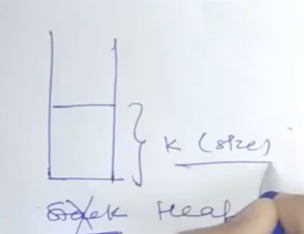
# ***HEAPS DATA STRUCTURES :***

Heap we can see that as “STACK”



Keywords to find heap   
Either K we have in question || we have smallest/largest

K + smallest = max heap

K + largest = min heap

We have to heap

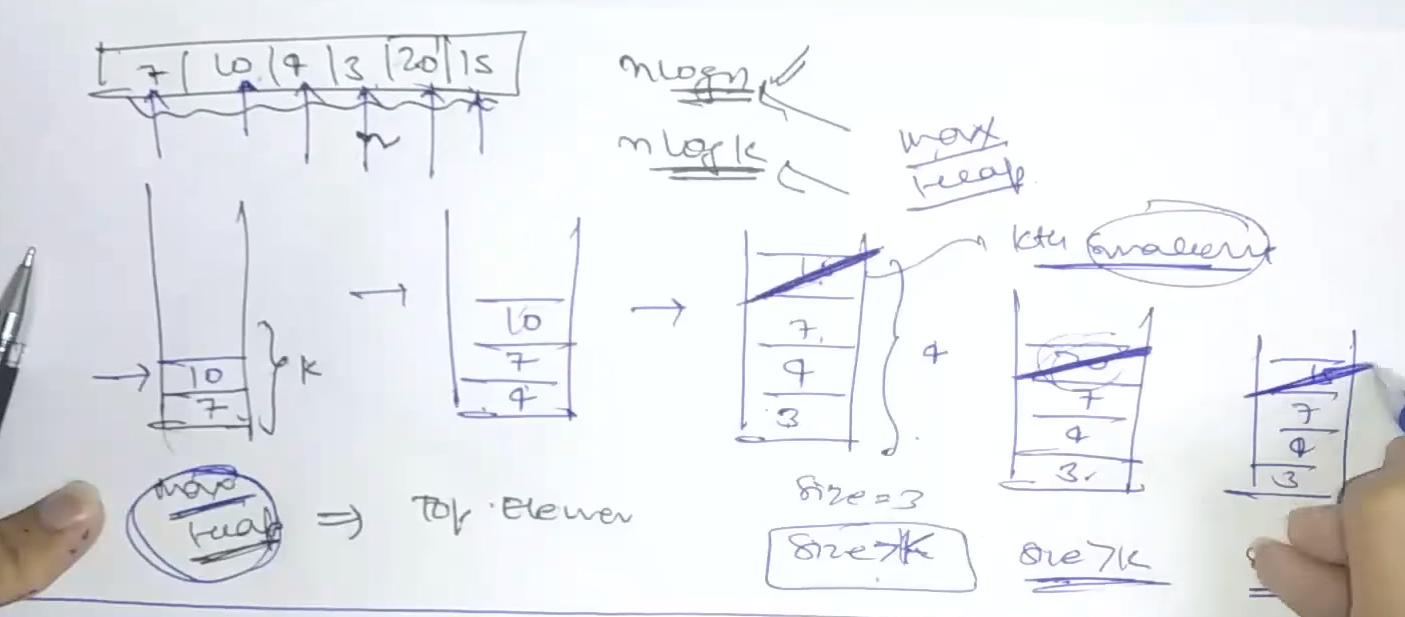
1. MAX heap
2. MIN heap

At top of the heap we will get the answer for any typ of questions and size of the heap is always will be “K”.

All the sorting questions we have in heap with TC = O(nlogn)

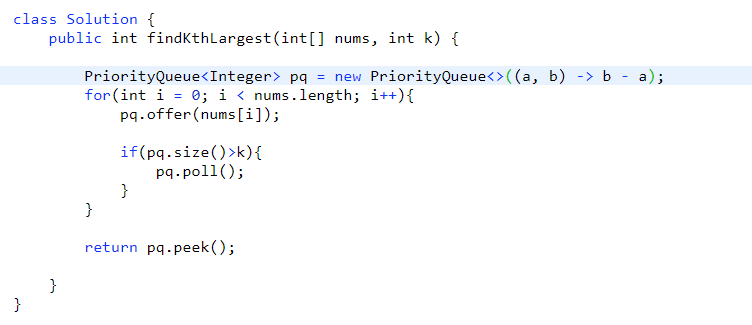
With heap it will reduce to O(nlogK).

**Question** Kth Smallest Element

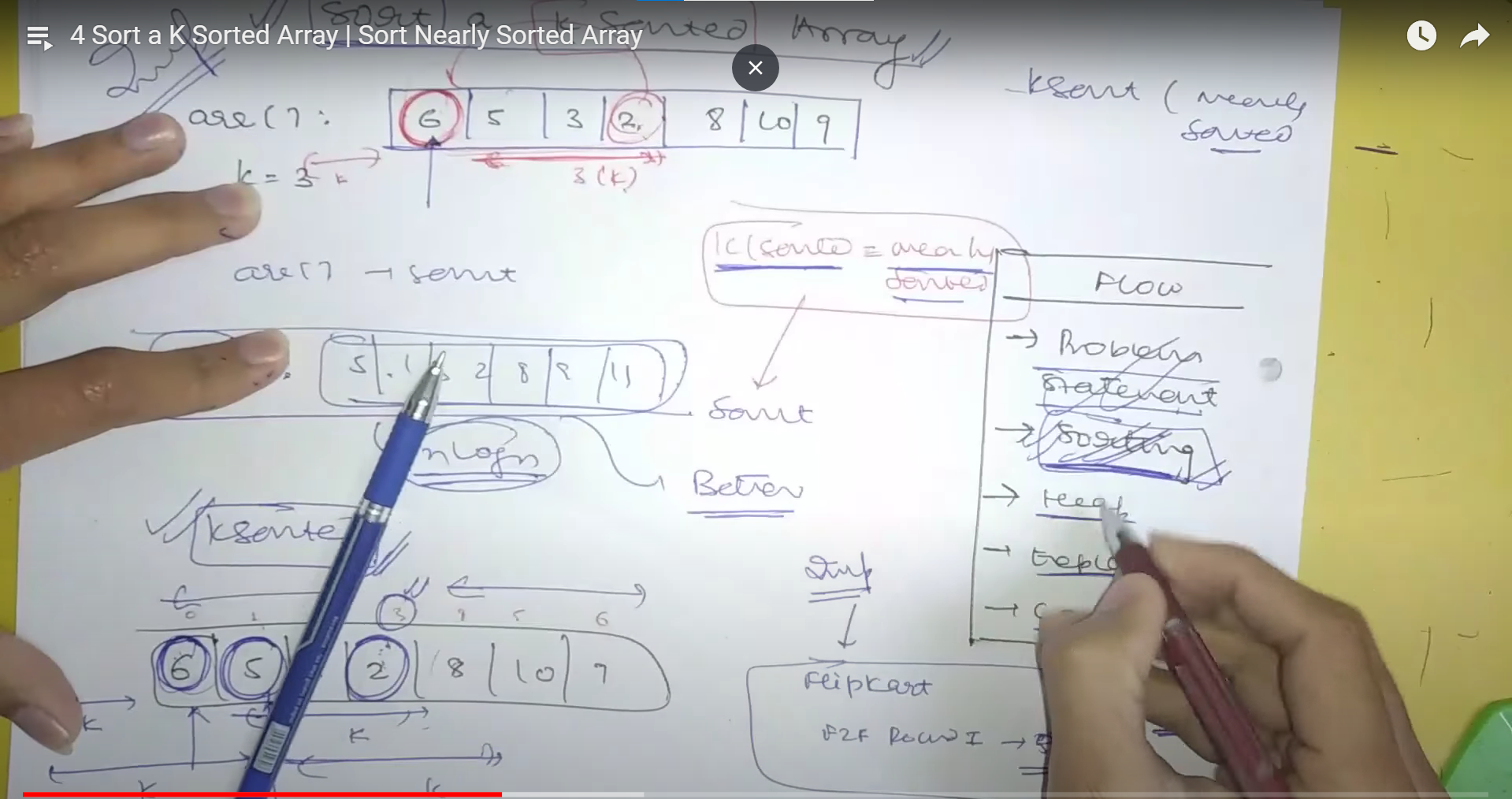


Whatever we have at top in max heap is the answer of smallest value at the Kth index!!

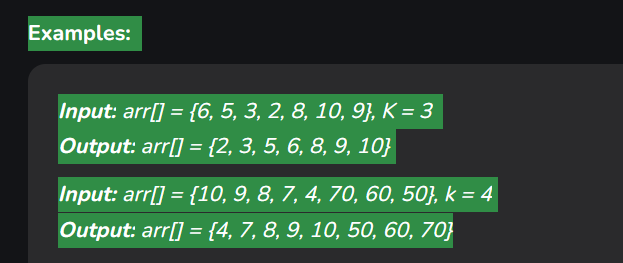
MAX HEAP In java :



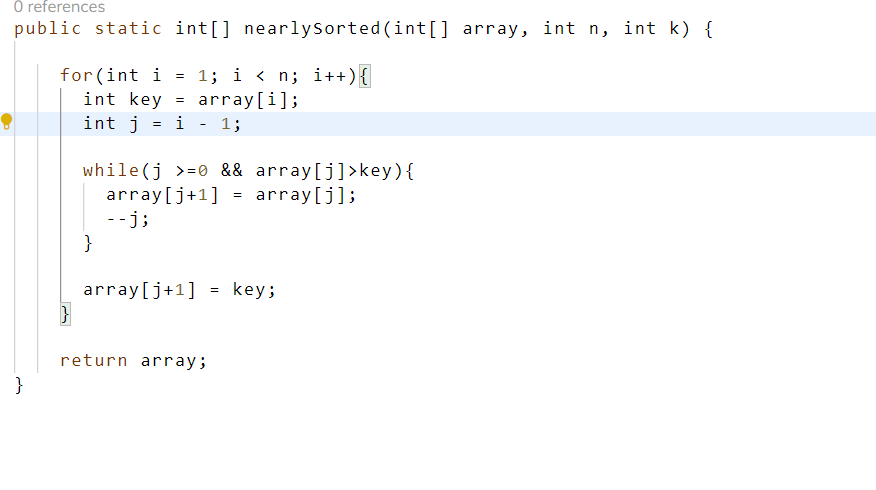
***Question: Sort a k – Sorted Array || nearly sorted array***

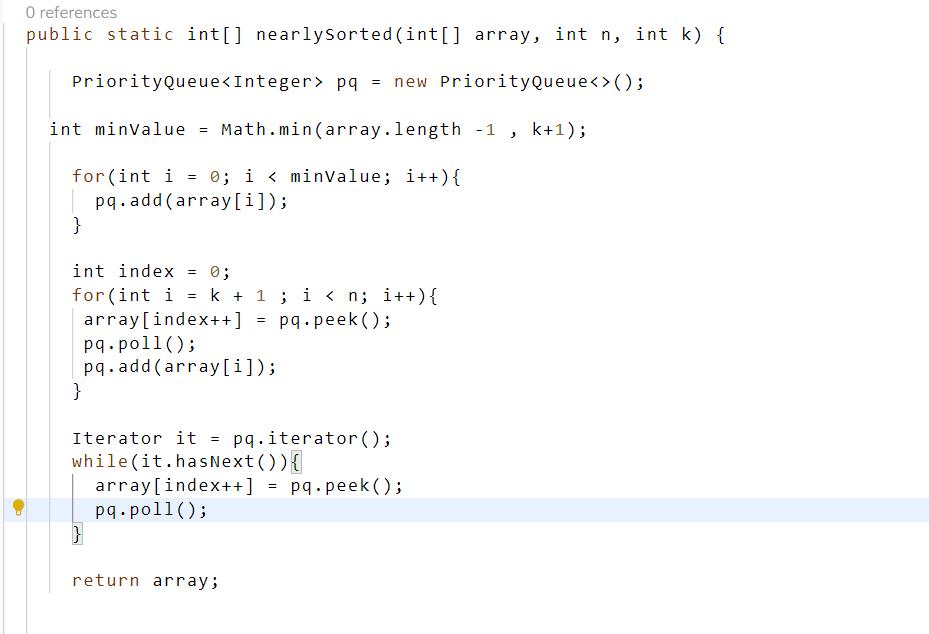
******

*In nearly sorted array* Given an array of **N** elements, where each element is at most K away from its target position

**

First approach we will try to solve this using insertion sort :

1. Take first element as sorted “Insertion sort”

Optimal Solution using min heap:  
  


Here what we are doing adding the k element into the min heap now after 2nd loop we have the min at the top of the heap we will peek() that and that last element into the array and we will return the array value;

***Question: K – Closest Number***

Given a **sorted** integer array arr, two integers k and x, return the k closest integers to x in the array. The result should also be sorted in ascending order.

An integer a is closer to x than an integer b if:

* |a - x| < |b - x|, or
* |a - x| == |b - x| and a < b

**Example 1:**

**Input:** arr = [1,2,3,4,5], k = 4, x = 3

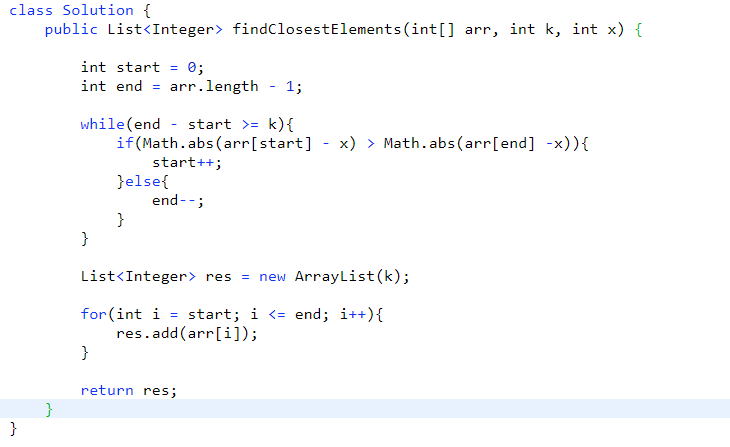
**Output:** [1,2,3,4]

*Solution:*

*We have K means total k number we need to find which is close to 3 !!*

*We have done that same thing in K – largest number from array*

*Here we have to find that*

*Optimal Solution 1: Two Pointer Approach *

*Optimal Solution 2: Priority Queue*