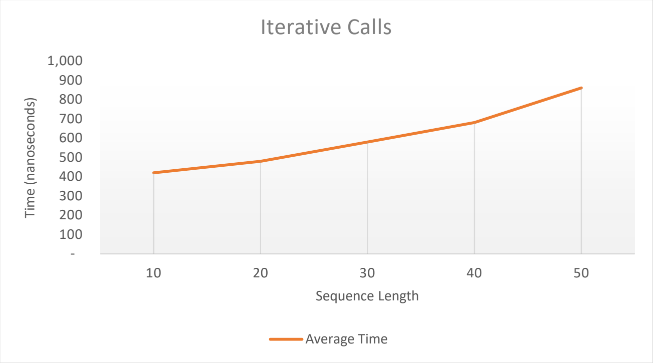
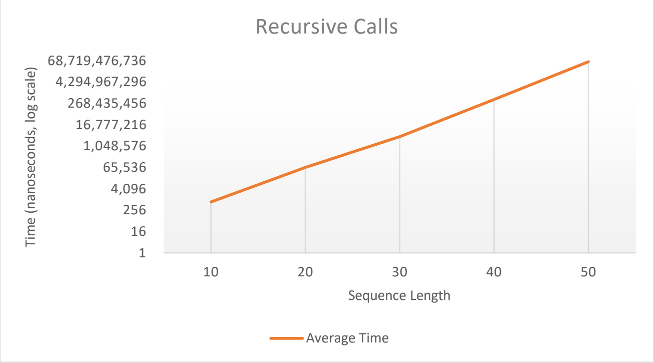
Call Type	Term Length	Run #1	Run #2	Run #3	Run #4	Run #5	Average Time
Iterative	10	1,100	300	200	300	200	420
Iterative	20	1,000	400	400	300	300	480
Iterative	30	900	500	500	500	500	580
Iterative	40	900	700	600	600	600	680
Iterative	50	1,100	800	800	800	800	860
Recursive	10	700	1,700	500	400	400	740
Recursive	20	66,100	103,900	69,900	45,400	43,900	65,840
Recursive	30	3,902,000	3,372,200	3,490,900	3,371,400	3,617,900	3,550,880
Recursive	40	436,355,800	442,960,700	448,699,700	444,652,700	448,958,700	444,325,520
Recursive	50	54,956,136,000	61,174,416,500	61,872,311,500	60,105,766,400	60,322,967,700	59,686,319,620





In all cases, the iterative call was faster than the recurisve ones. As the axes of the two graphs show, the iterative method increases in time linearly, but the recursive method increases exponentially. This is because the iterative method requires a single call, each with n runs through a loop, for any sequence of length n. For example, an n-value of 2 requires two runs through a for-loop; an n-value of 4 requiers four runs through a for-loop. The recursive method, however...

For an n-value of 2, it must calls itself once using $n-1=1$ and $n-2=0$, for a total of two calls. For an n-value of 4, it calls itself once using $n-1=3$ and $n-2=2$. The call which has $n=3$ passed into it will call itself with $n-1=2$ and $n-2=1$. Then, the two calls with $n=2$ with, as we calculated before, each call two more times. This means we have, for an n-value of 4, calls itself six times.

One of these is far more complex than the others. The iterative method is $O(n)$ complexity, and the recursive method is $O(n^2)$ complexity[https://goo.gl/BhfqUZ]. This is why one increases in time linearly, and the other exponentially.