02-613 Week 9

Algorithms and Advanced Data Structures

Aidan Jan

October 20, 2025

Suffix Tries

Problem: Given a large known, fixed **text**, e.g. a dictionary, and many unknowns of changing search queries, how do we design a data structure that can quickly search up terms in the structure, but also substrings?

- For example, if "uninterested" was in the dictionary, and we wanted to search for "interested" or "interest", they should both show up.
- Using any of our structures so far, there is not an efficient way to do this.
- What we *can* do is make a tree, where each node represents a letter, where every suffix is a path from the root to some leaf node.

[insert image here]

This structure allows for searching for any word in our dictionary, but also any suffix.

- This is known as a **trie**. (Not a tree, since trees don't allow internal nodes with only one child.)

Runtime of Tries

The runtime of a trie is $\Theta(\Sigma)$, where Σ is the size of our alphabet. For the English alphabet, $\Sigma = 27$ (26 letters, plus an end-of-word marker, \$.)

- Similarly, a DNA sequence would have $\Sigma = 5$ since it includes $A, C, T, G, \$$.
- We add a \$ to be able to distinguish when a valid string ends. If we want to find if a query is a suffix of our text, then the terminators become very important.

Searching in Tries

To search for a term q in our text T, we simply follow the path down from the root, since every substring would have a unique path from the root.

- If we want to know if q is a suffix of T, we can do that too since q would be a substring of T.
- If we want to know the number of times q is in T, then we follow the path down for q, and at the last character, count the number of leaf nodes in the subtree.
- To find the longest repeated substring, we return the deepest branching node.
- To find the lexicographically smallest suffix, from the root we follow the tree down always following the branch with the smallest character, assuming \$ is less than any other character in our alphabet.

Suffix Trees

To be continued...