**Big O Notation**

Jacob Johnson

Big O Notation is part of a family of notations that were invented by Paul Bachmann, Edmund Landau, and others. These notations are collectively known as Bachmann-Landau notation, or asymptotic notation. The symbol O was first introduced by number theorist Paul Bachmann in 1894, in the second volume of his book Analytische Zahlentheorie ("analytic number theory"), and later picked up by Landau. In computer science, big O notation is useful in the analysis of algorithms and is used to bound the growth of running time to a set of constant factors. Big O notation essentially states that running time grows at most a certain amount but could grow more slowly.

Big O is the most commonly used asymptotic notation for comparing functions. For asymptotically tighter bounds, big Theta Θ can be used.

In the case of binary searches, the longest runtime is Θ(lg n). It is not accurate to say that is the runtime of all binary searches, but rather that is the longest runtime a binary search can experience. To be more accurate in describing the runtime of binary searches in general, we can use big O notation. Binary searches run in O(lg n), as a blanket statement that is most accurate.

Big O usually describes worst case scenario of the execution time required or the memory space used by an algorithm. In computer science it can be used to compare the efficiency of different approaches to a problem. It is a bit more desirable than you would get form other mathematical functions as it focuses on what is basically happening. This can tell us how quickly the runtime grows relative to the input. Since it is the rate at which runtime grows, something other than a direct measurement is needed, this is commonly n.

Big O notation can help a programmer foresee excessive compiling and memory time/space, therefore notifying the programmer of potential problems that should be avoided. An undesirable growth in runtime could prompt the programmer to look into alternative ways to code a solution in a manner that increases efficiency and decreases undesirable increase in runtime or memory used. As programs grow larger and more complex, anticipating what problems can arise with a finished product becomes increasingly necessary. Big O notation can assist with this process.