Find the Kth Smallest Array Element [CodeStudio](https://www.codingninjas.com/studio/problems/kth-smallest-element_893056?leftPanelTab=1)

Given an integer array nums and an integer k, return the k th smallest element in the array. Note that it is the k th smallest element in the sorted order, not the k th distinct element.

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

Output: The 3rd smallest element: 3

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

* **Function Purpose:** To find the kth smallest element in an array using a brute force approach.
* **Explanation:**
  + Sort the array in ascending order and return the kth element.
* **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 smallest element in the array using a max heap**

* **Function Purpose:** To find the kth smallest element in an array using a max heap.
* **Explanation:**
  + Use a max heap to store elements.
  + Remove elements from the max heap until the kth smallest element is found.
* **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 smallest element in the array using a min heap**

* **Function Purpose:** To find the kth smallest 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 smallest.
* **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 smallest element in the array using a max heap (Optimized Approach)**

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

**Conclusion:**

* All four approaches yield the same result for finding the 3rd smallest element, which is 3.
* 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 Max Heap approach has a better time complexity of O(N \* log(K)) and requires O(K) space for the heap.**