Part 3:

\_\_\_\_\_\_\_HashTable - BIG arraysize of 2^10=1024

HashTable - BIG arraysize of 2^10=1024 - insertion runtime is 7.0

HashTable search runtime for aaaa is 0.0

HashTable search runtime for alice is 0.0

HashTable search runtime for zoe is 0.0

\_\_\_\_\_\_\_HashTable - MEDIUM arraysize of 2^7=128

HashTable - MEDIUM arraysize of 2^7=128 - insertion runtime is 3.0

HashTable search runtime for aaaa is 0.0

HashTable search runtime for alice is 0.0

HashTable search runtime for zoe is 0.0

\_\_\_\_\_\_\_HashTable - SMALL arraysize of 2^4=16

HashTable - SMALL arraysize of 2^4=16 - insertion runtime is 2.0

HashTable search runtime for aaaa is 0.0

HashTable search runtime for alice is 0.0

HashTable search runtime for zoe is 0.0

BST insertion runtime is 6.0

BST search runtime for aaaa is 0.0

BST search runtime for alice is 0.0

BST search runtime for zoe is 0.0

For both functions the search time was relatively instant. I was surprised that bst held up to this even though it has a O(log n) complexity. This was to be expected for the HashTable though. The Hash Tables did perform better when the array size was small too. Even though there were more collisions. This was most likely due to the fact that there was so much memory saved by making a smaller array and using most of the index’s vs. leaving so many empty. It was interesting to see that both performed around the same speed if the list was sorted or not.