Complexity Theory (Bronze)

Hariank Muthakana, adapted by Nate Foss

October 2, 2015

1 Introduction

Complexity theory is a way of describing how long your program will take to run as size of your data set increases, also called efficiency. It is important for every contest we do, because the complexity of your algorithm is crucial when it comes to data sets on the order of 10^8 items long. Most of the time, the obvious solution would get you the correct answer but is way to slow. The true challenge lies in coming up with a clever algorithm with low complexity.

2 Big-O Notation

Big-O is how we quantify this efficiency. It is written as a function *O(g)*, where g itself is a function of N. N is our magic number - it is the number of items we are operating on and is usually given as input for the problem. For example, a simple but important complexity is *O(N)*, or *linear time*. This means that as the input N increases linearly, the *maximum number of operations* taken by the program will also increase linearly. For more than one input, the Big-O function will involve multiple variables.