Big Oh

- running time is measured as a function of the size of the data on which it runs.
  - could be size of input
  - could be size of collected data

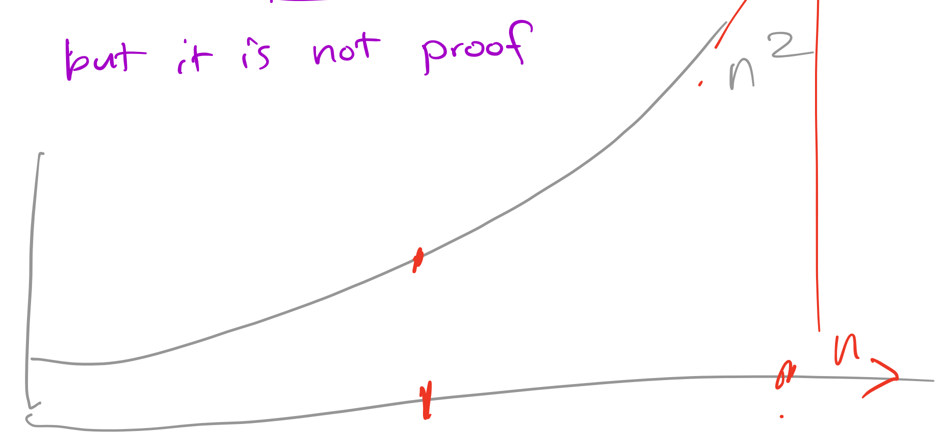
    e.g. database stored in a tree.

- We are interested in Upper Bounds

  like, "We can prove running time grows no more than quadratically with growth in input size"

  eg 1000 entries takes 1 sec
  2000 entries takes 4 sec

  that is consistent with $O(n^2)$
  but it is not proof



"Straight line code"

eg

$$\left.\begin{array}{l} x = x+1; \\ y = x+2; \\ z[6] = x+y; \end{array}\right\} O(1) \left\{\begin{array}{l} x = x^{\wedge}15; \\ y = 188^{\wedge}x; \\ z[6] = x+y; \end{array}\right.$$

We regard these code fragments as being constant time, though they may have very different constants....

Hence straight-line code is treated as if it is constant time...

..unless you use fancy containers eg heap or deque

Loops

A loop takes running time that is the sum of the running time of all its iterations.

```
for (int i=0; i<n; i++) {
    arr[i] = i*i;
}
```

$O(n)$

```
for (int i=0; i<n; i++) {
    for (int j=0; j<m; j++) {
        arr[i][j] = i*j;
    }
}
```

$O(m)$

$O(nm)$

```
for (int i=n; i>0; i--) {
    i = i/2;
}
```

$O(\lg n)$

$= O(\log n)$

Big - O

Def$\underline{n}$. for positive-valued functions $f(n), g(n)$

"Exists"

$f(n) \in O(g(n)) \iff \exists$ constant $c > 0$

and $\exists$ constant $n_0 \geq 1$

such that

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

"For all"

For example...

Claim: $\underbrace{5n}_{f(n)} \in O(\underbrace{n}_{g(n)})$

Proof: $5n \leq 5n, \;\; \forall \, n \geq 1$

$\therefore \;\; c = 5$ and $n_0 = 1$

demonstrate the inequality. $\blacksquare$

Indeed,

Rule#1 : "Removal of Constant factors"

$c \cdot f(n) \in O(f(n)) \quad \forall$ pos $f(n)$
and $c > 0$

Proof: $f(n) \leq \quad f(n) \quad \forall\, n \geq 1$
$\Rightarrow c\, f(n) \leq \quad c\, f(n) \quad \forall\, n \geq 1;$
So using $c = c$ and $n_0 = \underline{1}$
demonstrates the required inequality. $\blacksquare$

$1001\, n^2 \in O(n^2)$

$(n \log n)/40 \in O(n \log n)$

etc.

$\Uparrow$

If you are "allowed" to use the Rules
you can just state these claims, as follows:

$3n^2 \in O(n^2)$ by Removal of Constant Factors
(or "Constant Factors Rule")

Note that $\forall\, c_1, c_2$ both > 0,

$$c_1 f(n) \in O(c_2 f(n)) \text{ and } c_2 f(n) \in O(c_1 \cdot f(n))$$

Note $\lfloor n \rfloor \in O(n)$ and $\lceil n \rceil \in O(n)$.

What you will be asked to do:

1. Prove $5n^2 \in O(n^3)$, using the definition

Claim: $5n^2 \in O(n^3)$

Proof: $5n^2 \le n \cdot n^2 \quad \forall n \geq 5$

$\le n^3 \quad \forall n \geq 5$

$\therefore$ using $c = 1$, $n_0 = 5$, we demonstrate the inequality.

When using the definition to prove big-O, find a c and an n_0 that make the inequality work.

...n of ...tion.

2. Is $n^3 \in O(5n^2)$?

By way contradic

Ans: No. Suppose (BWOC) it were...

ie suppose $\exists\ c > 0,\ n_0 \geq 1$ such that

$$n^3 \leq c \cdot 5n^2 \quad \forall n \geq n_0$$

$$\Rightarrow \quad n \leq 5c \quad \forall n \geq n_0$$

But $n > \max(n_0, 5c)$ contradicts this

$\Rightarrow\Leftarrow$.

$\therefore\ n^3 \notin O(5n^2)$. $\blacksquare$

Similarly, $n \lg n \notin O(n)$.

$n^3 \notin O(n^2)$

$n^4 \notin O(n^3)$

etc.

3. Show $3n^2 - 2n + 6 \in O(n^2)$

Proof: $3n^2 - 2n + 6 \leq 3n^2 + 6 \quad \forall n \geq \underline{1}$

$\leq 3n^2 + 6n^2 \quad \forall n \geq \underline{1}$

$\leq 9n^2 \quad \forall n \geq 1$

$\therefore 3n^2 - 2n + 6 \in O(n^2)$ as demonstrated

by $c = 9$, $n_0 = \underline{1}$. ▨

That's how you use the <u>definition</u> to prove Big O.

4. Show $2n \lg n \in O(n^2)$ using $\text{def}^{\underline{n}}$.

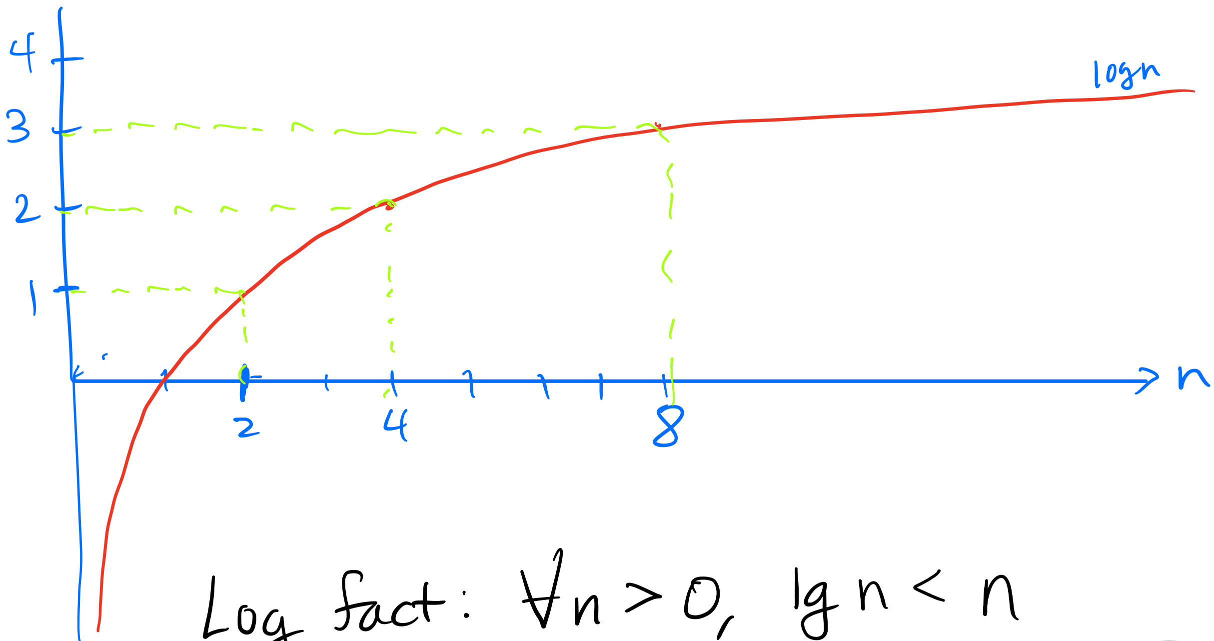
Proof: $2n \lg n \leq 2n^2 \quad \forall n \geq$ ____

$\therefore 2n \lg n \in O(n^2)$ as demonstrated by

$c = 2$ and $n_0 =$ ____. ▨

Reminder: if $x \leq y$ and both are positive

then $x \cdot f(n) \leq y \cdot f(n)$ $\forall$ positive-valued f



Log fact: $\forall n > 0$, $\lg n < n$

Rule #2: $\lg n \in O(n)$, [but $\lg n \notin O(n)$]

Rule#3: "Transitivity of Big-O"

If $f(n) \in O(g(n))$ and $g(n) \in O(h(n))$

then $f(n) \in O(h(n))$.

Proof: $\mathbb{S}$ $f(n) \in O(g(n))^{①}$ and $g(n) \in O(h(n))^{②}$.

From ①, $\exists$ c_1, n_1 such that $f(n) \leq c_1 \cdot g(n)$ $\forall n \geq n_1$

From ②, $\exists$ c_2, n_2 s.t. $g(n) \leq c_2 \cdot h(n)$ $\forall n \geq n_2$.

$\therefore$ $f(n) \leq c_1 \cdot (c_2 \cdot h(n))$, $\forall n \geq$ ______.

$\boxtimes$

Aside: $f(n) \in O(g(n)) \Longleftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} \leq$

for pos-valued functions $f(n), g(n)$.

You should now be able to do the

Self-test (HW) on the Week-by-Week

AND...

1. Using only the definition of Big-O and "$\lg n < n \ \forall n > 0$", show that

$$\lg^2 n \in O(n^2)$$

Recall $\log_b^c a$ means $(\log_b a)^c$

Log facts that will be useful:

$\lg n = \log_2 n$

$\lg n \leq n \quad \forall n > 0$

$\lg_b^c a = (\lg_b a)^c$

$2^{\lg n} = n, \quad \lg(2^n) = n.$

$\log_a n^b$ means $\log_a(n^b)$

$\log_a n^b = b \log_a n$

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

$$\log_b a = \frac{1}{\log_a b}$$

$a^{\log_b c} = c^{\log_b a}$

$\log_c(a * b) = \log_c a + \log_c b$

$$\log_4 5 = 1.161$$

$$\log_5 4 = 0.861$$

$$\log_4 3 = 0.792$$

$$\log_3 4 = 1.262$$