a Describe an optimal Divide-&-Conquer algorithm MAXMIN such that for array $A[1..n]$ and $1 \leq lo \leq hi \leq n$, the call MAXMIN($lo$, $hi$) returns the ordered pair $(max, min)$ which are respectively the maximum and the minimum elements of $A[lo..hi]$. So when invoked on array $(28, 210, 3, 812, 42)$, the call MAXMIN(1,5) would return the pair (812, 3), and MAXMIN(1, 3) would return (210, 3).

\[
\text{MAXMIN}(lo, hi) \\
\text{if } lo=hi \text{ then return } (A[lo], A[lo]) \\
\text{mid } \leftarrow \left\lfloor (lo + hi)/2 \right\rfloor \\
(min_{lo}, max_{lo}) \leftarrow \text{MaxMin}(lo, mid) \\
(min_{hi}, max_{hi}) \leftarrow \text{MaxMin}(mid + 1, hi) \\
\text{return } (\min(min_{lo}, min_{hi}), \max(max_{lo}, max_{hi}))
\]

b Show that your algorithm is optimal.

Letting $t(n)$ denote the time taken by MAXMIN(1,$n$), we note that it satisfies the recurrence

\[
t(n) = \begin{cases} 
2t(n/2) + c_0, & \text{if } n > 1 \\
c_1, & \text{if } n = 1
\end{cases}
\]

which yields $t(n) \in \Theta(n)$. To see that this is optimal, assume that some algorithm solves the problem in less than linear time. Since it doesn’t touch every element on any input, modify any input by changing an untouched element to $\infty$ or $-\infty$. The algorithm would still return the same answer, but it would be wrong. This contradicts the assumption of the existence of a sublinear time algorithm.