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, 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) → cannot contribute to diversity.

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

Total no. of pairs − duplicates
$nC_2 - D$

K = ~~3~~ ~~2~~ 1

1 2 5 3 ④ ⑦ ⑧ → Unique elements
↓ ↓ ↓ ↓ ↓ ↓ ↓
~~2~~ ~~3~~ 6 7 1 1 1
1 1

Total pairs − d = $18C_2 - 6C_2 - 7C_2$

1 → 1
2 → 1
5 → 6 } → $6C_2$
3 → 7 } → $7C_2$
4 → 1
7 → 1
8 → 1

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

K = 3

1 2 5 3
↓ ↓ ↓ ↓
2 3 6 ~~7~~ 4

7 − 3 = 4.

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

Max-heap PQ.

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

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

2. Height of Soldiers

Approach :-

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

2. For each element at index i :
   While stack not empty & arr [i] < arr [stack [top]]
   ⇒ pop from stack

   Push i to the stack

~~3. Find min in subarrays~~

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

4. Repeat for each window / distance.

<u>Exa</u>: Array : 1 5 2 9 6 3

S-1: stack [0]

S-2: 5 > 1 , push ⇒ [0, 1]

S-3: 2 < 5 , pop 1, push 2 ⇒ [0,2]

S-4: 9 > 2, push ⇒ [0, 2, 3]

S-5: 6 < 9 , pop 3, push 4 ⇒ [0, 2, 4]

S-6: 3 < 6 , pop 4, push 5 ⇒ [0, 2, 5]