Code Jam 2018 - Qualification Round

Analysis: Trouble Sort

Test set 1

Like bubble sort, Trouble Sort has O(**N**^{2}) time complexity; the proof is explained below. With **N** ≤ 100 for test set 1, we can run Trouble Sort to completion and simply iterate over the result list to find the first sorting error, if any (that is, a value that is greater than the value that follows it in the list).

Test set 2

Running O(**N**^{2}) Trouble Sort to completion is too slow for **N** ≤ 10^5.

Instead, let's break down what Trouble Sort is doing at each step. Let's consider an input list of 6 elements. Trouble Sort makes the following comparisons on each pass through the array:

- element 0 ↔ element 2
- element 1 ↔ element 3
- element 2 ↔ element 4
- element 3 ↔ element 5

Regardless of the length of the list, this table illustrates the fundamental flaw in Trouble Sort: even-index elements are compared with other even-index elements, and odd-index elements are compared with other odd-index elements, but even-index and odd-index elements are never compared with each other! This means that Trouble Sort is just bubble sort run separately on the even-index elements and the odd-index elements, interleaving them into the output list. Trouble Sort is correct only if interleaving the two sub-lists (the even-index list and the odd-index list) happens to produce another sorted list. Since there are O(**N**) even-index and O(**N**) odd-index elements, and since bubble sort is O(**N**^{2}), Trouble Sort is also O(**N**^{2}).

To solve test set 2, we can can run our favorite O(**N** log **N**) sorting algorithm independently on the two sub-lists described above, interleave the sorted sub-lists, and then find the first sorting error as in our solution for test set 1.