Big O

- 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

```
x = x+1;
y = x+2;
Z[6] = x+y;
```

O(1)

```
x = x^15;
y = (400 x^2 + 196 x)/2.2;
Z[6] = x+y;
```

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;
}
```

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

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

Big-O

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

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

and $\exists$ constant $n_0 \geq 0$

such that

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

For example...

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

Proof: $5n \leq \_\_\ n$, $\forall\ n \geq \_\_$

$\therefore$ $c =$ and $n_0 =$

demonstrate the inequality ▨

Indeed,

Rule #1 : "Removal of Constant factors"

$K \cdot f(n) \in O(f(n))$ $\forall$ positive K

Proof: $K \cdot f(n) \leq \underline{\quad}_{c} \cdot f(n)$ $\forall\ n \geq \underline{\quad}_{n_0}$

$\therefore$ $c = \underline{\quad}$ and $n_0 = \underline{\quad}$

demonstrates the required inequality. ▨

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

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

etc.

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,

$c_1 f(n) \in O(c_2 f(n))$ 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: $5n^2 \leq$ $\forall n \geq$

$\leq$. $\forall n \geq$

$\therefore$ using $c =$ __, $n_0 =$ __, 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 of Contradiction

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

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

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

$\Rightarrow$

But

$\therefore\ n^3 \notin O(5n^2)$. ▨

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: $3n^2 - 2n + 6 \leq$ $\forall n \geq$ ___

That's how you use the definition to prove Big O.

4. Show $2 n \lg n \in O(n^2)$ using def^n.

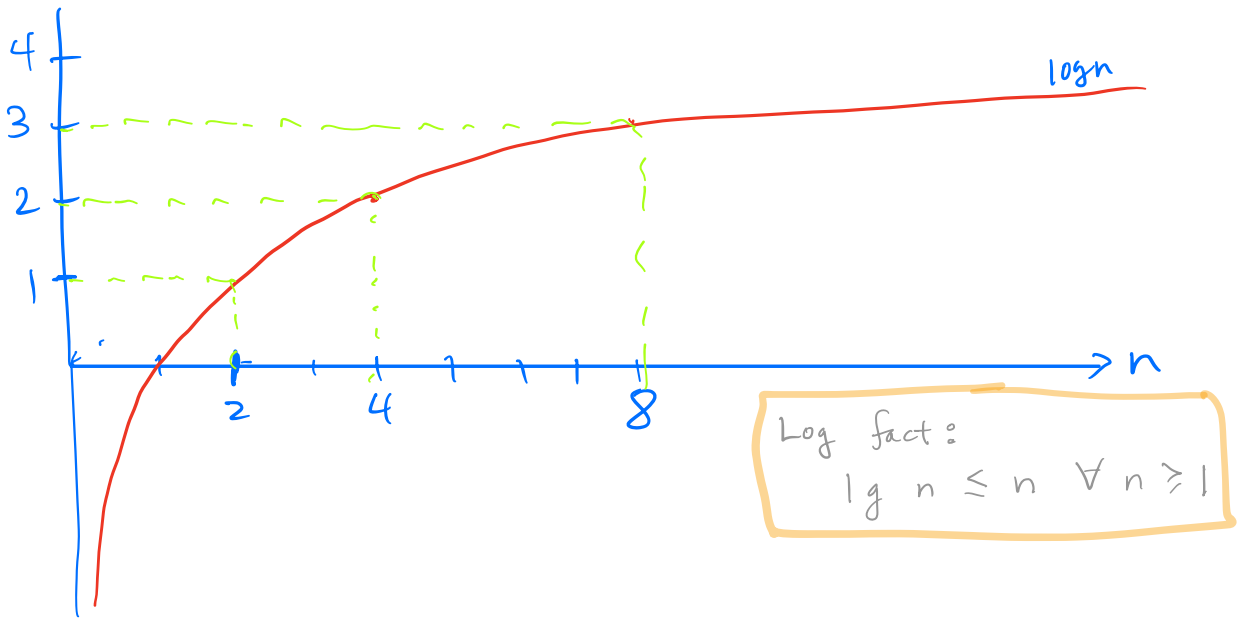
Proof: $2n \lg n \leq 2n^2$ $\forall n \geq$ ___

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

$c =$ ___ 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



Rule #2: $\lg n \in O(n)$

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: $\because$ $f(n) \in O(g(n))$ ① and $g(n) \in O(h(n))$ ②.

From ①

From ②.

$\therefore$

$\forall n \geq$ _____.

▨

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

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 "$\lg n < n \quad \forall n > 0$", 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 $\log_a (n^b)$

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

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

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

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

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

$\log_4 5 = 1.161$

$\log_5 4 = 0,861$

$\log_4 3 = 0.792$

$\log_3 4 = 1.262$