This program compares the runtime of two different data structures when searching for specific values. The two different types of list are the Ordered List and the Move to Front (MFT) list. First, the program reads in a specific file numbers and queries into arrays. Using this information, an orderedList and a MFT list are created from the array. The orderedList automatically sorts itself using mergeSort once all values are added from the input array. Next, the orderedList searches for the specified queries using binary search. The MFT uses the values from the input array to create a linked list. The search function removes the most recent query from its original location and moved it to the front of the linked list. Therefore, the most frequent and recent queries will readily accessible at the beginning of the list.

The mergeSort member function which has the runtime of O(n logn). When searching for queries, I implemented a binary search algorithm that has the runtime of O(logn). The mergeSort function has the space complexity of O(n) because we are constantly creating and deleting arrays that contain all of the elements. The binary search has the space complexity of O(1) because the space taken by the algorithm is independent of the size of the array.

For the MFT list, once a value is found in the search function, the value should be moved to the front of the list, therefore placing the most recent and popular queries at the front of the array. In the worst case, this search function will have a runtime of O(n) if the desired value is located at the end of the list. The MFT list search function will have constant space complexity because we are not allocating any new memory in the search.

Overall the orderedList searches almost always performs faster than the MFT list’s. For larger values, the runtime of O(logn) will always be faster than a time complexity O(n). The only time where MFT search performed faster was for smaller inputs. In terms of the space complexity, MFT has a slight edge since it does not allocate new memories to arrays, like the orderedList does for mergeSort.