COT 5407: Introduction to Algorithms

Author and Copyright: Giri Narasimhan
Florida International University
Lecture 2: August 30, 2007.

1 What is an algorithm?

An algorithm is a recipe, or a sequence of steps, to correctly solve a problem. But then, what is a "problem"? And what does it mean to "correctly" solve it? To define a problem, one needs to define all allowable inputs along with the corresponding outputs. The allowable inputs can be specified by identifying the input domain. The corresponding outputs can be specified by identifying the output domain and by providing a mapping from the input domain to the output domain. Note that the mapping guarantees that every element of the input domain has a defined corresponding output value in the output domain. A sequence of steps is said to be an *algorithm* for a specific *problem* if for **every** allowable input it (a) stops, and (b) produces the correct output for that input.

2 The Big-Oh notation

We say that $f(n) = O(g(n))$ if there exists positive constants c and n_0 such that for all $n \geq n_0$, we have $f(n) \leq c \cdot g(n)$. If $f(n) = O(g(n))$, then $g(n)$ is an asymptotic upper bound for $f(n)$, while $f(n)$ is an asymptotic lower bound for $g(n)$. For convenience, we define a separate notation for lower bounding a function. We say that $f(n) = \Omega(g(n))$ if there exists positive constants c and n_0 such that for all $n \geq n_0$, we have $f(n) \geq c \cdot g(n)$. Finally, we say that $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$. In other words, if $f(n) = \Theta(g(n))$, then there exist positive constants c_u, c_l, and n_0 such that for all $n \geq n_0$, we have $c_l \cdot g(n) \leq f(n) \leq c_u \cdot g(n)$.

In practical terms, if you want to show that $f(n) = O(g(n))$, then you need to find (i.e., specify) the values of the positive constants c and n_0, and then show that for all $n \geq n_0$, we have $f(n) \leq c \cdot g(n)$.

Exercise 2.1 *Most of you have had some logic course in the past. Can you write down the* negation *of the statement defining when $f(n) = O(g(n))$? In order to do it correctly, note these simple rule. The negation of "$\exists c > 0$ such that A is true" is "$\forall c > 0$, A is false". Similarly, the negation of "$\forall c > 0$, statement A is true" is "$\exists c > 0$, such that A is false".*

Exercise 2.2 *Furthermore, if you want to show that $f(n) \neq O(g(n))$, then you need to show what? Can you then show that $3n^2 - 7 \neq O(n)$?*

Suggestion 2.1 *The end of chapter 3 has many problems related to the definition of the big-Oh notation. Solve as many of these as possible.*

3 Summary

The big-Oh notation provides a notation and a framework to specify the asymptotic behavior (i.e., when the size of the input tends to ∞) of algorithms. It strips constant terms and factors and highlights the most dominant term in an expression.