Adam Kniffin
OSU ID# 931492308
email: kniffina@oregonstate.edu

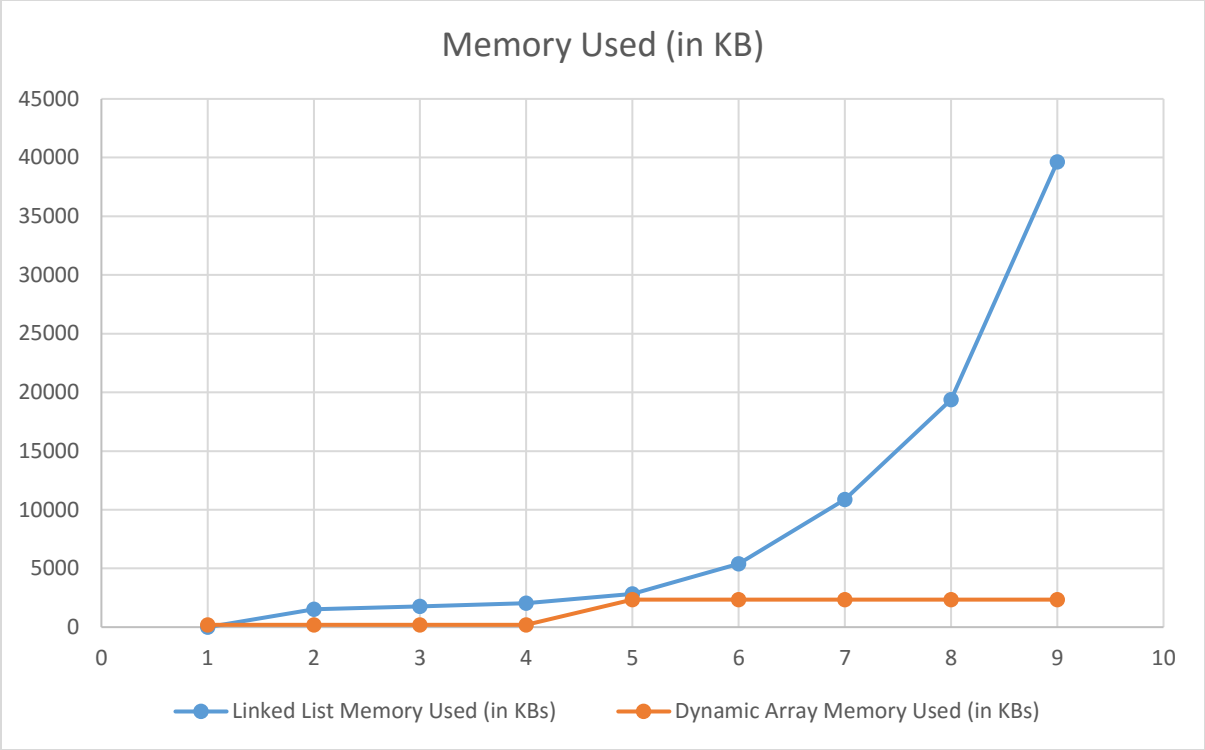
1) Running Time and Memory Usage for Linked List

- 2^10 = 1024 elements: 10 ms Memory used: 0 kb
- 2^11 = 2048 elements: 30 ms Memory used: 1508 kb
- 2^12 = 4096 elements: 110 ms Memory used: 1764 kb
- 2^13 = 8192 elements: 440 ms Memory used: 2028 kb
- 2^14 = 16384 elements: 1050 ms Memory used: 2824 kb
- 2^15 = 32768 elements: 3820 ms Memory used: 5400 kb
- 2^16 = 65536 elements: 15990 ms Memory used: 10872 kb
- 2^17 = 131072 elements: 59950 ms Memory used: 19372 kb
- 2^18 = 262144 elements 263600 ms Memory used: 39624 kb

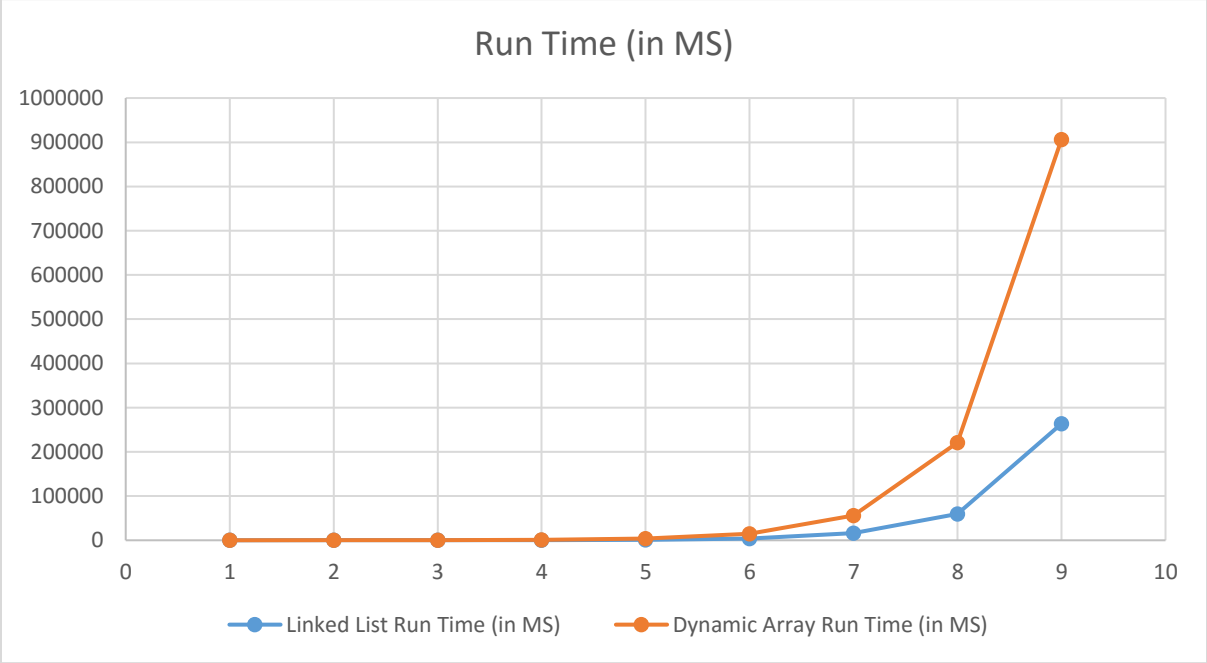
2) Running Time and Memory Usage for Dynamic Array

- 2^10 = 1024 elements: 30 ms Memory used: 196 kb
- 2^11 = 2048 elements: 100 ms Memory used: 196 kb
- 2^12 = 4096 elements: 410 ms Memory used: 196 kb
- 2^13 = 8192 elements: 1090 ms Memory used: 196 kb
- 2^14 = 16384 elements: 3770 ms Memory used: 2344 kb
- 2^15 = 32768 elements: 14920 ms Memory used: 2344 kb
- 2^16 = 65536 elements: 55740 ms Memory used: 2344 kb
- 2^17 = 131072 elements: 221130 Memory used: 2344 kb
- 2^18 = 262144 elements: Memory used: 2344 kb

Plot of Memory Used – Linked List vs Dynamic Array



Plot of Running Time – Linked List vs Dynamic Array



3) Answers to 3 Questions

i. *Question: Which of the implementations uses more memory? Why?*

Answer: The linked list implementation uses more memory if there is a large group of elements. Once a dynamic array starts to grow in size it is much better for a memory approach because the amortized runtime for a push or pop is 0(1). With the linked list there is additional overhead that takes place with moving the pointers around and is usually O(n) extra memory due to the storage of extra pointers in each element. The dynamic array does have the extra cost of adding the elements into the array but for the large numbers that started to be much less. Also, as we search the linked list using contains, we also have to move pointers around to the next element instead of moving memory locations very easily with the dynamic array.

ii. *Question: Which of the implementation is the fastest? Why?*

Answer: The linked list implementation was much faster. I think this is because the addition of an element is always around O(1). Depending on the type of act that is taking place it can cost extra to move the pointers around, but as the list gets bigger in the dynamic array it becomes more and more expensive to find the element that it is looking for.

iii. *Would you expect anything to change if the loop performed remove() instead of contains()? If so, why?*

Answer: I would expect the dynamic array to have a higher cost of memory. This is because the array would have to resize (depending on the beginning number of elements) and each resize would have a big-oh of O(n). The linked list should have a amortized efficiency of O(1).