Stefan Emmons

COSC-3020-01

Lab 03

9-26-2019

Part 2: Runtime analysis

Looking at our code, we have three recursive calls, that operate on three separate arrays. This can be can modeled with the following relation:

Recurrence relation:

$$T(n) = \begin{Bmatrix} 1 & if\ n = 0 \\ 1 & if\ n = 1 \\ 3T(\frac{n}{3}) & if\ n > 1 \end{Bmatrix}$$

Solve by substitution:

$$T(n) = 3t(\frac{n}{3})$$

$$= 3(3T\left(\frac{n}{9}\right))$$

$$= 9T(\frac{n}{9})$$

$$\ldots and\ so\ on, we\ can\ see\ a\ pattern\ here, so$$

$$3^i T(\frac{n}{3^i})$$

The ‘i” here is the recurrence relation representation.

When considering the searching efficiency of a binary tree, we know that this is represented by the value $log_2 n$. When looking at the recursive calls of this divide and conquer sum, we can see that we have three recursive calls, and so we can think of this like a trinary tree, which can be represented by the value $log_3 n$.

With this being in mind:

For i = $log_3 n$

$$= nT(1) + nT(1) + \ log_3 n = n + n + log_3 n\ \in\ \theta(log_3 n)$$

Drop the constants, because this is an asymptotic behavior analysis, and we are left with a time complexity of $O(log_3 n)$.