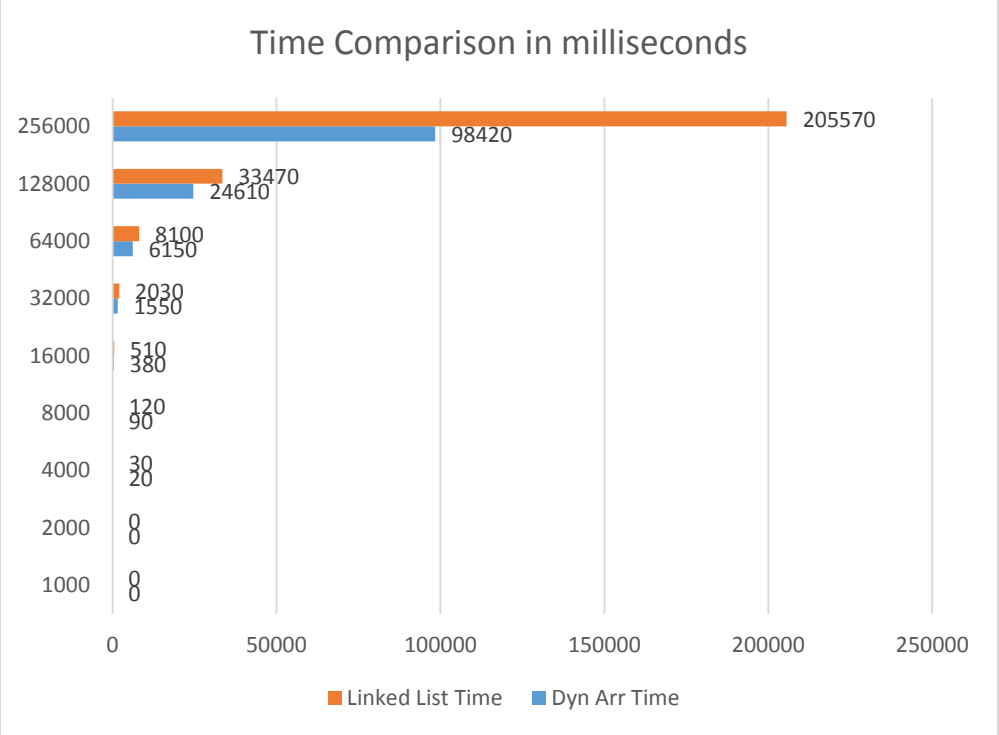
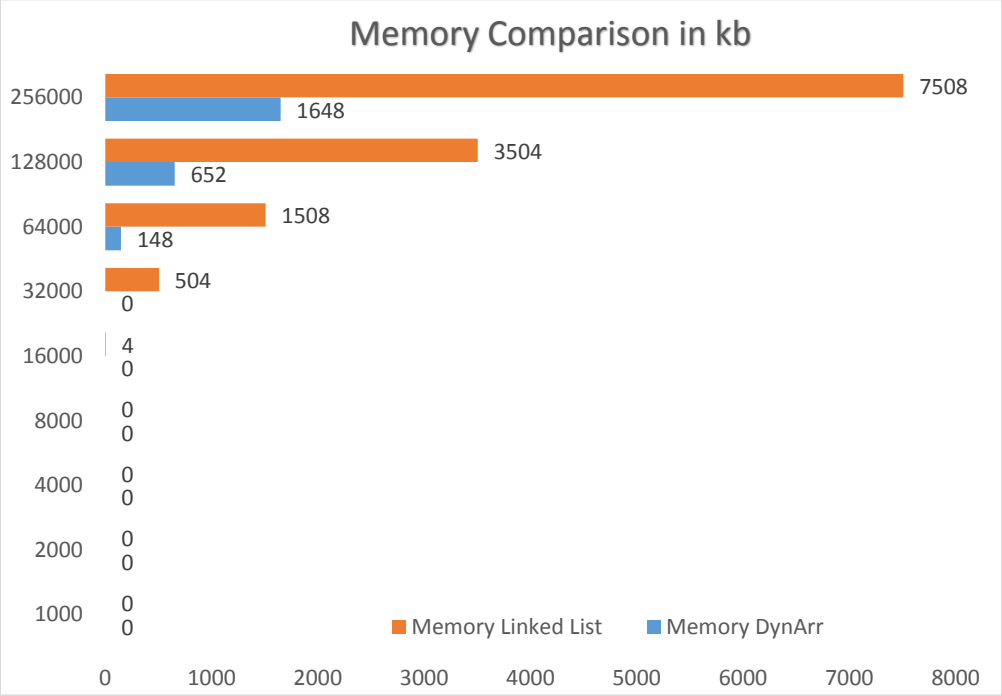
Susan Kuretski
kuretsks@onid.oregonstate.edu
CS261-400
Assignment 3

Comparison Analysis





Questions:

1. Which of the implementation uses more memory? Explain why.

   The linked list uses more memory. This is because each node in a linked list holds the data AND it also needs pointers to the other elements in the linked list. In addition, we implemented a doubly-linked list, so there are two pointers created for each node for the next and previous node, as compared to a singly-linked list. With a dynamic array, the memory is contiguous and each element holds just the data.

2. Which of the implementations is the fastest? Explain why.

   The dynamic array is faster compared to a linked list. To iterate through a linked list requires to find the memory location of each pointer. To iterate through a dynamic array requires accessing a contiguous piece of memory and that can also utilize data caching.

3. Would you expect anything to change if the loop performed remove() instead of contains()? Is so, what?

   This would depend on where/what we are removing. As the for-loop is set up in the test program for contains, it seems we would be removing the first element if keeping with the same setup (adjusting some code a bit albeit). If we were to remove the first element in the dynamic array, then the time complexity would be O(n) since all the elements afterward would be shifted down an index. The bigger the array became the longer it would take to remove that first element. For a dynamic array, removing a certain value at an unknown location would still require O(n) time due to searching for it, removing it, and then adjusting the indices of the rest of the elements.

   If we were to remove the first element in a linked list, then that would be O(1) since just the subsequent link's pointers would have to be assigned, and all the rest would remain the same. Since we're using a doubly-linked list, removing at the end would be O(1) as well since we have a pointer to the end. If we were searching for a certain element to remove in a linked list (not knowing where it is), then this would be O(n) because traversing the list would be required (similar to contains), though the same for the pointer adjustment from the deletion is the same: O(1).

   In short, if we were to do a side-by-side comparison of dynamic array and linked list to search for a value and delete it in an unknown place, then they both would be at O(n). The linked list remove would perform similar to contains. The dynamic array remove would take longer than contains since it must traverse the array to find it and then adjust the subsequent indices after deletion.