Asymptotic Analysis

- way to compare speed of two possible solutions to the same problem.

- Basic idea is to think of the rate of growth of the running time of an algorithm.
  - How much slower does the algorithm gets if we double the input size?
  - we try to ~~focus~~ focus on big inputs.

- for asymptotic analysis, we will first define what is the rate of growth of a function. (Big Oh and related defs)

- In other words, asymptotic analysis is a way to describe the efficiency of algorithms by examining how their performance grows as the input becomes very large. (Trying to get a big picture, ignoring small details)

Key concept of asymptotic analysis will be the definition of growth rates, of functions f and g, and how these growth rates compare.

Big O

Def: Let $f, g : \mathbb{R}^{\geq 0} \longrightarrow \mathbb{R}^{\geq 0}$, we say that f grows no faster than g if there exist constant $c > 0$ and $n_0 \geq 0$ s.t

$$\forall n \geq n_0 : f(n) \leq c \cdot g(n)$$

In this case, we write

$$f(n) \text{ is } O(g(n)) \quad \text{or} \quad f(n) = O(g(n))$$

The intuition behind $f(n) = O(g(n))$ is that, for all large enough n, we have $f(n) \leq c \cdot g(n)$

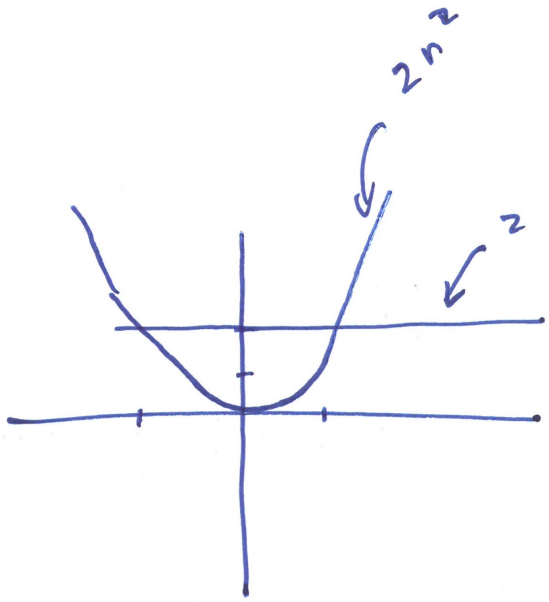
Example: $f(n) = 3n^2 + 2$ is $O(n^2)$

$$f(n) \leq c \cdot n^2$$

Def Big O says

If $f(n) = O(g(n))$, Then

$[\exists c>0, n_0 \geq 0 : \forall n \geq n_0 : f(n) \leq c \cdot g(n)]$



WTS $3n^2 + 2 \leq c \cdot n^2$

$\forall n \geq 1 : 2 \leq 2n^2$

$\forall n \geq 1$

$3n^2 + 2 \leq 3n^2 + 2n^2$

$\forall n \geq 1 : \underbrace{3n^2+2}_{f(n)} \leq 5\underbrace{n^2}_{g(n)}$

Pick $c = 5, n_0 = 1$, then

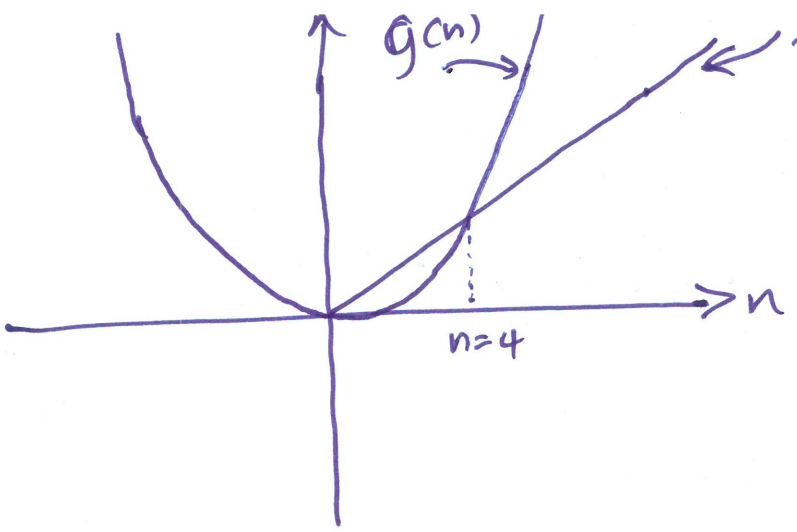
$\forall n \geq 1 : f(n) \leq c \cdot g(n)$

Ex: $f(n) = 4n$

Claim: $f(n) = O(n^2)$ ($f(n) = O(n)$)

we must show a $c>0, n_0 \geq 0$ s.t

$\forall n \geq n_0 : f(n) \leq c \cdot g(n)$



$f(n) = 4n$

$g(n) = n^2$

pick $c = 1$, $n_0 = 4$

$\forall n \geq 4 : 4n \leq 1 \cdot n^2$

Q: is $4n = O(4n^2)$ yes

is $3n^2 + 2 = O(\frac{1}{2} n^2)$

pick $c = 100$

$\forall n \geq 50$ $3n^2 + 2 \leq 100 \cdot \frac{1}{2} \cdot n^2$

Ex: ~~Is~~ Is $n^3 = O(n^2)$ (no)

can we find $c > 0, n_0 \geq 0$ s.t $\forall n \geq n_0 : n^3 \leq c \cdot n^2$ } we cannot

n^3 is not $O(n^2)$ ← How to prove this?

If $n^3 = O(n^2)$, then

$\exists c > 0, n_0 \geq 0 : \forall n \geq n_0 : f(n) \leq c \cdot g(n)$

since this ~~is~~ is incorrect, the negation of this ~~claim~~ claim should be true.

$\neg [\exists c > 0, n_0 \geq 0 : \forall n \geq n_0 : f(n) \leq c \cdot g(n)]$

$\neg [\exists x \in S : P(x)] \equiv [\forall x \in S : \neg P(x)]$ *

$\neg [\forall x \in S : P(x)] \equiv [\exists x \in S : \neg P(x)]$ * } Predicate logic theorems.

$[\forall c > 0, n_0 \geq 0 : \neg [\forall n \geq n_0 : f(n) \leq c \cdot g(n)]$

$[\forall c > 0, n_0 \geq 0 : [\exists n \geq n_0 : \neg (f(n) \leq c \cdot g(n))]$

$[\forall c > 0, n_0 \geq 0 : [\exists n \geq n_0 : f(n) > c \cdot g(n)]$

$\forall c > 0, n_0 \geq 0$: I need to show the existence of n s.t

$$f(n) > c \cdot g(n)$$

$$n^3 > c \cdot n^2$$

What if we pick $n = c+1$

$$(c+1)^3 > c \cdot (c+1)^2$$

$$n = \max(n_0, c+1)$$

When $n = \max(n_0, c+1)$, $\forall c > 0, n_0 \geq 0$, $f(n) > c \cdot g(n)$

$\therefore$ $f(n)$ is not $O(n^2)$

Note that

$f(n) = 4n^3 + 4n + 5$

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

$f(n) = O(n^4)$

$f(n) = O(n^5)$

However, the tightest $O(\cdot)$ function for $f(n)$ is $O(n^3)$

Note: When we write the Big O function for $f(n)$, we want to use the tightest bound, as it is more useful.

common distinct functions.

Name	Example f(n)	generic f(n)	tightest $O(\cdot)$
constant	$f(n) = 10$	$f(n) = c,\ c \in \mathbb{R}^{\geq 0}$	$O(1)$
log	$f(n) = 2\log_{10}(n)$	$c\log_b n,\ b \in \mathbb{R}^{>1},\ c \in \mathbb{R}^{>0}$	$O(\log n)$
linear	$f(n) = 10 \cdot n$	$cn,\ c \in \mathbb{R}^{>0}$	$O(n)$
nlogn	$f(n) = 8n\log_{12}(n)$	$cn\log_b n$	$O(n \log n)$
quadratic	$f(n) = 12n^2$	deg-2 polynomial	$O(n^2)$
cubic	$f(n) = 20n^3$	deg-3 polynomial.	$O(n^3)$
deg-k polynomial		$f(n) = a_1 n^k + a_2 n^{k-1} + \cdots + a_n \cdot 1$	$O(n^k)$
exponential	$f(n) = 3 \cdot 2^n$	2^n	$O(2^n)$
	$f(n) = 5 \cdot 3^n$	3^n	$O(3^n)$
		$\vdots$	

functions later on this list $\neq O$(functions earlier on this list)

$$2^n \neq O(n^{100})$$

$$3^n \neq O(n)$$

- However,

functions earlier $= O$(functions later on the list)

$$n = O(2^n)$$

$$n^{100} = O(2^n)$$

$$n^{100} = O(3^n)$$

Other asymptotic notations.

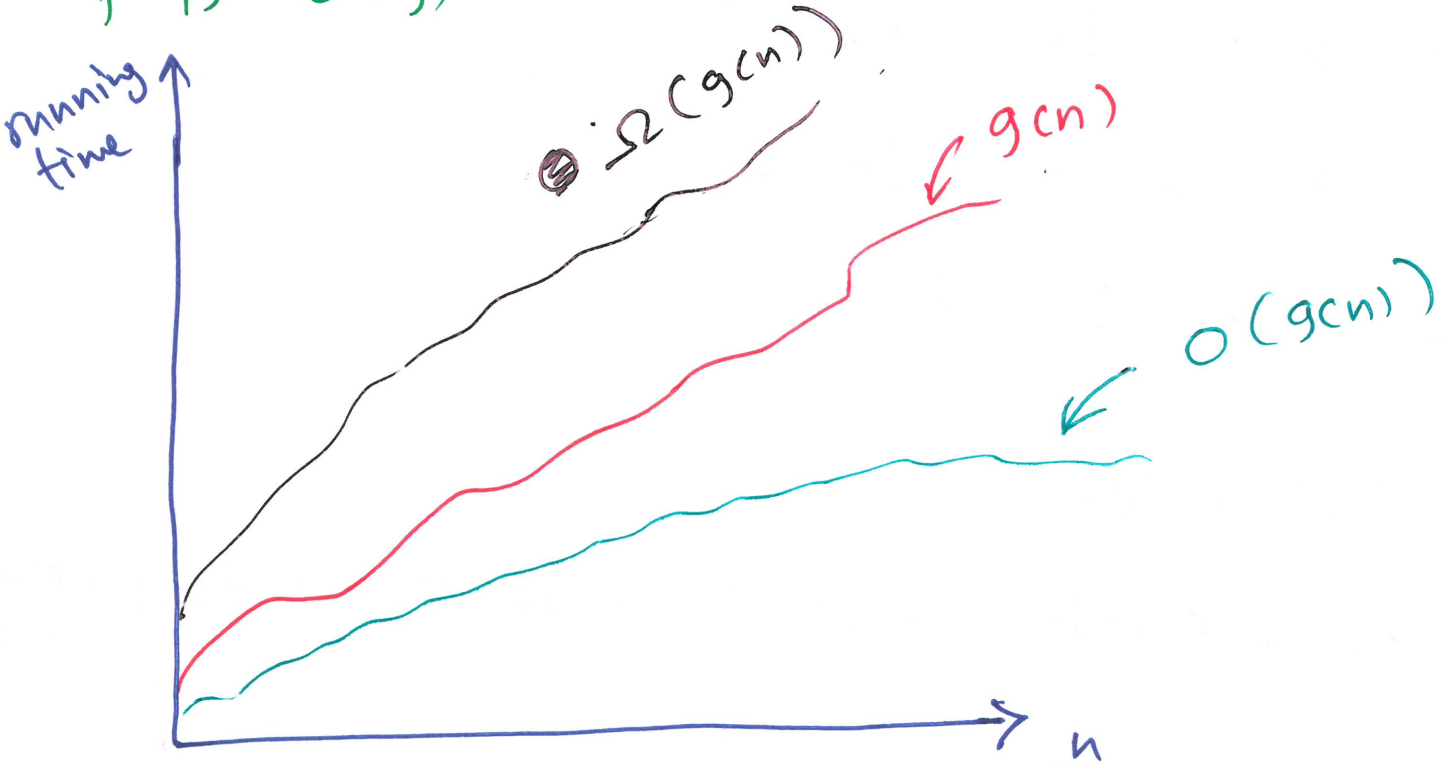
Def Big-Omega (Ω)

Given two functions f and g, we say f grows no slower than g, f is $\Omega(g)$ if,

$\exists d > 0, n_0 \geq 0 : \forall n \geq n_0 : f(n) \geqslant d \cdot g(n)$

Def Big-Theta (Θ)

Given two functions, f and g, we say f is $\Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$



claim! $f(n) = n^3 + 2n + 5$ is $\Omega(n^3)$

WTS: $\exists d > 0, n_0 \geq 0 : \forall n \geq n_0 : f(n) \geq c \cdot n^3$

Pick $d = 1$

$\forall n \geq 0 : n^3 + 5 > n^3$

$\forall n \geq 0 : n^3 + 2n + 5 \geq n^3 + 5 > n^3$

$\forall n \geq 0 : f(n) \geq n^3$

$\forall n \geq 0 : f(n) \geq 1 \cdot n^3$

so pick $d = 1$, $n_0 = 0$

$\forall n \geq n_0 : f(n) \geq 1 \cdot n^3$

$\therefore f(n)$ is $\Omega(n^3)$

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

$f(n) = \Theta(n^3)$ and $f(n) = \Omega(n^3) \Rightarrow f(n) = \Theta(n^3)$

Properties of Big O

lemma 6.2 Asymptotic equivalence of max and sum

we have.

$$f(n) = O(g(n) + h(n)) \iff f(n) = O(\max(h(n), g(n)))$$

Ex: $f(n) = n^2 + n = O(n^2 + n) \iff f(n) = O(\max(n^2, n)) = O(n^2)$

Lemma 6.3 Transitivity of $O(\cdot)$

If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then

$$f(n) = O(h(n))$$

In other words,

$$[f(n) = O(g(n))] \wedge [g(n) = O(h(n))] \Rightarrow [f(n) = O(h(n))]$$

Lemma 6.4

Addition & multiplication preserve $O(\cdot)$-ness

$$[f(n) = O(h_1(n))] \wedge [g(n) = O(h_2(n))] \Rightarrow f(n) + g(n) = O(h_1(n) + h_2(n))$$

and

$$f(n) \cdot g(n) = O(h_1(n) \cdot h_2(n))$$

ex: $f(n) = n^2 + 1$ $h_1(n) = n^3$

$g(n) = n^4 + 2$ $h_2(n) = n^4$

$f(n) = O(n^3)$ $g(n) = O(n^4)$

$f(n) + g(n) = n^3 + n^2 + 1 = O(n^3 + n^4) = O(n^4)$

$f(n) \cdot g(n) = (n^2 + 1)(n^4 + 2) = n^6 + 2n^2 + n^4 + 2 = O(n^3 \cdot n^4) = O(n^7)$

Lemm 6-5

Let $f(n) = \sum_{i=0}^{k} a_i n^i = a_0 n^0 + a_1 n^1 + a_2 n^2 + \cdots\cdots + a_k n^k$

be a deg-k polynomial, then

$$f(n) = O(n^k)$$

To measure runtime of an algorithm

1. find $f(n)$ by counting the # of steps
2. find the simplest $g(n)$ s.t $f(n) = \Theta(g(n))$

— Worst case running time analysis.

Example

```
sum = 0
for i = 1 to n do (1 + inner ops)
    for i = 1 to n do (1 + inner ops)
        for k = 1 to n (1 + inner ops)
            sum = sum + 1   <- 3 ops
return sum
```

$$f(n) = n \cdot (1 + n(1 + n(1 + 3)))$$

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

$$= n + 4n^2 + n^3$$

Ex 2

while $n > 1$ do $(n-1)$ (2 + inner ops)

$n = n - 1$ ← 3 ops

return n.

$f(n) = (n-1)(2+3)$

$= 5n - 1$

ex!

while $n > 1$ do $(\log_2 n)$ times

$n = n/2$ ← 3 ops

$f(n) = 4 \cdot \log n$

What if the running time depends on specific input of size n.

```
linearSearch(A[1 ... n], x):
Input: an array A[1 ... n] and an element x
Output: is x in the (possibly unsorted) array A?
1 for i := 1 to n:
2     if A[i] = x then
3         return True
4 return False
```

(a) Linear Search.

$A = (2, 3, 7, 8)$ $x = 5$ ← worst possible input.

$A = (1, 6, 2, 5)$ $x = 6$

$A = (7, 6, 5, 8)$ $x = 7$ ← Best possible input

All of these arrays are of same length.
However # of steps that we execute are different.

Note!