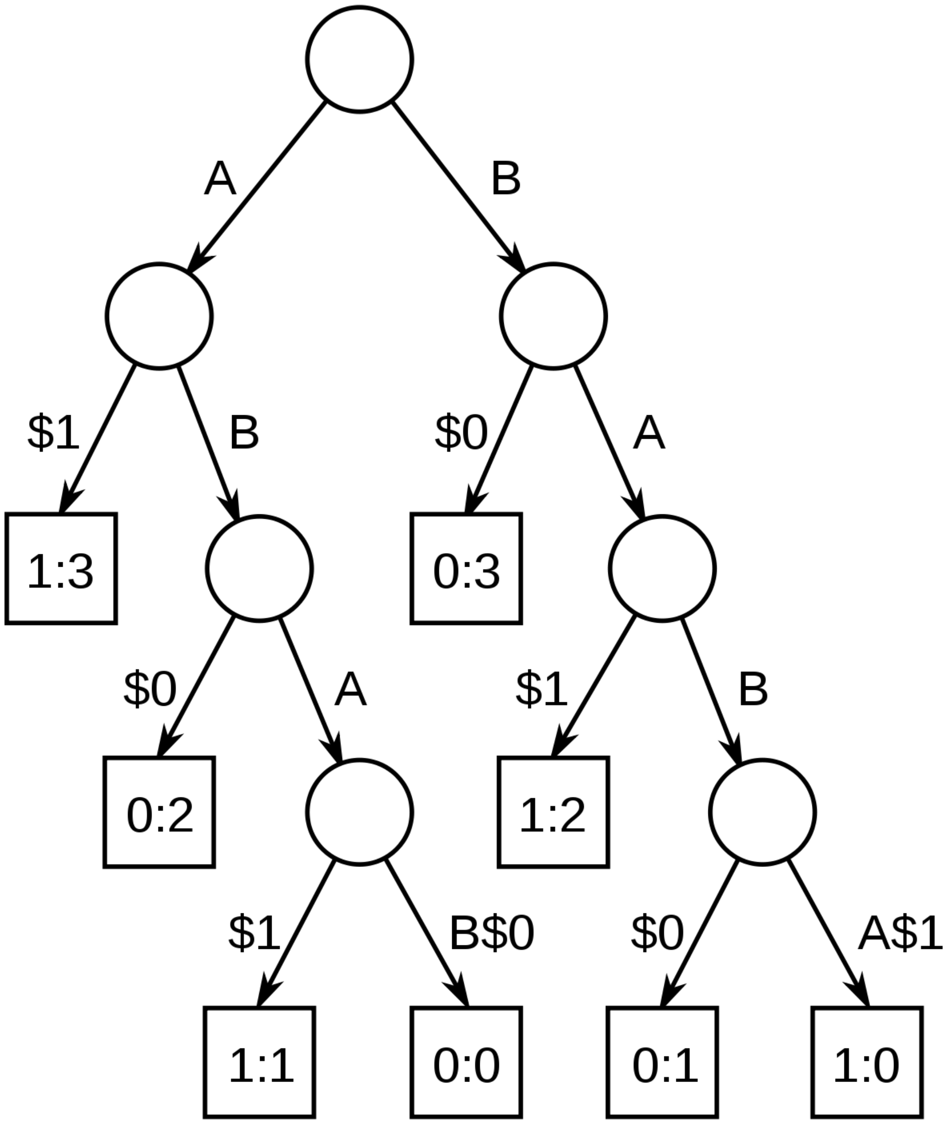
Anthony Jones
CPSC 4890/6890
December 6, 2021
Dr. Brian Dean

Exploring the Suffix Tree

For my area of improvement, I've been exploring the data structures involved with suffix trees. A **suffix tree** is a tree that compactly holds each of the suffixes of a particular word; it is used for many applications, with one being the auto-completion feature of many modern web searches. To understand what a suffix tree is, I began by researching through wikipedia and other educational resources dealing with suffix arrays, tries, and trees.

An example of a generalized suffix tree is shown below.



You read a suffix tree by starting at its root node. This is the node at the very top of the image above. The following nodes are each connected to the root by a series of edges; you will note that these edges, as well as the very bottom leaves of the tree, are all labeled. The label of an edge tells the reader how a particular suffix progresses as we move from the top to the bottom of the tree. The special terminal characters '\$0' and '\$1' in the example above denote the end of a suffix, for the words '0' or '1' respectively. In the case of a normal suffix tree, there is only a single terminal character '\$', and its termination represents a suffix for the singular word that constructs the tree. Let's go through the example above and read each of the suffixes of the two words involved. A suffix terminates only when a connecting edge is a terminal character; in that case, you can read the suffix by appending all the edge labels starting at the root and progressing to the terminating leaf. The leaf itself contains a pair of numbers; the first number represents the word that a suffix belongs to, and this is the same number as its terminating edge. The second number represents the position in that word where the suffix begins; in the case of the first leaf node shown above, we can see that the suffix 'A' terminates for the second word starting at position 3. This implies that the second word is a length 4 word ending with 'A'. Going through the list, we can read each node from left to right as follows: 'A' is a suffix of (1); 'AB' is a suffix of (0); 'ABA' is a suffix of (1); 'ABAB' is a suffix of (0); 'B' is a suffix of (0); 'BA' is a suffix of (1); 'BAB' is a suffix of (0); and 'BABA' is a suffix of of (1). Therefore, this tree is a generalized suffix tree for the words (0) : 'ABAB' and (1) : 'BABA'.

Generalized suffix trees are a special type of suffix tree, but their formation is not very complicated to understand. Provided with words (0) : a, (1) : b, (2) : c, etc.; the generalized suffix tree can be formulated by making a normal suffix tree of the concatenated word "a#0b#1c#2…", (where #n are the special non terminating characters denoting the arrangement of the words) and keeping only the prefixes prior to the '#' character of all the suffixes involved. The full process is better described in the following link:
https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/suffixtrees.pdf

As far as my improvements are concerned, I feel very confident with my understanding of the data structure, but I have yet to find a suitable implementation of a suffix tree for any of my three problems. The implementation of a suffix tree can be very challenging, as the best algorithm is a linear time algorithm that uses an advanced concept called **suffix links.** The basic principle involves building pseudo-trees of each suffix of a word, called implicit suffix trees, and circulating their respective nodes according to their suffix links. The algorithm is outlined here:
https://web.stanford.edu/~mjkay/gusfield.pdf

I am still working to implement a basic code for a suffix tree. I've only encountered one c++ implementation of the Ukkonen algorithm, but it's still beyond my comprehension. I hope to explore suffix trees more during Christmas break, because I feel very close to being able to use the data structure in actual problems for our competitions. Some implementations of suffix trees are provided in the links below.
https://cp-algorithms.com/string/suffix-tree-ukkonen.html
https://www.geeksforgeeks.org/ukkonens-suffix-tree-construction-part-1/\
https://codeforces.com/blog/entry/16780