Asymptotic Notation

CSC165 Week 8 - Part 2

Lindsey Shorser, Winter 2021

Upper, Lower, and Tight Bounds on a Function

$$\mathcal{O}(f) = \{g \mid g : \mathbb{N} \to \mathbb{R}^{\geq 0},\ \exists c, n_0 \in \mathbb{R}^+,\ \forall n \in \mathbb{N},\ n \geq n_0 \Rightarrow g(n) \leq c \cdot f(n)\}.$$

Definition 5.5. Let $f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$. We say that g **is Omega of** f if and only if there exist constants $c, n_0 \in \mathbb{R}^+$ such that for all $n \in \mathbb{N}$, if $n \geq n_0$, then $g(n) \geq c \cdot f(n)$. In this case, we can also write $g \in \Omega(f)$.

Definition 5.6. Let $f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$. We say that g **is Theta of** f if and only if g is both Big-O of f and Omega of f. In this case, we can write $g \in \Theta(f)$, and say that f is a **tight bound** on g.[7]

Example: $g \in \mathcal{O}(f) \implies f+g \in \Theta(f)$

WTS: $\boxed{g \in \mathcal{O}(f)} \implies f+g \in \Theta(f)$

Proof that $f+g \in \mathcal{O}(f)$: Assume that $g \in \mathcal{O}(f)$

$\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \implies g \leq cf$ by hypothesis

$\implies g(n) \leq cf(n)$

(add f to both sides) $\implies g(n) + f(n) \leq cf(n) + f(n)$

$g + f \leq (c+1) f$

Since $c \in \mathbb{R}^+ \implies c+1 \in \mathbb{R}^+$. $\boxed{\text{Let } m = c+1.}$

$\therefore \exists m, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \implies g+f \leq mf$ ■

Proof that $f+g \in \Omega(f)$:

WTS: $\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow f+g \geq cf$

$f : \mathbb{N} \to \mathbb{R}^{\geq 0}$
$g : \mathbb{N} \to \mathbb{R}^{\geq 0}$

$\underbrace{f(n)+g(n)}_{\text{sum of two positive numbers}} \geq f(n)$ by def'n of f, g.

Let $c = 1$ and $n_0 = 1 \Longrightarrow f(n)+g(n) \geq f(n)$
$f(n)+g(n) \geq cf(n) \quad \forall n \in \mathbb{N}, n \geq 1$

$\therefore f+g \in \Omega(f)$ We already know $f+g \in O(f) \therefore f+g \in \Theta(f)$ $\blacksquare$

Example: $\forall a \in \mathbb{R},\ af \in \Theta(f)$

WTS: $\forall a \in \mathbb{R}^{\geq 0},\ af \in \Theta(f)$ where af is $a \cdot f(n)$

Proof of $af \in O(f)$: WTS that $\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}$

$n \geq n_0 \implies af(n) \leq cf(n)$. Let $a \in \mathbb{R}$.

Since $a \leq |a|$

$af(n) \leq |a| f(n)$

$a = a$

$af(n) = af(n)$ $\left(\begin{matrix}\text{Let} \\ a = c\end{matrix}\right)$

$af(n) = cf(n)$

$\implies af(n) \leq cf(n)$

Let $c = |a|$. Then $a \leq |a|$

$a \leq c$

$af(n) \leq cf(n)$ since $f(n) \geq 0$

$\therefore af \in O(f)$.

3

Proof that $af \in \Omega(f)$:

WTS $\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow af \geq cf$.

If $a = 0$,

If $a > 0$, Let $c = a$. Then $af = cf$ so $af \geq cf$ is true.

Can we show $af \geq cf$ if $a < 0$? $c > 0$ $f(n) > 0$

If $a < 0$, $a < 0 \leq f(n)$
$af(n) < 0f(n) \leq f(n)$
$af(n) \leq f(n)$ so $c = 1$ works.

Let $c = 1$. $a < c \Rightarrow af(n) \leq cf(n)$ since $f(n) \geq 0$.

not for $\Omega(f)$ Only for $O(f)$

Moral of the story:

1. “Ignore lower order terms”

2. “Ignore constant factors”

Runtime Example

Goal: To find a simple function f(n) such that the "number of steps" function is an element of Θ(f).

How do we define input size n?

How do we define input size?

"Standard size" = total number of bits required to write down the input

- we will consider integers to have fixed size, but this is not always true. The same is true for floating point numbers.
- size of a string = its length, since characters do have a fixed size
- size of a list = sum of the lengths of its elements

  example: a list of integers would have size length(list) assuming fixed size for integers.
- EXCEPTION: simple algorithms with a single natural number as input have size = value of the number

What is a step?

1 step = any block of code whose runtime is independent of input size

Such code takes "constant time" because it does not change with the size of input.

Examples:

- arithmetic operations `+, -, x, \`
- comparisons `== < >`
- variable assignments `y = x+1`
- return statements `return x+1`

Which features have non-constant runtime?

- loops (to be discussed further)
- function call execution

  Note: making the function call is constant, but execution may not be.
- recursion (CSC236)
- complex data structures (CSC148, CSC263)

The plan for the example

1. Look at some code
2. Consider each loop and count the number of steps for each iteration
3. The total of all such counts is the "number of steps" function. We will call it g(n).
4. Use Θ notation so that $g \in \Theta(f)$ for a simple function f
5. The simplified function f is the goal for the exercise.

```
def f(n : int) -> int  # Assume n >= 0
    r = 0
    for i in range(10):            # Loop 1
        for j in range(n * n):     # Loop 2
            r = r + j
    for i in range(n // 2):        # Loop 3
        for j in range(i * i):     # Loop 4
            r = r - j
    return r
```

```
for i in range(10):          #Loop 1
    for j in range(n*n):     #Loop 2
        r = r+j
```

Loop 2: • iterations $= n^2$ $(j = 0, 1, \ldots, n^2-1)$
• each iteration is 1 step
• total steps $= \underbrace{1+1+\ldots+1}_{n^2 \text{ times}} = n^2$

Loop 1: • iterations $= 10$ $(i = 0, 1, \ldots, 9)$
• each iteration is n^2 steps
• total # steps for Loop 1 $= \overbrace{n^2+n^2+\ldots+n^2}^{10 \text{ times}} = 10n^2$

```
for i in range(n//2):        # Loop 3
    for j in range(i*i):     # Loop4
        r = r - j
```

Loop 4: • # iterations $= i^2$ $\quad j = 0, 1, \ldots, i^2-1$

• 1 step per iteration

• total # steps $= \underbrace{1+1+\ldots+1}_{i^2 \text{ times}} = i^2$

Loop 3: • # iterations $= \lfloor \frac{n}{2} \rfloor$ $\left(i = 0, 1, \ldots, \lfloor \frac{n}{2} \rfloor - 1\right)$

• i^2 steps per iteration

• total # steps $= 0^2 + 1^2 + 2^2 + \ldots + \left(\lfloor \frac{n}{2} \rfloor - 1\right)^2$

$$\sum_{k=0}^{N} k^2 = \frac{N(N+1)(2N+1)}{6}$$

Let $N = \lfloor \frac{n}{2} \rfloor - 1$

$$\text{\# of steps in total} = \frac{\left(\lfloor \frac{n}{2} \rfloor - 1\right)\left(\lfloor \frac{n}{2} \rfloor\right)\left(2\lfloor \frac{n}{2} \rfloor - 1\right)}{6}$$

Lines "r=0" and "return r" count as 1 step.

$$\text{Total \# steps in the program} = 1 + 10n^2 + \frac{\left(\lfloor \frac{n}{2} \rfloor - 1\right)\lfloor \frac{n}{2} \rfloor\left(2\lfloor \frac{n}{2} \rfloor - 1\right)}{6}$$

$1 \in \Theta(1)$, $10n^2 \in \Theta(n^2)$, $\frac{\left(\lfloor \frac{n}{2} \rfloor - 1\right)\left(\lfloor \frac{n}{2} \rfloor\right)\left(2\lfloor \frac{n}{2} \rfloor - 1\right)}{6} \in \Theta(n^3)$

$\therefore$ n^3 is a tight bound for the "number of steps" function for this code.