**Assignment 5**

2017-12-02

By: Bryan Greener

Language: C#

For this assignment, I took a slightly different approach to solving the problem in an attempt to push myself to write better scalable code. Instead of writing separate implementations for a 2-3 tree and a 2-3-4 tree, I wrote a single class for any degree of B-Tree. Along with that, I used generics for the first time to make the class work with any dataset. My implementation should be able to take in any text file in the proper format and build a B-Tree from it.

Because of this, the difference in runtimes between a tree with a degree of 3 and a tree with a degree of 4 is just a couple of fractions of a millisecond different. Using the ASCII table as input, the 2-3 tree had a total height of 8 and the 2-3-4 tree had a total height of 7. Both exhibited time complexities of O(log(n)) while searching however they could improve with a bit of restructuring since there is probably a bit of excess code here and there. Along with that, I had to work around a large number of edge cases which seemed a bit odd so I think I may have done something wrong. Regardless of which implementation I use, the number of leaves in the tree will be the same since they are storing all of our actual values. If we instead let the internal nodes store the values and handle the keys at the same time, the height of the tree could be reduced by 1 and the total number of leaves would be halved due to this. The time complexity would stay the same with this change but you would notice a large boost in speed either way.

In the end, I was surprised that this ended up working after all the hair pulling it put me through. I’d like to go back through this and rewrite it to see if I could make it run a bit faster.