ALGORITHMS, FALL 2019, HOMEWORK 4

– This assignment is worth 2 units.
– Due on Thursday, October 3, at noon.

1. **Note: This problem does not need to be typed.**
   I expect high quality figures, as described on the info page.
   For an array $A = [a_1, a_2, a_3, a_4]$ of distinct numbers, there are two main ways to build a heap, as described in class. In parts (a) and (b) if this problem you must show what comparisons each method will make, in the form of a binary decision tree. Each leaf should contain output in the form of some permutation of the input subscripts in A (e.g., if you write 3124 it means that after building the heap we have $A = [a_3, a_1, a_2, a_4]$.

   (a) Do the above (draw the decision tree) for the forward method.

   (b) Do the above (draw the decision tree) for the reverse method.

   (c) Describe your own heap-building algorithm that specifically handles inputs of size 4, and draw the corresponding decision tree that uses fewer decisions in the worst-case compared to the methods in (a) and (b). Your algorithm should be described in English, not pseudocode.

2. Suppose that we have a set of n distinct numbers that we wish to sort. The numbers are given in some arbitrary order. Each such order represents a possible input.

   Show that **every** comparison-sort algorithm takes $\Omega(n \log n)$ time, not just in the worst case, but for almost all inputs. Specifically show that the best conceivable algorithm can't even handle $\frac{1}{2^n}$ of all possible inputs in *little-*$o(n \log n)$ time.

   Recall that $f(n) = o(n \log n)$ means $f(n) = O(n \log n)$ and $f(n) \neq \Omega(n \log n)$.