Raw data:

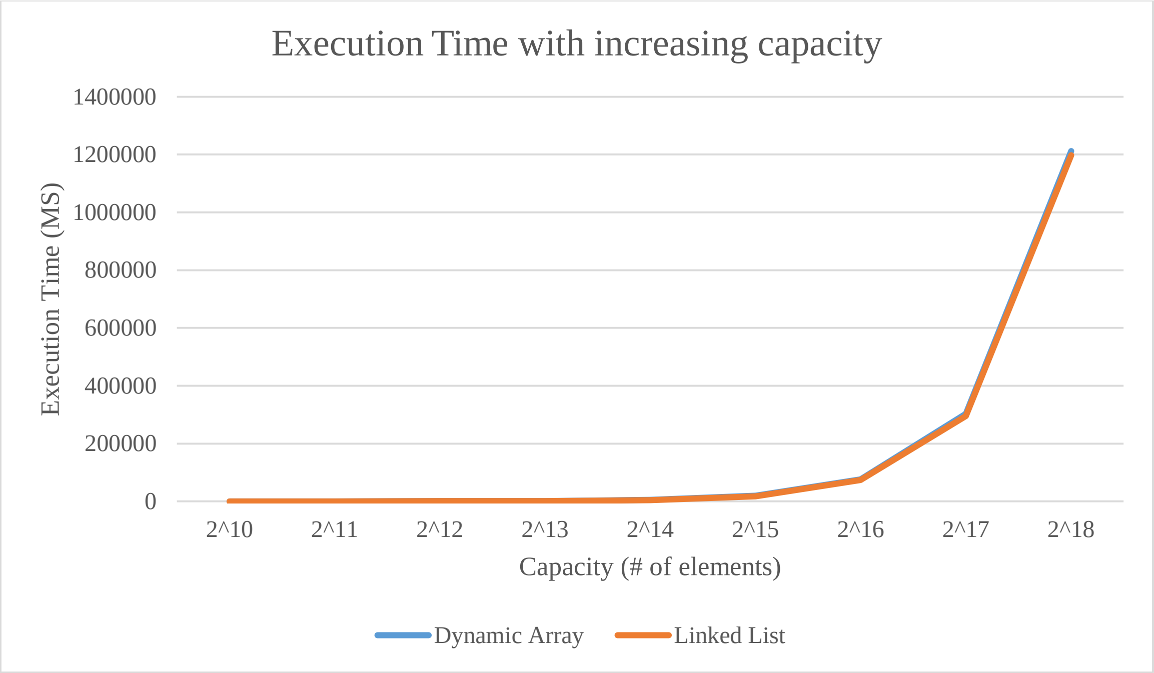
Linked List:

# of elements	**Memory Usage**	**Execution Time**
(2^10) 1,024	1,180KB	20ms
(2^11) 2,048	1,180KB	70ms
(2^12) 4,096	1,180KB	550ms
(2^13) 8,192	1,436KB	1,130ms
(2^14) 16,384	2,228KB	4,440ms
(2^15) 32,768	4,076KB	17,740ms
(2^16) 65,536	10,540KB	73,380ms
(2^17) 131,072	20,964KB	295,750ms
(2^18) 262,144	41,548KB	1,197,300ms

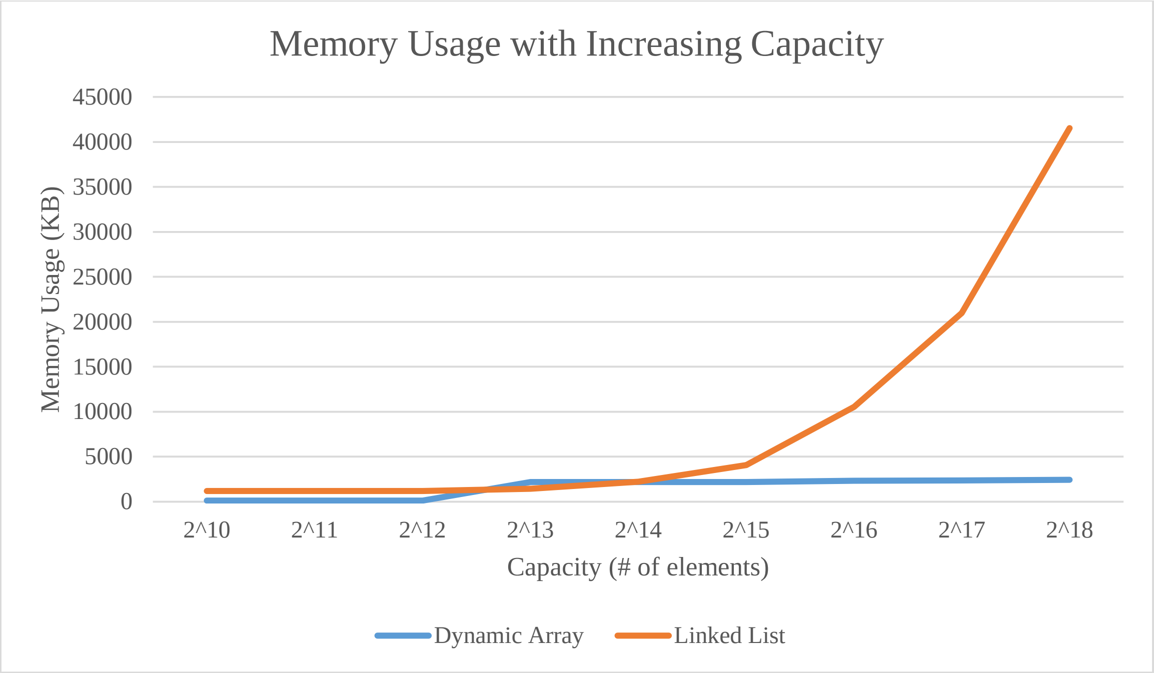
Dynamic Array:

# of elements	**Memory Usage**	**Execution Time**
(2^10) 1,024	124KB	20ms
(2^11) 2,048	124KB	80ms
(2^12) 4,096	124KB	460ms
(2^13) 8,192	2,172KB	1,230ms
(2^14) 16,384	2,172KB	5,170ms
(2^15) 32,768	2,172KB	19,170ms
(2^16) 65,536	2,308KB	75,840ms
(2^17) 131,072	2,356KB	302,850ms
(2^18) 262,144	2,436KB	1,212,254ms

Graphs begin on the next page.



The two methods are so close in execution time that a difference between them can barely be seen. As the number of elements double, the execution time quadruples, suggesting $O(n^2)$ complexity.



Dynamic array seems to be more constant with memory usage, while dynamic array is closer to linear as N grows large. Near the end, we see memory usage double as capacity doubles.

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

The Linked List implementation uses much more memory than the dynamic array. The linked list needs to hold space for two pointers and a value, while the dynamic array is just a contiguous block of data.

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

They are around the same speed, because with both operations, the contains() function is an O(n) complexity operation.

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

If the loop performed remove() instead of contains(), the linked list would take less time overall. Linked Lists can remove items from the data structure in O(1) time, while dynamic array removal is an O(n) operation in most cases, due to the shuffling required.