Kristen Orue

Exploration: Binary Tree

1. Compare and contrast time complexity for arrays, linked lists, and binary search trees.  Include both sorted and unsorted arrays.

Binary search tree time complexity is for sorted lists and it’s time complexity on average is O(logn) but can sometimes be O(1). However, if you were to search through an array it’s time complexity is O(n) and if you were to sort the array with MergeSort it would be O(nlogn). For linked lists and arrays searching is usually O(n) if there is no known reference for where the item is located. If the reference is know that it’s O(1) for arrays and O(n) for linked lists. Deletion for arrays is O(n) and linked lists it’s O(1)

1. In  a binary search tree, how is order maintained when inserting and deleting data? Be specific.

Each note contains one key, the key on the left of the subtree is less than the key in it’s parent key while the keys in the right subtree are greater than the key in it’s parent node. There are no duplicates allowed which means the when we go to insert we start from the root node and if the node about to be inserted I less than the root, we go to the left child, if it’s great we go to the right. We continue until we find a null node where we can’t continue and insert it depending on whether it’s greater or less than our it’s most recent parent node.

1. Explain the relationship between the density of a binary search tree and its worst case time complexity

The Density of a Binary tree let’s you know how balanced a binary tree is or whether it’s skewed or perfect. The density of a skewed tree is minimum while the density of the perfect tree is maximum. It measures the number of nodes relative to the height of the tree. Since density of a cinary tree refers to its size, of course the larger the tree, the more steps it will have to take which determines time complexity.