For this assignment we were asked to pick a term from our area of expertise and provide a Parenthetical, Sentence and Expanded definition for it. I chose the term *Computational Complexity* which is a term frequently used by computer scientists in the area of algorithm analysis. These definitions assume the audience has no technical knowledge of algorithm analysis.

**Parenthetical Definition:**

An algorithm’s computational complexity (how much work the algorithm must do) is an important factor to consider when deciding which algorithm to use.

**Sentence Definition:**

Computational complexity is a metric used by computer scientists to measure the amount of work an algorithm must do in order to perform its operation on a set of data. Computational complexity is a very important metric when comparing two similar algorithms or when designing a new algorithm. Computational complexity is also often referred to as the running time or time complexity of an algorithm.

**What does computational complexity look like?**

On paper computational complexity appears as a mathematical function which can be used to show how the algorithm will perform given a certain input size. When the input size is unknown or irrelevant it is commonly denoted as *n.* The letter or symbol in front of the function, most commonly a O if the analysis was done assuming the worst-case for the algorithm’s running time, Ω for the best-case running time or Θ if the worst and best cases are the same. The image from Wayne (2006) below lists several computational complexity functions along with the expected running time for different input sizes.



*Fig. 1* The approximate running times of algorithms with different computational complexities as input sizes increase. (Wayne, 2006, p. 9)

**How is computational complexity used?**

Computational complexity is used in computer science to compare two or more algorithms in order to determine which one is the most efficient. In order to do this the complexity of all algorithms in question is calculated and the algorithm that has the lowest computation complexity is the most efficient. This metric can be used to not only measure time efficiency of an algorithm but also how much space an algorithm uses (Kleinberg & Tardos, 2006, p. 29).

**What is computational complexity?**

Computational complexity is a metric used by computer scientists to measure how long or how much space an algorithm with require in order to compare it with similar algorithms to aid in design decisions.

**What is an example of computational complexity?**

There are many different algorithms for sorting data these days and it is important to know which algorithm is going to perform the best given your needs. Two sorting algorithms are Quicksort and Insertion Sort. The average time complexity of Quicksort is O(n log(n)) (Kleinberg & Tardos, 2006, p. 734) which means that for a list with 100 items in it Quicksort will take 200 operations to complete. Insertion sort on the other time has an average time complexity of O(n^2) (Cormen, 2001, p. 7) which means on 100 items it will take 10,000 operations to complete. By comparing the computational complexity of these two algorithms we can see that Quicksort is the better option in most cases.

**Works Cited:**

Cormen, T. H. (2001). *Introduction to algorithms.* MIT Press.

Kleinberg, Jon, & Tardos, Eva (2006). *Algorithm Design.* Boston : Pearson/Addison-Wesley.

Wayne, Kevin (2006). *Lecture slides for algorithm design: algorithm analysis* [PDF document]. Pearson/Adison-Wesley. Retrieved from<http://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/02AlgorithmAnalysis.pdf>