

## Lab 3

**Problem 1. Goofy Sort.** Goofy has thought of a new way to sort an array `arr` of  $n$  distinct integers:

- (a) Step 1: Check if `arr` is sorted. If so, return `arr`.
- (b) Step 2: Randomly arrange the elements of `arr` (using your work in Problem 2 of this lab)
- (c) Step 3: Repeat Steps 1 and 2 until there is a return.

Answer the following:

- (A) Will Goofy's sorting procedure work at all? *Solution:* Yes, most of the time (see analysis below).
- (B) What is a best case for GoofySort? *Solution:* The best case is when the input array is already sorted.
- (C) What is the running time in the best case? *Solution:* The running time in the best case is  $O(n)$  (it takes  $O(n)$  time to verify that `arr` is in sorted order).
- (D) What is the worst-case running time? *Solution:*  $\infty$  – this happens if no random arrangement ever occurs in sorted order.
- (E) Is the algorithm inversion-bound? *Solution:* No. Consider the case in which the input array is in reverse-sorted order (so there are  $n(n-1)/2$  inversions) and, after the first trial, the randomly generated array produced is in sorted order. In that case, checking array is sorted twice required approximately  $2n$  comparisons; but  $2n < n(n-1)/2$ , so, in this case, fewer comparisons are done than there are inversions in the input array.