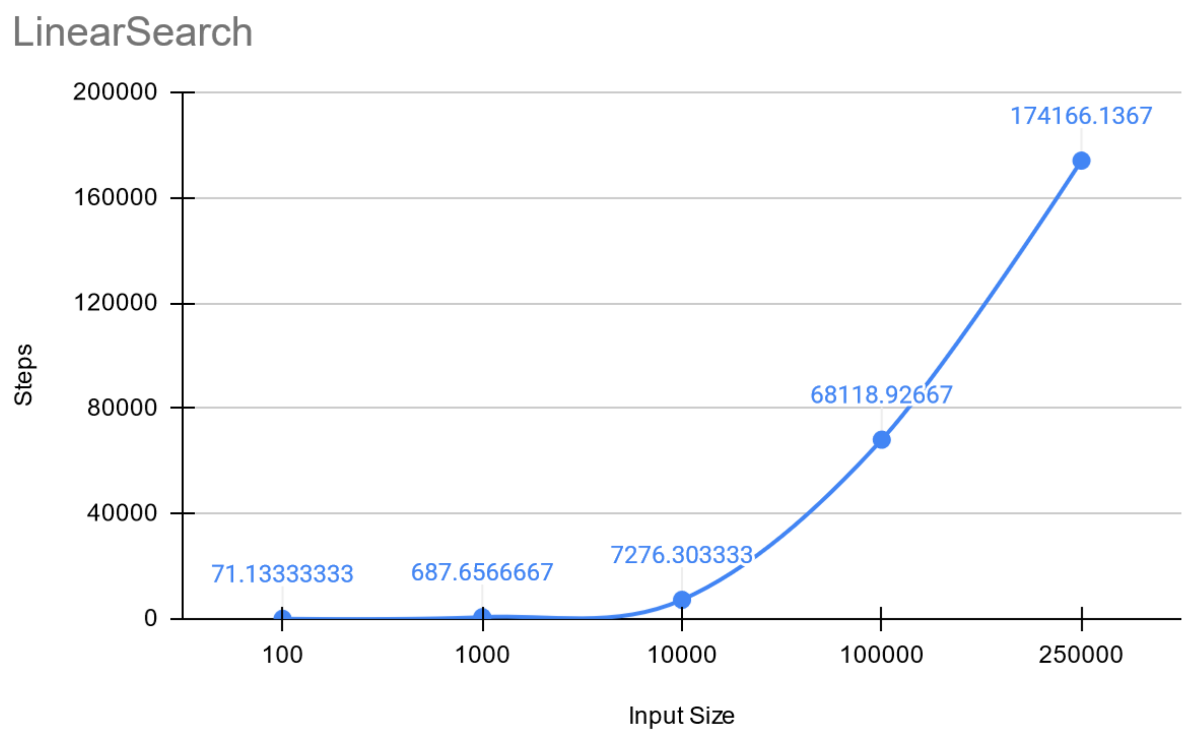
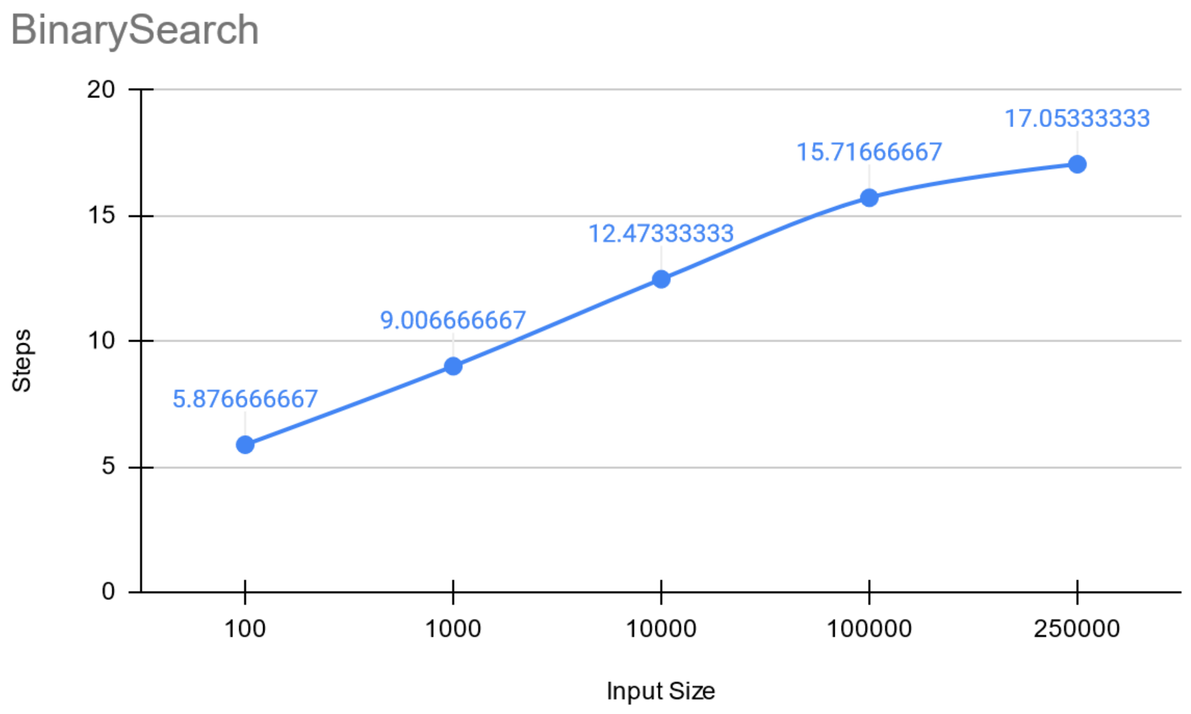
Conner McKevitt
Question 2

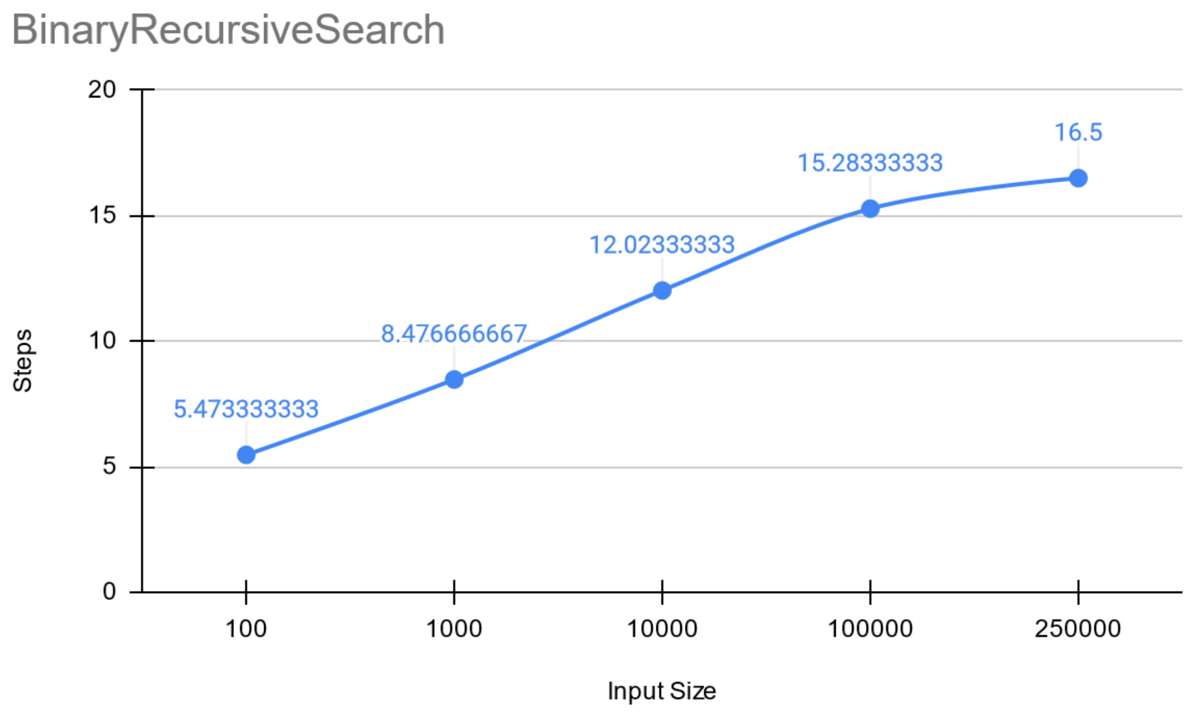
For each experiment I used 100 randomly generated keys within the range of 1->input size that were searched for in increasing array sizes. This was done 3 times and the total steps were then averaged for each input size then divided by 100. This gives the average time to search for 1 key in the input size.



Linear search searches through the entire array in order, checking each value for the desired key. This gives it a worst case runtime of O(n) and a best case of O(1).



Binary Repetitly divides the array in half. Assuming that the input is sorted, it finds the midpoint of the array and if the key is larger than the midpoint it will then search the top half of the array and vice versa. It does this by setting a new lowest array index or high index for the input array within it loop. It will keep dividing the array in 2 until the midpoint is the key or there are no values left in which case the search fails. This will give an O(log n) time complexity.



Binary recursive search acts on the same idea of binary search however instead of looping and setting a new low or high index. It calls itself using the new low and high indexes and inputs to its call. This also functions in O(log n).