The sublists must be consecutive items (a segment).

Let \( L \) be the list, and let \( j : \text{nat} \) be an index variable, and let \( m : \text{nat} \) be a variable recording the length of the longest sorted segment ending at index \( j \), and let \( n : \text{nat} \) be a variable recording the length of a longest sorted segment anywhere in the interval \( 0;..j \). The result is the final value of \( n \). Define \( \text{sorted} \) so that \( \text{sorted} \ h k \) tells whether the segment \( h;..k \) is sorted,

\[
\text{sorted} = \langle h, k : \text{nat} \rightarrow \forall i, j : h,..k : i \leq j \Rightarrow Li \leq Lj \rangle
\]

and define \( \text{llss} \) so that \( \text{llss} \ j \) is the length of a longest sorted segment ending at index \( j \)

\[
\text{llss} = \langle j : 0;..\#L+1 \rightarrow \text{MAX} \ i : 0;..j+1 : \text{if} \ \text{sorted} \ i \ j \ \text{then} \ j-i \ \text{else} \ -\infty \ \text{fi} \rangle
\]

Then the problem is \( P \), defined as

\[
P \iff \text{if} \ \#L \leq 1 \ \text{then} \ n:= 0 \ \text{else} \ Qi \ \text{fi} \]

\[
Q \iff \text{if} \ L(j-1) \leq Lj \ \text{then} \ j:= j+1 \ \text{else} \ Qi \ \text{fi}
\]

The timing is \( t' \leq t + \#L \) and \( t' \leq t + \#L - j \).

This time let \( \text{llss} \ n \) be the length of a longest sorted sublist of list \( L[0;..n] \). We can represent a sublist of \( L \) by a set \( S \) of indexes, which is a subset of \( \{0;..n\} \). Formally,

\[
\text{llss} \ n = \text{MAX} \ S : (\langle S' : \langle 0;..n \rangle \ \forall i, j : S' \ i \leq j \Rightarrow Li \leq Lj \rangle \cdot SS)
\]

And this time I'll need a \textbf{for-loop},

\[
\begin{align*}
n' \ = \ & \text{llss} \ (#L) \iff s:= 0. \ I0 \Rightarrow I'(\#L) \\
I0 \Rightarrow I'(\#L) \iff n:= 0;..\#L \text{ do } In \Rightarrow I'(n+1) \text{ od}
\end{align*}
\]

where \( In = s = \text{llss} \ n \) and we have yet to refine \( In \Rightarrow I'(n+1) \). As we go from \( n \) to \( n+1 \), the new sublists are \( [Ln] \) whose length is \( 1 \), and for each sorted sublist \( S \) in \( L[0;..n] \) whose last item is less than or equal to \( Ln \), the list \( S'(Ln) \) whose length is \( \#S + 1 \). To calculate that, we will form a new list \( M \) such that \( Mk \) is the length of the longest sorted sublist whose last item is \( Lk \). We strengthen invariant \( I \).

\[
\begin{align*}
In \ = \ & s = \text{llss} \ n \\
& \wedge \forall k : 0;..n : Mk = \text{MAX} \ S : (\langle S' : \langle 0;..k+1 \rangle \ \forall i, j : S' \ i \leq j \Rightarrow Li \leq Lj \rangle \cdot SS)
\end{align*}
\]

Note that \( s = \text{MAX} \ (M[0;..n]) \) except when \( n = 0 \). The remaining refinement will also use a \textbf{for-loop}, for which the invariant is

\[
\begin{align*}
Jm \ = \ & (\forall k : 0;..n : Mk = \text{MAX} \ S : (\langle S' : \langle 0;..k+1 \rangle \ \forall i, j : S' \ i \leq j \Rightarrow Li \leq Lj \rangle \cdot SS) \\
& \wedge Mn = \text{max} \ 1 \ (\text{MAX} \ k : (\langle k : 0;..m : \text{Lk} \leq Ln \rangle \cdot Mk + 1))
\end{align*}
\]

And the remaining refinements are

\[
\begin{align*}
In \Rightarrow I'(n+1) \iff Mn:= 1. \ (J0 \Rightarrow J'n) \wedge s' = s. \ s:= \text{max} \ s \ (Mn) \\
J0 \Rightarrow J'n \wedge s' = s \iff \text{for} \ m := 0;..n \text{ do } (Jm \Rightarrow J'(m+1)) \wedge s' = s \text{ od} \\
(Jm \Rightarrow J'(m+1)) \wedge s' = s \iff \text{if} \ Lm \leq Ln \text{ then } Mn:= \text{max} \ (Mn) \ (Mn+1) \text{ else ok fi}
\end{align*}
\]

If you object that the \textbf{for-loop} specification \( J0 \Rightarrow J'n \) \wedge s' = s \ is not exactly in the right form, I could use \textbf{frame} to put it in the right form, or use the more general \textbf{for-loop} rule
that does not use an invariant. The solution just given has running time \((#L)^2/2\). For a solution with running time bounded by \((#L) \times \log (#L)\), instead of maintaining the list \(M\) of lengths of longest sorted sublists, maintain the list of minimum last items for each length, and replace the inner loop with a binary search.