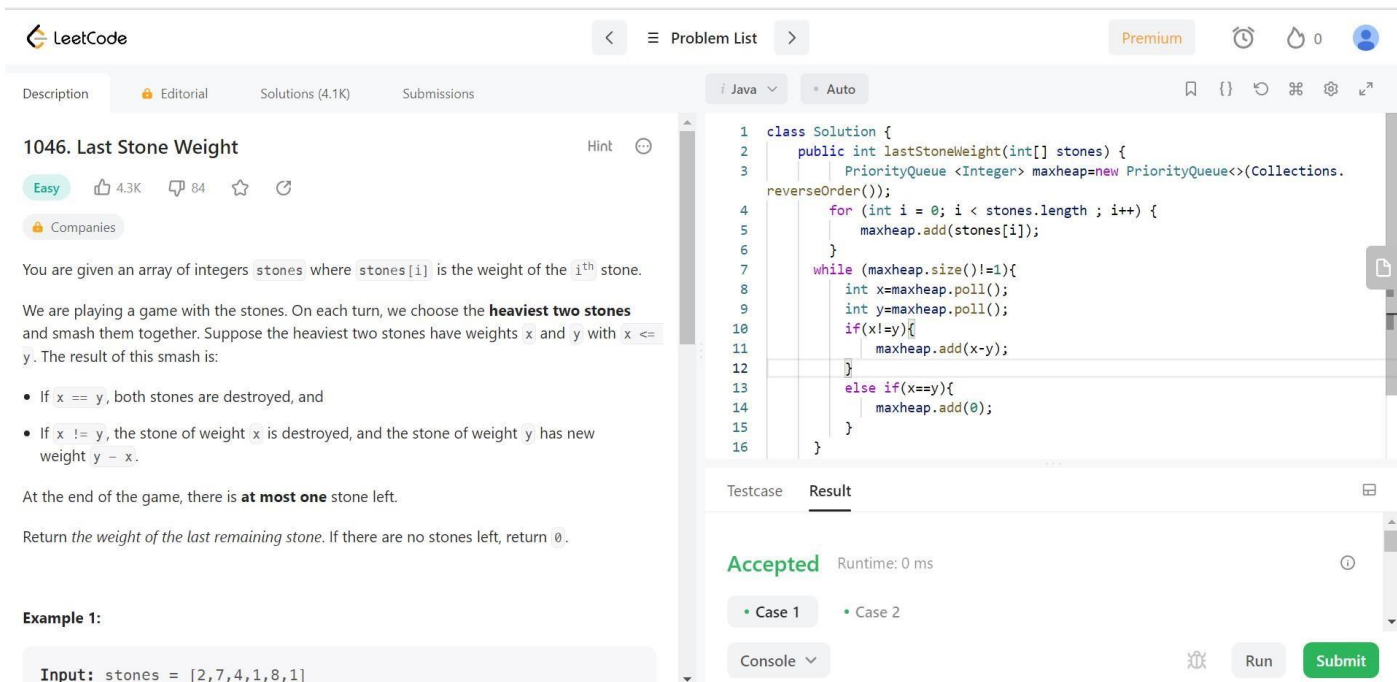
```
class Solution {
    public int lastStoneWeight(int[] stones) {
        PriorityQueue<Integer> maxheap=new PriorityQueue<>(Collections.reverseOrder());
        for (int i = 0; i < stones.length ; i++) {
            maxheap.add(stones[i]);
        }
        while
        (maxheap.size()!=1){
            int x=maxheap.poll();
            int y=maxheap.poll();
            if(x!=y){
                maxheap.add(x-y);
            }
        }
        return maxheap.poll();
    }
}
```

```

    }
    else
    {
        if(x==y){
            maxheap.add(0);
        }
    }
    return
    maxheap.poll();
} }

```

OUTPUT:



The screenshot shows the LeetCode interface for the problem "1046. Last Stone Weight". The problem is marked as "Easy" with 4.3K likes and 84 comments. The description states: "You are given an array of integers stones where stones[i] is the weight of the ith stone. We are playing a game with the stones. On each turn, we choose the **heaviest two stones** and smash them together. Suppose the heaviest two stones have weights x and y with x ≤ y. The result of this smash is:

- If x == y, both stones are destroyed, and
- If x != y, the stone of weight x is destroyed, and the stone of weight y has new weight y - x.

At the end of the game, there is **at most one** stone left. Return the weight of the last remaining stone. If there are no stones left, return 0.

Example 1:

Input: stones = [2,7,4,1,8,1]

The code editor shows a Java solution using a PriorityQueue (maxheap) to simulate the game. The code is as follows:

```

1 class Solution {
2     public int lastStoneWeight(int[] stones) {
3         PriorityQueue<Integer> maxheap=new PriorityQueue<>(Collections.
reverseOrder());
4         for (int i = 0; i < stones.length ; i++) {
5             maxheap.add(stones[i]);
6         }
7         while (maxheap.size()!=1){
8             int x=maxheap.poll();
9             int y=maxheap.poll();
10            if(x!=y){
11                maxheap.add(x-y);
12            }
13            else if(x==y){
14                maxheap.add(0);
15            }
16        }
17    }
18 }

```

The "Result" section shows "Accepted" with a runtime of 0 ms. There are two test cases: "Case 1" and "Case 2". The "Console" is visible at the bottom.