Adam Rich
EN.605.202.87.SP18 Data Structures
Max Array Problem
Feb 14, 2018

Question: what is the Big O complexity of a "divide and conquer" algorithm for finding the maximum integer in an array of n integers?

Start with investigating small cases. How many operations does it take for an array of 1, 2, or 3 elements?

(x)	->	No operations, just return x
(x, y)	->	One to get max(x, y)
(x, y, z)	->	Split to (x, y) and (z) 1 for (x, y), 0 for (z), 1 to compare max(x, y) and (z) = 2 total
(w, x, y, z)	->	Only 1 op more than (x, y, z) Split to (w, x) and (y, z) 1 + 1 + 1 to check on the re-combine = 3 total

It looks like the pattern will be

1. Split array recursively till getting either (x, y) or (z)
2. 1 op for base case
3. + 1 for each split (to compare on the recombine)

So, +1 op for each pair or single from splits which will be n/2 in the best case, i.e. n = 2^k, or 2*n/3 as the worst case. Either way, it is a multiple of n, so O(n) without operations for recombine.

Splits will be log(n).

The complexity will be O(n + log n) = O(n).

Which basically means that this algorithm is no more efficient, really, than the iterative pairwise algorithm discussed in the lecture. I would prefer the iterative algorithm since the overhead of recursion doesn't add any efficiency.