Readme

Data Structures:
I implemented a Binary Tree combined with a Linked List to create a program that is more cost efficient than just using arrays.

3 Run Time Analysis

BST Insertion - $O(n^2)$
$O(n^2)$ occurs when the tree is skewed and nodes are all inserted on one side. If the tree is not skewed then O(logn) will be the running time.

BST Search - O(n)
Traversing though a tree is O(n) because we are going through a tree with n items

BST Delete - O(n + l)
This is always O(n + l) because it is the size of the tree, every node in the linked list is deleted.

Linked List Search - O(n)
Finding an item in the linked list of size n takes worst case O(n)

Linked List Insertion - O(1)
Adding an extra node to a linked list is done in linear time

Challenges:

Some challenges in coding this program included picking a data structure and learning how pointers function. I previously only knew java so understanding pointers and certain C functions was a little difficult. Figuring out that Binary Trees was a good data structure helped my program in terms of running time. Writing this program did help me learn more about C and using new functions such as malloc and fopen. This assignment was not easy, but it was necessary to learn proper C language mechanics.