Asymptotics

Spencer Hirsch, Tyler Gutowski, Remington Greko

February 6, 2023

You have been randomly assigned to teams. Work together to write a report crossing this first bridge on algorithmic quest.

Submit the team's report on Canvas. Include a task matric indicating who did what.

Asymptotic Quest

After successful completion of these exercises you will understand the topic of *Asymptotics* and be able to explain and correctly answer questions about the topic.

The Pieces and their relationships

The pieces are functions which we will call f, g, and h, should we need others they can be named.

Standard relations include:

less than, equal, greater than, etc.

Relations can have properties such as:

Reflexing, Symmetric, Transitive

Quantifiers are also needed

For all, There exists...

Write precise (mathematical) definitions of the following relations:

1. Big-O:

   Big-O is used as a general equation in order to find the time complexity of a function. It is hardly ever exact, however, it is unnecessary for it to be exact as a general estimate is sufficient in determining the complexity of an algorithm based on n, the number of input.

T(n) and f(n) are two positive funtcions. We can write $\mathbf{T(n) \in O(f(n))}$, and say that T(n) has order of f(n), if there are positive constants M and n_0 such that T(n) $\leq$ M * f(n) for all n $\geq$ n_0

(`https://yourbasic.org/algorithms/big-o-notation-explained/`)

2. Big-Ω:

   Big Ω is used to give a lower bound for the growth of a function. It is very similar to traditional Big-O, however the inequality is different.

   T(n) and f(n) are two postivie funtions. We write $\mathbf{T(n) \in \Omega(f(n))}$, and say that T(n) is Big-Ω of f(n), if there are positve constants m and n_0 such that T(n) $\geq$ m(f(n)) for all n $\geq$ n_0

   (`https://yourbasic.org/algorithms/big-o-notation-explained/#omega-and-theta-notation`)

3. Big-Θ:

   Now that general complexity and lower bound complexity have been defined, we can now look at Big-Θ which is used to determine both the upper and the lower bound of the time complexity function.

   $\mathbf{T(n) \in \Theta(f(n))}$ if T(n) is both O(f(n)) and Ω(f(n))

Give examples of functions that satisfy these relations.

Note: **All bounds are defined as n** $\rightarrow \infty$

1. Big-O:

   The function $\mathbf{f(n) = 3n^2 + 2n + 1}$ is defined by $\mathbf{O(n^2)}$ because there exists a constant $\mathbf{c = 3}$, $\mathbf{n_0 = 1}$ such that for all $\mathbf{n > 1}$, $\mathbf{f(n) \leq c \cdot g(n)}$, where $\mathbf{g(n) = n^2}$. This means that the function $\mathbf{f(n)}$ has a growth rate no worse than a quadratic function (n^2).

   The function $\mathbf{f(n) = 5n \cdot log(n) + 1000}$ is defined by $\mathbf{O(n \cdot log(n))}$ because there exists a constant $\mathbf{c = 5}$, $\mathbf{n_0 = 1}$ such that for all $\mathbf{n > 1}$, $\mathbf{f(n) \leq c \cdot g(n)}$, where $\mathbf{g(n) = n \cdot log(n)}$. This means that the function $\mathbf{f(n)}$ has a growth rate no worse than a log-linear function (n $\cdot$ log(n)).

   (`https://jarednielsen.com/big-o-log-linear-time-complexity/`)

2. Big-Ω:

   The function $\mathbf{f(n) = 2n^3 + 100}$ is defined by $\Omega\mathbf{(n^3)}$ because there exists a constant $\mathbf{c = 1}$, $\mathbf{n_0 = 1}$ such that for all $\mathbf{n > 1}$, $\mathbf{f(n) \geq c \cdot g(n)}$, where $\mathbf{g(n) = n^3}$. This means that the function $\mathbf{f(n)}$ has a growth rate no better than a cubic function (n^3).

   The function $\mathbf{f(n) = 2n^2 + 4n + 2}$ is defined by $\Omega\mathbf{(n)}$ because there exists a constant $\mathbf{c = 1}$, $\mathbf{n_0 = 1}$ such that for all $\mathbf{n > 1}$, $\mathbf{f(n) \geq c \cdot g(n)}$, where $\mathbf{g(n) = n}$. This means that the function $\mathbf{f(n)}$ has a growth rate no better than a linear function (n).

   (`https://course.ccs.neu.edu/cs5002f18-seattle/lects/cs5002_lect9_fall18_notes.pdf`)

3. Big-Θ:

   The function $\mathbf{f(n) = 2n + 100}$ is defined by $\Theta\mathbf{(n)}$ because there exists constants $\boldsymbol{c_1 = 1}$, $\boldsymbol{c_2 = 1}$, and $\boldsymbol{n_0 = 1}$ such that for all $\mathbf{n > 1}$, $\boldsymbol{c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)}$ where $\mathbf{g(n) = n}$. This means that the function $\mathbf{f(n)}$ has both an upper and lower bound defined by $\mathbf{n}$.

   The function $\mathbf{f(n) = n^2 + 3n + 2}$ is defined by $\mathbf{\Theta(n^2)}$ because there exists constants $\boldsymbol{c_1 = 1}$, $\boldsymbol{c_2 = 1}$, and $\boldsymbol{n_0 = 1}$ such that for all $\mathbf{n > 1}$, $\boldsymbol{c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)}$ where $\mathbf{g(n) = n^2}$. This means that the function $\mathbf{f(n)}$ has both an upper and lower bound defined by $\mathbf{n^2}$.

   (`https://www.geeksforgeeks.org/analysis-of-algorithms-big-%CE%B8-big-theta-notation/`)

Explain how these relations describe bounds on running time (or other resources) expended when an algorithm is executed on input of size n:

The relationship between these asymptotic upper bounds is not an asymptotically tight bound, meaning that it can be (imprecisely) stated that the running time for a binary search is $\mathbf{O(n)}$. This is because $\mathbf{O(\log n)}$ will always be less than $\mathbf{O(n)}$, $\mathbf{O(n!)}$, $\mathbf{O(2^n)}$, etc.

Because Big-Θ defines both the upper and lower bounds for running time, it can be stated that in an algorithm that has a running time of $\Theta\mathbf{(f(n))}$, then it must also be $\mathbf{O(f(n))}$.

However, the opposite is not always true. It is important to note that the relationships between each asymptotic notation are not conversely proportional. Simply because a bound imposed onto Big-Θ results in the bounds of Big-O changing does not mean that a bound imposed onto Big- O will result in the bounds of Big-Θ changing.

Use binary search as an example. The worst-case time complexity of a binary search is $\Theta(\mathbf{log^2 n})$, the $\mathbf{O(log^2 n)}$. But simply because binary search runs in $\mathbf{O(log^2 n)}$, it does not mean that it will run in $\mathbf{O(log^2 n)}$.

(`https://www.khanacademy.org/computing/computer-science/algorithms/asymptotic-notation/a/big-o-notation`)

Name	**Section**
Remington Greko	Examples of functions
Tyler Gutowski	Relationship of bounds on running time
Spencer Hirsch	Mathematical definitions and explanations