Sean Connor Discussion Prompt 1

1. What is the design value in writing an ADT?

An ADT specifies what something should do. It specifies what something should do, and how information is used and accessed. It seems to me that it allows one to get a broad view of what something does without worrying about how it does said thing.

2. When assessing complexity, what do we measure (or, what do we not measure)?

Complexity gives us an idea of how well something performs. We do not measure, rather provide an estimate on how long something takes to complete as a function of number of cases. We can determine best-case, worst-case, and average-case. This gives us a better idea of whether a method is good or bad, and provides some basis for comparison of different methods.

3. Does an upper bound for a function apply in all cases? Explain.

Yes. There will always be some bound, regardless of complexity (O(n^3), O(logn), O(1),etc). Now, the upper and lower bound may be the same or different, but there does not exist anything that can be completed in zero time (and I believe even that is a bound).

4. Why might it be important to have different measure of complexity (e.g. upper bounds, lower bounds, …)?

To assess the best and worst cases for a particular algorithm.

5. Under what conditions might you not be concerned about upper and lower bounds?

One case is if the upper and lower bound are the same. Else, if one is not concerned about performance.

6. The lecture notes state, “Sometimes the ‘worst’ algorithm is the best choice.” What does this mean, and how might it apply in specific situation?

In some cases, one algorithm may be better than another until a certain n-value, at which it is always worse. If you know the limits of your application, you can determine which is the best algorithm to use.

7. How important is an understanding of space complexity in an era of cheap memory?

Generally not very important.