note that $(i+1)^2 - i^2 = i^2 + 2i + 1 - i^2 = 2i+1$.

For $i = 0, 1, 2, \ldots$,
start with an empty sequence T_0 of LC trees and $i = 0$.

After $2i+1$ operations, construct T_{i+1} from T_i and make $i \leftarrow i+1$.

The number of operations when T_i is built is exactly i^2.
So we will have i LC trees in T_i, each with i more operations.

Specifically call the operations $0, 1, \ldots, i^2 - 1$. The jth LC tree in T_i includes the operations $0, 1, \ldots, ji - 1$ for $j = 1, \ldots, i$.

To go from $m_i = i^2$ operations to $m_{i+1} = (i+1)^2$ note that the new $2i+1$ operations can be intertwined with the old i^2 in an arbitrary order in the operations sequence.

Call t_j the position of the $j.i-1$ old operation for $j = 1, \ldots, i$.
This is the last operation in the j-th LC tree of T_i.

Let $t_0 = -1$. We want to know the interval in which each t_j falls with respect to the $k(i+1)$-th current $(i+1)^2$ operations.
The $k(i+1)$-th current operation is the first one in the k-th LC tree in T_{i+1} that is not in the $(k-1)$-st LC tree in T_{i+1}.

Claim 1: $t_j < (j+2)(i+1)$ for $i \geq 0$ and $j = 0, 1, \ldots, i$.

Pf: note that

$$t_j \leq ji - 1 + 2i + 1 = (j+2)i < (j+2)(i+1). \quad \square$$

Therefore we can build the $(j+2)$-th LC tree for the $(i+1)^2$ operations from the jth LC tree for the i^2 old operations.

Specifically we build the two first LC trees from scratch, with the $i+1$ and the $2(i+1)$ first of the $(i+1)^2$ operations.

To build the jth LC tree for the $(i+1)^2$ operations, for $j = 3, \ldots, i+1$, we start from the $(j-2)$th LC tree for the i^2 operations, which already include $(j-2)i$ operations, and add the missing $j(i+1) - (j-2)i = j + 2i$ operations (some new ones from the past and some old ones in the sequel).

This amounts for $j + 2i \leq i + 1 + 2i = 3i + 1 < 3(i+1) = 3\sqrt{m_{i+1}}$ insertions in an LC tree with less than m_{i+1} elements.

So the total amount of insertions we would have to do, to update the $\sqrt{m_i}$ LC trees into the $\sqrt{m_{i+1}}$ new LC trees, is less than $3m_{i+1}$, for a total amortized cost of $O(m_{i+1} \cdot \lg m_{i+1})$.

Amortizing by the $2i + 1 < 2(i+1) = 2\sqrt{m_{i+1}}$ new operation, we get an amortized cost of $O(\sqrt{m_{i+1}} \lg m_{i+1})$ per operation.

So clarifying the algorithm behind these ideas,
we keep track of a sequence of edge insertions,
each with on associated time stamp, sorted by time stamp.

Every time the number of operations in the sequence reaches a perfect square, we reorganize the data structure we will keep to answer queries on the MST cost at any instant in time.

Let $m_i := i^2$. We will describe how to reorganize the data structure from when the sequence of edge insertions had m_i operations to when it reached m_{i+1} operations.

The data structure for a sequence of m_i edge insertions consists of a sequence of i LC trees.
The j-th LC tree contains an MST for the operations from 0 to $j \cdot i - 1$.

One could think of building the jth LC tree from a copy of the $(j-1)$st LC tree adding the next i operations. But this is too costly.

We will instead update one of the LC trees from the previous sequence, namely the $(j-2)$th one, to obtain the jth LC tree of the new sequence.

So we will add two empty LC trees T_{-1}, T_0 to the beginning of the old sequence $T_1, \dots, T_i$, then we will obtain T_j from T_{j-2} for $j = 1, \dots, i+1$ by applying the $j+2i$ operations from the current sequence that are missing for T_{j-2} include the first $j(i+1)$ operations of the current sequence of edge insertions.