**Analysis and Evaluation Results**

There was a lot to consider when making the evaluation of which type of structure would fit best for this assignment. In reviewing the material and pseudocode for each of the different structures, each of them seemed to run around the same speed with the longest being the hash table having to iterate over several cases in worst case scenario and then outputting based on a larger number of searched elements. The fastest running was the Binary Search tree with a larger number of available searches with a significantly smaller return each time through the way that the BST parses through data. The Linked Lists would have been a good way to go about this project as we find linear probing going on in this structure which generally is efficient with small data sets, but inefficient as data sets get larger. Additionally, with the linked lists, we would find a much simpler case of adding and deleting elements within the array. The largest drawback which is also the biggest thing going for it, is that linear probing. In terms of speed, this becomes largely inefficient when looking for specific data elements.

The next structure to talk about is the Hash table. Like the linked lists array, this structure would also have worked fine in the long run, boasting the ability to search quadratically rather than linearly. This data structure also has the ability to have several linked lists tied to a hash which bodes well when selecting courses that have prerequisites. The major detractor for this structure comes in the form of how information is inserted or added, often needing to go through several different hashes before being able to insert new information either into a hash’s linked list or a new hash all together.

All of that being said, for the assignment and the continuity of the project we will be going with the Binary Search Tree structure as it seems as though this will be the fastest running and most efficient method to outputting the information we are looking for. The BST parses through data logarithmically and has a much faster overall run time compared to the methods of parsing for the other structures. This method is also incredibly efficient for the sake of memory as each node has a parent child relationship with consistent left to right translations of value.