Find the Kth Largest Array Element [LeetCode](https://leetcode.com/problems/kth-largest-element-in-an-array/description/)

Given an integer array nums and an integer k, return *the* kth *largest element in the array*.

Note that it is the kth largest element in the sorted order, not the kth distinct element.

Example: [3,2,1,5,6,4], k = 3

Output: The 3rd largest element: 4

**Approach 1: Function to find the kth largest element in an array using a brute force approach**

* **Function Purpose:** To find the kth largest element in an array using a brute force approach.
* **Explanation:**
  + Sort the array in ascending order and return the kth element from the end.
* **Time Complexity:** **O(N \* log(N)), where N is the number of elements in the array.**
* **Space Complexity: O(1) since it operates in-place.**

**Approach 2: Function to find the kth largest element in an array using a max heap**

* **Function Purpose:** To find the kth largest element in an array using a max heap.
* **Explanation:**
  + Use a max heap to store elements.
  + Remove the k - 1 largest elements, leaving the kth largest.
* **Time Complexity:** **O(N \* log(N)), where N is the number of elements in the array.**
* **Space Complexity: O(N) due to the max heap.**

**Approach 3: Function to find the kth largest element in an array using a min heap**

* **Function Purpose:** To find the kth largest element in an array using a min heap.
* **Explanation:**
  + Use a min heap to store elements.
  + Remove elements until the heap size is k, leaving the kth largest.
* **Time Complexity:** **O(N \* log(N)), where N is the number of elements in the array.**
* **Space Complexity: O(N) due to the min heap.**

**Approach 4: Function to find the kth largest element in an array using a min heap (Optimized Approach)**

* **Function Purpose:** To find the kth largest element in an array using an optimized min heap approach.
* **Explanation:**
  + Create a min heap to maintain the k largest elements.
  + Insert the first k elements into the min heap.
  + For the remaining elements, if an element is greater than the current minimum in the heap, replace the minimum with the larger element.
  + The top element of the min heap is the kth largest element.
* **Time Complexity:** **O(N \* log(K)), where K is the value of k.**
* **Space Complexity: O(K) for the min heap.**

**Conclusion:**

* All four approaches yield the same result for finding the 3rd largest element, which is 4.
* The Brute Force approach is straightforward but has a time complexity of O(N \* log(N)).
* The Max Heap and Min Heap approaches also have a time complexity of O(N \* log(N)) but require O(N) space for the heap.
* **The Optimized Min Heap approach has a better time complexity of O(N \* log(K)) and requires O(K) space for the heap.**