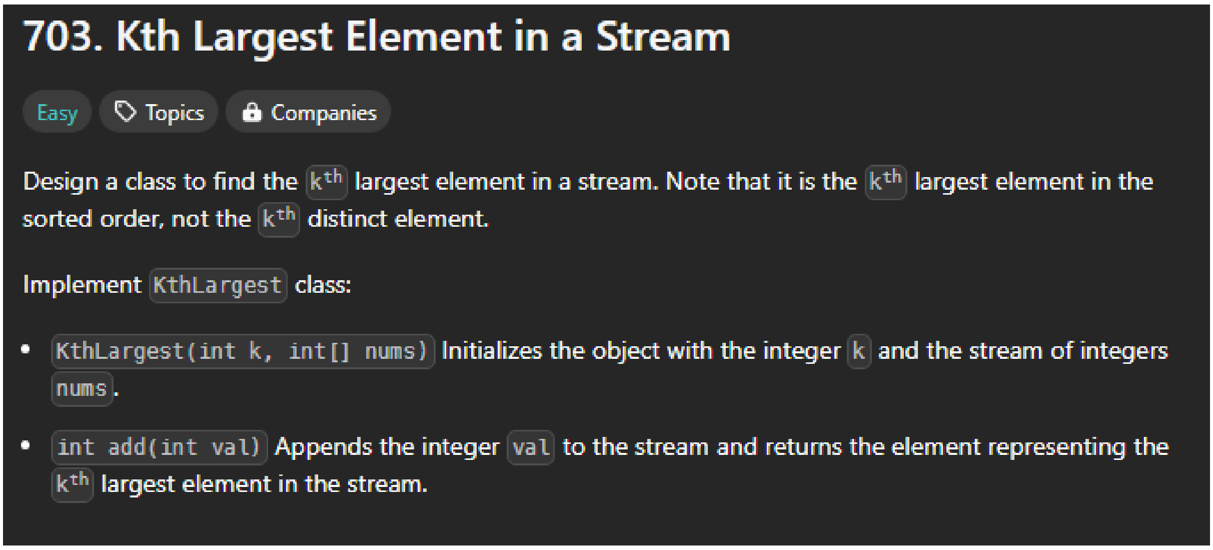
703. Kth Largest Element in a Stream - 12/08/24 (easy)

Heap or Priority queue



703. Kth Largest Element in a Stream

Design a class to find the `k^th` largest element in a stream. Note that it is the `k^th` largest element in the sorted order, not the `k^th` 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 `k^th` largest element in the stream.

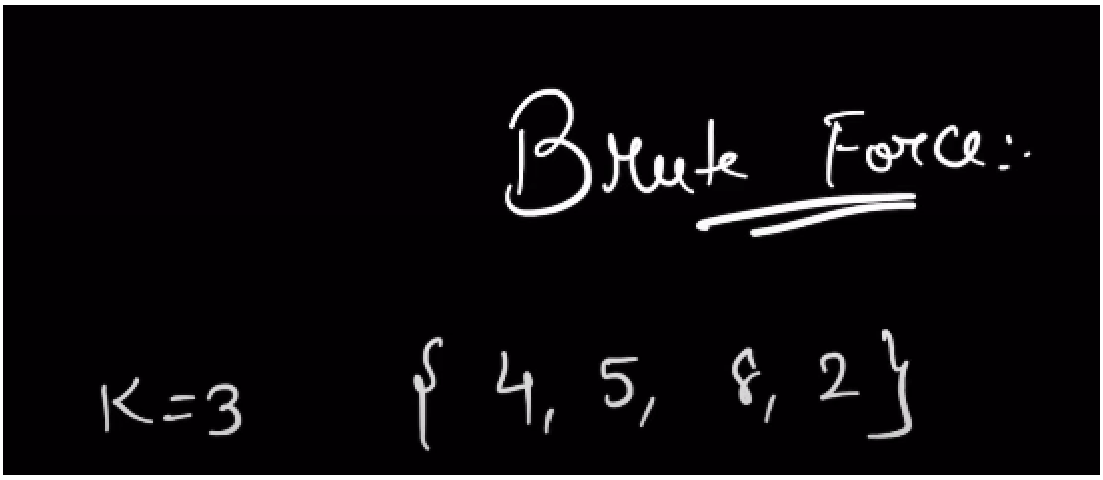
stream - data will be added time to time

largest in sorted order not in distinct

means tha

```
1,2,2,2,2,3
3 largest element is - 2 not 3
becuase in order 2 is largeest
```

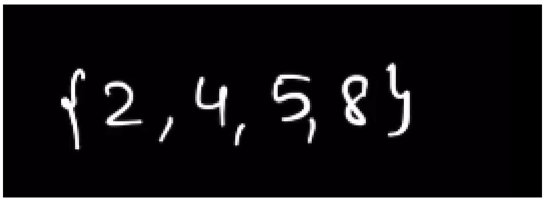
brute force:



Brute Force:

K=3 {4, 5, 8, 2}

sort it in starting



{2,4,5,8}

we sue binary search to insert new element

but it would take n log(n) time complexity

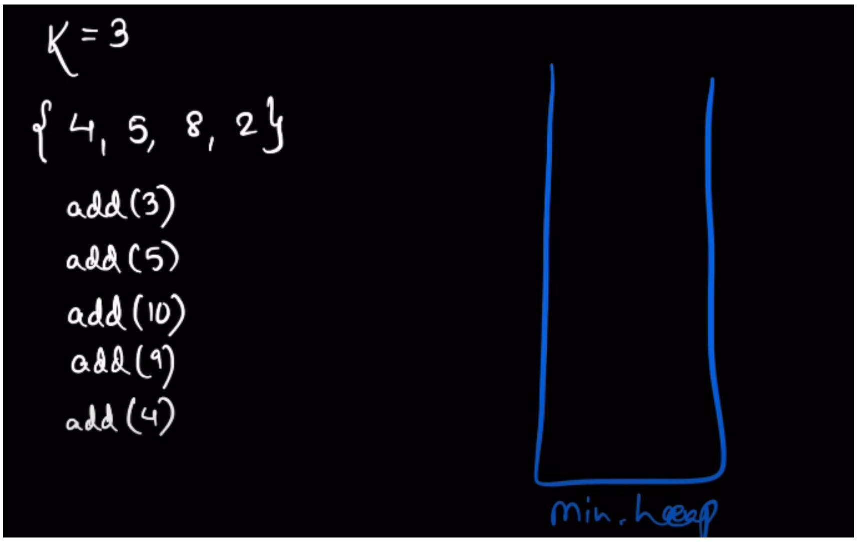
Hint whenever there are n th largest or nth smallest use heap to solve it:

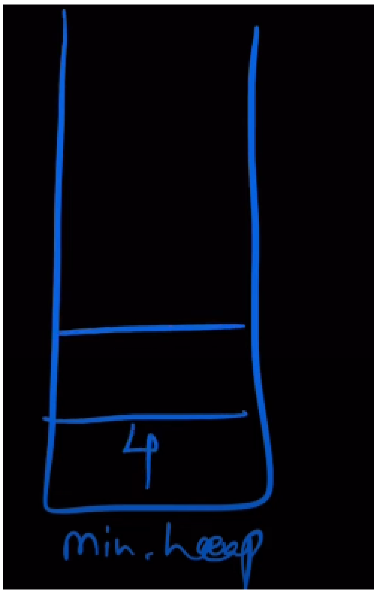
hint:

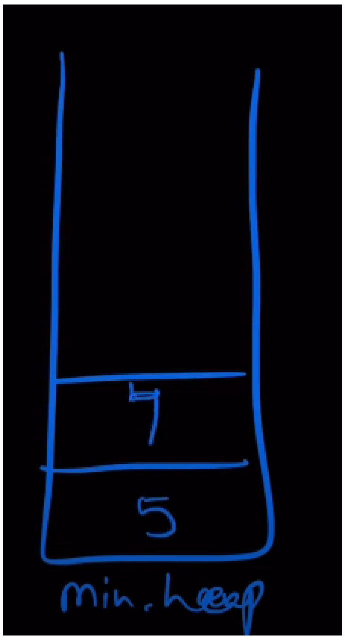
when to find largest → use min heap

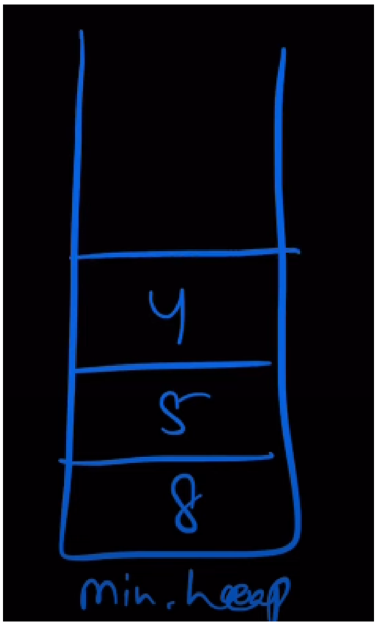
when to find smallest → us max heap

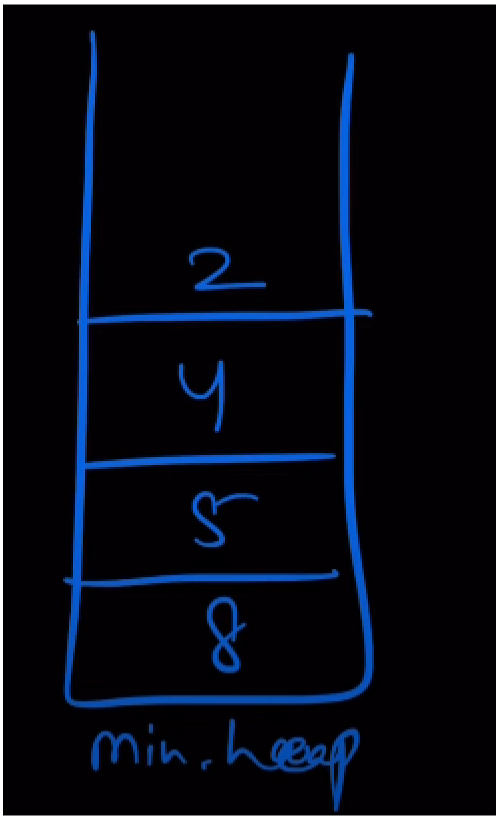
we solve it using min heap


K = 3
{4, 5, 8, 2}
add(3)
add(5)
add(10)
add(9)
add(4)
min. heap

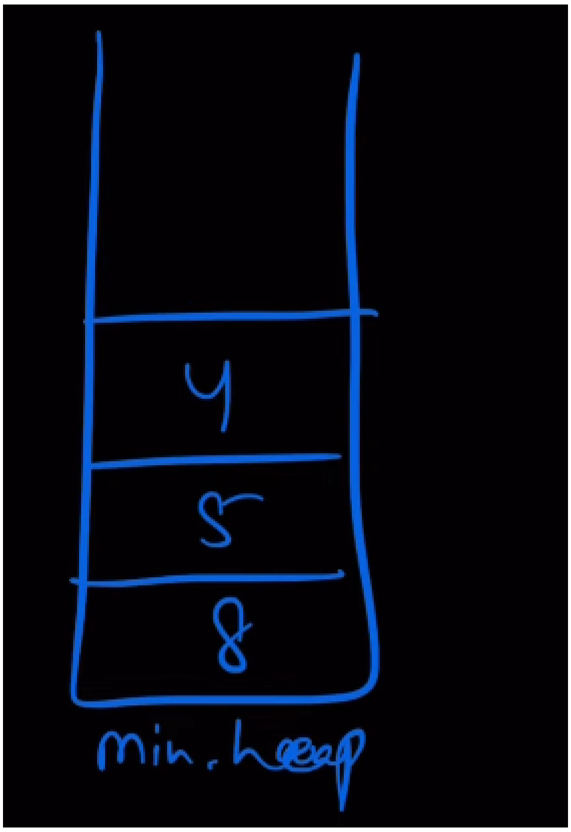

4
min. heap


4
5
min. heap

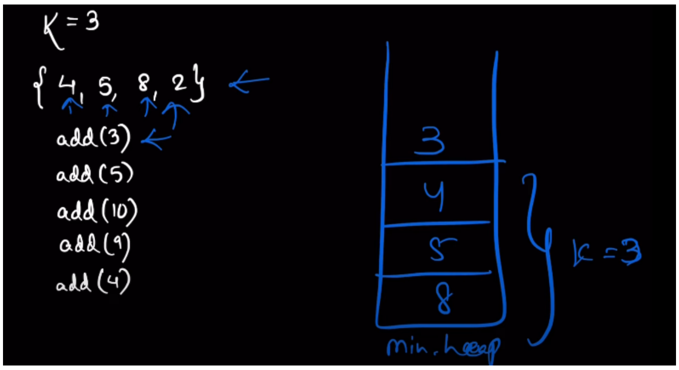

4
5
8
min. heap



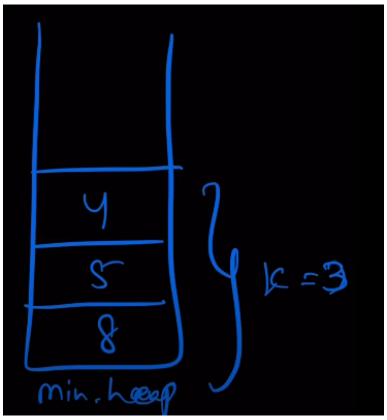
now we will pop the 2



now add 3 in min heap



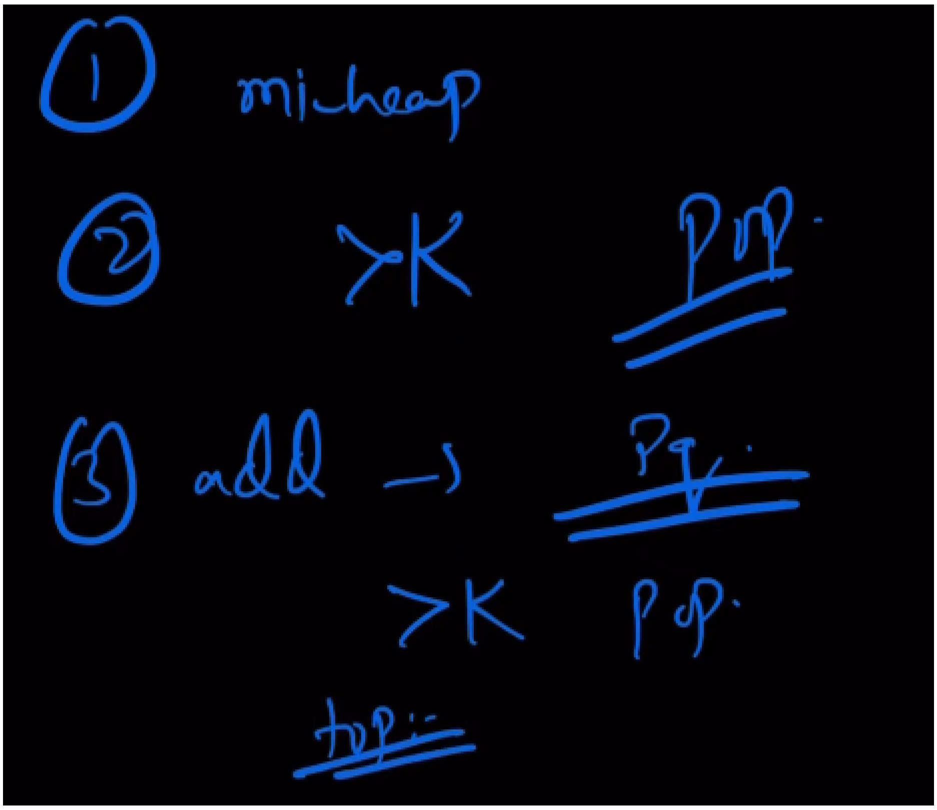
but now size is greater than 3 so pop the min one so we pop 3



priority queue take **log(k)**

the total T.C is **O(n(log(k))**

story wise explanation



Final code:

```
class KthLargest {
public:

    int K;
    priority_queue<int , vector<int> , greater<int>> pq;

    KthLargest(int k, vector<int>& nums) {
        K =k;
        for(int &num:nums){
            pq.push(num);

            if(pq.size()>K){
                pq.pop();
            }

        }

    }

    int add(int val) {
        pq.push(val);
        if(pq.size()>K){
            pq.pop();
        }

        return pq.top();
    }
};

/**
 * Your KthLargest object will be instantiated and called as
 * KthLargest* obj = new KthLargest(k, nums);
 * int param_1 = obj->add(val);
 */
```