The following problems are for those of you looking to challenge yourself beyond the required problem sets and programming questions. Most of these have been given in Stanford's CS161 course, Design and Analysis of Algorithms, at some point. They are completely optional and will not be graded. While they vary in level, many are pretty challenging, and we strongly encourage you to discuss ideas and approaches with your fellow students on the "Theory Problems" discussion form.

1. You are given as input an unsorted array of n distinct numbers, where n is a power of 2. Give an algorithm that identifies the second-largest number in the array, and that uses at most $n + \log_2 n - 2$ comparisons.
2. You are a given a *unimodal* array of n distinct elements, meaning that its entries are in increasing order up until its maximum element, after which its elements are in decreasing order. Give an algorithm to compute the maximum element that runs in O(log n) time.
3. You are given a sorted (from smallest to largest) array A of n distinct integers which can be positive, negative, or zero. You want to decide whether or not there is an index i such that A[i] = i. Design the fastest algorithm that you can for solving this problem.
4. You are given an n by n grid of distinct numbers. A number is a *local minimum* if it is smaller than all of its neighbors. (A neighbor of a number is one immediately above, below, to the left, or the right. Most numbers have four neighbors; numbers on the side have three; the four corners have two.) Use the divide-and-conquer algorithm design paradigm to compute a local minimum with only O(n) comparisons between pairs of numbers. (**Note:** since there are n^2 numbers in the input, you cannot afford to look at all of them. **Hint:** Think about what types of recurrences would give you the desired upper bound.)

Mark as completed