# Top K problems - Sort, Heap, and QuickSelect

since2020 ★ 798 Last Edit: March 2, 2021 8:59 PM 5.5K VIEWS

## Overview

Top K problem asks to find the Kth largest/smallest element or the top K largest/smallest items from an unsorted array. It's one of the questions that you may encounter during an interview. Fortunately, the solutions to it is quite standard, which are mainly: 1) **sort**; 2) **using a heap**; 3) **Quick Select**.

Sorting first and then returning the Kth item definitely work, yet it yields O(N^2) or O(NlogN) time complexity, depending on which sorting algorithm you will choose. However, with the help of heap or QuickSelect, better time complexties could be achieved. This post will try to help you nail this kind of questions by introducing all of the solutions one by one.

> Let's take Leetcode 215. Kth Largest Element in an Array as an example.

### Sort - O(NlogN)

Sorting algorithms could be applied first before returning the Kth item. Time complexities for the following code would be O(NlogN).

---

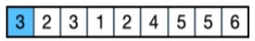
### Heap - O(NlogK)

We can implement a size-`K` Min-heap, and `heap[0]` would be the answer. More specifically:

- We start with an empty heap, and `push` items into it before the size equals to `K`.
- Then for the remaining items:
  - we ignore those which are smaller than `heap[0]`, because they won't contribute to the top `K` largest elements;
  - when the new item is larger than `heap[0]`, we would switch it with `heap[0]` first, and call `heapify(0)` so that the heap structure could be maintained.

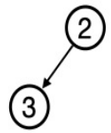
The height for a heap of size `K` is O(logK), therefore the total time complexity would be O(NlogK) since we need to `push` O(N) times.


Find the 4th largest element

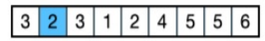



A min-heap with size of 4

Input array

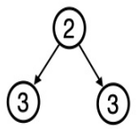



Find the 4th largest element

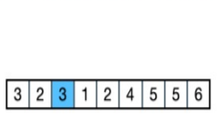



A min-heap with size of 4

Input array

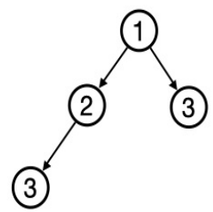



Find the 4th largest element

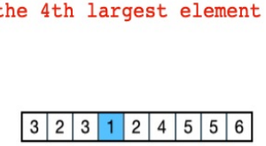



A min-heap with size of 4

Input array

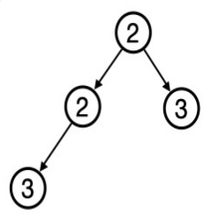



Find the 4th largest element

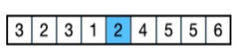



A min-heap with size of 4

Input array

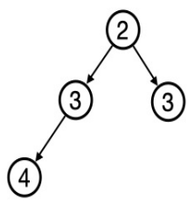



Find the 4th largest element

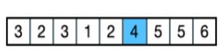



A min-heap with size of 4

Input array

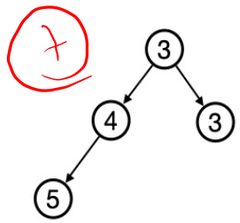



Find the 4th largest element



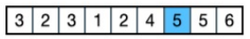

A min-heap with size of 4

Input array

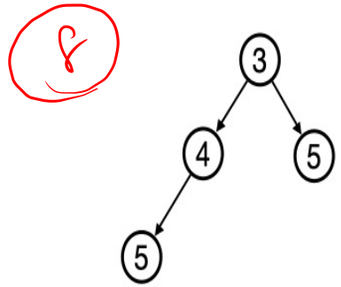



A min-heap with size of 4

Find the 4th largest element

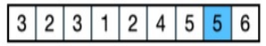



Input array

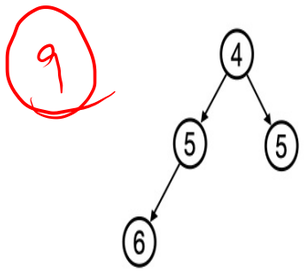



A min-heap with size of 4

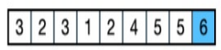
Find the 4th largest element




Input array




Find the 4th largest element




**Quick Select - O(N)**

Quickselect takes advantage of the `partition` function from Quick Sort. An explanation for `partition` in Leetcode:

> One chooses a pivot and defines its position in a sorted array in a linear time using so-called partition algorithm.

However, after we place the pivot in the right place by calling `partition(left, right)`, we won't recursively call both `partition(left, pivot - 1)` and `partition(pivot + 1, right)`, because we don't need to! Instead, we discuss the relation between `K` and `pivot`:

- If `K < pivot`, then the Kth largest item must be in `(left, pivot - 1)`.
- Else if `K > pivot`, then the Kth largest item must be in `(pivot+1,right)`.
- Else, we know `K == pivot`, we get the answer!

If we recursively run both halves like in Quick Sort, it would take O(NlogN). However, this approach keeps throwing away one half (in average case) of the array, which leads to O(N) time complexity in the average case, though it might reach O(N^2) in the worst case due to the instability of quicksort. (Wikipedia)

Give a Real life sorting algorithm in Surgery a Lecturer Data Pelp & Surgery interrelated