

### Lab 3

1. 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.
  - b. Step 2: Randomly arrange the elements of `arr` (Hint: this can be done in  $O(n)$ )
  - 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? Explain
  - B. What is a best case for GoofySort?
  - C. What is the running time in the best case?
  - D. What is the worst-case running time?
  - E. Is the algorithm *inversion-bound*?
2. *Interview Question.* An array  $A$  holds  $n$  integers, and all integers in  $A$  belong to the set  $\{0, 1, 2\}$ . Describe an  $O(n)$  sorting algorithm for putting  $A$  in sorted order. Your algorithm may not make use of auxiliary storage such as arrays or hashables (more precisely, the only additional space used, beyond the given array, is  $O(1)$ ). Give an argument to explain why your algorithm runs in  $O(n)$  time.
3. *BubbleSort*
  - a. Improve the BubbleSort implementation so that in the best case (which means here that the input is already sorted), the algorithm runs in  $O(n)$  time. Explain why your new version works --- in other words, prove that the best case running time of your code is  $O(n)$ . Call your new Java file `BubbleSort1.java`.
  - b. Recall that in BubbleSort, at the end of the first pass through the outer loop, the largest element of the array is in its final sorted position. After the next pass, the next largest element is in its final sorted position. After the  $i$ th pass ( $i=0,1,2,\dots$ ), the largest, second largest, ...,  $i+1$ st largest elements are in their final sorted position. Use this observation to cut the running time of BubbleSort in half. Implement your solution in code, and prove that you have improved the running time in this way. Call your new Java file, which contains the improvements from this problem and the previous problem, `BubbleSort2.java`.
  - c. In this lab folder, I have given you an environment for testing sorting routines. Insert into this environment the original BubbleSort file along with your new BubbleSort1 and BubbleSort2 classes, and run the `SortTester` class. What are the results? Are the results what you expected? Explain why the running times turned out the way they did.