1. Consider the following QuickSort algorithm (at a high level):

1: function QuickSort(A)  \(\triangleright\) A is an unsorted list of size \(n > 1\)
2: Pick “pivot” element \(p \in A\) (e.g., let \(p = A[0]\))
3: Partition \(A\) into \(B = \{x \in A : x \leq p\}\) and \(C = \{x \in A : x > p\}\) (in place)
4: Call QuickSort recursively on \(B\) and \(C\) (in place)
5: end function

Analyze the running time of this algorithm.

2. Find a good (as efficient as possible) divide and conquer algorithm that given an unsorted list \(A\) of distinct numbers and a “rank” \(k \in \{1, 2, \ldots, |A|\}\) returns the \(k^{th}\) smallest element of \(A\).