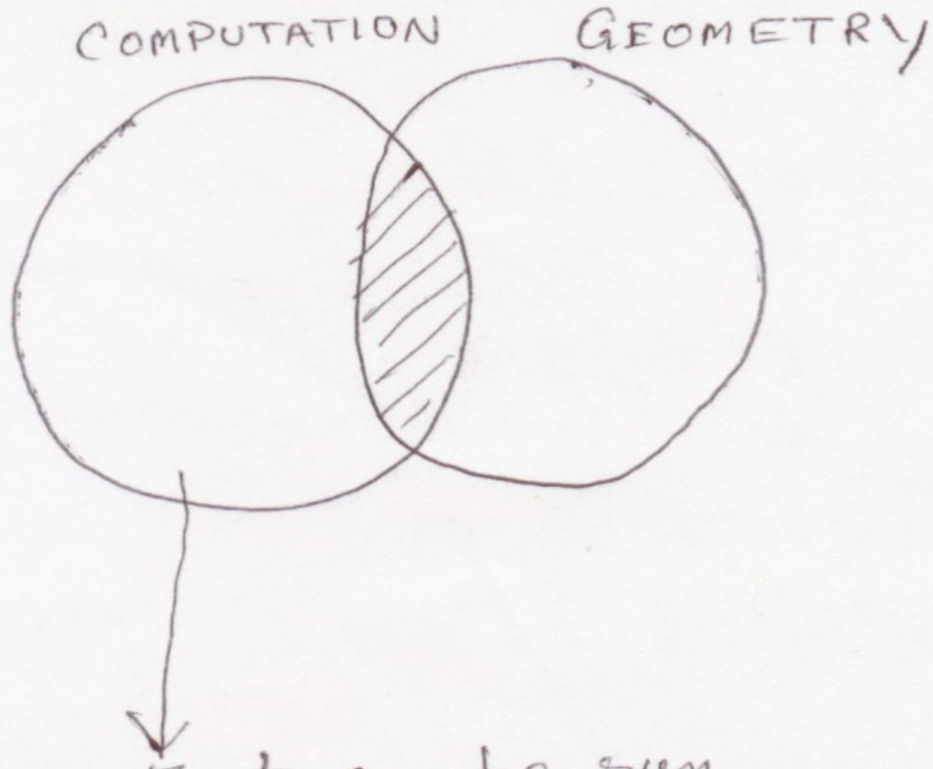
COMPUTATIONAL GEOMETRY
LECTURE #0: THE BASICS

OBSERVATION 1:



Procedure(s) that can be run on a COMPUTER or any abstract model of a computes.

Such as

- Turing Machines
- Post machines
- Normal Markov Algorithms

⋮

- We shall term such procedures as Algorithms

Computational geometry involves design, study, and analysis of efficient algorithms for geometric reasoning.

Key points: The geometry is typically Euclidean

Question: what is efficient algorithms ?

An Algorithm:

> A Computational procedure that ~~that~~ takes some value ~~or~~ or set of values as *input* and produces some values or set of values as *output*.

INPUT → [Algorithm] → output

- How do I think about efficiency of algorithms.

- Lets play a game

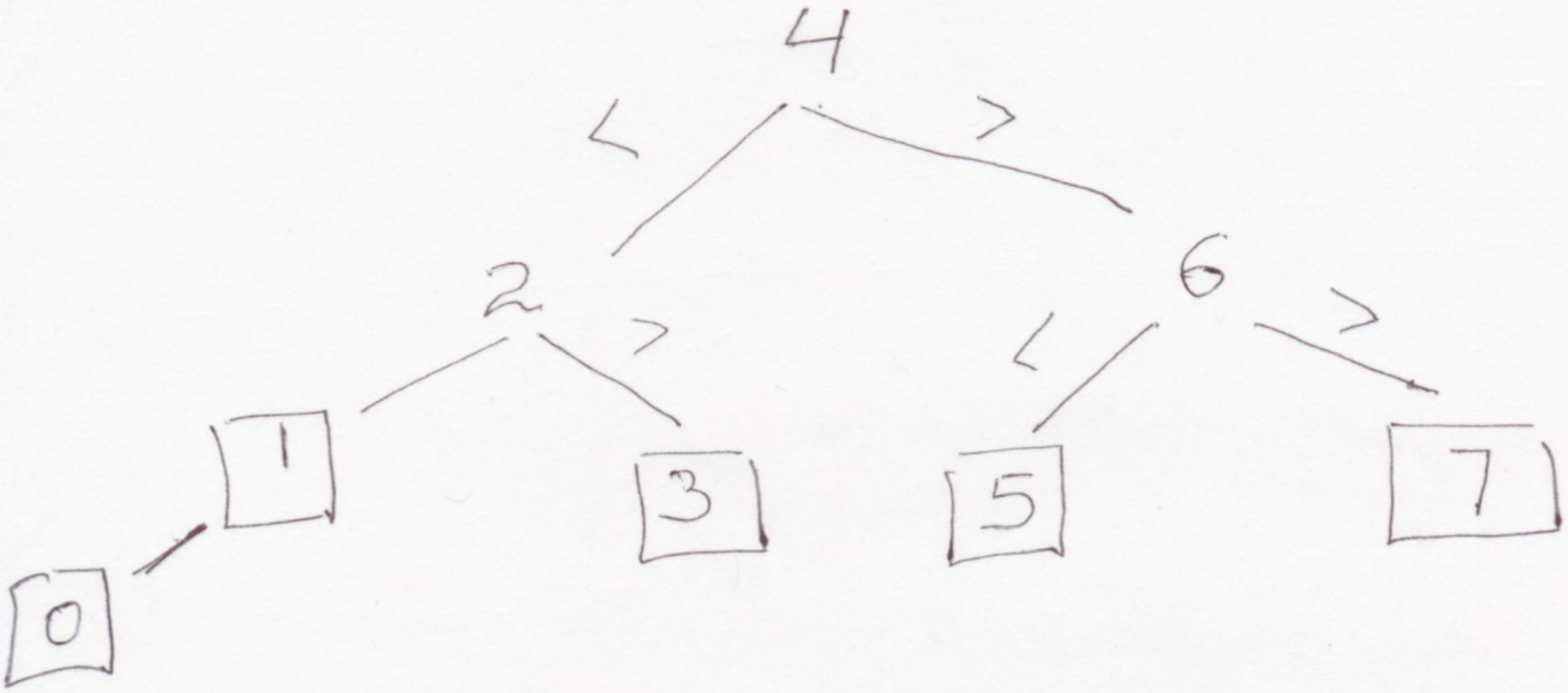
  I will guess a number between [0, n]
  e.g (n = 4, 7). Tell me what it is.
  For each of your guesses I will tell you if it is
  < or > or ✓

Strategy 1 : Enumerative

Your guess	① 0	② 1	③ 2	④ 3	⑤ 4	} Good enough?
my answer	✗	>	>	>	✓	

0	1	2	3	4	5	6	7
>	>	>	>	>	>	>	✓

Another strategy (2)



what is the worst I can do with

strategy 1 $\Rightarrow$ $\begin{cases} 4/4 \\ 8/8 \\ \vdots \\ n/n \end{cases}$ or in general n

strategy2 $\Rightarrow$ $3/8 = \log_2 8$

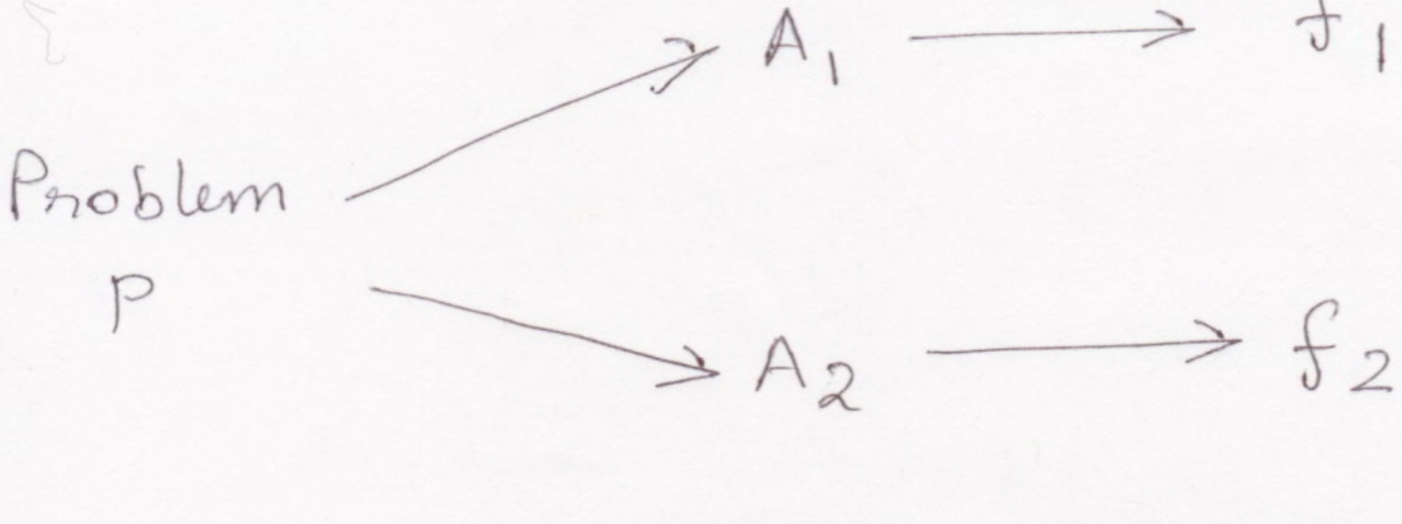
or in general $\underline{\lg n}$

n	2	4	8	16	32	64	128	256	512	1024
$\lg n$	1	2	3	4	5	6	7	8	9	10

- $\lg n$ grows much more slowly than n.

Key point

- The running time of an algorithm is a function of the problem size



Assuming A_1 and A_2 are both correct, we compare them by comparing f_1 and f_2

- what could be other ways of comparing them?
  - Amount of memory they take
  - How easy is it to implement them.
  - Is the implementation available publicly

⋮

What could be the possible relations between two algorithms (in this sense)?

1 A_1 takes about the same time as A_2

$f_1 \approx f_2$: The Θ notation
$f_1(n) = \Theta(f_2(n))$

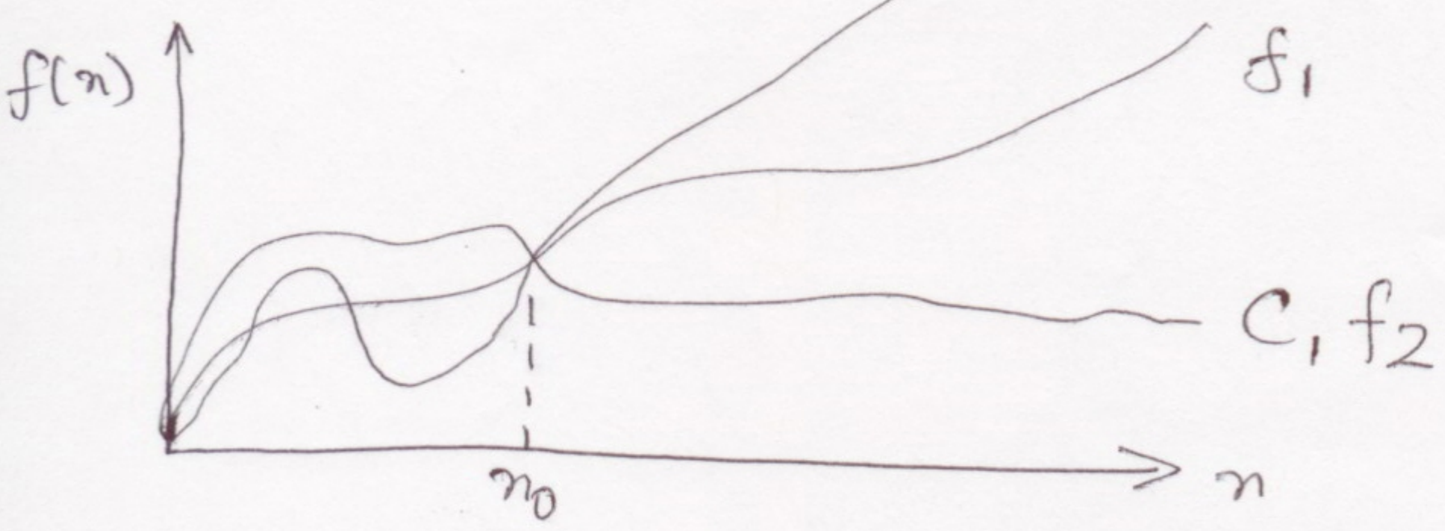
2. A_1 is much faster than A_2

$f_1 < f_2$ The O notation
$f_1(n) = O(f_2(n))$

3. A_1 is much slower than A_2

$f_1 > f_2$ The Ω notation
$f_1(n) = \Omega(f_2(n))$

1. $f_1 \approx f_2$

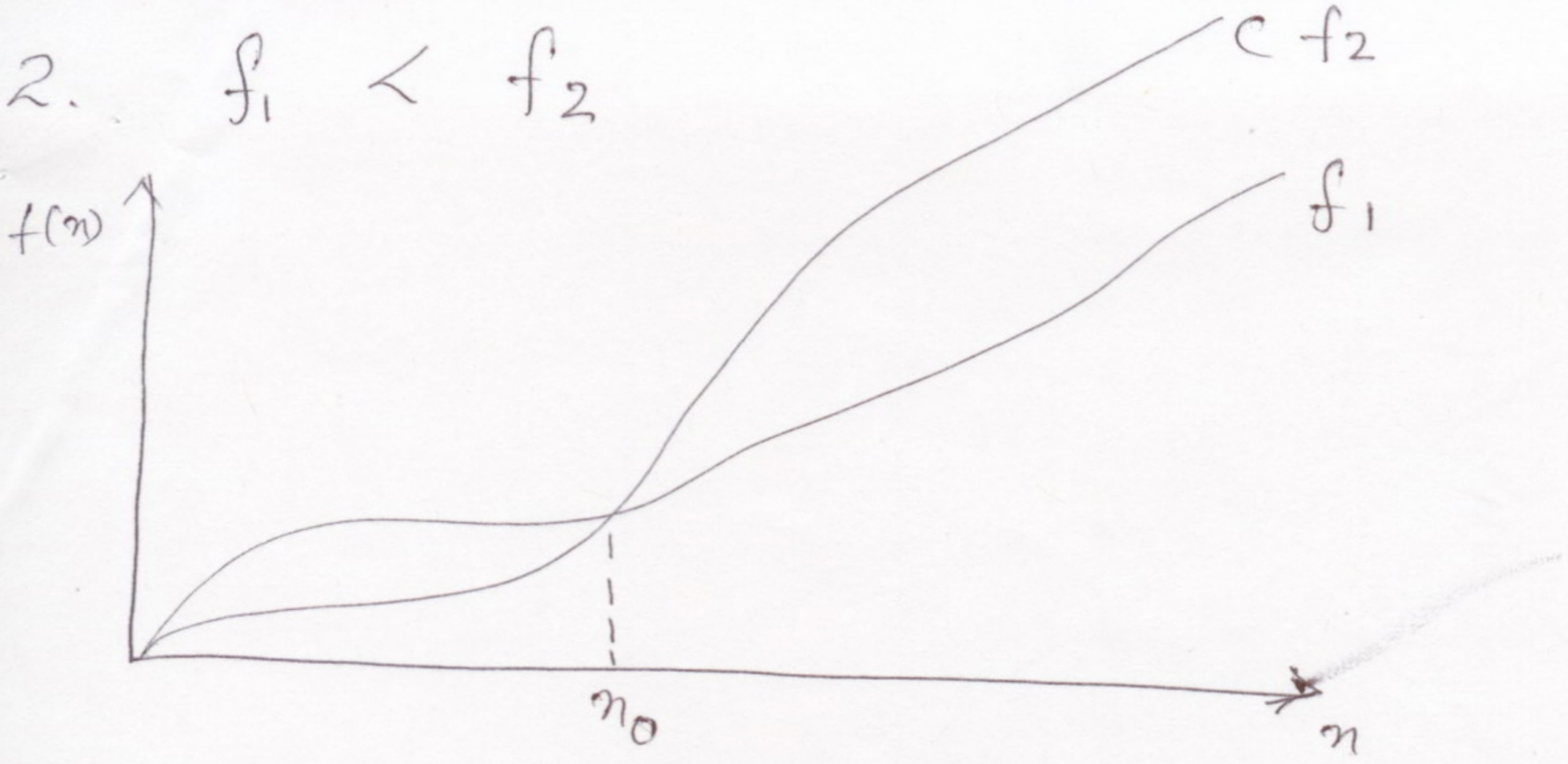


$f_1(n) = \Theta(f_2(n))$ if $\exists\ c_1, c_2 > 0$ (constants)
and $n_0 > 0$: $\forall\ n > n_0$

$$c_1 f_2(n) \leq f_1(n) \leq c_2(f_2(n))$$

- f_1 is "sandwiched" by f_2 for som n_0, c_1, c_2

2. $f_1 < f_2$

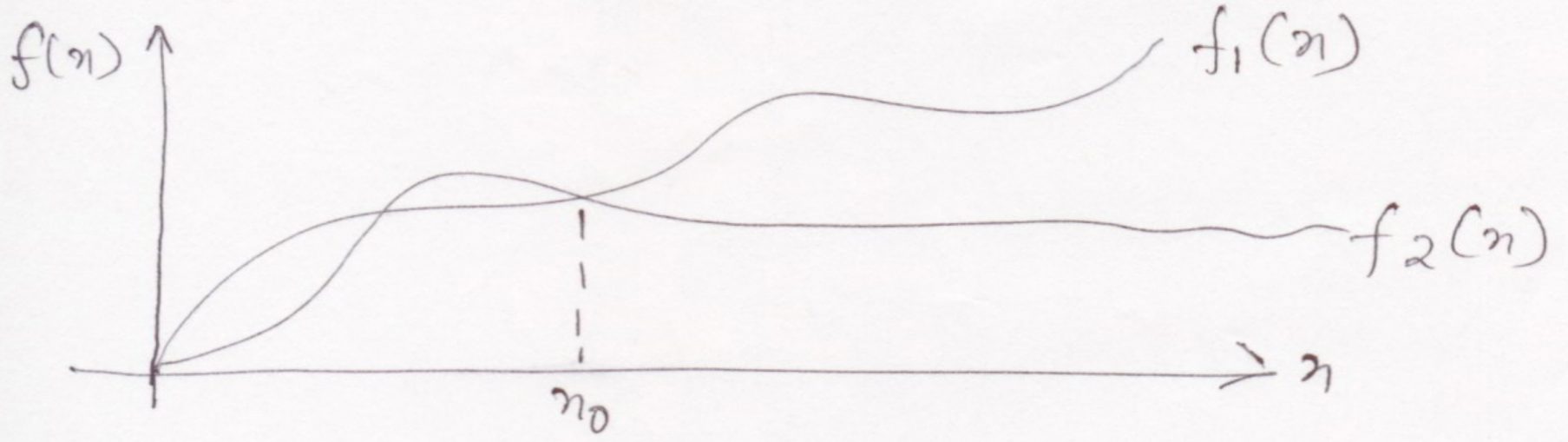


$f_1(n) = O(f_2(n))$ if $\exists$ constant c and n_0:

$\forall n > n_0 \quad c \cdot f_2(n) \geq f_1(n) \geq 0$.

– Asymptotic upper bound.

3. $f_1 > f_2$



$f_1(n) = \Omega(f_2(n))$ if $\exists\ c, n_0 > 0$:

$\forall n > n_0, \quad 0 \leq c \cdot f_2(n) \leq f_1(n)$

Example

Let's show

$$\frac{1}{2}n^2 - 3n = \Theta(n^2)$$

- we need to determine $c_1, c_2, n_0 > 0$:

$$c_1 \cdot n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2 \cdot n^2 \quad \forall\ n > n_0$$

$$\Downarrow$$

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2 \quad \forall\ n > n_0$$

$\Downarrow$ ①

$$\frac{1}{2} - \frac{3}{n} \leq c_2 \quad \forall\ n > n_0$$

for any $n \geq 1$ this is true for $c_2 \geq 1/2$

②

for $n \geq 7$ this is true for $c_1 \leq 1/14$

∴ By choosing $c_1 = 1/14$, $c_2 = 1/2$ and $n_0 = 7$ we can verify $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

Note: other choices of constants exist. But the key is that some choice exists!

Some more Examples

$10n^2 + 20n + 100 = \Theta(n^2)$

$5n^3 + 10000n^2 + 10000000n = \Theta(n^3)$

$\lg n + 5 = \Theta(\lg n)$

$n + \lg(n) = \Theta(n)$

Rule: Drop the lower order terms

Ignore the leading constant of the highest order term.

Properties

1. $\forall f(n)$ and $g(n)$

$f(n) = \Theta(g(n))$ iff

$f(n) = O(g(n))$ AND

$f(n) = \Omega(g(n))$

2. $f(n) = \Theta(g(n))$ and $g(n) = \Theta(h(n)) \Rightarrow$

$f(n) = \Theta(h(n))$ - Transitivity.

3. $f(n) = O(g(n))$ and $g(n) = O(h(n)) \Rightarrow$

$f(n) = O(h(n))$ - transitivity.

4. $f(n) = \Omega(g(n))$ and $g(n) = \Omega(h(n)) \Rightarrow$

$f(n) = \Omega(h(n))$ - transitivity.

Reflexivity

5. $f(n) = \Theta(f(n))$

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

$f(n) = \Omega(f(n))$

Symmetry

6. $f(n) = \Theta(g(n))$ iff

$g(n) = \Theta(f(n))$

Transpose Symmetry.

7. $f(n) = O(g(n))$ iff $g(n) = \Omega(f(n))$

Question

Are all functions asymptotically comparable?
(elementary functions)

ie. is it possible for two functions $f(n)$ and $g(n)$

neither $f(n) = O(g(n))$ nor

$f(n) = \Omega(g(n))$ hold?

eg. $f(n) = n$

$g(n) = n^{1+\sin n}$

Are not asymptotically comparable since $1+\sin n$ oscillates between 0 and 2 taking all intermediate values.

Some more Questions.

TRUE or False

1. $3^n = \Theta(2^n)$

False.

2. $(n + \lg n)^5 = \Theta(n^5)$

True

3. $n^{\lg_2(\lg n)} = O(\lg^2 n)^{\lg n}$

True.

Why.

we know from basic math.

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

LHS

$$n^{\lg_2(\lg n)} = \lg n^{\lg_2 n}$$

RHS

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

$$n^{\lg_2(\lg n)} = O(\lg^2 n)^{\lg n}.$$

4. $n! = O(2^n)$

False

why.

By Stirling's inequality

$$n! = \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \cdot \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

$$\lg(n!) = \lg_2 \sqrt{2\pi n} + n \cdot \lg n - \lg_2 e + \lg(\cdot)$$

$$= \Theta(n \lg n)$$

$$\lg 2^n = n.$$

$n \lg n = O(n)$? . Hence the proof.

<u>what grows faster</u>

n.	$n!$	2^n
1	1	2
2	2	4
3	6	8
4	24	16
5	120	32
	⋮	

5. $\sqrt{n} = \Omega(\lg n)$. True.

Why?

n	$\sqrt{n}$	$\lg n$
1	1	0
2	1.4	1
4	2	2
16	4	4
25	5	4. something.
36	6	5. something

6. $4^{\lg_2 n} = \Theta(n^2)$

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

$4^{\lg_2 n} = n^{\lg_2 4} = n^2 = \Theta(n^2)$

True.