CSC236 Week 4

Larry Zhang

Announcements

- PS2 due on Friday
- This week's tutorial: Exercises with big-Oh
- PS1 feedback
  - People generally did well
  - Writing style need to be improved. This time the TAs are lenient, next time they will be more strict.
- Read the feedback post on Piazza.

Recap: Asymptotic Notations

- **Algorithm Runtime**: we describe it in terms of the **number of steps** as a **function** of **input size**
  - Like **n^2, nlog(n), n, sqrt(n), ...**
- **Asymptotic notations** are for describing the **growth rate** of **functions**.
  - constant factors don't matter
  - only the highest-order term matters

Big-Oh, Big-Omega, Big-Theta

O(f(n)): The set of functions that grows **no faster** than **f(n)**

- asymptotic **upper-bound** on growth rate

Ω(f(n)): The set of functions that grows **no slower** than **f(n)**

- asymptotic **lower-bound** on growth rate

Θ(f(n)): The set of functions that grows **no faster and no slower** than f(n)

- asymptotic **tight-bound** on growth rate

growth rate ranking of typical functions

$$f(n) = n^n$$

$$f(n) = 2^n$$

$$f(n) = n^3$$

$$f(n) = n^2$$

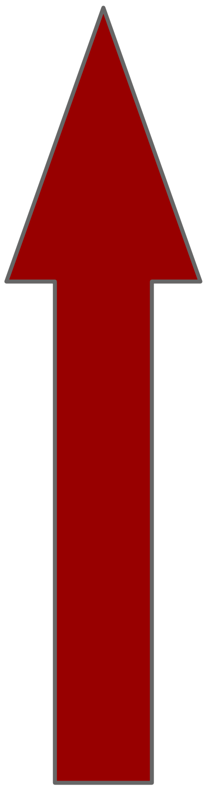
$$f(n) = n \log n$$

$$f(n) = n$$

$$f(n) = \sqrt{n}$$

$$f(n) = \log n$$

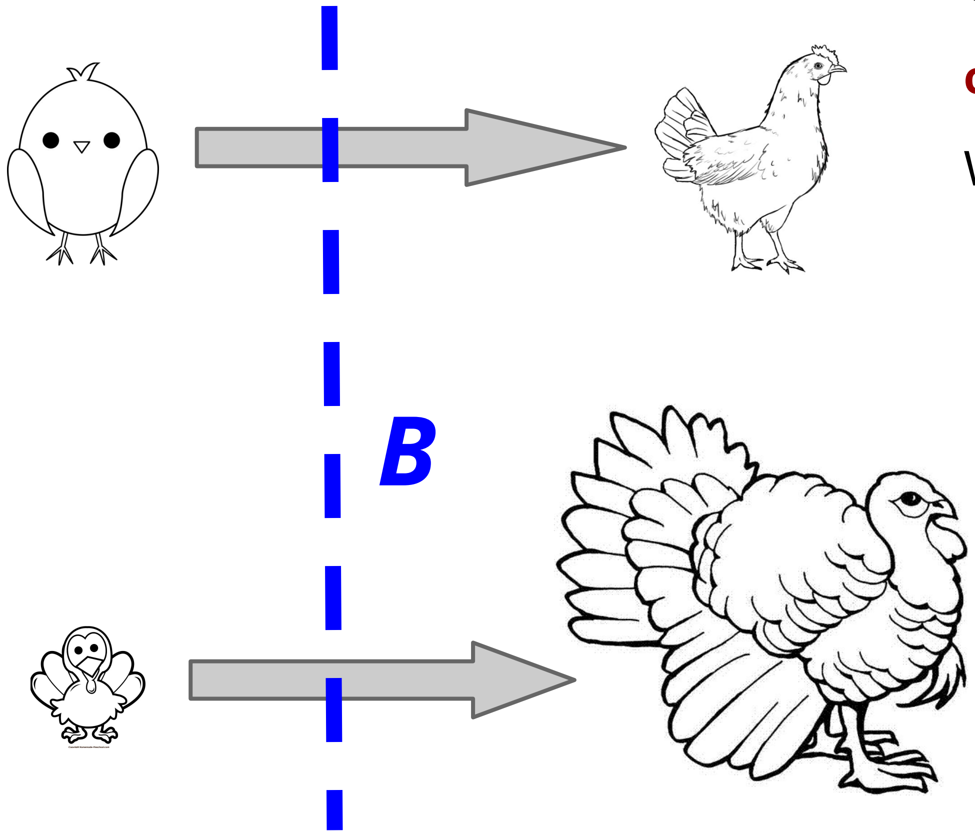
$$f(n) = 1$$



grow fast

grow slowly

The formal mathematical definition of big-Oh



Chicken **grows slower** than turkey, or

chicken size is in O(turkey size).

What it really means:

- Baby chicken might be larger than baby turkey at the beginning.
- But after certain "**breakpoint**", the chicken size will be **surpassed** by the turkey size.
- From the **breakpoint on**, the chicken size will **always** be smaller than the turkey size.

Definition of big-Oh

Function $f(n) = O(g(n))$ ("f is big oh of g") iff

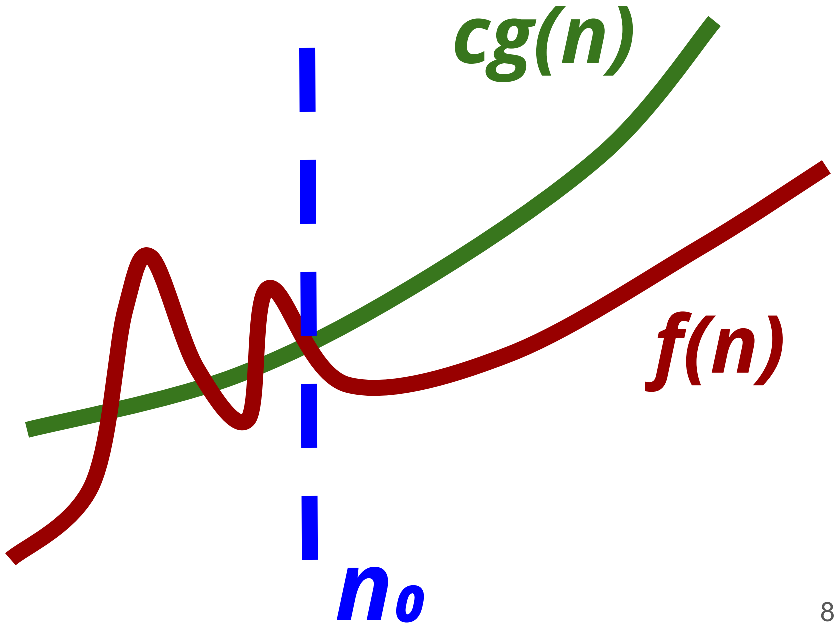
(i) There is some positive $n_0 \in N$

(ii) There is some positive $c \in R$

such that

$\forall n \geq n_0, f(n) \leq cg(n)$

*Beyond the **breakpoint n_0**, **f(n)** is **upper-bounded** by **cg(n)**, where **c** is some wisely chosen constant multiplier.*



Side Note

Both ways below are fine

- $f(n) \in O(g(n))$, i.e., $f(n)$ is **in** $O(g(n))$
- $f(n) = O(g(n))$, i.e., $f(n)$ is $O(g(n))$

Both means the same thing, while the latter is a slight abuse of notation.

Function $f(n) = O(g(n))$ ("f is big oh of g") iff

(i) There is some positive $n_0 \in N$

(ii) There is some positive $c \in R$

such that

$\forall n \geq n_0, f(n) \leq cg(n)$

Knowing the definition,
now we can write proofs for big-Oh.

The key is finding n_0 and c

Example 1

Prove that $100n + 10000$ is in $O(n^2)$

Need to find the n_0 and c such that $100n + 10000$ can be upper-bounded by n^2 multiplied by some c.

under-estimate and simplify

c n^2

Pick c = 10100

$10100n^2$

$100n^2 + 10000n^2$
 # coz n >= 1

$100n^2 + 10000$
 # coz n >= 1

100n + 10000

over-estimate and simplify

large

small

Function $f(n) = O(g(n))$ ("f is big oh of g") iff
(i) There is some positive $n_0 \in N$
(ii) There is some positive $c \in R$
such that
$\forall n \geq n_0, f(n) \leq cg(n)$

Pick $n_0 = 1$

Proof that $100n + 10000$ is in $O(n^2)$

Function $f(n) = O(g(n))$ ("f is big oh of g") iff
(i) There is some positive $n_0 \in N$
(ii) There is some positive $c \in R$
such that
$\forall n \geq n_0, f(n) \leq cg(n)$

Write up the proof

Proof:

Choose n_0=1, c = 10100,

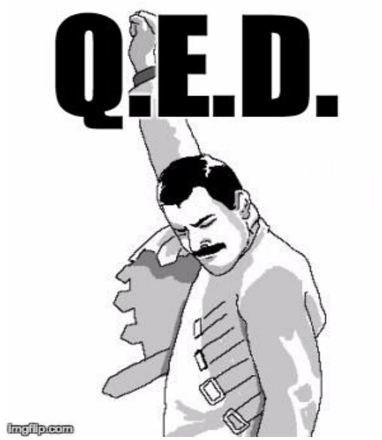
then for all n >= n_0,

$100n + 10000 <= 100n^2 + 10000n^2$ **# because n >= 1**

$= 10100n^2$

$= cn^2$

Therefore by definition of big-Oh, $100n + 10000$ is in $O(n^2)$



Quick note

The choice of n_0 and c is not unique.

There can be many (actually, infinitely many) different combinations of n_0 and c that would make the proof work.

It depends on what inequalities you use while doing the upper/lower-bounding.

Example 2

Prove that $5n^2 - 3n + 20$ is in $O(n^2)$

under-estimate and simplify

$c\,n^2$

Pick c = 25

$25n^2$

$5n^2 + 20n^2$

\# coz n >= 1

$5n^2 + 20$

\# remove a negative term

$5n^2 - 3n + 20$

over-estimate and simplify

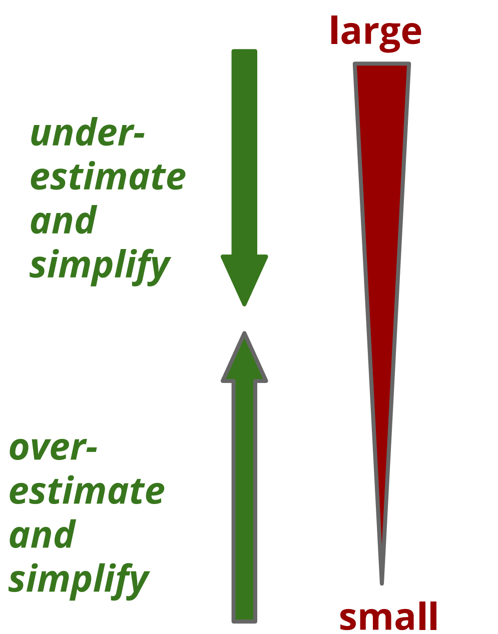
large

small

Pick $n_0 = 1$

Takeaway

choose some breakpoint (**n_0=1 often works**), then we can assume **$n \geq 1$**



under-estimation tricks

→ **remove** a **positive** term

- $3n^2 + 2n \geq 3n^2$

→ **multiply** a **negative** term

- $5n^2 - n \geq 5n^2 - n*n = 4n^2$

over-estimation tricks

→ **remove** a **negative** term

- $3n^2 - 2n \leq 3n^2$

→ **multiply** a **positive** term

- $5n^2 + n \leq 5n^2 + n*n = 6n^2$

After simplification, **choose a *c*** that connects both sides.

The formal mathematical definition of big-**Omega**

Definition of big-Omega

Function $f(n) = \Omega(g(n))$ iff

(i) There is some positive $n_0 \in N$

(ii) There is some positive $c \in R$

such that

$\forall n \geq n_0, \boxed{cg(n) \leq f(n)}$ The only difference

This means that $g(n)$ is a **lower bound** on $f(n)$.

Example 3

Prove that $2n^3 - 7n + 1 = \Omega(n^3)$

Prove that $2n^3 - 7n + 1 = \Omega(n^3)$

under-estimate and simplify

large

$2n^3 - 7n + 1$

$\boxed{n^3 + n^3 - 7n + 1}$

$n^3 + 1$

\# $n^3 - 7n > 0$

\# coz $n >= 3$

Pick $n_0 = 3$

$\boxed{n^3}$

over-estimate and simplify

Pick c = 1

$c\,n^3$

small

Prove that $2n^3 - 7n + 1$ is in $\Omega(n^3)$

Write up the proof

Proof:

Choose $n_0 = 3$, $c = 1$,

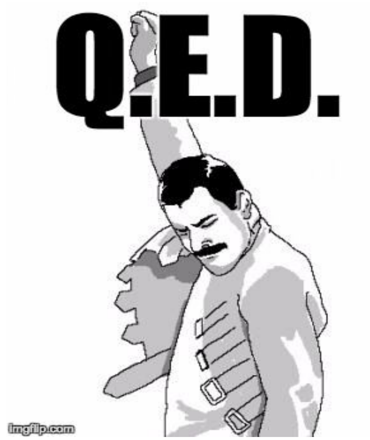
then for all $n >= n_0$,

$2n^3 - 7n + 1 = n^3 + (n^3 - 7n) + 1$

$>= n^3 + 1$ **# because n >= 3**

$>= n^3 = cn^3$

Therefore by definition of big-Omega, $2n^3 - 7n + 1$ is in $\Omega(n^3)$



Takeaway

Additional trick learned

- Splitting a higher order term
- Choose n_0 to however large you need it to be

$$\boxed{n^3 + n^3 - 7n + 1}$$

The formal mathematical definition of big-**Theta**

Definition of big-Theta

Function $f(n) = \Theta(g(n))$ iff $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.
This means that $g(n)$ is a **tight bound** on $f(n)$.

In other words, if you want to prove big-Theta, just prove **both** big-Oh and big-Omega, separately.

Exercise for home

Prove that $2n^3 - 7n + 1 = \Theta(n^3)$

Summary of Asymptotic Notations

- We use **functions** to describe algorithm runtime
  - Number of steps as a function of input size
- Big-Oh/Omega/Theta are used for describing function growth rate
- A proof for big-Oh and big-Omega is basically a chain of inequality.
  - Choose the n_0 and c that makes the chain work.