Module 1 - Discussion Prompt Questions
Adam Rich
Section EN.605.202.87
Spring 2018

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

   We measure the asymptotic upper bound or lower bound of the function. We do not measure the time requirements for input sizes near the origin.

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

   Yes, every function has an upper bound. Define g(n) = f(n) + 1. The function g(n) is *always* greater than f(n), and therefore meets the definition for an upper bound. However, "big O" notation uses only the dominant term of the upper bound function and strips constant coefficients. It may be non-trivial to simplify g(n) to meet these conventions but that doesn't mean it does not exist.

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

   A function f(n) represents an algorithm. Assume that it has an upper bound and lower bound that are not the same. Would the presence of this less expensive lower bound open the possibility that a different implementation of the algorithm might have better performance? Or that there is a way to re-write the function to at least perform better for some inputs?

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

   If you know that your inputs are always going to be close to the origin then the asymptotic behavior of a function is not that important. For example, if β is 1000 and your inputs are always going to be less than that then algorithmic complexity can probably be ignored.

   Another scenario when you may not be concerned is when the inputs might be greater than β but the function is still so fast in the range of possible inputs that it doesn't really matter.

5. **The lecture notes state, "Sometimes the 'worst' algorithm is the best choice." What does this mean, and how might it apply in specific situation?**

   Close to the origin some asymptotically better performing algorithm might perform poorly when compared to something that is more expensive in "big O" notation.

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

   Not as important as it used to be. However, it still must be considered especially when working with lots of data. Storage is still finite for the time being, so it cannot be ignored altogether.