Date Page

Day - 188

Heap - 4

* k^{th} Smallest Element:

=) | 10 | 3 | 7 | 4 | 8 | 9 | 2 | 6 | , $k = 4$

=) In Brute force approach, we can sort the array & return that $(k-1)$ element.

=) So, T.C. → $O(n\log n)$

S.C. → $O(1)$ if you use Heap sort.

=) In the next approach, we can use min heap. Then, pop the top element k times.

=) And return that popped k^{th} element.

T.C. → $\underset{\text{Creation}}{O(n)} + \underset{\text{Deletion}}{k\log n}$

S.C. → $O(1)$.

=) In the next approach, we will select the 4 elements & find the 4^{th} smallest element.

=) Now, select the next element and check that element can become the part of that group of 4 by checking it with the largest element.

=) If ~~th~~ it is smaller than that, it will come into the group.

Date Page

⇒ So, we will use max heap. for creating group.

⇒ So, T.C. will be as —

→ $k + (n-k) * 2\log k$

→ $k + n\log k - k\log k$

→ $O(n\log k)$

* k^{th} Largest Element:

⇒

6	8	2	10	5	7	4	3

⇒ We will do same things as we do in last question.

⇒ But instead of using max heap, we will use min heap.

⇒ And if the upcoming is greater than the top element, we will add it to the heap.

⇒ We can also solve this question by using quick select method that uses quick sort. (Explore it on their own).

* k^{th} Largest Element in a stream:

1	2	3	4	5	6

Date ______ Page ______

⇒ We will have a stream of numbers & in that stream, we have to find k^{th} largest element.

⇒ Suppose, the value of $k = 4$ then in the starting 1 will come but here there are only one element, so answer is -1.

⇒ In the same, we have find all the answers & return it in vector.

* Sum of element b/w ~~b~~ k1 & k2 smallest element.

20	8	22	4	12	10	14

⇒ First approach, we can sort the array and then find the answer.

⇒ We don't have to include k1 & k2.

⇒ In the second approach, we will make two heap that contains k1 smallest elements & (k2 - 1) smallest elements.

⇒ Then do the sum of both heaps.

⇒ Now, subtract both the sum & return the result.

T.C. → O(n log k2)

S.C. → O(k1 + k2), if you are using priority queue or O(1) if you are using ~~bottom~~ step-down approach.