10 practice problems

Some of these problems would be quite challenging on an exam, or have solutions that are simply too lengthy for an exam. Others are relatively easier. They are all good to get you thinking about the material.

1. Someone hacks into your sorted array of n distinct elements, and randomly displaces one element (shifting everything in between its initial and final position). For example, displacing 2 in 1,2,3,4,5 and positioning it between 4 and 5 would result in 1,3,4,2,5. Between standard Insertion-sort and Merge-sort, which would you use to minimize the worst-case time of getting the array back in sorted order? Why?

2. The analysis of Merge sort in the course notes assumes that n is even. Write down a recurrence for Merge sort that is more accurate, without this assumption. Without actually analyzing the more accurate recurrence directly, explain why the time complexity is still $O(n \log n)$.
   In other words, don't use induction or trees, instead use one of my favorite techniques.

3. Given a list of n distinct real numbers and a value k, how can you report all pairs in the list that have a difference of k? Find two different ways to solve this.

4. You are given an array A, containing n real numbers without duplicates. For every pair of elements, we say that the pair is "in order" if the smaller value is found somewhere to the left of the larger one. For example, in the array $[13, 11, 14, 12]$ there are three pairs that are in order: $\{13, 14\}$, $\{11, 14\}$, $\{11, 12\}$. Show how to find the number of such pairs in A, in $O(n \log n)$ time.

5. You have k sorted lists, each of size n. How fast can you sort all this data, in terms of k and n?

6. Show that we don't really need to worry about the Case 3 technicality of the master method (see course notes), if $f(n)$ is n^k or 2^n or $\log^k n$. Here, k is a positive constant.

7. *This problem is described in section* 4.1 *of CLRS (p.68). Here's a summary:*
   You are given an array A of n real numbers. You get to select two indices of A, $i \leq j$, and your score will be the sum of all values stored between A[i] and A[j] (inclusive). For example if you select 3 and 5, your score will be A[3] + A[4] + A[5].
   (a) The book says that a brute-force algorithm would run in quadratic time. I disagree. What do you think brute-force would be?
   (b) Can you solve the problem in quadratic time?
   (c) Can you solve the problem in $O(n \log n)$ time?

8. Considering the two heap-build methods covered in class:

   (a) Given an array with distinct elements, will both methods give the same heap, not give the same heap, or does it depend on the input? Justify your answer.

   (b) We say an algorithm is *stable* if any pair of elements with equal value appear in the same order in both the input and output. Assume that both heap-building methods will not swap elements that have equal value. Which heap-building method is stable, if any? What about the extraction phase of heapsort that follows heap-building?

9. Suppose that you have n real numbers, listed in groups of size k. The groups are already in "sorted" order in the sense that every number from one group is smaller than every number from the next. However within each group nothing is sorted. For example, [3,6,2,5], [12,15,11,19], [25,21,28,27], etc.
   You must return all numbers in sorted order.
   (a) What would you do to sort this input, and what would the time complexity be?
   (b) How many possible outputs (permutations of the input) are there?
   (c) Provide a lower bound for the worst case time complexity of fully sorting the input, using your answer from (b). Do this without Θ-notation.
   (d) Convert the answer that you get in (c) to an expression that is more "user-friendly", using Θ-notation.

10. Provide your own binary decision tree that can sort any 5 distinct numbers, with the smallest possible depth.
    Note: this tree will be large, so if there are similar subtrees, just draw one of them and explain that others are similar. For instance if two subtrees have exactly the same structure, but their indices are a permutation of each other, then it suffices to provide one subtree and the permutation that encodes the similarity. For example, see Figure 1.

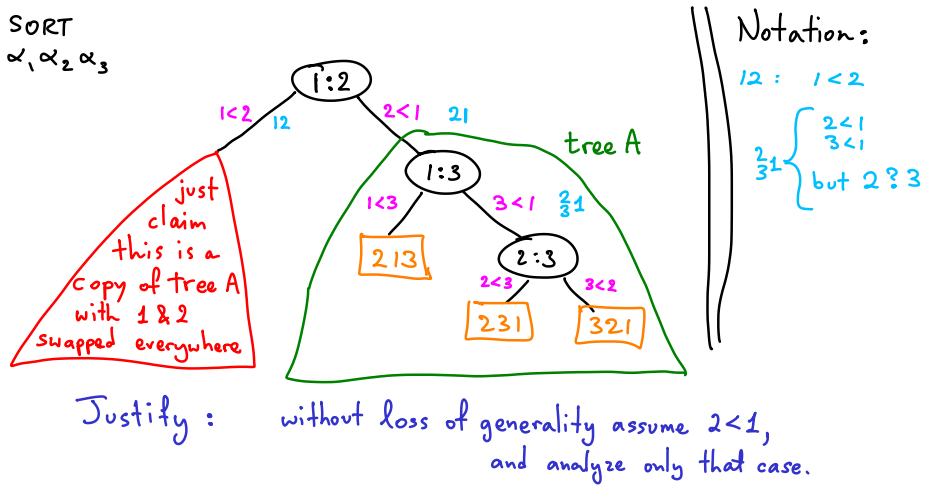

Figure 1: Some warmup and notation that can be used for the problem of sorting 5 numbers.