(5) **(Reducing kth-smallest to median-finding)** (15 points) If we have an algorithm that finds the kth-smallest element of an n element set, we can obtain an algorithm for finding the median element simply by calling the kth-smallest algorithm with $k = \lceil (n+1)/2 \rceil$. This question asks you to do the converse.

Given an algorithm that finds the median element of an n element set in $\Theta(n)$ time, design an algorithm that finds the kth-smallest element for arbitrary k in $\Theta(n)$ time using the median-finding algorithm. Be sure to analyze the running time of your algorithm.

(Note: Do *not* invoke the linear-time kth-smallest algorithm presented in class and the book. You must reduce the problem of finding the kth-smallest element to the problem of finding the median.)

Note that Problems (3)(c) and (4)(e) are *bonus* questions, and are *not* required.

① Since we are given an algorithm to find the median we will call it to find the median element of the array.

② Once we have the median we can create two new arrays - One that holds all the elements less than the median value and the other to hold the elements greater than the median.

Thus, we'll have

min A $\boxed{*|*|\cdots|*}$ median max A $\boxed{*|*|\cdots|*}$

③ From there we can get the index of the median where it would be in a sorted array by getting the length of min A. The length of min A will give us the position of the median element in a sorted array.

④ Now that we have the index of the median if that index equals k then we return the median element. If the median index is greater than k then we recursively call the algorithm on min A and if its less than k we recursively call the algorithm on max A.

<u>Time analysis</u>

① Getting the median $\rightarrow O(n)$ time

② In order to create the two extra arrays we simply iterate over every element of the original array and sort them into the two lists and skip over the median value $\rightarrow O(n)$ time

③ Finding the index of the median will be constant time of finding the length of min A $\rightarrow O(1)$ time

④ Recursively calling on min A or max A will be recursively calling on arrays that are the size of $\lceil \frac{n+1}{2} \rceil - 1 \rightarrow T(n) = T(\lceil \frac{n+1}{2} \rceil - 1)$

Total = $O(n) + O(n) + O(1) + T(\lceil \frac{n+1}{2} \rceil - 1)$

We can approximate $T(\lceil \frac{n+1}{2} \rceil - 1)$ to $T(\frac{n}{2})$

so $T(n) = T(n/2) + 2O(n) = T(n/2) + O(n)$

and using the master's theorem $a = 1$ $b = 2$ $c = 1$ $d = 0$

$\log_b a = \log_2 1 = 0 < c$

$\therefore T(n) = \Theta(n^c \log^d n) = \Theta(n)$ //