

Experiment 1.3

Student Name: Anant Bhardwaj

UID: 21BCS8910

Branch: BE-CSE

Section/Group: 21BCS_CC-648-B

Semester: 6th

Subject Code: 21CSP-351

Subject Name: Advance Programming Lab-2

Date of performance: 02-06-2024

Aim: To demonstrate the concept of Heap model.

Objective: To understand the Heap model as a data structure and its applications, including efficient storage and retrieval of priority-based information, and to explore its implementation, operations, and complexities to facilitate effective problem-solving and algorithmic design.

PROBLEM-1: Design a class to find the kth largest element in a stream. Note that it is the kth largest element in the sorted order, not the kth distinct element. Implement KthLargest class:

- KthLargest(int k, int[] nums) Initializes the object with the integer k and the stream of integers nums.
- int add(int val) Appends the integer val to the stream and returns the element representing the kth largest element in the stream.

CODE:

```
class KthLargest {
public:
    priority_queue<int> q;
    int size;

    KthLargest(int k, vector<int>& nums) {
        size = k;
        for(int i : nums) add(i);
    }

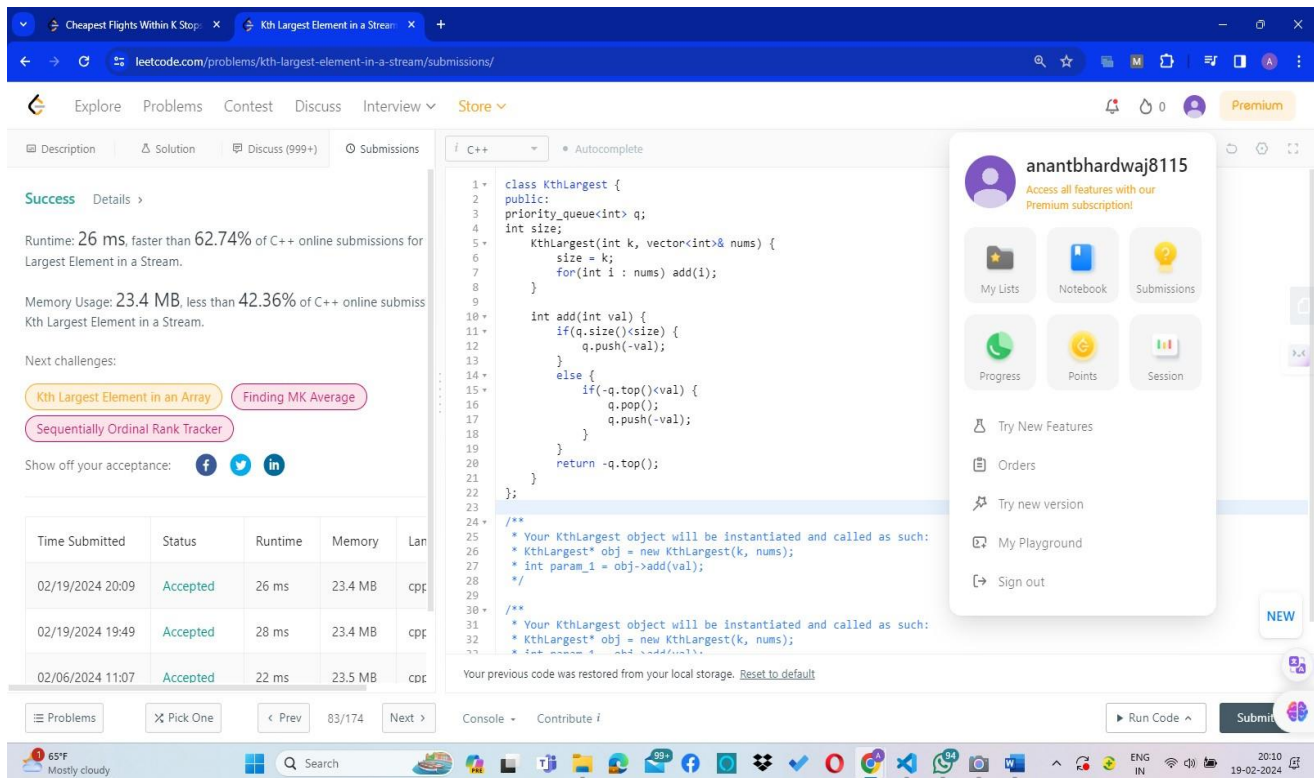
    int add(int val) {
        if(q.size()<size) {
            q.push(-val);
        }
    }
};
```

```

else {                if(-
q.top()<val)          {
q.pop();
                q.push(-val);
                }
}
return -q.top();
}
};

```

OUTPUT:



The screenshot shows a web browser displaying a C++ solution for the problem "Kth Largest Element in a Stream" on LeetCode. The solution is accepted with a runtime of 26 ms and memory usage of 23.4 MB. The code uses a priority queue to maintain the k largest elements.

Success Details

Runtime: 26 ms, faster than 62.74% of C++ online submissions for Largest Element in a Stream.

Memory Usage: 23.4 MB, less than 42.36% of C++ online submissions for Kth Largest Element in a Stream.

Next challenges:

- Kth Largest Element in an Array
- Finding MK Average
- Sequentially Ordinal Rank Tracker

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
02/19/2024 20:09	Accepted	26 ms	23.4 MB	c++
02/19/2024 19:49	Accepted	28 ms	23.4 MB	c++
02/06/2024 11:07	Accepted	22 ms	23.5 MB	c++

```

1 class KthLargest {
2 public:
3     priority_queue<int> q;
4     int size;
5     KthLargest(int k, vector<int>& nums) {
6         size = k;
7         for(int i : nums) add(i);
8     }
9
10    int add(int val) {
11        if(q.size()<size) {
12            q.push(-val);
13        }
14        else {
15            if(-q.top()<val) {
16                q.pop();
17                q.push(-val);
18            }
19        }
20        return -q.top();
21    }
22 };
23
24 /**
25  * Your KthLargest object will be instantiated and called as such:
26  * KthLargest* obj = new KthLargest(k, nums);
27  * int param_1 = obj->add(val);
28  */
29
30 /**
31  * Your KthLargest object will be instantiated and called as such:
32  * KthLargest* obj = new KthLargest(k, nums);
33  * int param_1 = obj->add(val);
34  */

```

Your previous code was restored from your local storage. [Reset to default](#)

Run Code Submit

PROBLEM-2: 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 $x \leq y$. The result of this smash is:

If $x == y$, both stones are destroyed, and

If $x \neq y$, the stone of weight x is destroyed, and the stone of weight y has new weight $y - x$.

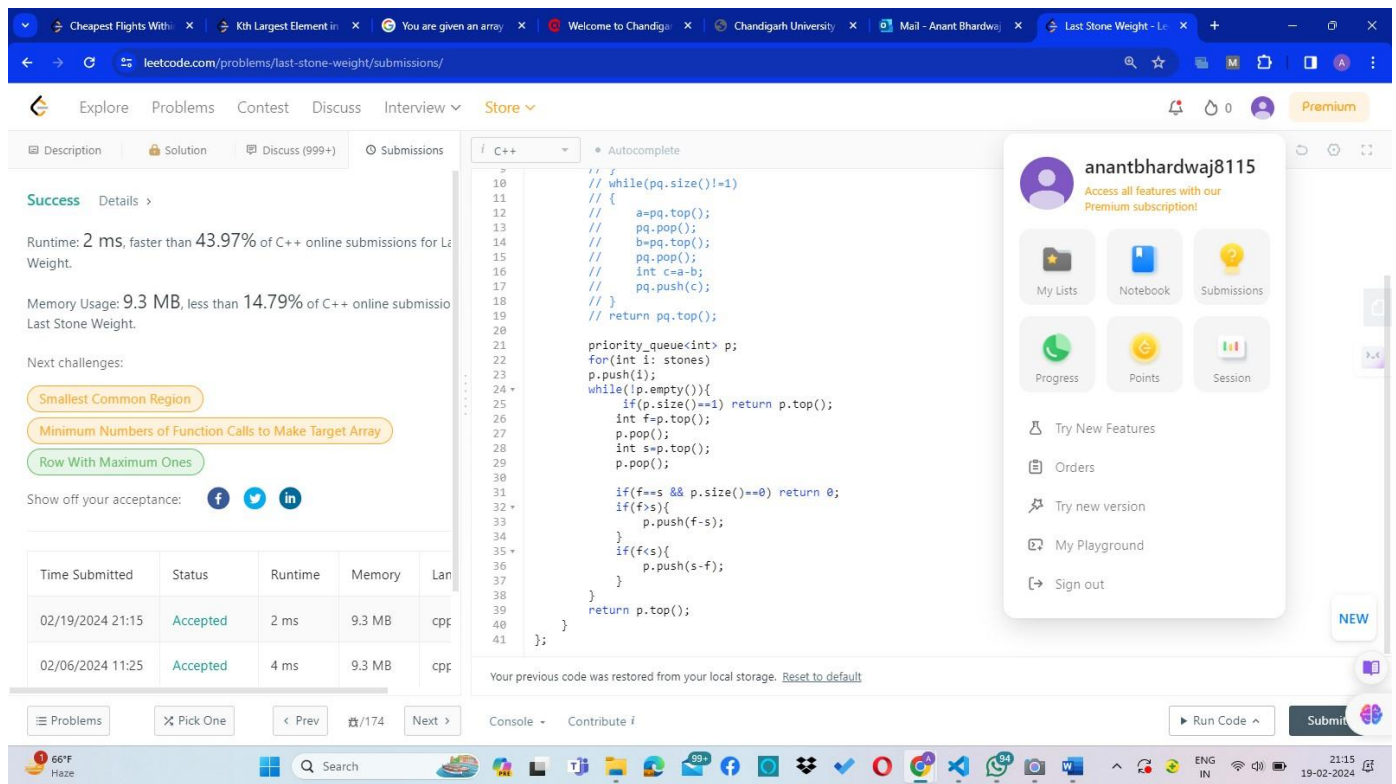
At the end of the game, there is at most one stone left.

Return the weight of the last remaining stone. If there are no stones left, return 0.

CODE:

```
class Solution { public:      int
lastStoneWeight(vector<int>&
stones) {
priority_queue<int> p;
for(int i: stones)
    p.push(i);
while(!p.empty()){
if(p.size()==1) return p.top();
    int f=p.top();
    p.pop();
int s=p.top();
    p.pop();
if(f==s && p.size()==0)
return 0;
    if(f>s){
        p.push(f-s);
    }
    if(f<s){
        p.push(s-f);
    }
}
return p.top();
}
};
```

OUTPUT:



Success Details >

Runtime: 2 ms, faster than 43.97% of C++ online submissions for Last Stone Weight.

Memory Usage: 9.3 MB, less than 14.79% of C++ online submissions for Last Stone Weight.

Next challenges:

- Smallest Common Region
- Minimum Numbers of Function Calls to Make Target Array
- Row With Maximum Ones

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
02/19/2024 21:15	Accepted	2 ms	9.3 MB	c++
02/06/2024 11:25	Accepted	4 ms	9.3 MB	c++

```

10 // while(pq.size() != 1)
11 // {
12 //     a=pq.top();
13 //     pq.pop();
14 //     b=pq.top();
15 //     pq.pop();
16 //     int c=a-b;
17 //     pq.push(c);
18 // }
19 // return pq.top();
20
21 priority_queue<int> p;
22 for(int i: stones)
23     p.push(i);
24 while(!p.empty()){
25     if(p.size() == 1) return p.top();
26     int f=p.top();
27     p.pop();
28     int s=p.top();
29     p.pop();
30
31     if(f==s && p.size() == 0) return 0;
32     if(f>s){
33         p.push(f-s);
34     }
35     if(f<s){
36         p.push(s-f);
37     }
38 }
39 return p.top();
40 }
41 };

```

Your previous code was restored from your local storage. [Reset to default](#)

Run Code Submit

Learning Outcomes:

- Understanding the properties and characteristics of a Heap, including its structure, ordering, and representation.
- Implementing Heap data structure and related algorithms in programming languages such as Java, or C++.