Growth of Functions

Yumo Bai, Sayeed Ghani

August 2022

1 Introduction

We learned about the Big O, Big Θ and Big Ω notations during our first 2 classes. In this document we will be presenting some supplementary material for those of you who are interested in getting a deeper understanding of these notations. Please note that this supplementary material, unless otherwise specified, will not be covered in the assignments, quizzes or exams.[1][1]

2 Big O O(g(n))

In the algorithm analysis we use O-notation when we only have the **asymptotic--upper bound**. Mathematically we define O as

$$O(g(n)) = \{f(n) : \exists c, n_0 \in R^+ \quad \forall n \geq n_0 \ : 0 \leq f(n) \leq cg(n)\}$$

Or if we described this in plain English, a function f(n) is in $O(g(n))$ if there exists positive constants c and n_0 such that for all $n \geq n_0$ the relation $0 \leq f(n) \leq cg(n)$ is always true.

For example, we have a function $f(n) = 2n^2 + n + 10$. We can prove that is f(n) is $O(n^2)$ as follows

$$2n^2 + n + 10 \leq cn^2$$

$$2 + \frac{n}{n^2} + \frac{10}{n^2} \leq c$$

$$2 + \frac{1}{n} + \frac{10}{n^2} \leq c$$

By the definition above, we could easily see that a c=3 and $\forall n \geq n_0 = 4$ fulfill the above condition. Then we have $\forall n \geq 4, \quad f(n) \leq 3n^2$. Thus we proved that the f(n) is bounded by $O(n^2)$. The following graph also demonstrate that such relation is true.

[1] Please see [1] chapter 2 for further details on the topic

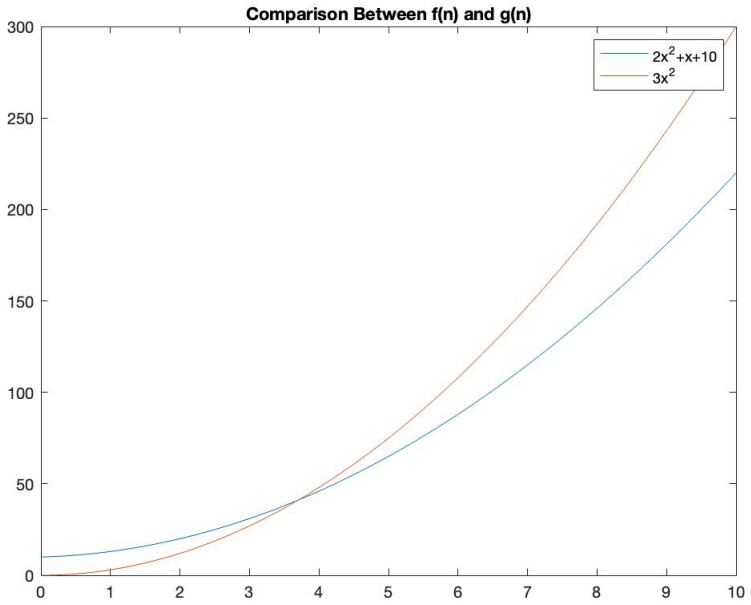

Figure 1: f(n) and g(n)

However, we may soon realize some problems with the BigO notation, in that is can also be a loose upper bound. For example, if a function is $f(n) = 3n^3+10$, we find that $f(n) = O(n^3)$ and $f(n) = O(n^4)$ are also true based on the above definition(Note: the equal sign here does not implies equality relation and all it actually means is that $f(n) \in O(n^3)$ and $f(n) \in O(n^4)$). Therefore, in order to be more precise when describing the asymptotic performance of an algorithm or data structure we use the Big Theta notation.

3 Big Theta $\Theta(g(n))$

We can use the Θ-notation if we want to have a more precise complexity of an algorithm. Hence $\Theta(g(n))$ is used as an **asymptotically tight bound**. Mathematically we define Θ as:

$$\Theta(g(n)) = \{\exists c_1, c_2, n_0 \in R^+ \quad \forall n \geq n_0 \ : 0 \leq c_1 g(n)) \leq f(n) \leq c_2 g(n)\}$$

which compared with O notation, just includes one more lower bound for the growth of the function.

Using the example in the O notation, we can prove that $f(n) = 2n^2 + n + 10$ is also $\Theta(n^2)$ as follows

$$c_1 n^2 \leq 2n^2 + n + 10 \leq c_2 n^2$$
$$c_1 \leq 2 + \frac{n}{n^2} + \frac{10}{n^2} \leq c_2$$
$$c_1 \leq 2 + \frac{1}{n} + \frac{10}{n^2} \leq c_2$$

If we can take $c_1 = 2 \quad c_2 = 3$ and $\forall n \geq n_0 = 4$, we can verify that the above condition holds and hence$f(n) = 2n^2 + n + 10$ is $\Theta(n^2)$.

By using the Θ notation, we will be able to describe the asymptotic performance of an algorithm more precisely and by definition we could easily prove:

$$\Theta(g(n)) \subseteq O(g(n))$$

which shows $\Theta(g(n))$ is a more precise indication than $O(g(n))$

4 Big Omega $\Omega(g(n))$

Sometimes in algorithm analysis, we will also care about the lower bound of the growth of the function. For example, when we introduce the binary search tree later on in this course, we will wonder what would be the best case that we could convert a linear list to a Binary Search Tree(BST). So, we introduce the Ω-notation which shows the **asymptotic lower bound** for the growth of a function. Mathematically we define Ω as:

$$\Omega(g(n)) = \left\{ f(n) : \exists c, n_0 \in R^+ \quad \forall n \geq n_0 \ : 0 \leq cg(n) \leq f(n) \right\}$$

Using the example above, We can prove that $f(n) = n^2 + n + 10$ is $\Omega(n^2)$

$$cn^2 \leq 2n^2 + n + 10$$
$$c_1 \leq 2 + \frac{n}{n^2} + \frac{10}{n^2}$$
$$c_1 \leq 2 + \frac{1}{n} + \frac{10}{n^2}$$

If we can make $c = 2 \quad \forall n_0 \geq n_0 = 1$, we can verify that $f(n) = 2n^2 + x + 10$ is $\Omega(n^2)$.

By using the Ω notation, we will be able to describe the best possible asymptotic performance of an algorithm and by definition we could easily prove:

$$\Theta(g(n)) \subseteq \Omega(g(n))$$

5 Question:

If we have f(n) and g(n). can you prove the following relation based on above definitions?

$$\{f(n) = \Theta(g(n))\} \Leftrightarrow \{f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))\}$$

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition.* The MIT Press, 3rd edition, 2009.