Assignment 3 Writeup - Jack Vento

Recursive Solution

Due to my algorithm calling itself twice with n-1 times, its recurrence relation ends up being *$T(n) = 2T(n-1)$*, which when converted to Big O notation, simplifies down to a time complexity of *$O(2^n)$*. As for space complexity, recursive functions have a complexity of O(depth) and since we have a depth of n, our final memory complexity is *$O(n)$*.

Iterative Solution

Similarly to the recursive solution, my iterative algorithm has a time complexity of *$O(2^n)$* and a memory complexity of *$O(n)$*. Our moves loop executes 2^n-1 times, giving us our aforementioned exponential time complexity (which is seemingly the point of this problem: to get insurmountably large fast). Furthermore, each stack has a capacity of n items, so we have a memory complexity of *$O(3n)$* which simplifies down to *$O(n)$*.

Comparing the Two

Although our Big O complexities are the same, the recursive solution is both the less complex from a comprehension and a development standpoint and uses slightly less memory. Implementing the recursive solution took sub-5 minutes whereas the iterative solution took multiple hours to design, implement, and debug alongside requiring the creation of a new data structure and two helper functions.

All in all, the recursive solution is the significantly less complex solution, although the space & memory complexities of the two end up being the same.