

COS 226 – Data Structures and Algorithms
Fall 2014 – Flipped Lecture Section
Group Worksheet – 09.25.14
20 minutes

1. Awakening drenched in sweat one night, you clearly see your path to fame and fortune. You will build a robotic rhinoceros and tour the country singing songs about nature to children, who will be allowed to play and interact with the rhinoceros. While a real rhinoceros would be too dangerous, you believe a robotic rhinoceros can be kept in check. In each of the situations below, which sort would you use? In all cases, assume memory is not an issue, and that the goal is to minimize run time so that the rhino can react as quickly as possible to any potential trouble. Answers may be used many times.

----- The rhinoceros is outfitted with a large number of sensors, each of which generates objects of type `Observation`. Observations include many instance variables, taken at fixed time intervals, including importance, timestamp, pressure, temperature, light intensity, etc. These are placed in an unsorted array, and every time 1,000,000 `Observations` are generated, they are delivered to a central processing unit that sorts, the `Observations` by the `importance` field, which is of type `double`. What sort should you use to minimize the run time required to sort all `Observations` by `importance`?

- A. Mergesort
- B. Insertion sort
- C. Selection sort
- D. Knuth shuffle

----- Due to some close calls, you're going to refactor the sorting process to deal with a rare but dangerous situation where some `Observations` are generated with an incorrect `importance` value. For engineering reasons not described here, you can detect these by sorting by the `timestamp` and `importance` of each `Observation`.

Instead of `importance`, you first want to sort by the `timestamp` of each `Observation`. The `timestamp` is of a comparable type called `DateTime`. What sort should you use to minimize the run time required to sort all 1000000 `Observations` by `timestamp`?

----- After sorting by `timestamp`, you want to sort by `importance` such that all the objects of the same `timestamp` stay clustered. What sort should you use to minimize the run time while maintaining

3. **Inversions.** Design a subquadratic algorithm that counts the number of inversions in an array.
4. **Linked List Scramble.** Given a singly-linked list containing N items, rearrange the items uniformly at random. Your algorithm should consume a logarithmic (or constant) amount of extra memory and run in time proportional to $N \log N$ in the worst case.