A priority verifier supports the operations, Insert(x), Delete(x), and Not-all-bigger(z), being applied to a dynamic set. For the insert operation, both the name and key-value of an item are provided as input. For the deletion operation, the name of the item to be deleted is provided as input. For the not-all-bigger operation, the input consists of a value from the key space; and the operation returns “yes” provided that some item currently in the set has a key value not greater than z, and “no” otherwise.

We proceed to demonstrate that these operations cannot be supported in $o(\log n)$ amortized time when $n$ items are present, under the comparison based model of computation. The comparison based model of computation in the context of this problem allows for comparisons, $x_i : x_j$ between two items in the set, as well as comparisons, $z : x_i$, where $z$ is the input supplied to a not-all-bigger operation underway. First, we define an adversary $A_\pi$, where $\pi$ is a permutation on $[1, n]$, which performs $n$ insertions, $n$ deletions, and $n$ not-all-bigger operations. Second, we show that for some permutation $\pi$, the combined number of comparisons executed by the $3n$ operations performed by $A_\pi$ is at least $\log_2 n!$, and the desired conclusion follows.

The adversary $A_\pi$ proceeds as follows:

```
for i = 1 to n insert(x_i with key \(\pi(i)\))
for i = 1 to n {
    call not-all-bigger(i + 1/2);
    delete(x_j) where j is chosen so that \(\pi(j) = i\), so that the key of x_j is i.
}
```

We readily observe that with the given choices for the arguments, each call to not-all-bigger returns “yes”, justified by exactly one item currently in the set, the one having the smallest key. This smallest item is then immediately deleted.

Next, we consider the computation tree $T$ that reflects and is generated by the possible executions of the $A_\pi$ computations across the ensemble of the $n!$ possible permutations $\pi$. $T$ consists of circular nodes, each designating a particular comparison, and each having at most two outcome edges (less or greater, there being no pairs of equal values at play); and square nodes, each indicating the completion of some not-all-bigger operation. To each square node there are attached out-going edges, each labeled with an item name $x_j$, to be interpreted as indicating that $x_j$ is being supplied as the argument for the follow-on delete command. We show that at least one path through $T$ (therefore some $A_\pi$ computation) entails at least $\log_2 n!$ comparisons.

The following two claims will be demonstrated.
Claim 1. A square node of $T$ has only one out-going edge.

Claim 2. The $n!$ paths corresponding the possible executions $A_\pi$ are all distinct.

Now observe that as a consequence of Claim 1, the branchings that occur among the paths of $T$, which by Claim 2 are $n!$ in number, take place exclusively at binary nodes, so that at least one path in $T$ passes through at least $\log_2 n!$ binary decision (comparison) nodes, establishing the main conclusion.

The second claim follows from the observation that the labels of the edges proceeding from square nodes along the computation path for $A_\pi$ in $T$ uniquely determine $\pi$.

To establish the first claim we observe that during an embedded call (within $T$) to not-all-bigger with parameter $z = i + 1/2$, there can be at most one identifier $x_j$ for which a comparison establishes that $z > x_j$, since only one identifier has its key being the current minimum value $i$. Moreover, prior to the completion of the call, some comparison $C$ must actually take place showing an outcome $z > x_j$ because otherwise in the absence of any such comparison, all of the other comparison outcomes are consistent with the possibility that $z < i$, in which instance not-all-bigger should return “no”. So without having performed the critical comparison $C$, the execution (upon completion) would not have faithfully determined the proper outcome of the associated not-all-bigger operation. Thus, the path arriving at any particular square node is consistent with only one item identifier being paired with the minimum key value – the unique identifier $x_j$ for which $z > x_j$ is a preceding comparison outcome (within the just completed call execution). And $x_j$ can be the only label appearing on an outgoing edge from the square node. Therefore the ensemble of $A_\pi$ computations can cause but one outgoing edge to materialize from any square node.