Date _____ Page _____

Day - 190

Heap - 6

* k^{th} Element in a Matrix:

$k = 6$

16	28	60	70
22	41	63	91
27	50	87	93
36	78	87	94

⇒ We have given a N×N matrix that is row wise sorted & column wise sorted.

⇒ And we have to find the k^{th} smallest element of matrix.

⇒ In first approach, we can create a 1D array then sort it & then return the k^{th} element from starting. → $O(n^2 \log n)$

⇒ In the second approach, we can create a min heap then pop the k^{th} k elements and return the answer → $O(k \log n)$

⇒ In the the above approach, we don't use the sorted property of our matrix.

Date _____ Page _____

⇒ We know that, the first smallest will be 16. So, second smallest will be 28 or 22.

⇒ Then, second smallest will be 22 so, next smallest will be 28, 27, or 41.

⇒ Now, in this way, we can find k^{th} smallest element.

⇒ But this will make code complex.

⇒ So, we will take one direction → row or col.

⇒ Suppose, we choose col, then, we will push all the 1^{st} col elements into the min heap.

⇒ Then pop the first min element & add its adjacent row element.

⇒ In this way, we can easily find the k^{th} smallest element.

⇒ Now, ~~space~~ time complexity for creating heap will be — O(N) & if you are using priority_queue then you can also get O(N) T.C. by using this code —

```
priority_queue<int> p(v.begin(), v.end());
priority_queue<int> p(arr, arr+N);
```

Date _____ Page _____

⇒ It is necessary to pass the iterator.

⇒ Now, we have to delete & insert elements, so, T.C. for this will be $2 \log n$.

⇒ For k elements → $k * 2 \log n$.

⇒ So, total T.C. → $n + k \log n$

⇒ We will push data, row & col → all three things in the priority queue for every element.

⇒ For this, we will use pair.
pair<int, pair<int, int>> p;

⇒ In the next approach, we can select first element & then pop it & add its ~~te~~ adjacent element.

⇒ Also, we don't have to add already ~~a~~ pushed element.

⇒ Now, suppose, if we have to find X → 6^{th} smallest element

⇒ For that, we can use Binary search.

⇒ We select first & last element the apply binary search on that.

⇒ we will get an element & then we count how many no. are smaller than that no.

Date _____ Page _____

⇒ And then we apply the properties of BS.

* Find Median in a Stream:

7	11	4	9	15	2	1	18

⇒ We will get elements in a stream & we have to find median of every stream.

⇒ 7 → 7

⇒ 7 11 → 9

⇒ 4 7 11 → 7 ①

⇒ 4 7 9 11 → 8 & so on.

⇒ We will use heap here. ②

⇒ Suppose.

2	4	6	7	8	10	12	14

Max heap Min heap ③

⇒ We can see that, for finding median, we will require 7 & 8.

⇒ So, as we can see 2, 4, 6, 7 are max heap and 8, 10, 12, 14 are min heap.

⇒ So, we will use top element of both heap & perform our calculation.

Date _____ Page _____

⇒ Now if next element is coming then we can check with max heap top for their correct position.

⇒ Any side is only +1 greater than other side.

⇒ If any side becomes greater +2 then shift one element to the other side.

Cases

	Max heap		Min heap
①	Left side	==	Right side

$$\frac{\text{Top} + \text{Top}}{2}$$

② Left side - 1 == Right side

left side top

③ ~~Left~~ Right side > left side

Left side > Right side + 1