Result $f_1(n)$ is $O(g_1(n))$ and $f_2(n)$ is $O(g_2(n))$. Then,

i) $(f_1+f_2)(n)$ is $O(\max\{g_1(n), g_2(n)\})$ for all n.

$$\rightarrow (f_1+f_2)(n) \leq c_1 g_1(n) + c_2 g_2(n)$$

H.W

$$\leq c_3\, g_3(n) \qquad g_3 = \max\{g_1, g_2\}$$

ii) $(f_1 \cdot f_2)(n)$ is $O(g_1(n) \cdot g_2(n))$

H.W.

$\underline{\underline{Ex^n}}$ Give the Big oh estimate of the function $f(n)$ where, $f(n) = \underbrace{(n^2+2n+5)\log(n!)} + n\log n + \underbrace{(5n^2+\log n)(n^2+2)}$

$(n^2+2n+5)(n\log n) \approx \underbrace{n^2 n\log n}_{n^3\log n} + 2n.n\log n + 5n\log n.$

$= n^2\, n\log n$

$= n^3\log n.$

$5n^4$

$5n^4$

$\boxed{O(n^4)}$

$\underline{\underline{Ex^m}}$ Give Big oh estimate of f(n) where,

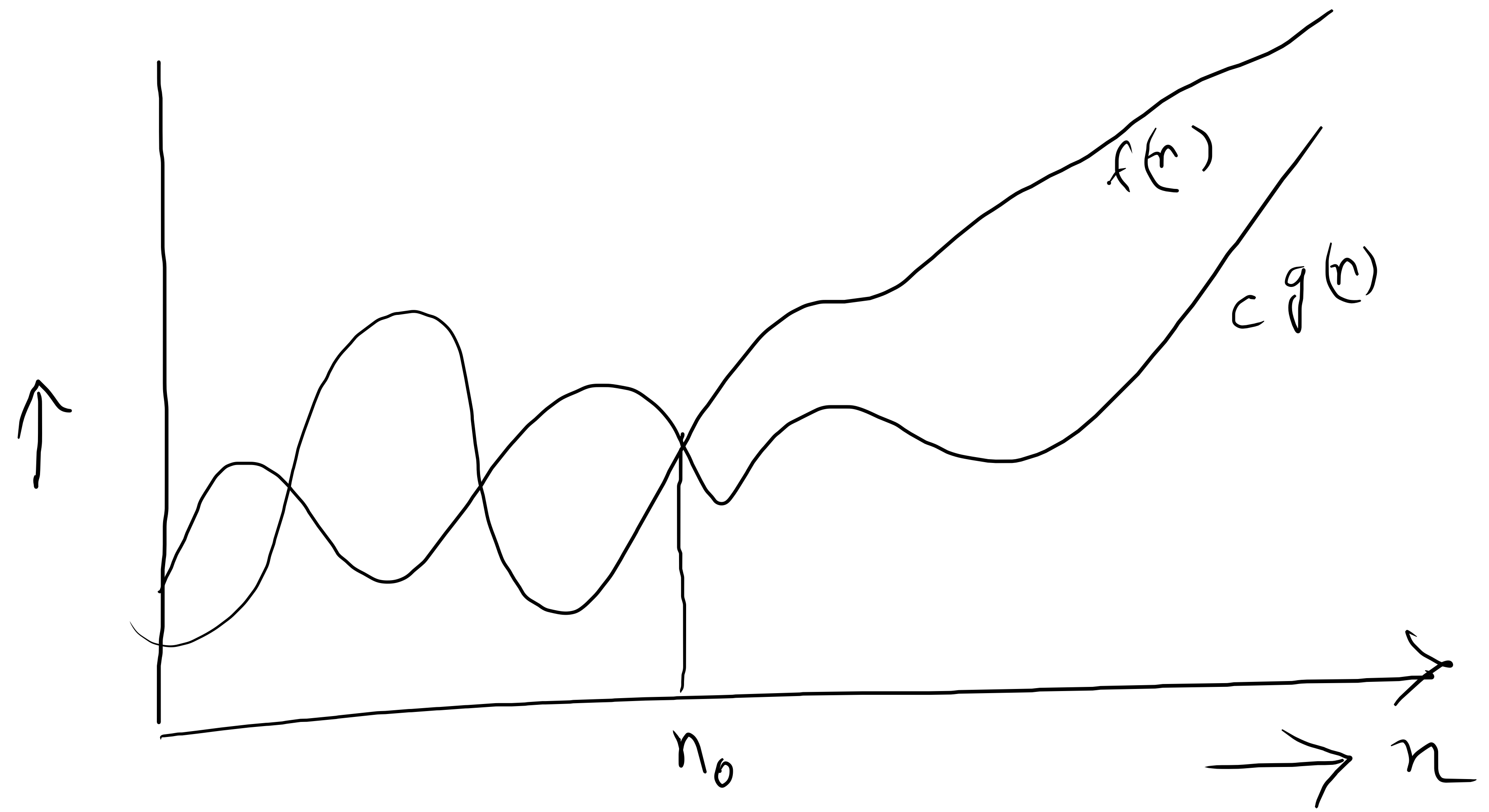
$$f(n) = n\log n \log\log n + (2n+5)\log n! + (3n^2+2n+9)\log n$$

$$f(n) = O(n^2 \log n)$$

H.W Practice

Big omega 'Ω' notation (lower bound)

$f(n) = \Omega(g(n))$ if there exists constants $c > 0$ and $n_0 > 0$ such that $0 \le c\,g(n) \le f(n) \quad \forall\, n \geq n_0$



$\underline{\underline{Ex^m}}$ Is $2n^2+5n$ is $\Omega(n^2)$

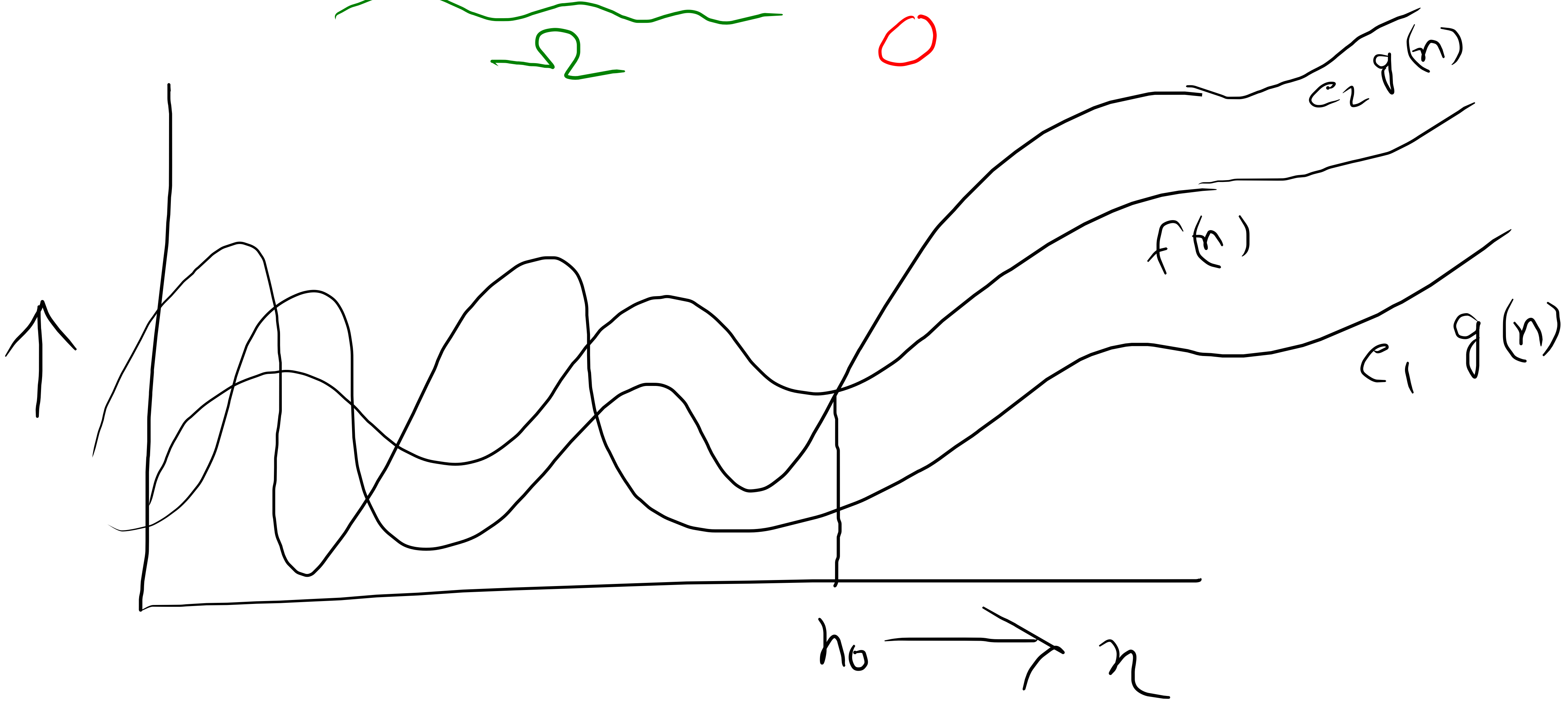
Assume $n_0 = 1$ then $\underline{\underline{c \le 7}}$ $\boxed{0 < c \le 7}$

Theta notation 'θ' tight bound

$f(n) = \theta(g(n))$ if there exists constants $c_1 > 0$, $c_2 > 0$, $n_0 > 0$

such that $\underbrace{c_1 g(n) \le f(n)}_{\Omega} \le c_2 g(n)$ $\forall\ n \geq n_0$

$f(n) \le c_2 g(n)$: O



Ex^m Is $5n^2+2n+3 = \theta(n^2)$

Assume $n_0 = 1$ then $0 < c_1 \leq 10$

$c_2 \geq 10$

Small 'o' and small omega 'ω'

$f(n) = o(g(n))$ if there exist constants $c > 0$, $n_0 > 0$

such that $0 \le f(n) < c\,g(n) \quad \forall\, n \ge n_0$

$f(n) = \omega(g(n))$ if there exist constants $c > 0$, $n_0 > 0$

s.t. $0 \le c\,g(n) < f(n) \quad \forall\, n \ge n_0$

Solving recurrences

- A recurrence is an equation or inequality that describes a function in terms of its value on smaller inputs.

- It is useful to analyse the complexity of Divide and conquer algorithm.

Methods to solve recurrences

1. Substitution method

2. Recursion tree method

3. Master method

substitution method

step 1: Guess the form of the solution

step 2: verify the guess by mathematical induction

step 3: solve for some constants.

$\underline{Ex^m}$ solve $T(n) = 4T(\frac{n}{2}) + n$

[Assumption $T(1) = \theta(1)$]

$\underline{Step 1}$: Guess: $T(n) = \theta(n^3)$ ✓ $T(n) \le cn^3$

$\underline{Step 2}$: Assume $T(k) \le ck^3$ for $k < n$

our goal is to prove, $T(n) \le cn^3$

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

$$\leq 4 \cdot c \cdot \left(\frac{n}{2}\right)^3 + n$$

$$= \frac{c}{2} n^3 + n$$

$$= \underbrace{cn^3}_{\text{desired}} - \underbrace{\left[\frac{c}{2} n^3 - n\right]}_{\text{residual}}$$

$T(n) \leq$ desired provided residual ≥ 0

$T(n) \leq cn^3$ provided $\frac{c}{2} n^3 - n \geq 0$

<u>step 3</u>: residual is true when $n \geq 1$, $c \geq 2$

H.W

$$T(n) = 4T\left(\frac{n}{2}\right) + n$$

Guess: $T(n) = O(n^2)$

Guess $T(n) = O(n)$