PH502: Scientific Programming Concepts

Irish Centre for High End Computing (ICHEC)

September 23, 2020

Overview

- There are some metrics that are used in formal time analysis.
- These are big-Ω, big-Θ and Big-O.

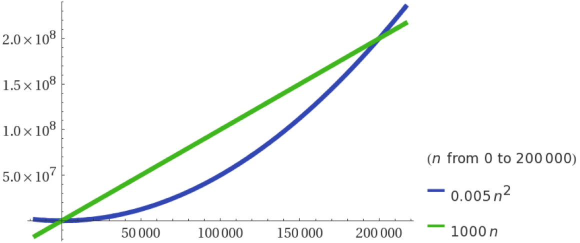
Big-Theta Notation: how it works?

- Complexity is expressed using Θ notation and is written as $\Theta(t(n))$ where n is the input size.
- $t(n)$: the number of elementary operations performed by the algorithm.
- The part of $t(n)$ that increases the fastest as the value of n increases.
- Drop lower terms, Ignore leading constants.

  - $t(n) = 100n \Rightarrow \Theta(n)$
  - $t(n) = 2n^2 + 55n + 10 \Rightarrow \Theta(n^2)$
  - $t(n) = 10 \Rightarrow \Theta(1)$

Big-Theta Notation: how it works?

- Efficiency relates to the general case for a large enough input set.
  - $t_1(n) = 1000n\ (\Theta(n))$
  - $t_2(n) = 0.005n^2\ (\Theta(n^2))$
- For small input sizes t_2 is faster than t_1, but for values above a certain threshold this is not true.



- $\Theta(n)$ always beats $\Theta(n^2)$.

Big-O, Big-Omega and Big-Theta

Big-O Notation

- Let t and g be positive functions of a single positive integer argument n. Then $t(n)$ is said to be $O(g(n))$ if and only if there are positive integers c and n_0 such that for every integer $n \geq n_0$,

$$t(n) \leq cg(n)$$

- t is (asymptotically) less than or equal to g.
- For example: $t(n) = 4n^2 + 10n + 78$ is $O(n^3)$.
- As $10n^2 > t(n)$ for $n \geq 5$, $t(n)$ is also $O(n^2)$.
- Big-O notation is an upper bound, expressing the worst-case time required to run an algorithm on various inputs.

Big-O, Big-Omega and Big-Theta

Big-Omega Notation

- Let t and g be defined as before. Then $t(n)$ is said to be $\Omega(g(n))$ if and only if there are positive integers c and n_0 such that for every integer $n \geq n_0$,

$$t(n) \geq cg(n)$$

- t is (asymptotically) greater than or equal to g
- For example: $t(n) = 400n + 23$ is $\Omega(1)$
- Big-Omega is a lower bound, expressing the best-case time.

Big-O, Big-Omega and Big-Theta

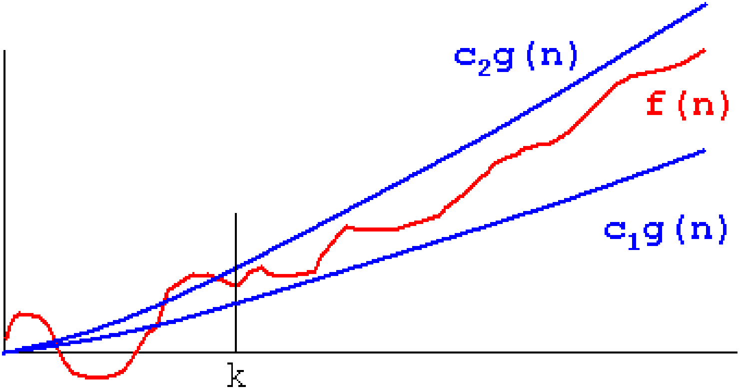
Big-Theta Notation

- Let t and g be defined as before. Then $t(n)$ is said to be $\Theta(g(n))$ if and only if there are positive integers c_1, c_2 and n_0 such that for every integer $n \geq n_0$,
$$c_1 g(n) \leq t(n) \leq c_2 g(n)$$
- t is bounded above and below by g.
- Big-Theta combines both upper and lower bounds; gives an asymptotic equivalence.

Big-O, Big-Omega and Big-Theta

Big-Theta Example

- After $n \geq k$ the red curve falls between the blue lines.
- However the curves need to converge assymptotically to define Θ.



Analysing per Case

- Very commonly an algorithm exhibits different resource consumption complexity depending on the scenario:
  - Worst-Case scenario: Refers to a case where the algorithm performs especially poorly for a particular data set. The solution is found with difficulty and many steps are executed. The algorithm consumes the most resources.
  - Best-Case scenario: A different data set for the exact same algorithm might have extraordinarily good performance. The solution is easily achieved and the algorithm executes very few of its steps. The algorithm consumes the least resources.
  - Average-Case scenario: The algorithm performs somewhere in between these two extremes. The solution is achieved after an expected number of steps. The algorithm consumes average resources.