Suffix trees (**?**) are a representation of a string designed for performing search related algorithms. They are based on a suffix trie of a given string **T**. The trie is constructed using all of the suffixes of **T** with a terminator character not used in the alphabet appended to the end of **T**. The terminator is deemed to be lexicographically of lower priority than all of the other characters to ensure that a prefix of a string would appear before that string (such as in *as* before *ash*). In the following example, the suffixes of *T: abaaba* would produce the following suffixes with **$** applied as a terminator.

Figure 1: Example Suffix Trie

This allows for efficient searching and counting substrings of **T** by following the trie from the root node. Each substring **S** of **T** is a prefix of a path starting at the root. It allows for counting occurrences of the substring by finding the number of leaf nodes that can be reached from the end of the substring. This format is not particularly space efficient as it has an upper bound storage in the order of $O(n^2)$.

A far more efficient way to store and query the trie is to store it as Suffix Trees. In order to produce a suffix tree from a trie, the non-branching nodes are firstly reduced (or coalesced) in to a single edge. The original string **T** is then stored along side the tree and the labels further reduced to a starting position and offset within **T**. The leaf nodes become labels to the offset of the suffix in the string. This reduces the upper bound storage to $O(n)$ and results in a tree for our example **T** as seen in **??** .

Figure 2: Example Suffix Tree (edges shown as text and as positions/offsets)

This technique of building the tree is considered naive though as it is again very inefficient (being computationally in the order of $O(n^3)$). It is far more desirable to achieve time-linear *O(n)* construction of the tree. An example of this is the Ukkonens Algorithm (**?**). The algorithm is too long for inclusion but effectively starts with the implicit suffix tree of a string of length 1 and conditionally extends the tree on each iteration of adding a character.

At Birkbeck, an infrastructure has been developed to load and query Suffix Trees (**?**). At a high level it is based on language models but should be extendable to time series by the use of SAX. There is current work in progress to utilise external memory (in this case Solid State Drives) to back the loaded Suffix Trees. This would massively increase the maximum size of a tree to be queryable. The current libraries to utilise this are written in Python however there is currently a refactor happening to C due to Python not being particularly computationally efficient because of its interpreted nature. The trees in this implementation are stored as k-truncated trees for practical reasons.