Yusuf Bahadur
CPE-103
May 10, 2017

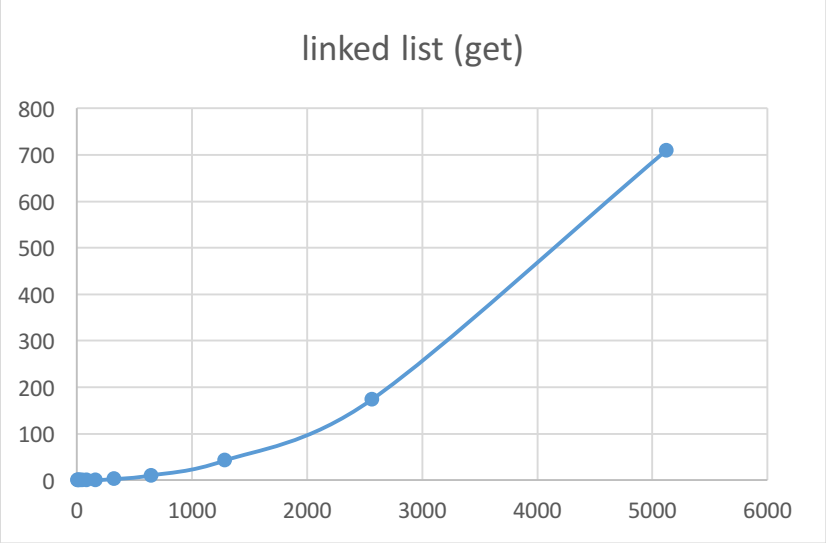
Empirical Study Report – Music Catalog Project

Introduction:

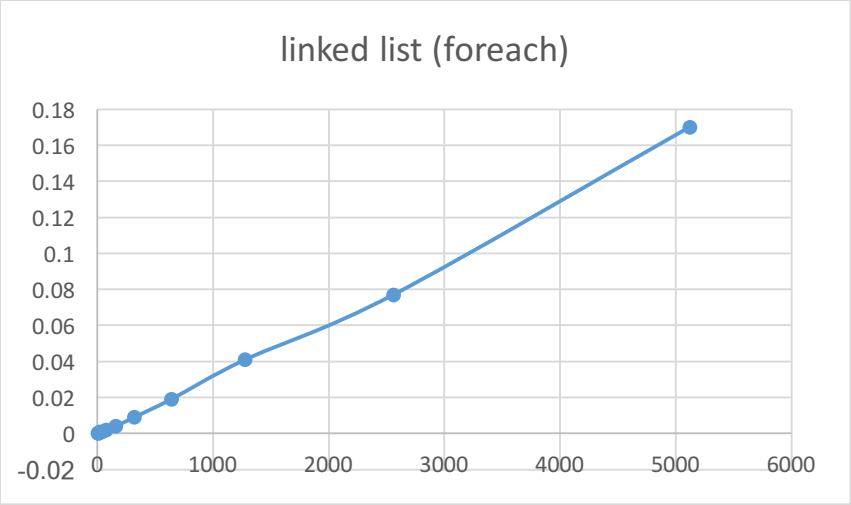
When manipulating and creating new lists, it is crucial to take into account the effects of time and memory. In order to measure the performance of various list operations we used a timeit library which outputted the size of the list, the number of iterations, and the time in seconds. The size of the given list doubled each time as it ran through the varying list sizes: [5, 10, 20, 40, 80, 160, 320, 640, 1280, 2560, 5120]. The number of iterations stayed constant throughout the entire experiment at 100 iterations. The time in seconds outputted is then recorded with the varying list size to output a graph which depicts the Big O Notation.

Experiment Data:

4.1 Linked List Foreach vs. Get



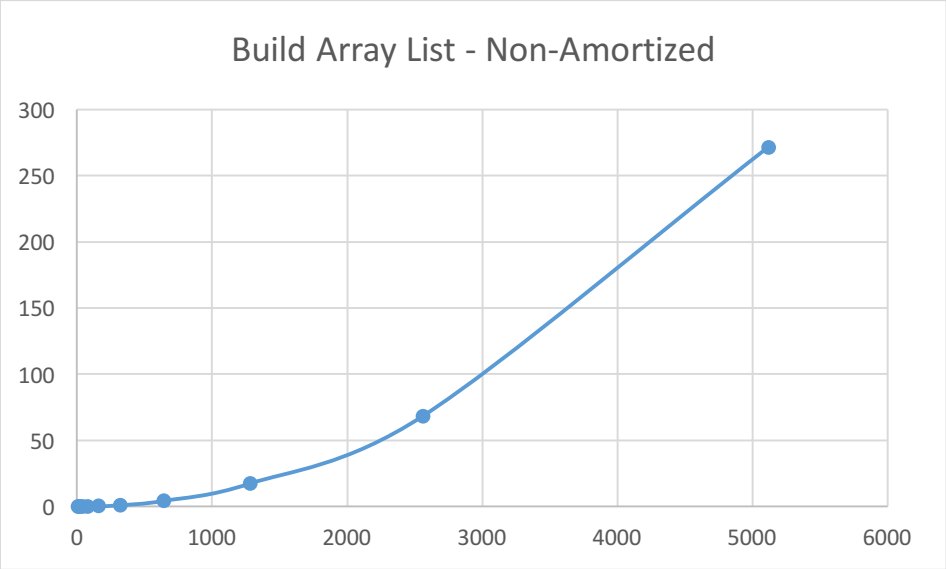
Running a get function on a linked list results in a Big O Notation of O(N^2) as for a linked list the get function has to go through each element in the list until it reaches the proper index. As such, the longer the list the longer the time to access each element.



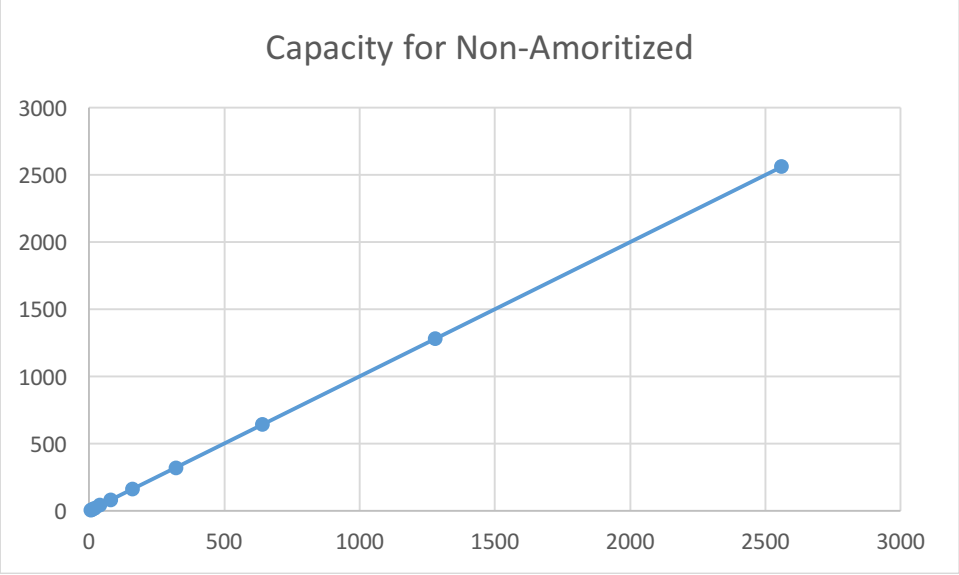
Running a foreach function on a linked list results in a Big O Notation of O(N) as the function simply goes through each value in the list and performs a function on the given value. The foreach function simply continues down the list, resulting in a linear time growth.

4.2 Array List

Array List Non – Amortized Data

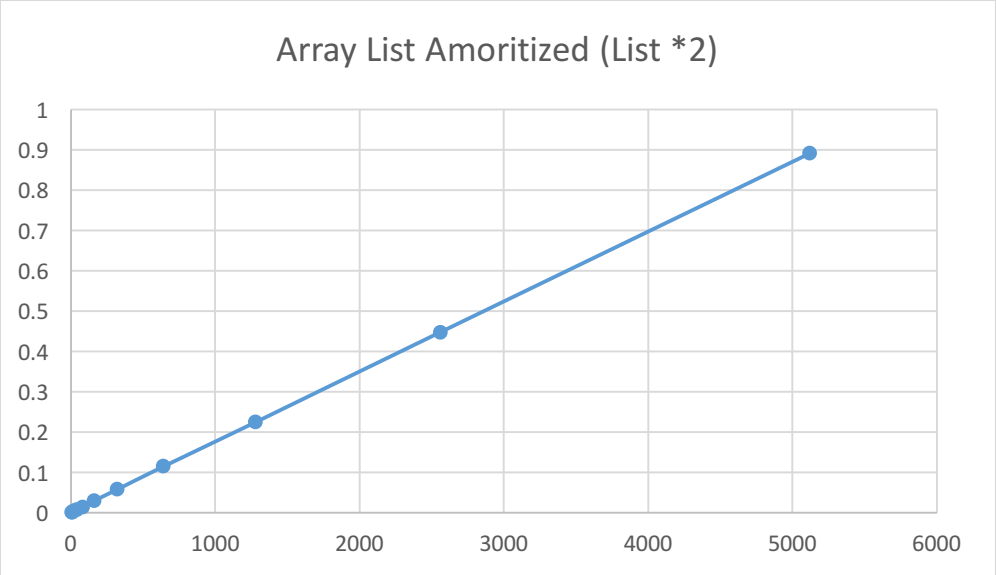


Running the add function to build an array list resulted in a Big O Notation of O(N^2) for a non-amortized list.

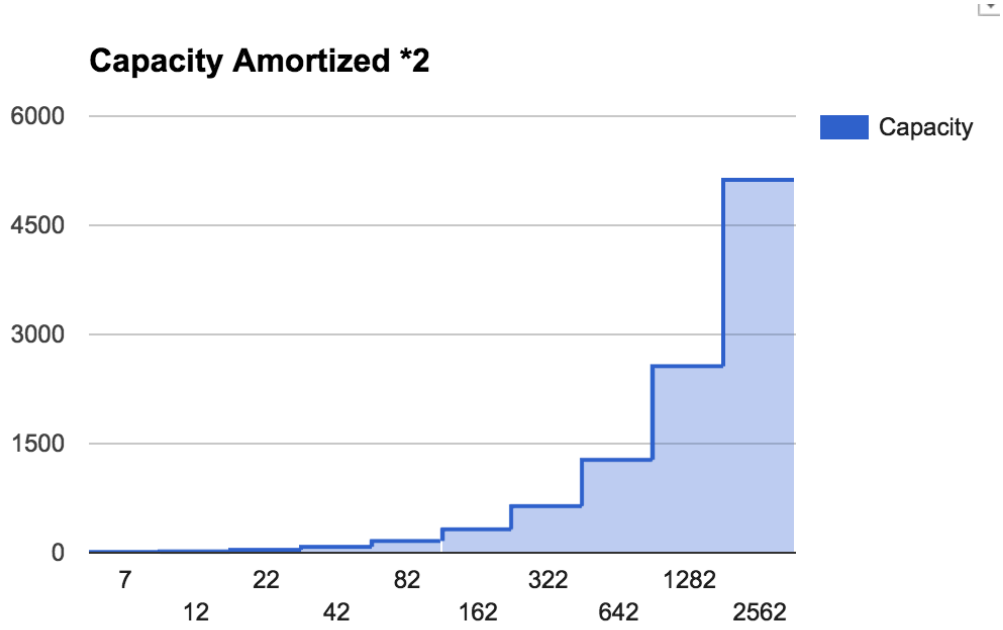


The capacity for a non-amortized increases with each new individual value and as such grows with the length of the array list.

Array List Amortized Data Multiplied by 2

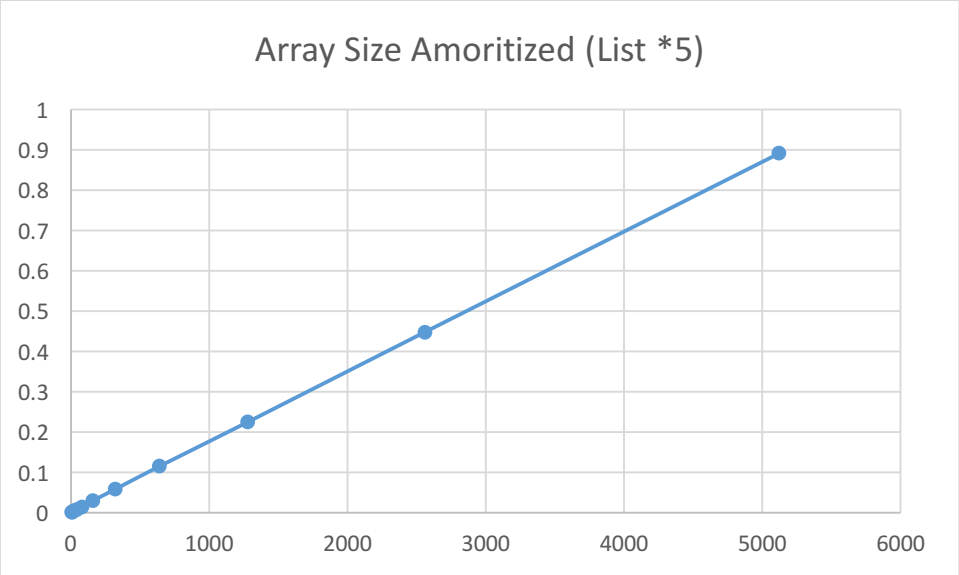


Running the add function on an array list results in a Big O Notation of O(N) for an Amortized List which doubles its capacity. It is O(N) as the add function has to continue adding elements to the end of the array list and for a large list it will inherently take longer to add such elements.

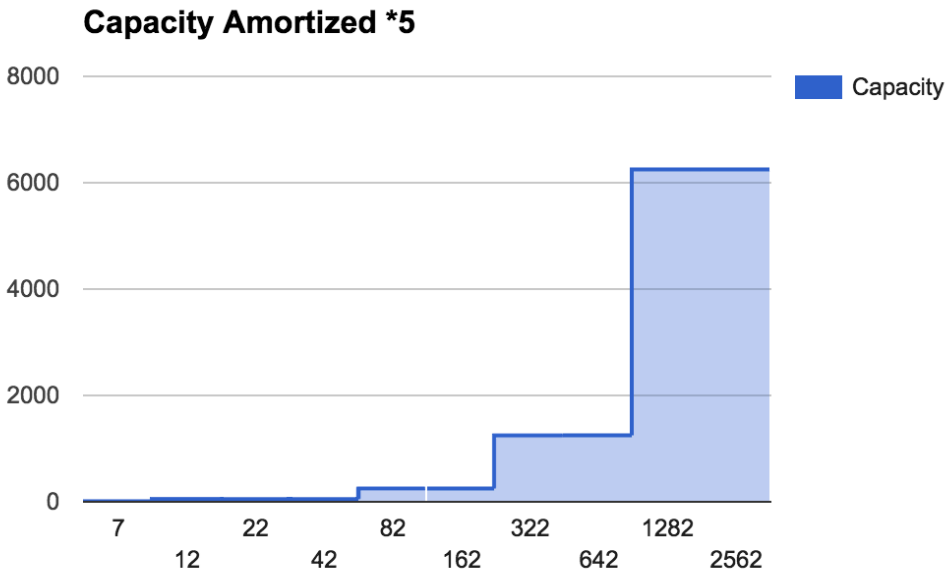


The capacity for an amortized list will double every time the size of the list reaches the capacity.

Array List Amortized Data Multiplied by 5



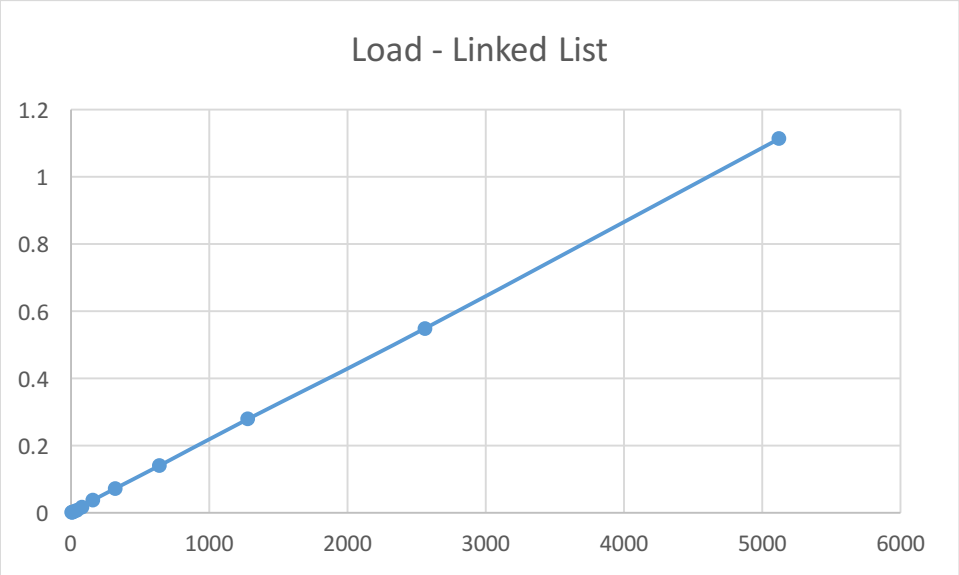
Running the add function on an array list results in a Big O Notation of O(N) for an Amortized List which multiplies the capacity by 5. It is O(N) as the add function has to continue adding elements to the end of the array list and increasing the capacity once it reaches the threshold.



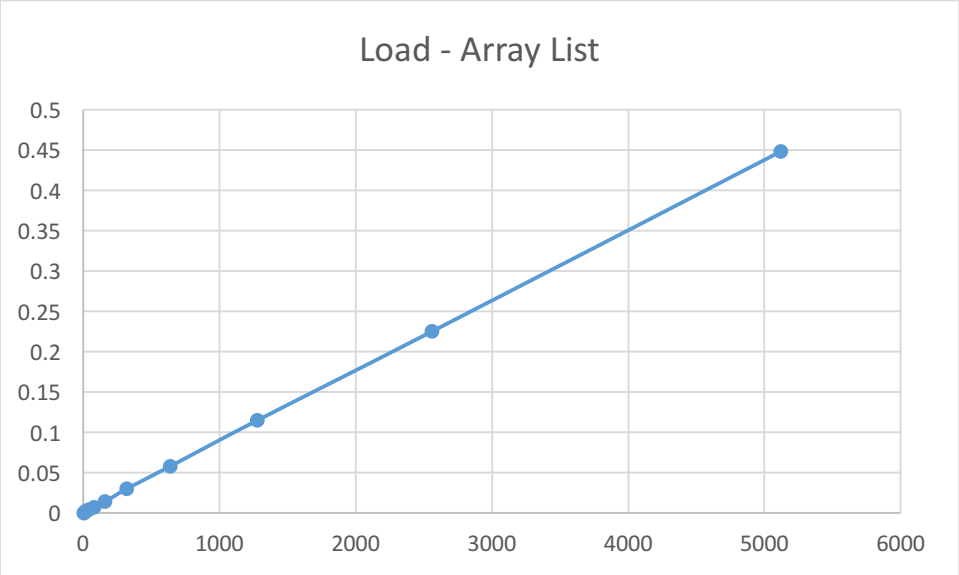
The capacity continues to be increase by 5 once the capacity is reached.

4.2.1 Array List vs Linked List

"Load" Linked List vs. Array List Data

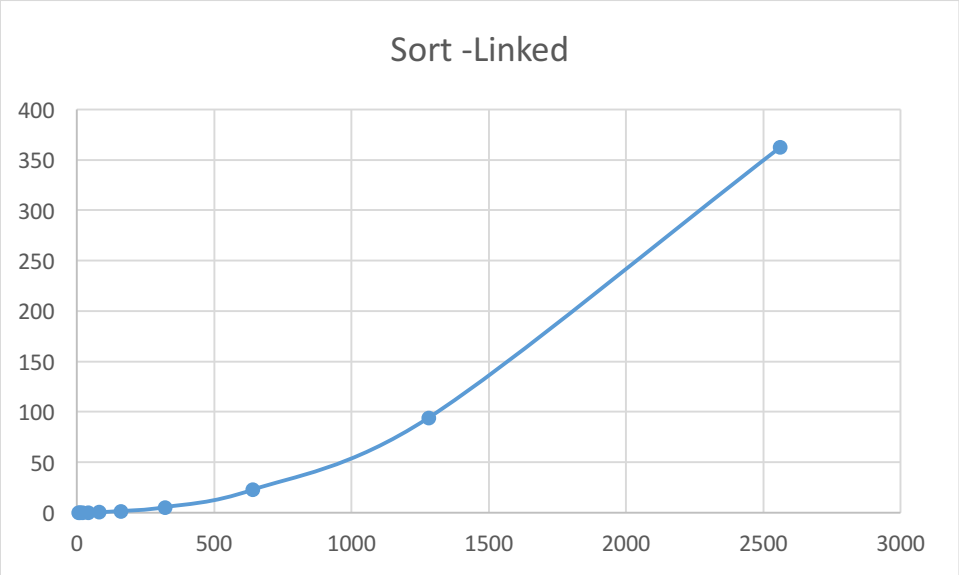


When "loading" on a linked list, which essentially means adding to a growing linked list, the Big O Notation will be O(N).

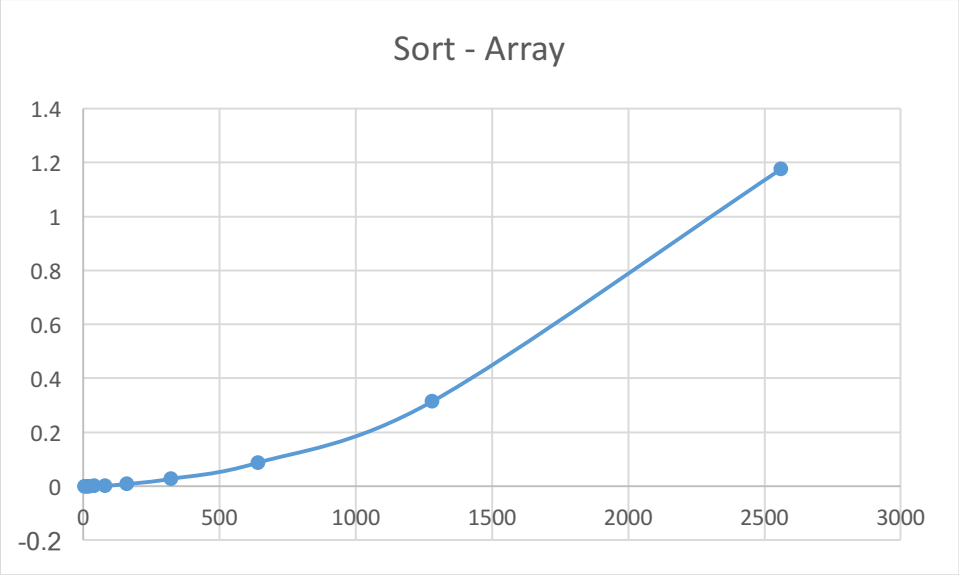


When "loading" on an array list, which means to add to a growing array list, the Big O Notation will be O(N).

"Sort" Array List vs. Linked List Data

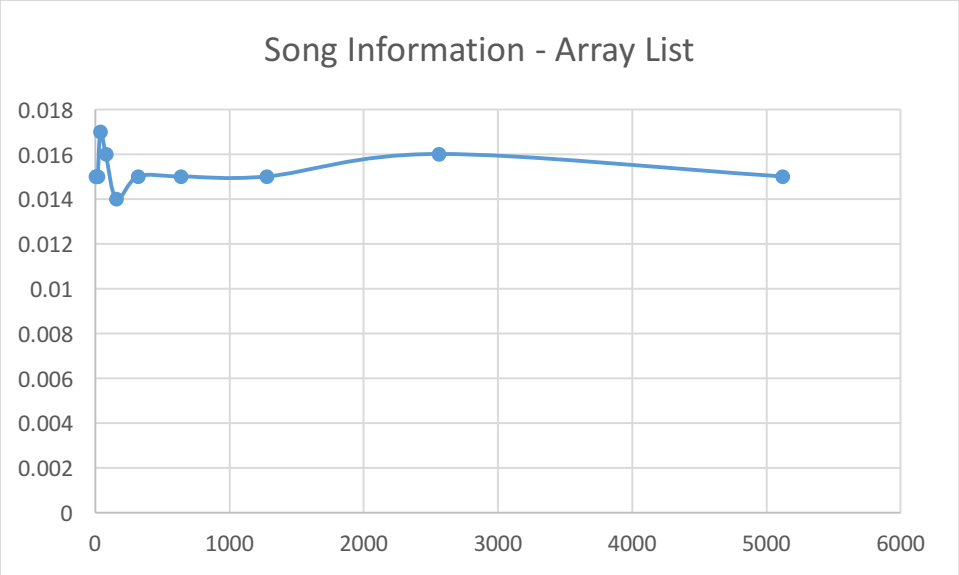


The Big O Notation of a Sorted Linked List will be O(N**2) as was discussed in class.

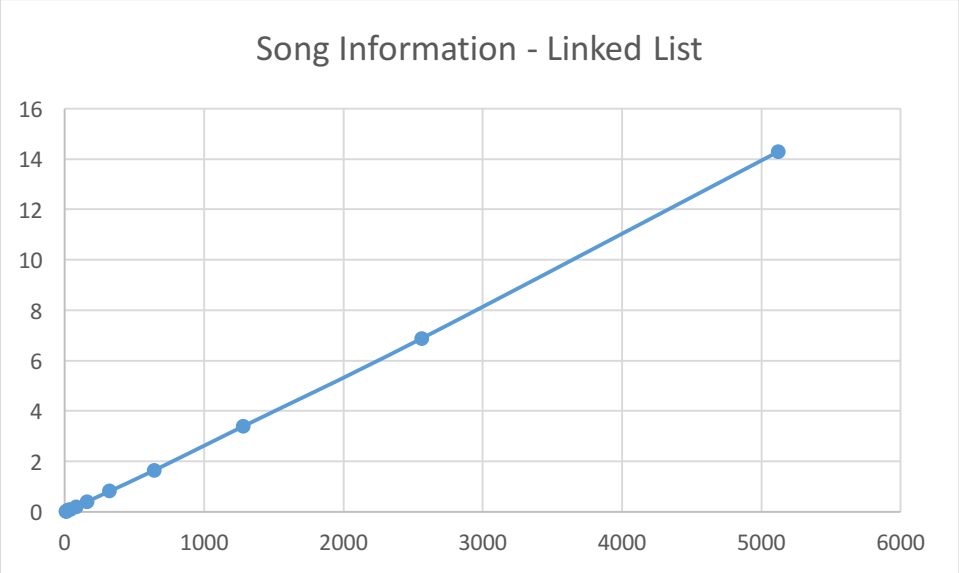


The Big O Notation of a Sorted Array List will be O(N**2) as was discussed in class.

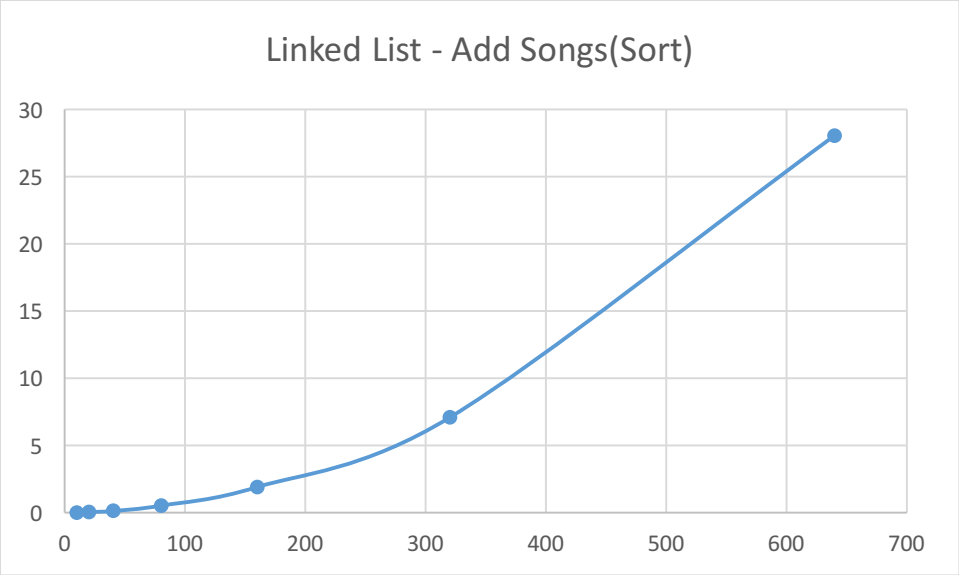
“Song Information” Array List vs. Linked List Data



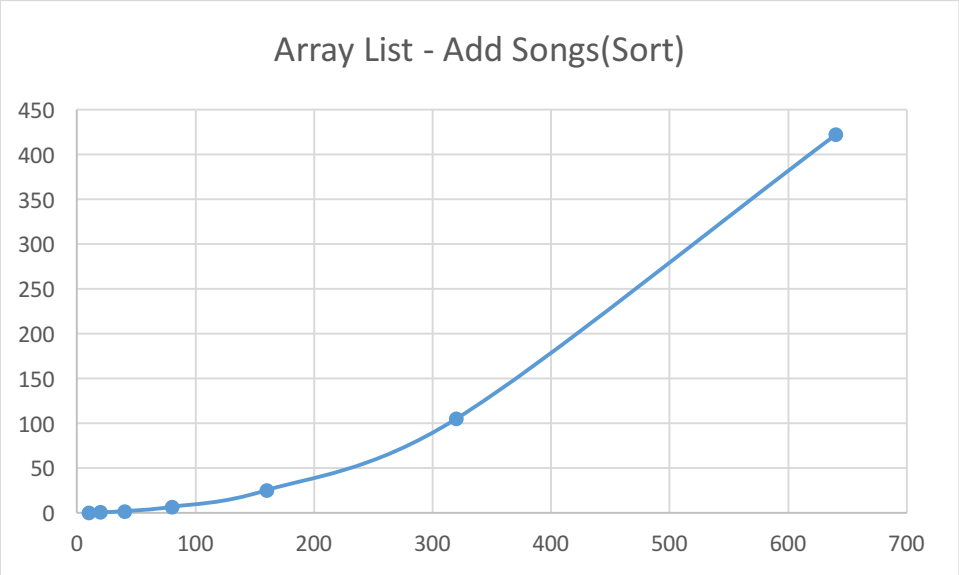
The Song Selection for an Array List, which essentially means to use the get function on an array list has a Big O Notation of O(1) as the get function simply accesses the index of the array list.



The Song Selection for a Linked List, which essentially means to use the get function on a linked list has a Big O Notation of O(N) as the get function has to traverse each element in the list until it reaches the specified index.



The linked list has a Big O Notation of O(N**2) as we have discussed in class. The function builds the list and then sorts the list.



The linked list has a Big O Notation of O(N**2) as we have discussed in class. The function builds the list and then sorts the same list.