The longest common extension (LCE) of two indices in a string is the length of the longest identical substrings starting at these two indices. The LCE problem asks to preprocess a string into a compact data structure that supports fast LCE queries.

In this paper we generalize the LCE problem to trees and suggest a few applications of LCE in trees to tries and XML databases. Given a labeled and rooted tree T of size n, the goal is to preprocess T into a compact data structure that support the following LCE queries between subpaths and subtrees in T. Let v1, v2, w1, and w2 be nodes of T such that w1 and w2 are descendants of v1 and v2 respectively.

- **LCEPP(v1, w1, v2, w2):** (path-path LCE) return the longest common prefix of the paths v1 ~→ w1 and v2 ~→ w2.

- **LCEPT(v1, w1, v2):** (path-tree LCE) return maximal path-path LCE of the path v1 ~→ w1 and any path from v2 to a descendant leaf.

- **LCETT(v1, v2):** (tree-tree LCE) return a maximal path-path LCE of any pair of paths from v1 and v2 to descendant leaves.

We present the first non-trivial bounds for supporting these queries. For LCEPP queries, we present a linear-space solution with O(log* n) query time. For LCEPT queries, we present a linear-space solution with O((log log n)²) query time, and complement this with a lower bound showing that any path-tree LCE structure of size O(n polylog(n)) must necessarily use Ω(log log n) time to answer queries. For LCETT queries, we present a time-space trade-off, that given any parameter τ, 1 ≤ τ ≤ n, leads to an O(nτ) space and O(n/τ) query-time solution. This is complemented with a reduction to the set intersection problem implying that a fast linear space solution is not likely to exist.