Comparing Algorithms

One of the primary goals of this course is to develop and apply analytic techniques to enable comparing two or more algorithms. The purpose of such a comparison is to help computer scientists determine the "best" algorithm for their particular problem. Typically, such a comparison focuses on one of two possible areas—time and space. In other words, from a resource optimization perspective, we are concerned with minimizing the amount of time required for an algorithm to run, as well as minimizing the space required to store the data for processing.

How should we go about comparing algorithms? In general, we may want to assess performance empirically where we implement the algorithms and run a series of experiments. This is an excellent way to assess performance on specific implementations of algorithms but can be problematic when attempting to draw general conclusions about the underlying algorithms themselves. To make such generalizations, careful experimental design is a must! A second approach is to apply formal mathematical analysis to the algorithm to determine what the complexity bounds are. Here, mathematical expressions characterizing time and space requirements are formulated and then solved.

Given the need to compare algorithms, we also need to ask appropriate questions to guide that comparison. For example, what performance bounds should be the focus of the analysis? Are we interested in the worst-case conditions for the algorithm and the associated complexity? Perhaps what we really want to know is how the algorithm can be expected to perform "on average." But this raises issues of what we mean by "average" in that it suggests knowledge of some distribution of the data can be determined. At times we might be interested in the best case conditions of the algorithm, perhaps because such analysis tells us the theoretical limit of what we can expect.

For best and worst case analysis, we often do not care about the true, closed-form solution to the mathematical expression we devised, but we don't want to rely on empirical assessment either. This leads to the goal of finding an "asymptotic bound" on performance where we find the complexity within some constant multiplier. In other words, we seek to find an expression that "bounds" the performance as tightly as possible without necessarily providing a detailed mathematical expression. This is because many of the constants and "low-order" terms in the expression are driven by specific implementation of the algorithms … something of little interest to us when attempting to determine the theoretical properties of the algorithm.

Analyzing Algorithms

A major component of this class will be developing the skills necessary to analyze algorithms in support of the above comparisons. In addition to analyzing the resource requirements (i.e., time and space) for the algorithm, we will often be concerned with determining the correctness of the algorithm. In other words, we will want to prove that the algorithm does what we think it does. Elements of such analysis include determining the conditions under which we can expect the algorithm to terminate, and verifying the extent to which we get the right answer on data provided as input.