# University of Missouri – Kansas City

# CS 303 – Data Structures

# Project 2 – Morse Code Translation

# July 28th, 2017


**Group Members:**

Aaron Lloyd James

Alexander Larios

Cameron James Conant


**This project can be found on Github**

https://github.com/LogicLotus/Project2

# Efficiency of Algorithms

**Big-O of Functions**

| Function | Best | Worst |
|---|---|---|
| addNode(); | O (2) | O(h) |
| codeSearch(); | O(2) | O(n) |
| morseTraversal(); | O(2) | O(h) |

**Analysis**

The best of all of our functions is O(2). This is because the root node is blank, so we have to move to at least the next height to find what we are looking for. Height is the worst big O for all except codeSearch. codeSearch has to recursively search the tree and could end up navigating the hole tree just to find the matching letter. There isn't really a faster way to do this, since searching by letter has no mapping to the tree.

**Can some functions perform better?**

I don't believe these functions can really improve beyond their current state. Binary trees already make it so efficient that its hard to think what could improve, if anything. Most of the work in designing efficient code is mostly making sure that you don't over complicate a problem. The only code that I think might be improved is codeSearch, but that would require adding more to project than was likely necessary. We would have needed to invent a way to navigate the tree by letter, but this would require more work than the reward would be worth.

**References**

BTNode file on blackboard, modified for our own uses.

The other binary tree files on blackboard which we referenced but did not use.