**Runtime Evaluations**

**Vector search runtime**

The Big O value for finding a course in a vector would be O(n), so it is a linear time algorithm. The search function for the vector simply goes through each course and only prints the one that shares a matching id with the function's parameter.

**Hash search runtime**

The Big O value for a hash table search is O(1). To find the course in a hash table, all we have to do is hash the id and return what is found.

**BST search runtime**

The Big O value for a BST search is O(log(n)), making it faster than the vector and hash table search times. Using the BST cuts the search time in half by starting from the root and choosing to go left or right routinely all the way down until the course is found. Doing this each time throughout the tree cuts down search time significantly.

Each data structure's average runtime shows its own respective speed. The vector search needs to go through each object in the vector list to finally output the course, which makes it not very effective. The hash table has to hash the input and pull up the course information using that hash id. Finally, the BST is balanced and can be searched for the id in a relatively quick amount of time. Due to the runtime of the BST, I would definitely recommend it as a fit for the advisor's requirements.