Transition: DS + Algorithms

Runtime Analysis

Last time: Big-O Proofs

$\sum_{i=1}^{n} i$

Worksheet review:

$$\sum_{i=1}^{n} \left(\sum_{j=1}^{i} 1 \right) = \sum_{i=1}^{n} \left(\underbrace{1 + 1 + \cdots 1}_{j=1 \quad j=2 \quad j=i} \right)$$

$$= \sum_{i=1}^{n} (i)$$

```
for i ← 1 to n
    for j ← 1 to i
        add 1
```

$$= \sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots n$$

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

$$= \Theta(n^2)$$

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2$$

aside

$$\lg 2^n = \log_2 2^n = n$$

def of lg: = exponent we raise 2 to in order $= 2^n$

$$2^k = 2^n$$

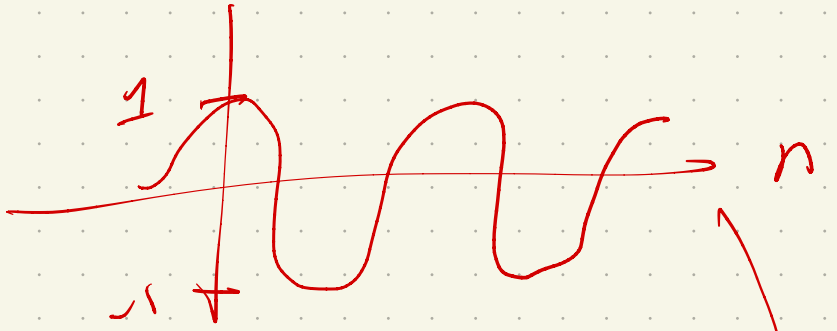
exponents: $x^a \cdot x^b = x^{a+b}$

$$(x^a)^b = x^{ab}$$

Next: $2^{2\lg n}$

$$= (2^{\log_2 n})^2 = n^2 = \Theta\left(\sum_{i=1}^{n}\sum_{j=1}^{n} 1\right)$$

$\cos n + 2$



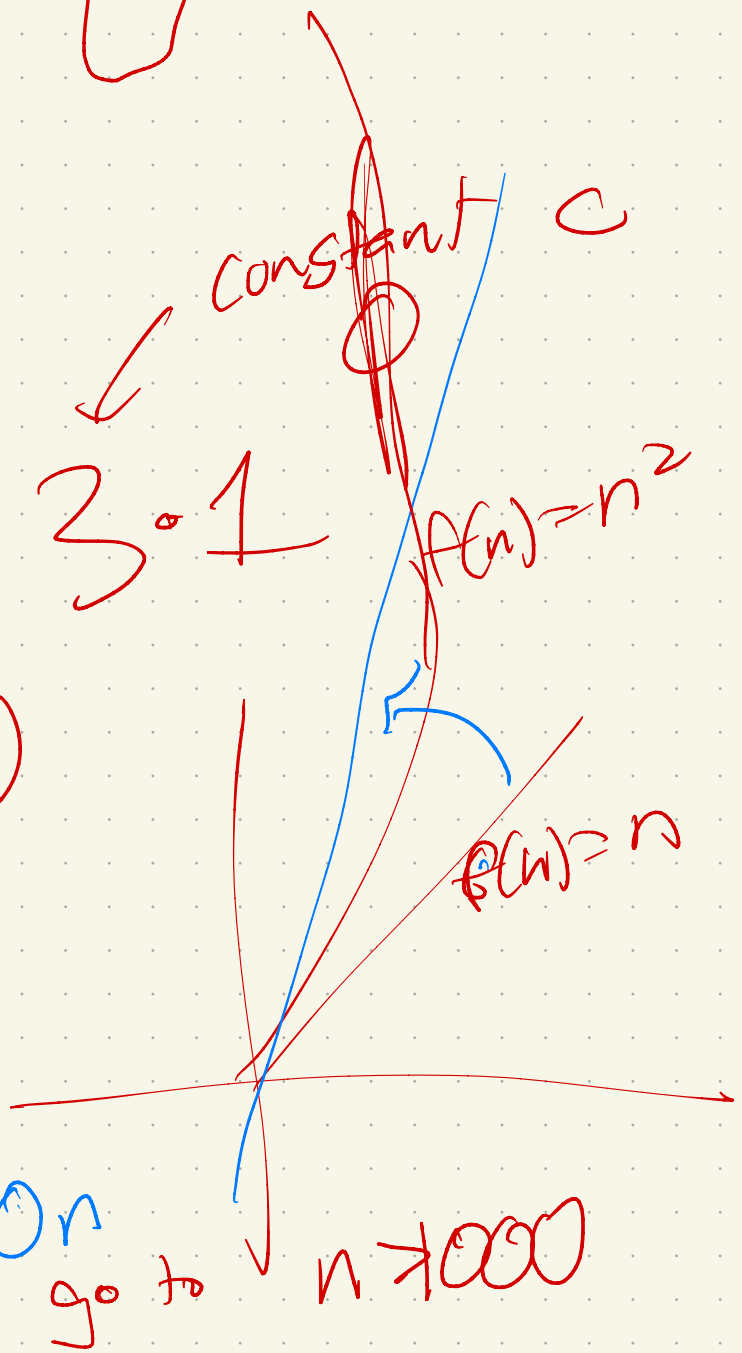
$\cos n \leq 1$

$= O(1)$

$\cos n + 2 \leq 3 = 3 \cdot 1$

$\Rightarrow O(1)$

$\cos n + 2 \neq 0$



Recap Let f and g be functions from $\mathbb{Z}^+$ to $\mathbb{R}^+$

why? → counting things

We say $f(n) = O(g(n))$ if $\exists$ constants c and n_0 s.t. $\forall n > n_0$,

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

Pick a worksheet one:

That limit definition

I did $f(n) = O(g(n))$ if $\exists n_0$ & c s.t.

$\forall n > n_0$, $f(n) \leq c \cdot g(n)$

Take that inequality:

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

$$\Rightarrow \frac{f(n)}{g(n)} \leq c \quad \text{if } n > n_0$$

So $f(n) = O(g(n)) \Leftrightarrow \lim_{n \to \infty} \frac{f(n)}{g(n)} \leq c$

Big Ω

Theta

$\exists c, n_0$ s.t. $\forall n > n_0$, $f(n) \leq c \cdot g(n)$

Let f and g be functions from $\mathbb{Z}^+ \to \mathbb{R}^{\geq}$

We say f is $\Omega(g(n))$ if

$\exists$ positive constants c and n_0

such that $\forall n > n_0$,

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

"f is big-omega of g"

$5n$ is $\Theta(n)$
pick $c = 6$
$5n \leq 6n$

Example: $5 \cdot n$ is $\Omega(n)$:

Pick c & n_0 st. $5n \geq c \cdot n$

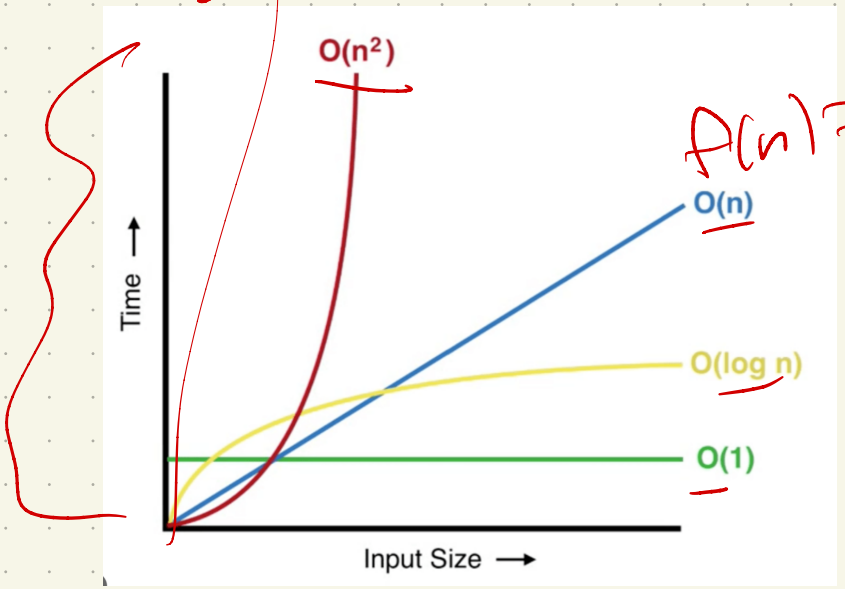
pick $c = 2$, & $\forall n > 1$, $5n \geq 2n$ ✓

Big-Θ

If f is $O(g)$ and $\Omega(g)$ then f is $\Theta(g)$.

(big-O, Theta)

This gives a way to nicely "bucket" function size:



2^n

$f(n) = n$

Ex: S_n is $\Theta(n)$

Take $n \log n$:

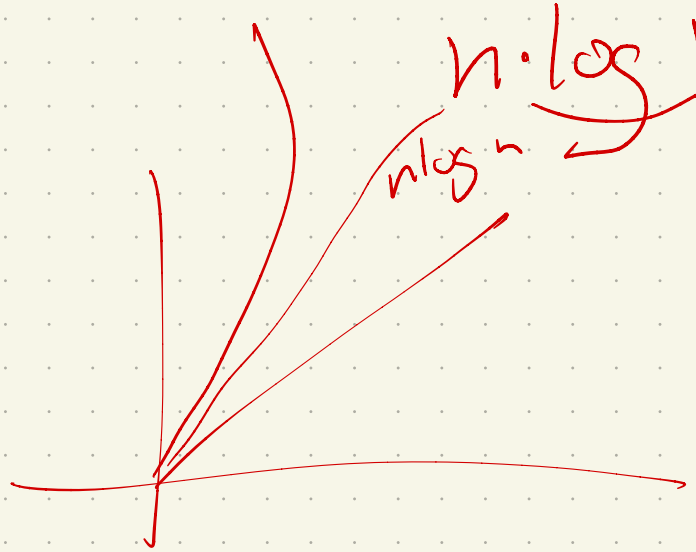
$n \log_2 n$ is $\Omega(n)$:

$\log_2 n \geq 1 \quad \forall \; n \geq 2$

so $n \cdot \log n \geq n \cdot 1$

$n \log_2$ is $O(n^2)$

$n \cdot \log n \leq n \cdot n$



Theorem $3x^2 + 2x - 5$

Let $p(n)$ be a degree k polynomial, so

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$$

for constants a_i and $a_k \neq 0$.

Then $p(n) = \Theta(n^k)$.

Proof:

Big-O:

Idea: $c = \sum a_i$

use this c: $p(n) \leq c \cdot n^k$

Big-Ω:

Choose some fraction "small enough"

$$p(n) \geq (\text{frac})\, n^k$$

$$\underbrace{n^2} - 2n + 5$$

$$\geq c \cdot n^2$$

$$\frac{1}{2} n^2 \quad \text{+ pick } n_0 > 5$$

Using this theorem;

Give big-Θ estimates for:

- $f(x) = \frac{1}{8}x^5 + x^3 + 2$

$= \Theta(x^5)$

- $f(x) = 20000x^2 - 1000000x$

$= \Theta(x^2)$

- $f(x) = \frac{2x^3}{3000} - x + 2$

$= \Theta(x^3)$

Other tricks:

Sometimes $f(n) = \sum_{i=1}^{n} \log_2 n$

How to get Θ-bound?

$$f(n) = \sum_{i=1}^{n} \log_2 n$$

$$= (\underbrace{\log_2 n}_{i=1} + \underbrace{\log_2 n}_{i=2} + \cdots + \underbrace{\log_2 n}_{i=n})$$

$$= n \log_2 n$$

Another: $\log_2 n$ versus $\log_{10}(n)$

What is a log?

exp you raise base to in order to get ~~the~~ inside argument

$x^a \cdot x^b = x^{a+b}$

$\log_2 a + \log_2 b = \log_2 abc$

$\log_2 2^3 + \log_2 2^4 = 3+4 = 7$

$\log_2 2^3 \cdot 2^4 = \log_2 2^7$

Identities:

$\log_2 a^b = b \log_2 a$ $\quad a = (x^a)^b = (x^b)^a$

$\log_b a = \frac{\log_x a}{\log_x b}$

$\log_2 n$ versus $\log_{10} n$

goal: big-O!

$\log n$

$$\log_2 n \overset{\text{identity}}{=} \frac{\log_{10} n}{\log_{10} 2}$$

$$= \left(\frac{1}{\log_{10} 2}\right) \cdot \log_{10} n$$

$$= O(\log_{10} n)$$

$\log_7 n$ and $\log_{13} n$

$\hookrightarrow \log_7 n = \frac{\log_{13} n}{\log_3 7}$

Another: ?

$$\sum_{i=1}^{n} \log_2 i = \log_2 1 + \log_2 2 + \cdots \log_2 n$$

$$= \log_2 (1 \cdot 2 \cdot 3 \cdots n)$$

$$= \log_2 (n!)$$

$$\leq \log_2 n + \log_2 n + \cdots \log_2 n$$

$$\leq n \log_2 n$$

$$= O(n \log n)$$

Next time:

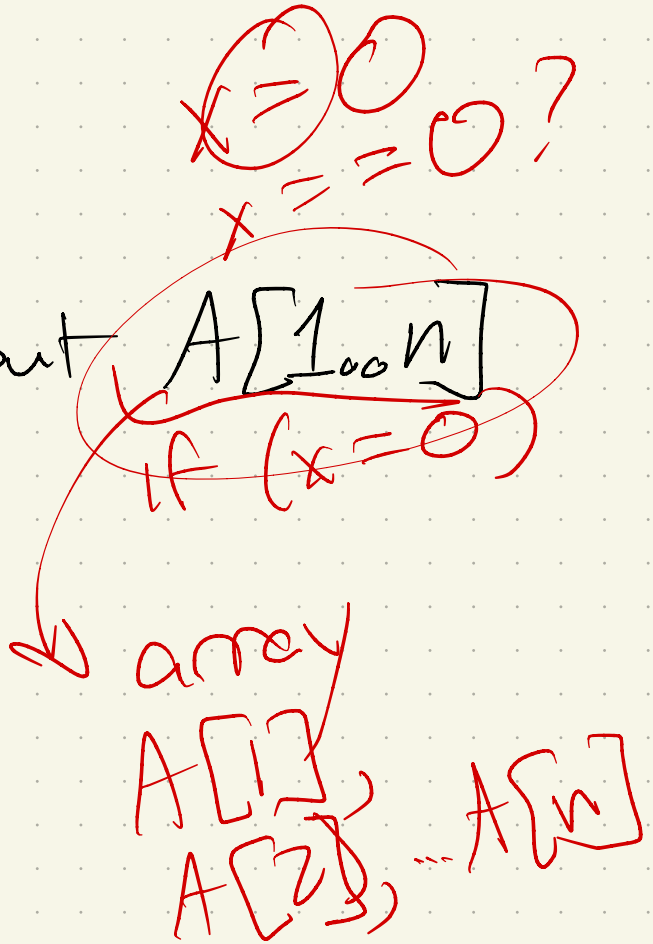
Analyzing algorithms with big-O.

Reading: recaps the definition

↳ then uses big-O

Example: Function with input $A[1 \ldots n]$

```
sum ← 0
For i ← 1 to n
    sum ← sum + A[i]
return sum
```



Runtime: count operations!

Can't count directly - depends on A, how large.

So, use a function: input array $A[1..n]$ $a_1, \ldots, a_n$

```
sum ← 0          ← O(1)
For i ← 1 to n
    sum ← sum + A[i]   ← one + one store: O(1)
return sum
```

End result:

Runtime $T(n) = O(1) + \left(\sum_{i=1}^{n} 2 \right) + 1$ ← 2? 2 operations

$= \Theta(1) + \Theta(n) + O(1) = \Theta(n)$

main:

function(A send in array)

fu