

## Doubt Session

1. Diversify the array.

There are  $k$  operations only.

Each decrement of a duplicate gives a chance to introduce a new unique.

Max. new unique elements you can add =  $\min(K, \text{no. of duplicates})$

Total pairs in  $N$  elements,  $nC_2 = \frac{n \times (n-1)}{2}$

Exa :- Arr : 1 2 5 3  
Freq : 2 3 6 7

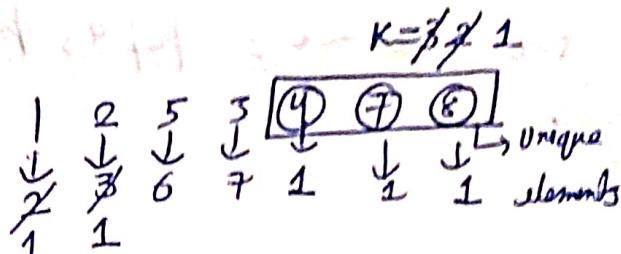
(2,2), (3,3) & (4,4)  $\rightarrow$  cannot contribute to diversity.

Out of  $K$  operations we can dec.  $K$  from existing duplicates and add to unique.

Total no. of pairs - duplicates

$$nC_2 - d$$

$$\text{Total pairs} = 18C_2 - 6C_2 - 7C_2$$



1  $\rightarrow$  1

2  $\rightarrow$  1

5  $\rightarrow$  6 }  $\rightarrow 6C_2$

3  $\rightarrow$  7 }  $\rightarrow 7C_2$

4  $\rightarrow$  1

7  $\rightarrow$  1

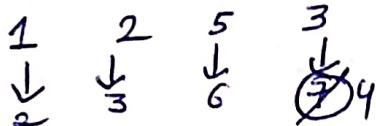
8  $\rightarrow$  1

Max - freq PQ.

(7,3)
(6,5)
(2,1)
(3,2)

Instead of target min freq element, target max freq element.

$K = 3$



$7-3=4$ .

$$\Rightarrow 18C_2 - 2C_2 - 3C_2 - 4C_2 - 6C_2$$

1. Use max priority queue (max, element).

2. Height of soldiers

Approach 8 -

1. Use a stack to keep indices of array elements. It will be a monotonic inc. stack.

2. For each element at index  $i$ :

while stack not empty & arr  $[i] < \text{arr}[\text{stack}[\text{top}]]$   
 $\Rightarrow$  pop from stack

Push  $i$  to the stack

3. Find min in subarrays

calculate

3. Each time you pop, ^ the distance where the element is min.

4. Repeat for each window / distance.

Ex: Array : 1 5 2 9 6 3

s-1: stack [0]

s-2:  $5 > 1$ , push  $\Rightarrow [0, 1]$

s-3:  $2 < 5$ , pop 1, push 2  $\Rightarrow [0, 2]$

s-4:  $9 > 2$ , push  $\Rightarrow [0, 2, 3]$

s-5:  $6 < 9$ , pop 3, push 4  $\Rightarrow [0, 2, 4]$

s-6:  $3 < 6$ , pop 4, push 5  $\Rightarrow [0, 2, 5]$