Building the Word Tree

Last update - February 6, 2013

We need to consider compositions of the four functions T, S^{-1}, T^{-1} and S, or equivalently, words made up of the four letters T, S^{-1}, T^{-1} and S. To make our discussion a little easier to understand, we will give each of these functions a corresponding number and direction:

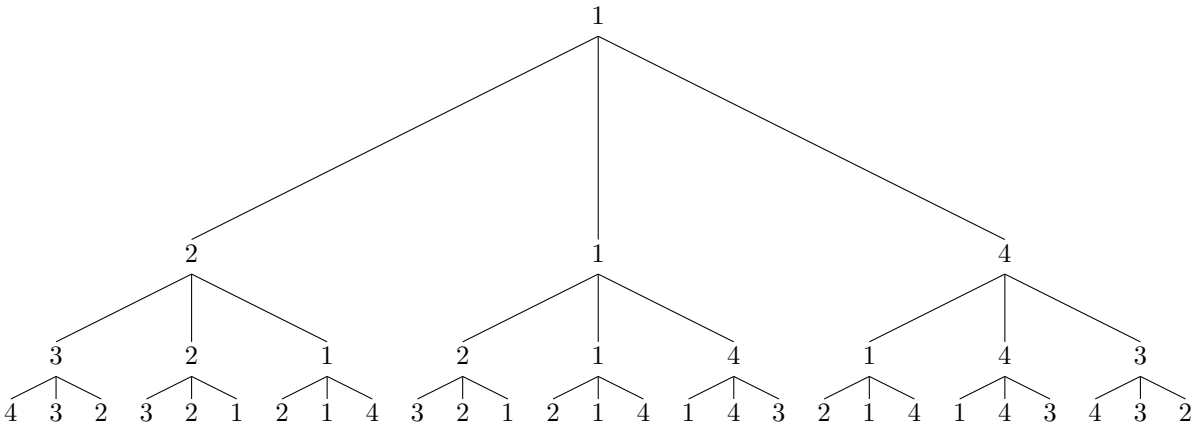
function	number	direction
T	1	east
S^{-1}	2	south
T^{-1}	3	west
S	4	north

From the above table, we note that T and S are the positive x and y axes on a grid, (hence S^{-1} is in the $-y$ direction, and T^{-1} in the $-x$ direction). Thus the path $1 \to 2 \to 3 \to 4 \to 1$ is a full clockwise rotation about the origin (which we denote I, the identity map), yielding the word $T\,S^{-1}\,T^{-1}\,S$.

We need to find a method for going through many permutations of the letters in a very methodical fashion. The rule of thumb is to always turn right, then straight, then left. For instance, if I am at a point on the grid, after having applied the function T, (which we call 1), then in order, my next move will be to the right (S^{-1}, or 2) etc... However if it has been determined that we have gone far enough down the current path, instead of turning right (applying S^{-1}), we would instead go straight, which in this case means applying T again (or 1), etc... As a rule, here are how the steps work:

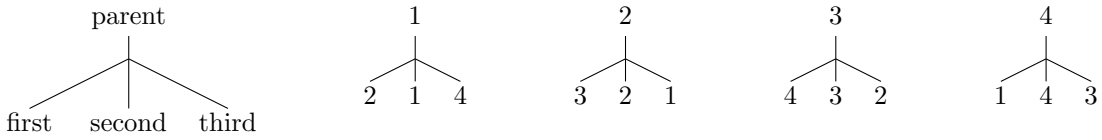
current location	order of steps
1	$2, 1, 4$
2	$3, 2, 1$
3	$4, 3, 2$
4	$1, 4, 3$

As a result, we can think of the set of words as a ternary structure. To make this more amenable to coding, we need to construct a tree. Since each parent node will have three child nodes, this will be a ternary tree. However, based on the parent, the children will be different. If you examine the following tree, which can be read from left to right, you will find all possible paths to arrive at the 40 words of length 4 or less, in order for the right half of the $TS-$plane. You should work out the tree for the left half as well (those starting with T^{-1} as opposed to T which are given in the tree below). The order requires that you start at the top of the tree, and work your way down on the left.



For instance, the first word (after the identity), will simply be 1. The next word, will be 12, the next 123, then 1234, at which point, we have reached as far down the tree as we want to go. Thus, we back up one and go to the next child of 3, which results in the word 1233. Hopefully you can see how to travel to all points on this tree, IN ORDER. *We will need to traverse this tree in order in the future, so this is an important concept to fully understand.*

As a result, we always read down and to the left, when possible. Note that depending on the value of the current node, the children will look different, but the pattern is always as follows:



Now, our first assignment, is to create a program which will travel to all words of 4 numbers or less, in order. To do this, we are suggesting a tree transversal approach, using classes/structures (important: C has no classes, only structs). If used correctly, with pointers, we should be able to construct a ternary tree transversal program, given all of the information above which details how the tree is built. Furthermore, the tree transversal program will have to be made in the pre-order traversal sequence (wiki if you are unfamiliar). One final comment: The tree that we will eventually be building/transversing, will have branches of varying lengths, which makes this problem very stack-unfriendly. Furthermore, if you look at the 40-word tree drawn on the previous page, there are times where word length can drop by more than one letter, this causes a yet another problem for the stack-approach. When we get to the tree with varying branch lengths, it was pointed out that we may have word lengths change in length by more than a thousand digits! If you find a way to do this in a stacking method, please enlighten us, give as an approach. Otherwise, perhaps the tree transversal with structs/classes and pointers is the way to go.

Below is the left half of the tree: