Juhwan Lee

CS-163

August 9th 2020

Program #4 Efficiency Write Up

The goal of the fourth program is to move from a hash table and implement a binary search tree. The advantage of a binary search tree is the ability to retrieve our data using a logarithmic performance if the tree is relatively balanced and be able to search for a range of information and obtain our data in sorted order. There are two main points that I found when I was programming program number four.

First, unlike a hashtable, a binary search tree is easier than a hashtable when listing data in order because, depending on the data value, if it is smaller than the root data, the data is stored to the left and if it is larger than the root data, the data is stored to the right. When the hashtable was used, the key value of the data was determined and the key value was applied to the hash function to obtain the index value of the array and store the data in the array with the index value obtained. Therefore, it was very difficult to list the data in order when using hash table. However, if we use the binary search tree, we first compare the data that we want to add with the root data and store the data to the left when it is smaller than the root data and store the data to the right if the data is greater than the root data so if we try to list the data in order, we can easily list the data in order using the in-order traversal. From this point of view, the binary search tree is a better data structure than the hash table.

However, the binary search tree was not only advantageous compared to the hash table. What I felt during the program number four was that when data were added in order, problems arose. In the case of hash table, the performance of the data structure was always the same, regardless of the order of the data being added. But the case of binary search tree is different. When data is added randomly regardless of the order, the binary search tree is not perfect but still shows a pretty good balance structure. However, if the datas are added to the binary search tree in the perfect order, the structure of the binary search tree will have the same structure as the linear linked list. This results in O(N) performance of the data structure. O(N) performance is what we want to avoid the most. Therefore, the ideal form of the binary search tree is to achieve a perfect balance, and in that case, the data structure performance will be logarithmic. So how do we balance the binary search tree? Because of this problem, I felt that 2-3 tree, 2-3-4 tree, Red-Black tree, and AVL tree that I learned in this class existed for this reason.

Overall, I think I learned a lot about binary search tree through this programming assignment. To add one more thing, since the binary search tree uses a recursive function, coding was a bit simpler than the previous programming assignment when programming. Through this programming assignment, I could definitely feel the power of the recursion.