Due at the start of class Wednesday, June 24, 2015.

1. Draw the Decision Tree for Bubble Sort on three elements A, B, C (which start in positions indexed by 1, 2, 3 of the array, respectively). Note that Bubble Sort is inefficient so it does some redundant comparisons; some comparisons will not have two children. In those cases just show the child that can actually occur.

2. Assume you are given an array $A[0, \ldots, n-1]$ of n numbers, and a function $k(n)$. You can assume $k(n)$ is a slowly growing function such as a constant or $\log(n)$. You are told that every number is within $k(n)$ positions of its correct location using mod n arithmetic.
   (a) Give an algorithm that sorts this list with as few comparisons as possible (as a function of n and k). How many comparisons does your algorithm use? Just get the high order term right.
   (b) Show that your algorithm is optimal using a decision tree argument. Your combinatorics can be a little loose (and may not be quite legitimate), but it should give the correct bound.

3. Assume you are given an array $A[0, \ldots, n-1]$ of n numbers, and a function $k(n)$. You can assume $k(n)$ is a slowly growing function such as a constant or $\log(n)$. You are told that there is some integer r such that if the array is rotated r positions (mod n), every number will be within $k(n)$ positions of its correct location. In other words, the number in location i belongs in some position between $r + i - k(n) \bmod n$ and $r + i + k(n) \bmod n$ (inclusive). You do not know the value of r.

   Another way of saying this is that you start with an array that is almost sorted: every element is with $k(n)$ positions of its sorted location (using mod n arithmetic). Then some joker comes along and rotates the array by some unknown amount.
   (a) Give an algorithm that sorts this list with as few comparisons as possible (as a function of n and $k(n)$). Exactly how many comparisons does your algorithm use? Just get the high order term right.
   (b) Show that your algorithm is optimal using a decision tree argument. Your combinatorics can be a little loose (and may not be quite legitimate), but it should give the correct bound.