When encountering a problem that needs sorting array or list, and you are sure there is a much more efficient algorithm than O(nlogn), then you should consider all sorting alrogithms and see whether we can devise one efficient solution from one of them.

Especially, the most oftenly used algorithms include:

* Insertion Sort
* Bubble Sort
* Selection Sort
* Merge Sort
* Quick Sort
* Heap Sort
* Radix Sort

Examples:

**§Sort a nearly sorted (or K sorted) array**

Given an array of n elements, where each element is at most k away from its target position, devise an algorithm that sorts in O(n log k) time.   
For example, let us consider k is 2, an element at index 7 in the sorted array, can be at indexes 5, 6, 7, 8, 9 in the given array.

1) Create a Min Heap of size k with first k elements. This will take O(k) time

2) One by one remove min element from heap, put it in result array, and add a new element from remaining elements.

3) Removing an element and adding a new element to min heap will take Logk time. So overall complexity will be O(k) + O((n-k)\*logK)