For this problem, will use proof by contradiction. For any correct, deterministic algorithm, to even read through the entire input array would take time $\Omega(n)$. So, if there's an algorithm for the problem that runs in time o(n), there must exists some $n_0 < n$ and array $A[1, \ldots, n_0]$ for which the algorithm does not even read all the array entries.

However, in order to guarantee the algorithm is correct, one must check each entry in A in order to make sure that they are sorted in non-decreasing order, which means that for each entry $x_i \in A$, $A[x_i] \leq A[x_{i+1}]$. And this is the only way to make sure that the array is sorted in non-decreasing order as if the algorithm fails to check any entries in the input array, it can not guarantee that the array is sorted in non-decreasing order, for that missed entry x_i may be smaller than x_{i-1}.

As a result, any correct, deterministic algorithm must check all n entries in array A. Hence, there does not exist $n_0 < n$ and array $A[1,\ldots,n_0]$ for which the algorithm does not even read all the array entries. And this is a contradiction, so for any correct, deterministic algorithm for this problem, it would run in time $\Omega(n)$.