Thinking about algorithm complexity

and estimating it

Sukrit Gupta

October 13, 2023

Outline

Acknowledgement and disclaimer

All mistakes (if any) are mine.
I have used several other sources which I have referred to in the appropriate places.

Section 1

Asymptotic Notation

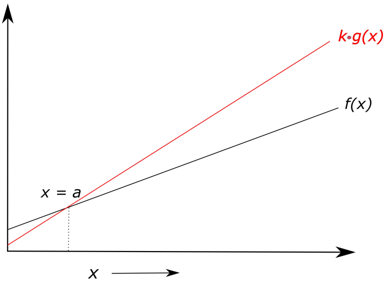
Asymptotic notation

- The *asymptotic notation* provides a formal way to talk about the relationship between the running time of an algorithm and the size of its inputs.
- As a proxy for “very large”, asymptotic notation describes the complexity of an algorithm as the size of its inputs approaches infinity.
- Let’s quickly look at Big O notation again.

The Big O Notation (loose upper bound)

Definition

For functions f and g, we say that $f \in O(g)$ when there exists **at least one** choice of a constant $k > 0$, where you can find a constant a such that the inequality $0 \leq f(x) \leq k \cdot g(x)$ holds for all $x > a$.



Warm Up.

If $f(x) = a_m x^m + \cdots + a_i x + a_0$, prove that $f(x) = O(x^m)$
Proof:

- $f(x) \leq \sum_{i=0}^{m} |a_i| x^i$
- $f(x) \leq x^m \sum_{i=0}^{m} |a_i| x^{i-m}$
- $f(x) \leq x^m \sum_{i=0}^{m} |a_i|$ for $x \geq 1$.
- So, $f(x) = O(x^m)$.

Section 2

Small O Notation

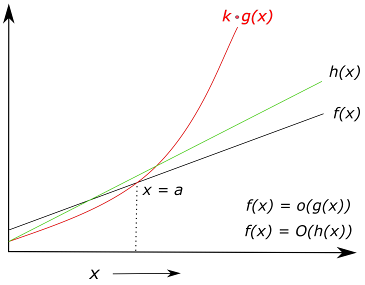
Imagine ...

- You're a **small** kid who is fascinated with his career and likes to dream big.
- You see your father doing his job and drawing a salary $f(x)$, where x denotes his age.
- You imagine that your salary $g(x)$ will be better than your father's at the same age, irrespective of the external factors k that crop up.
- All the external factors contribute a factor k in your life.
- If your salary is $f(x) = O(g(x))$ this may not be true, depending upon external factors. How?
- But if it is $f(x) = o(g(x))$, like your "small" child brain imagined, your salary will always exceed your father's salary no matter what the external factors k are.

The Small O notation (strict upper bound)

Definition

For functions f and g, we say that $f \in o(g)$ when for **every** choice of a constant $k > 0$, you can find a constant a such that the inequality $0 \leq f(x) \leq k \cdot g(x)$ holds for all $x > a$.



Example: $f(n) = 5n^3 + 2n^2 + 7$

- f(n) is $O(n^3)$ because it is bounded by n^3 for sufficiently large n.
- f(n) is not $o(n^3)$ because f(n) can grow at the same rate as n^3 or even faster. However, it is $o(n^4)$ because for all n greater than a certain value, $f(n)$ is always less than n^4.

Example: $f(n) = 2n + 10$

- $f(n)$ is $O(n)$ because for sufficiently large n, it grows linearly.
- $f(n)$ is not $o(n)$ because $f(n)$ can grow at the same rate as n. However, it is $o(n^2)$ because for all n greater than a certain value, $f(n)$ is always less than n^2.

Example: $f(n) = 100$

- $f(n)$ is $O(1)$ because it is a constant function and does not depend on n.
- $f(n)$ is $o(n)$ because it grows slower than any linear function. However, it is not $o(1)$ because $f(n)$ is not significantly smaller than 1 for large n.

Comments on the Small O notation

- For the big O, the inequality $0 \leq f(x) \leq k \cdot g(x)$ has to hold for *at least one* constant k.
- For small O, the inequality $0 \leq f(x) \leq k \cdot g(x)$ has to hold for *all* constants k. k can be very small. Each k can have a corresponding a for the condition $x > a$.
- Thus, small O makes a stronger statement than the corresponding big O notation, i.e., $o(g(x)) \subset O(g(x))$.
- Coming back of our example of the salaries.
- If your father worked in India with salary $f(x)$ and you move to US with a salary $g(x)$, it is likely that your salary $f(x) = o(g(x))$.
- However, in India, the relationship would most likely be $f(x) = O(g(x))$.
- One thing's for sure, Happiness $= O(f(x))$. Can anyone explain why I say so?
- After a certain threshold, happiness is independent of the salary. ;-)

Section 3

Big Ω Notation

The Big Ω Notation (loose lower bound)

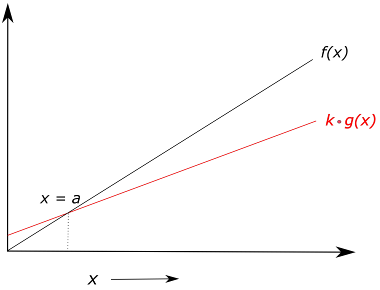
Definition

For functions f and g, we say that $f \in \Omega(g)$ when there exists **at least one** choice of a constant $k > 0$, where you can find a constant a such that the inequality $f(x) \geq k \cdot g(x) \geq 0$ holds for all $x > a$.

- The big Omega gives a lower bound of a function.
- For example, the formula $f(x) \in \Omega(x^2)$ means that the function f grows faster than the quadratic polynomial x^2.

Comments on the Big Ω notation

- Big Ω notation is used to give a lower bound on the asymptotic growth (often called the order of growth) of a function.
- The Big Ω notation is loose and can often be *abused* by making statements like, "the complexity of f(x) is $\Omega(x)$". This means that in the best case f will take $\Omega(x)$ steps to run.
- However, it can be much more than $\Omega(x)$.
- The big Ω represents the best case performance.



Example: $f(n) = 5n^3 + 2n^2 + 7$

$f(n)$ is $\Omega(n^3)$ because, for sufficiently large n, $f(n)$ is greater than or equal to $5n^3$.

Q for U: $f(n) = 2n + 10$

$f(n)$ is $\Omega(n)$ because, for sufficiently large n, $f(n)$ is greater than or equal to $2n$.

Q for U: $f(n) = 100$

$f(n)$ is $\Omega(1)$ because it is a constant function and, for any positive constant c, $f(n)$ is greater than or equal to c for all n greater than a certain value.

Coming back to the looseness of the bounds

- The big O and the big Ω notations are loose.
- For $f(x) = x^2$, the loose bounds can often be *abused* by making statements like "$f(x)$ is $O(x^3)$". This means that in the worst case f will take $\mathrm{O}(x^3)$ steps to run. True, but it can be much less than $\mathrm{O}(x^3)$.
- Or "$f(x)$ is $\Omega(x)$". This means that in the best case f cannot take less than $\mathrm{O}(x)$ steps to run. True, but it can be much more than $\mathrm{O}(x)$.
- What do we do?
- The big $Theta$!

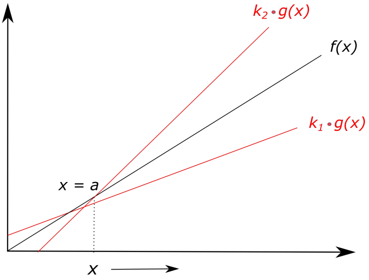
Section 4

Θ Notation

The Θ Notation

Definition

For functions f and g, we say that $f \in \Theta(g)$ when there exists **at least one** choice of a constants $k_1 > 0$ and $k_2 > 0$ where you can find a constant a such that the inequality $0 \leq k_1 \cdot g(x) \leq f(x) \leq k_2 \cdot g(x)$ holds for all $x > a$.



Example: $f(n) = 5n^3 + 2n^2 + 7$

$f(n)$ is $\theta(n^3)$ because, for sufficiently large n, $f(n)$ is both upper bounded by $5n^3$times a constant and lower bounded by $5n^3$ times a constant.

Q for U: $f(n) = 2n + 10$

$f(n)$ is $\theta(n)$ because, for sufficiently large n, $f(n)$ is both upper bounded by $2n$ times a constant and lower bounded by $2n$ times a constant.

Q for U: $f(n) = 100$

$f(n)$ is $\theta(1)$ because it is a constant function. In this case, the function is both the upper and lower bound of a constant times a constant.

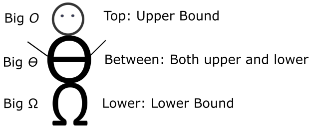
Comments on the Θ notation

- The Big Θ sandwiches $f(x)$ between $k_1 g(x)$ and $k_2 g(x)$.
- It is the tightest bound on the performance of a function, i.e. the best of all the worst case times that the algorithm can take.

How do you remember this?

Table: Summary. Remember $\exists$ means "there exists" and $\forall$ means "for all".

Notation	Name	Description	Formal definition
$f(x) = O(g(x))$	Big O	$\lvert f\rvert$ is bounded *above* by g asymptotically	$\exists k > 0\, \exists a\, \forall x > a:$ $\lvert f(x)\rvert \leq k\, g(x)$
$f(x) = o(g(x))$	Small O	f is *dominated* by g asymptotically	$\forall k > 0\, \exists a\, \forall x > a:$ $\lvert f(x)\rvert < k\, g(x)$
$f(x) = \Omega(g(x))$	Big Omega	f is bounded *below* by g asymptotically	$\exists k > 0\, \exists a\, \forall x > a:$ $f(x) \geq k\, g(x)$
$f(x) = \Theta(g(x))$	Big Theta	f is bounded both above and below by g asymptotically	$\exists k_1 > 0\, \exists k_2 > 0$ $\exists a\, \forall x > a:$ $k_1\, g(x) \leq f(x) \leq k_2\, g(x)$



What did we learn today?

1. Asymptotic Notation
2. Small O Notation
3. Big Ω Notation
4. Θ Notation

Thank you!

Figure: Need your feedback for better delivery.