# Transition: DS + Algorithms

Big-O
(really this time!)

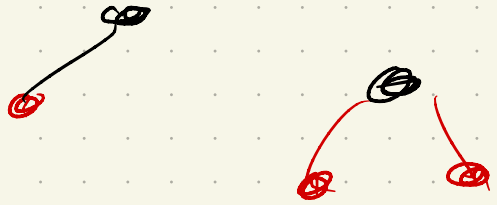
First: Worksheet

How did induction go?

$\forall n$ $P(n-1) \to P(n)$
$P(n) \to P(n+1)$

(1) For any rooted binary tree, can properly 2-color the vertices

By induction $n$ = #vertices



BC: $n=1$ ~~(or 0)~~ ✓

(0)

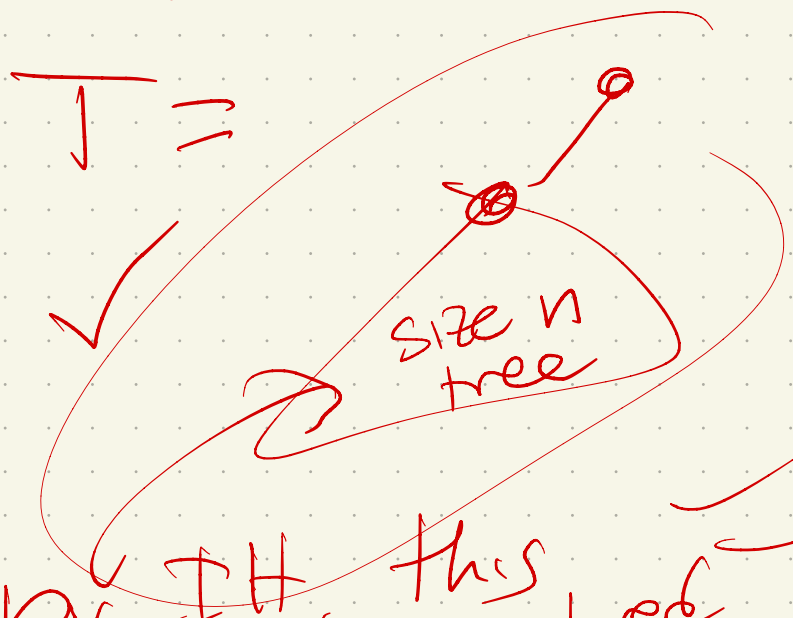
→ IH: Assume binary tree of size $\leq n$ can be properly 2-colored.

IS: Consider $T$, rooted bin tree of size $n+1$.

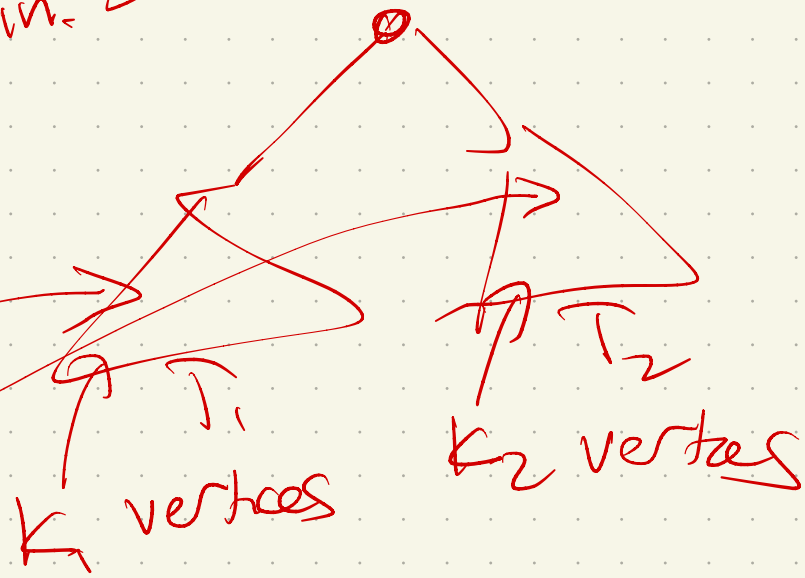
$T \to$ size $n+1 > 1$

↳ What is $T$? either root w/ size n tree, or root w/ 2 bin. subtrees, $n = k_1 + k_2$

$T =$




or



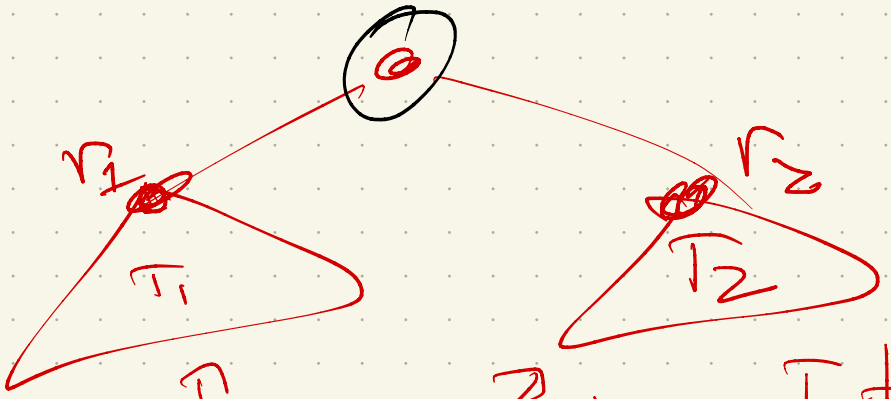

by IH, this can be 2-colored

Need to color all of $T$:

1st case 


← color last vertex the opposite as root of subtree

other case

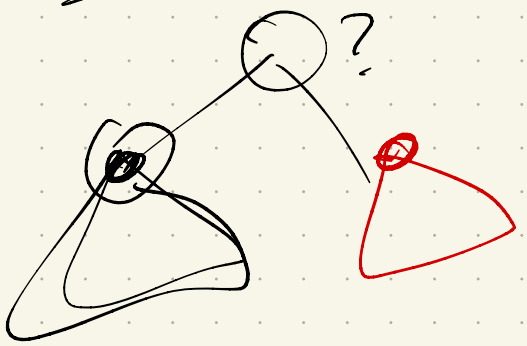



colored by IH

If $r_1$ is same color as $r_2$

↳ done: just color new root the opposite

If $r_1$ & $r_2$ have different colors:




Can't choose a color.

Instead, need to change one of the subtrees

Today: Big-O

The why:

Given all the low level things we can't control, can be hard to perfectly count "# of operations".

Big-O ignores constants, to focus on "long term" behavior.

Being twice as fast is still good!

But, if $10n\log n$ versus $\frac{1}{10} \cdot n^2$:

want $10n\log n$ → faster for large $n$

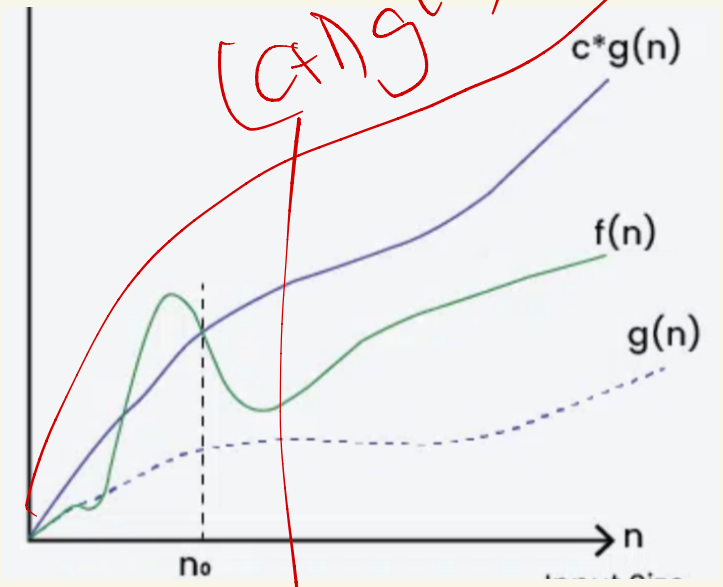
# Big-O

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. $\forall n > n_0$,

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




Note: $O(g(n))$ is a set.

Example

$f(x) = x^2 + 2x + 1$ is $O(x^2)$

proof: Find $c$ and $n_0$, then show inequality holds.

Consider $x^2 + 2x + 1$: if $x \geq 1$,

$$x^2 + 2x + 1 \leq x^2 + 2x^2 + x^2$$

$$= 4x^2$$

Set $n_0 = 1$ and $c = 4$

## General strategy

- Start with $n_0$: pick it so can estimate size of $f(n)$ for $n > n_0$.
  - often based on coefficients
- Then look for a constant $c$ so inequality holds.

**Example**: Show $f(x) = 5x^2 + 25x + 3$ is $O(x^3)$.

$5x^2 + 25x + 3 \leq 5x^2 + 25x^2 + 3x^2$

set $n_0 \geq 1$ ($x \geq 1$)

$\leq 33x^2$

$\leq (33x^2) \cdot x = 33x^3$ ($x \geq 1$)

set $c = 33$, ineq holds

We'll write $f(x) = O(g(x))$.

NOT AN EQUALITY!

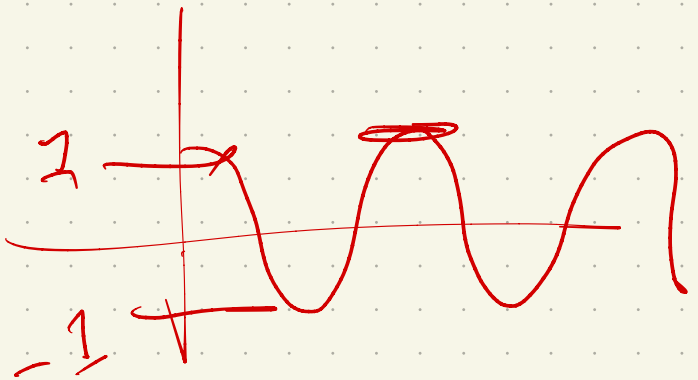
- $x^2 + 2x + 1$ is $O(x^2)$
- $x^2 + 2x + 1$ is $O(x^3)$

Really, means $f(x) \in \{$ functions that are $O(g(x))\}$

Polynomials are usually nice → but can do many functions!

Ex: $\sin(n)$ is $O(1)$.

Fact: $\sin(n) \leq 1$




Ex: $n$ is $O(n \log_2 n)$.

Find $c$ & $n_0$ such that $\forall n > n_0$ $\quad n \leq c \cdot n \log n$

$\log_2 n \to$ always $\geq 1$

if $n > 2$

let $c = 1$

$n \log_2 n \geq n$

$\log_2 n = x$ if $2^x = n$

e.g. $\log_2 2^3 = \log_2 8 = 3$

since $\log_2 n \geq 1$,

**Ex** Show that $n^2$ is **not** $O(n)$

**proof:** harder! need to show

$\neg(\exists c\ \exists n_0$ s.t. $\forall n > n_0,\ n^2 \leq c \cdot n)$ ← def of $O(\ )$

$\Downarrow$

$\forall c$ and $\forall n_0$, $\exists$ some $n > n_0$ with $n^2 > c \cdot n$

So: Fix some $c$ & $n_0$.

Find an $n > n_0$ that violates the ineq.

$n^2 > c \cdot n$ $\quad$ $1^2 > \frac{1}{c}$

↳ take $n$ bigger than $c$

Ⓜ

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:

So $f(n) = O(g(n)) \iff$

# Big $\Omega$

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 $\forall n > n_0$,

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

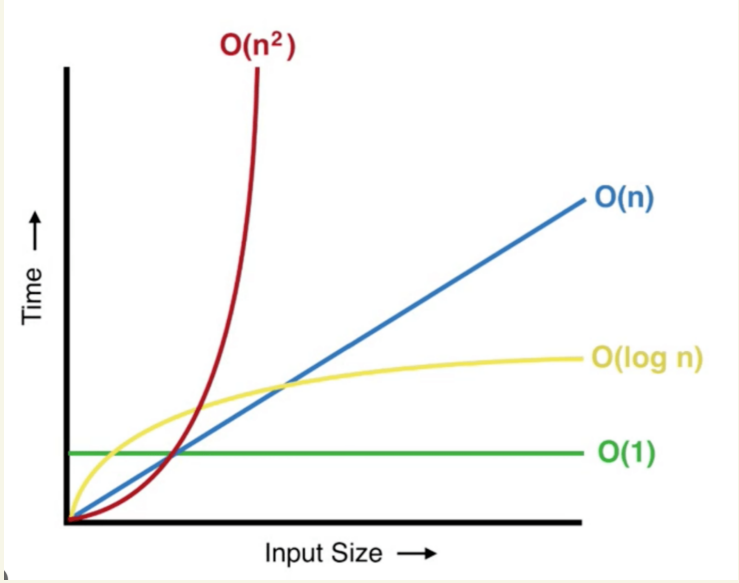
"$f$ is big-omega of $g$"

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

# Big-$\Theta$

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

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



# Theorem

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:

Big-O:

Using this theorem;

Give big-O estimates for:

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

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

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

Other tricks:

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

How to get $\Theta$-bound?

Another: $\log_2 n$ versus $\log_{10} n$.

What is a log?

Identities:

Another:

$$\sum_{i=1}^{n} \log_2 i =$$

Next time:

Analyzing algorithms with big-O.