1984. Minimum Difference Between Highest and Lowest of K Scores

Hint

Easy 852 193

Companies

You are given a **0-indexed** integer array `nums`, where `nums[i]` represents the score of the `i^th` student. You are also given an integer `k`.

Pick the scores of any `k` students from the array so that the **difference** between the **highest** and the **lowest** of the `k` scores is **minimized**.

Return *the **minimum** possible difference*.

Example 1:

```
Input: nums = [90], k = 1
Output: 0
Explanation: There is one way to pick score(s) of one student:
- [90]. The difference between the highest and lowest score is 90 - 90 = 0.
The minimum possible difference is 0.
```

Example 2:

```
Input: nums = [9,4,1,7], k = 2
Output: 2
Explanation: There are six ways to pick score(s) of two students:
- [9,4,1,7]. The difference between the highest and lowest score is 9 - 4 = 5.
- [9,4,1,7]. The difference between the highest and lowest score is 9 - 1 = 8.
- [9,4,1,7]. The difference between the highest and lowest score is 9 - 7 = 2.
- [9,4,1,7]. The difference between the highest and lowest score is 4 - 1 = 3.
- [9,4,1,7]. The difference between the highest and lowest score is 7 - 4 = 3.
- [9,4,1,7]. The difference between the highest and lowest score is 7 - 1 = 6.
The minimum possible difference is 2.
```

Constraints:

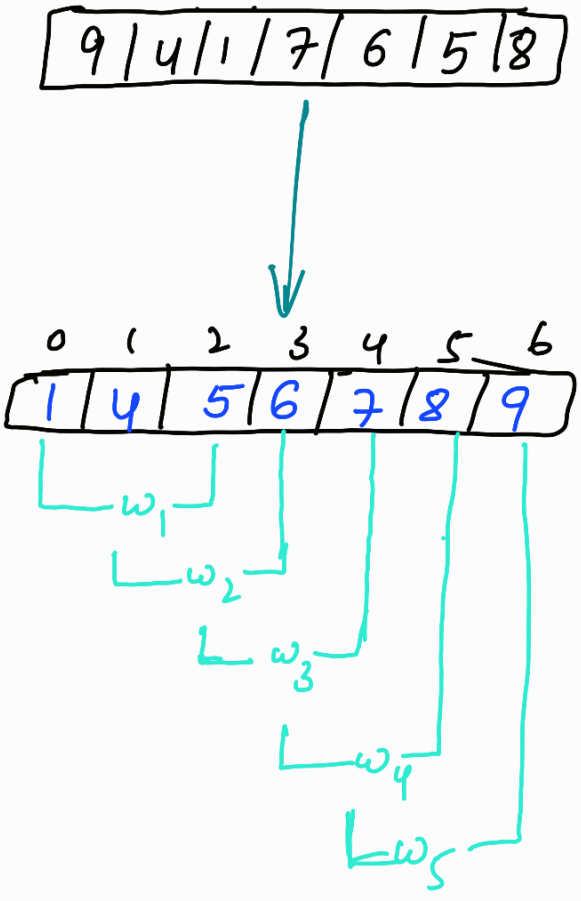
- `1 <= k <= nums.length <= 1000`
- `0 <= nums[i] <= 10^5`

Accepted **54K** | Submissions **97.6K** | Acceptance Rate **55.3%**

This is a classic example of sliding window technique.

Window Technique

→ Sort the array.
→ Then check all the k sized windows and get the minimum.



$l = 0$

$r = k - 1$

Out of all windows difference

$(\omega_2 = \omega_3 = \omega_4 = \omega_5) < \omega_1$

and min is 2.

```
while( r <= n-1)
{
    if ( nums[r]-nums[l] < min)
        min = nums[r] - nums[l]

    l++
    r++
}
```

$T(n) = O(n \log n)$

TAKEAWAY:

If we are given a k, and asked to find something in the k no. of elements then see if we could use "Sliding window technique".