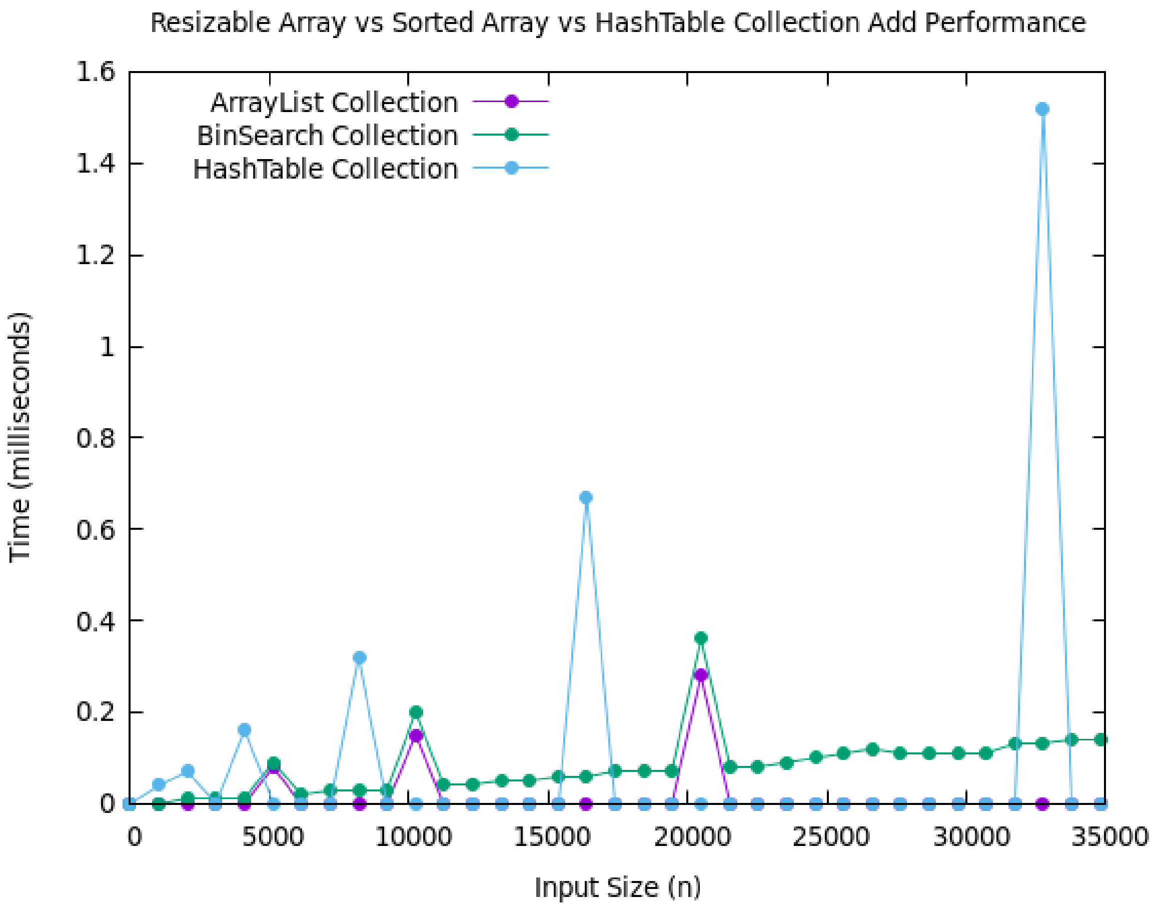
Scott Tornquist

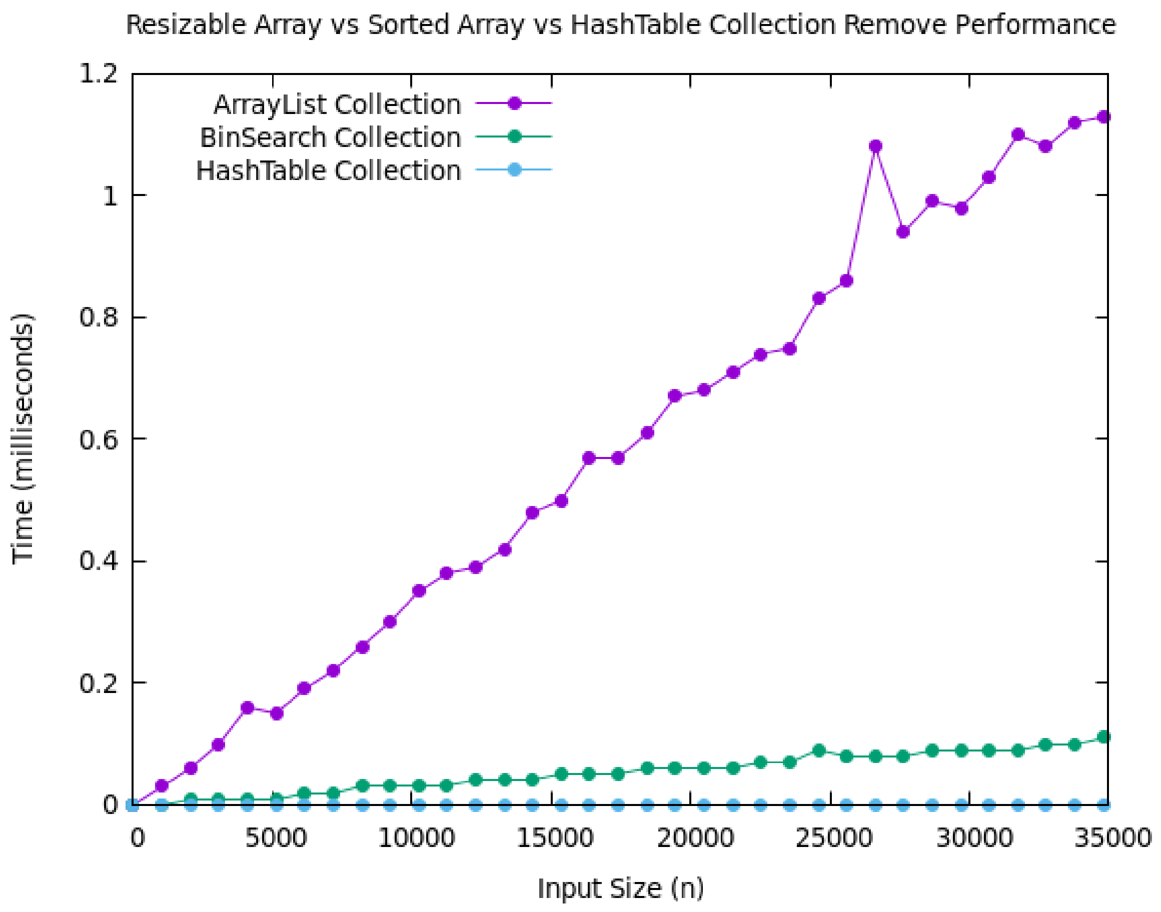
Abstract Data Structures

HW6 write up



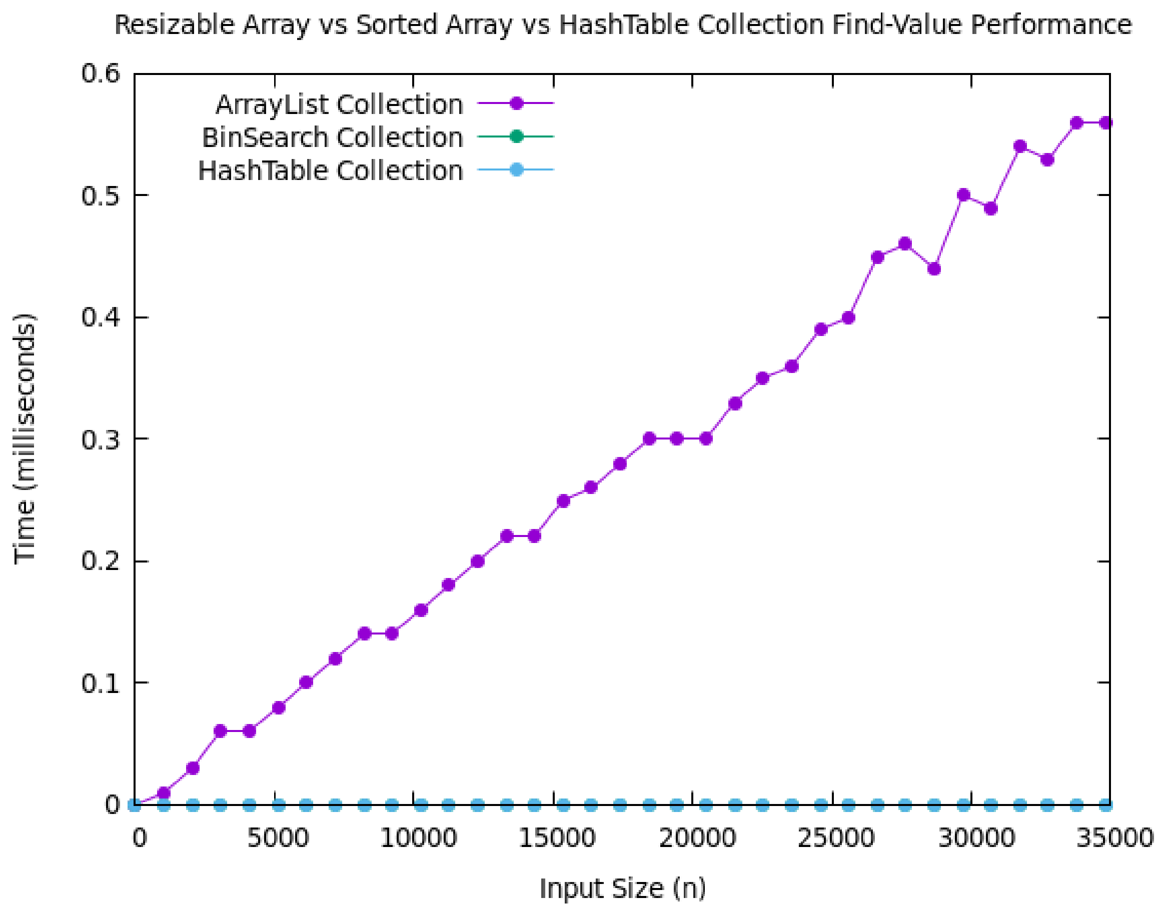
The Hash Table collection Add() Function came out this way because the it is O(1) in the average case. The spikes are caused by resize and hash when the table gets too full. However, we account for these spikes by using amortized analysis which essentially proves that the spikes can be averaged out over the course of time. This gives a slightly higher cost 3n instead of 3n-1. However, this is still constant time.

Tn the worst case Add() is still O(1) because it adds to the front of the chain, so it does not matter how long the chain. There is not much of a worst case.



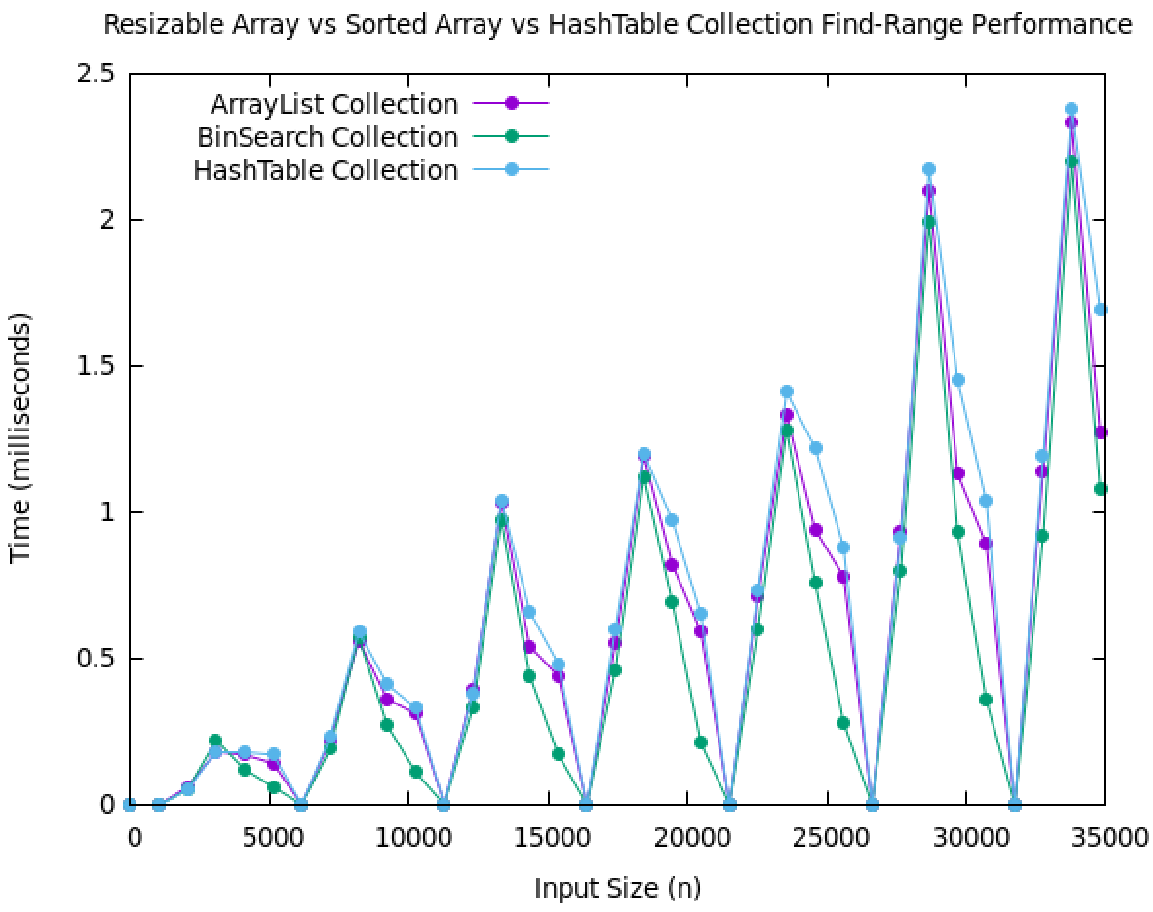
The Hash Table collection Remove() Function came out this way because the it is O(1) in the average case. To remove we hash to the value, find it in the chain, and remove.

However, in the worst case that all the remove() values hashed to the same index, the hash table becomes a linked list. If the last value in the chain is to be removed, Removed() becomes O(n) in the worst case because it must iterate through the entire list every time.

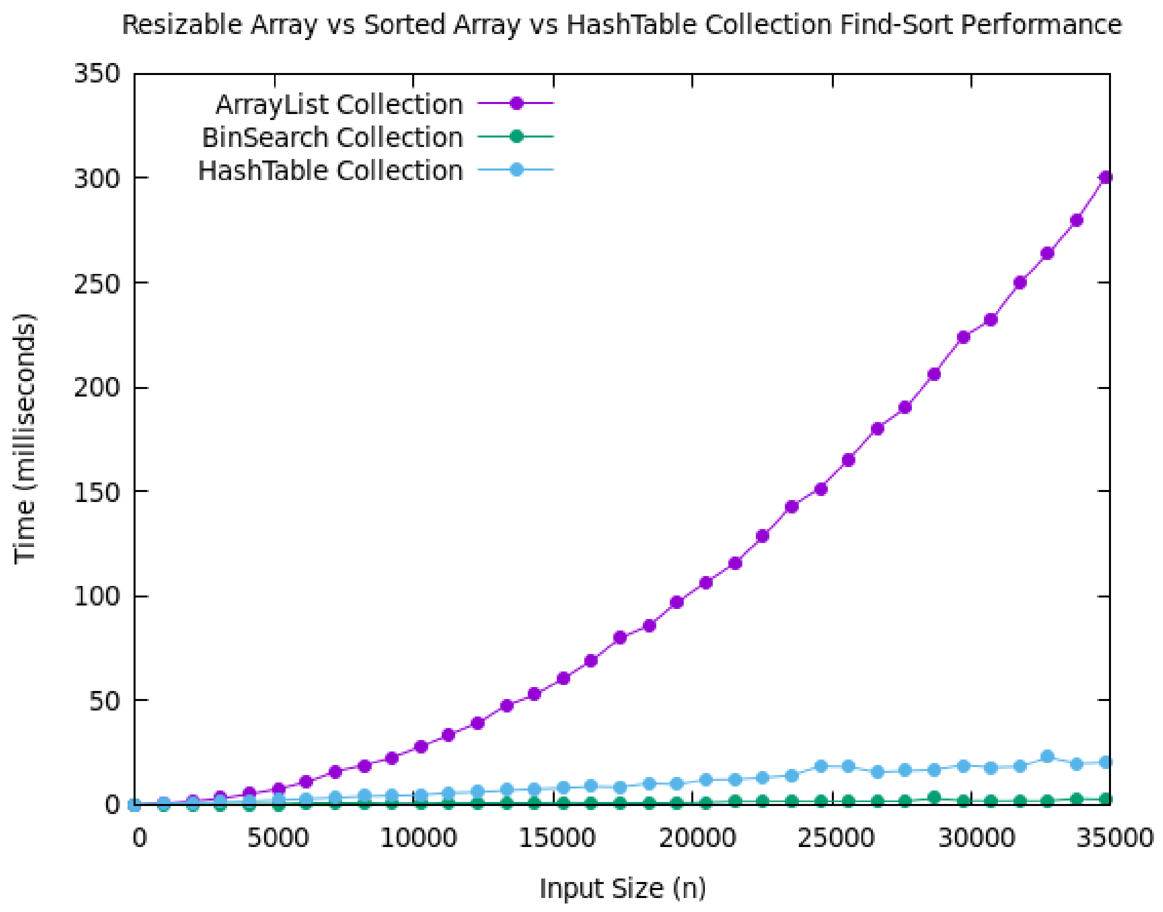


The Hash Table collection Find-Value() Function came out this way because the it is O(1) in the average case. To Find we hash to the value, find it in the chain, and remove.

However, in the worst case that all the Find-Values() hashed to the same index, the hash table becomes a linked list. If the last value in the chain is to be found, Find-Value() becomes O(n) in the worst case because it must iterate through the entire list every time.

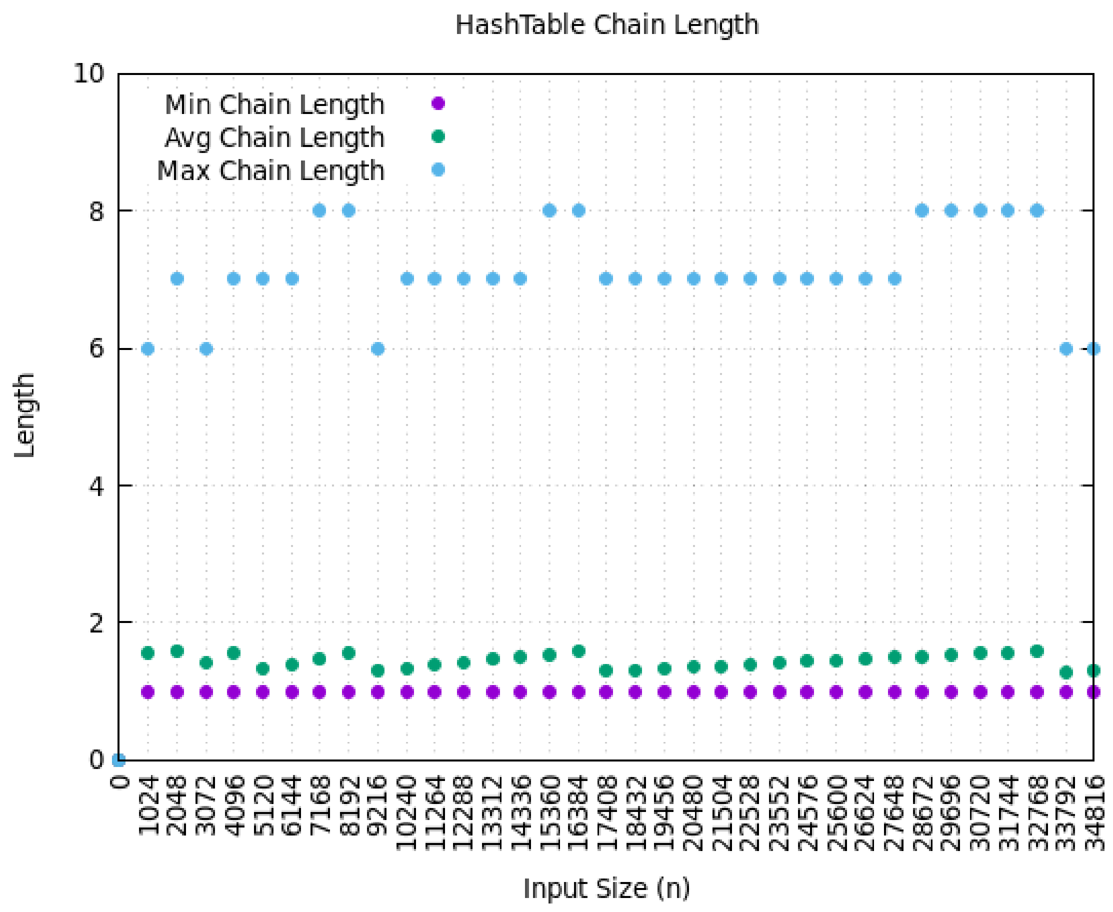


Waiting for discussion in hw7.



The Hash Table collection Sort() Function came out this way because the it is O(n) in the average case. To Sort we hash to the value, find it in the chain, and add to the array. To do this for every value is O(n). The array is then sorted using quick sort which is log(n) in the average case. O(n)+O(log(n)) = O(n)

However, in the worst case that all the Sort() values hashed to the same index, the hash table becomes a linked list. To add all the values to the array is O(n). The sorting is quick sort which is O(n) in the worst case. O(n) + O(n) = O(n). Sort() becomes O(n) in the worst case because it must iterate through the entire list to add the values and quicksort is O(n) in the worst case.



The Min chain length is always 1 because unless it is a very bad hash function the values will be distributed evenly and most definitely there will be at least one chain of one.

The Max chain length fluctuates because longest chain just depends on what values happen to be put into the hash table that have collisions. There will most likely be some collisions. The amount is hard to predict.

The Average chain length is mostly constant but follows the trend of the max chain length because it is determined by the number of nodes/number of chains. So, if the max chain length is higher there is probably more nodes per chain.

Worst Case Algorithm Analysis

| Operation | Collection Implementation | | | |
|---|---|---|---|---|
| | Array List | Linked List | Sorted Array | Hash Table |
| Add | O(1) | O(1) | O(n) | O(1) |

| | | | | |
|---|---|---|---|---|
| Remove | O(n) | O(n^2) | O(n) | O(n) |
| Find-Value | O(n) | O(n^2) | O(n) | O(n) |
| Find-Range | O(n) | O(n^2) | O(n) | O(n) |
| Sort | O(n^2) | O(n^2) | O(n) | O(n) |

This assignment was a very good practice of how to implement a collection and a refresher on the constructor, destructor, copy constructor, and assignment operator overload. The assignment operator overloading was the hardest part. The resize and rehash was also tough. This data structure was very cool and powerful. I think this is used in industry a lot, so I am glad to have an understanding and be able to analyze it.