Phân tích & Thiết kế thuật toán (Algorithms Design & Analysis)

L/O/G/O

GV: HUỲNH THỊ THANH THƯƠNG
Email: thuonghtt@uit.edu.vn

PHÂN TÍCH THUẬT TOÁN

CHƯƠNG 1: TỔNG QUAN

L/O/G/O

Nội dung

1. Vấn đề và xác định vấn đề
2. Thuật toán
3. Phân tích thuật toán
4. Độ phức tạp

Đặt vấn đề

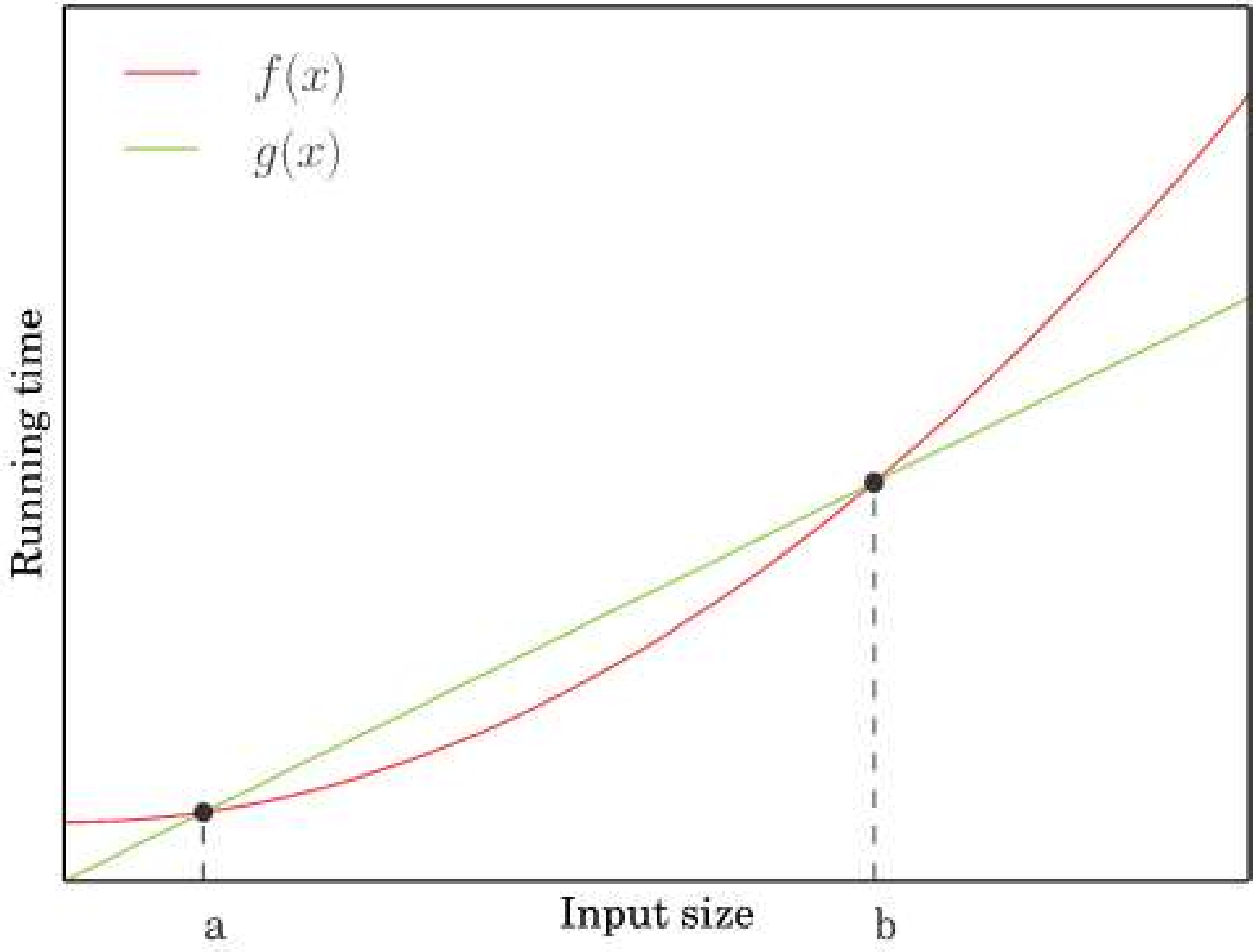
- Giả sử bạn đang thiết kế 1 trang web để xử lý số liệu (VD số liệu tài chính). Ta có 2 chương trình có thể xem xét:

$T_1(n)=30n+8$

$T_2(n)=n^2+1$

Ban sẽ chọn chương trình nào nếu muốn phục vụ cho khách hàng?

Compare 2 algorithms



$T_2(n)=n^2+1$

$T_1(n)=30n+8$

So sánh 2 thuật toán

- Cách tốt nhất: xác định chính xác thời gian thực hiện T(n) (the exact running time) của 2 GT rồi so sánh chúng
- Thực tế:
  - Rất khó để tính T(n) chính xác

```
int BSearch_Recursion (int list[], int key, int left, int right)
{
   if (left <= right)
   {
        int mid = (left + right)/2;
        if (key == list[mid])
                return mid;       // trả về vị trí tìm thấy key
        else if (key < list[mid])
                return BSearch_Recursion (list, key, left, mid-1);
        else return BSearch_Recursion (list, key, mid+1, right);
   }
   return -1;     // không tìm thấy
}
```

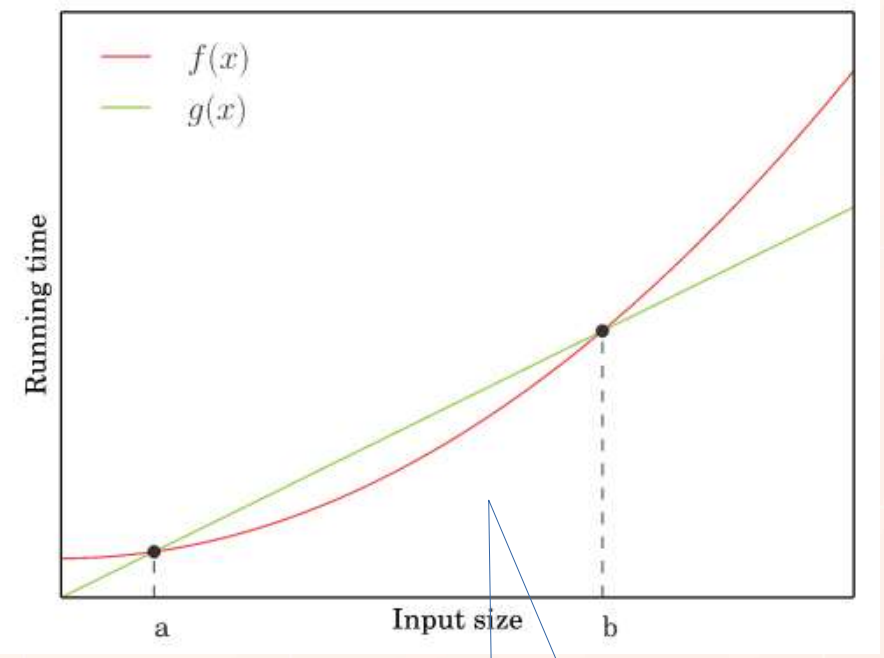
```
1   i = 1; count =0;
2   while ( i ≤ 4n)
3   {   x=(n-i)(i-3n) ;
4       y=i-2n;
5       j=1;
6       while (j  ≤ x )
7       {   count = count - 2;
8           j = j  + 2;
9       }
10      if (x>0)
11          if (y>0)
12              count = count +1;
13      i = i + 1;
14  }
```

Giải phương trình đệ quy để tìm T(n)

Dùng kỹ thuật toán "Xét dấu hàm" để xử lý if...else

So sánh 2 thuật toán

- Cách tốt nhất: xác định chính xác thời gian thực hiện - T(n) của 2 GT rồi so sánh chúng
- Thực tế:
  - Rất khó để tính T(n) chính xác
  - Nếu tính được T(n) rồi thì có thể gặp khó khăn khác khi so sánh 2 hàm số với nhau



Nếu hàm T(n) đơn giản thì vẽ hình và quan sát hoặc xét dấu hàm

Đặt vấn đề

$T(n) = 3n^2\log(n) - 12n^2 + 19$

$T(n) = 3e^n - 10000n^2 + 2$

Giải thuật nào tốt hơn (nhanh hơn)?

Tìm cách so sánh khác

Compare 2 algorithms

Mục tiêu mới:

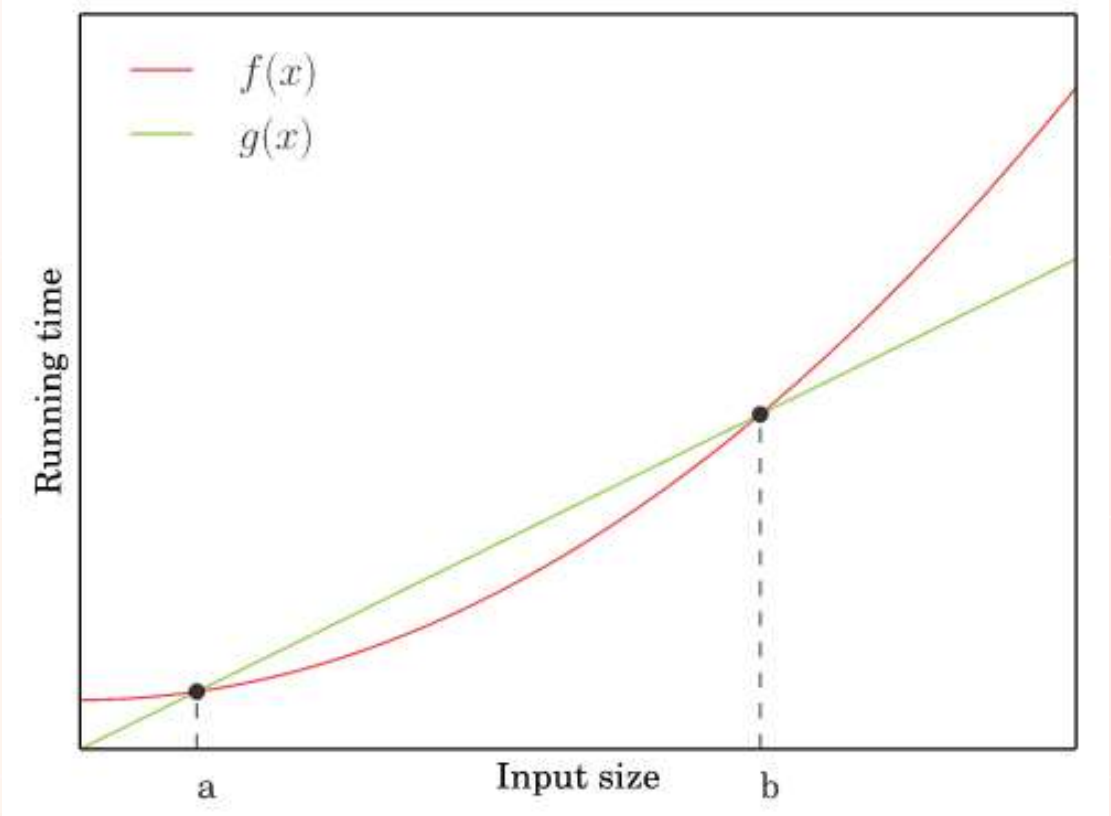
- 1. So sánh tương đối: hàm không sai biệt nhiều thì xem như sấp xỉ nhau về độ lớn
- 2. Chỉ quan tâm đến những giá trị n đủ lớn -->lớn như thế nào?
- 3. Xét n tiến tới ∞

So sánh "độ lớn" của T_1 và T_2 bằng cách nào?

Độ lớn của T1 là cỡ khoảng 2^n, độ lớn của T2 là cỡ khoảng $3^{\log n}$

- Khi n càng lớn (tiến tới ∞) thì hàm nào sẽ lớn hơn?

- "nếu f (x) tăng nhanh hơn g(x) thì f(x) luôn lớn hơn g(x) ở ∞"



Làm sao biết được làm nào tăng nhanh hơn?

Dựa vào “order of growth” (bậc tăng trưởng)

- Hàm có bậc tăng trưởng lớn hơn → hàm đó tăng nhanh hơn → sẽ lớn hơn với các giá trị n dủ lớn
- Những hàm cùng bậc tăng trưởng → cùng độ lớn

Ví dụ: $T(n)= 30n+8$ có bậc tăng trưởng là n

$1{,}000{,}001 \approx 1$

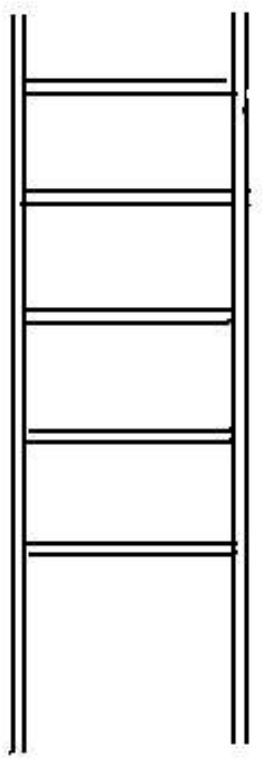
$3n^2 + 5 \approx n^2$

Aymptotic Notation

- Ký hiệu/ký pháp tiệm cận: O, Ω, Θ
  - Cho ta 1 phương pháp để phân loại các hàm "chặt chẽ hơn" dựa theo bậc tăng trưởng của chúng

Độ phức tạp

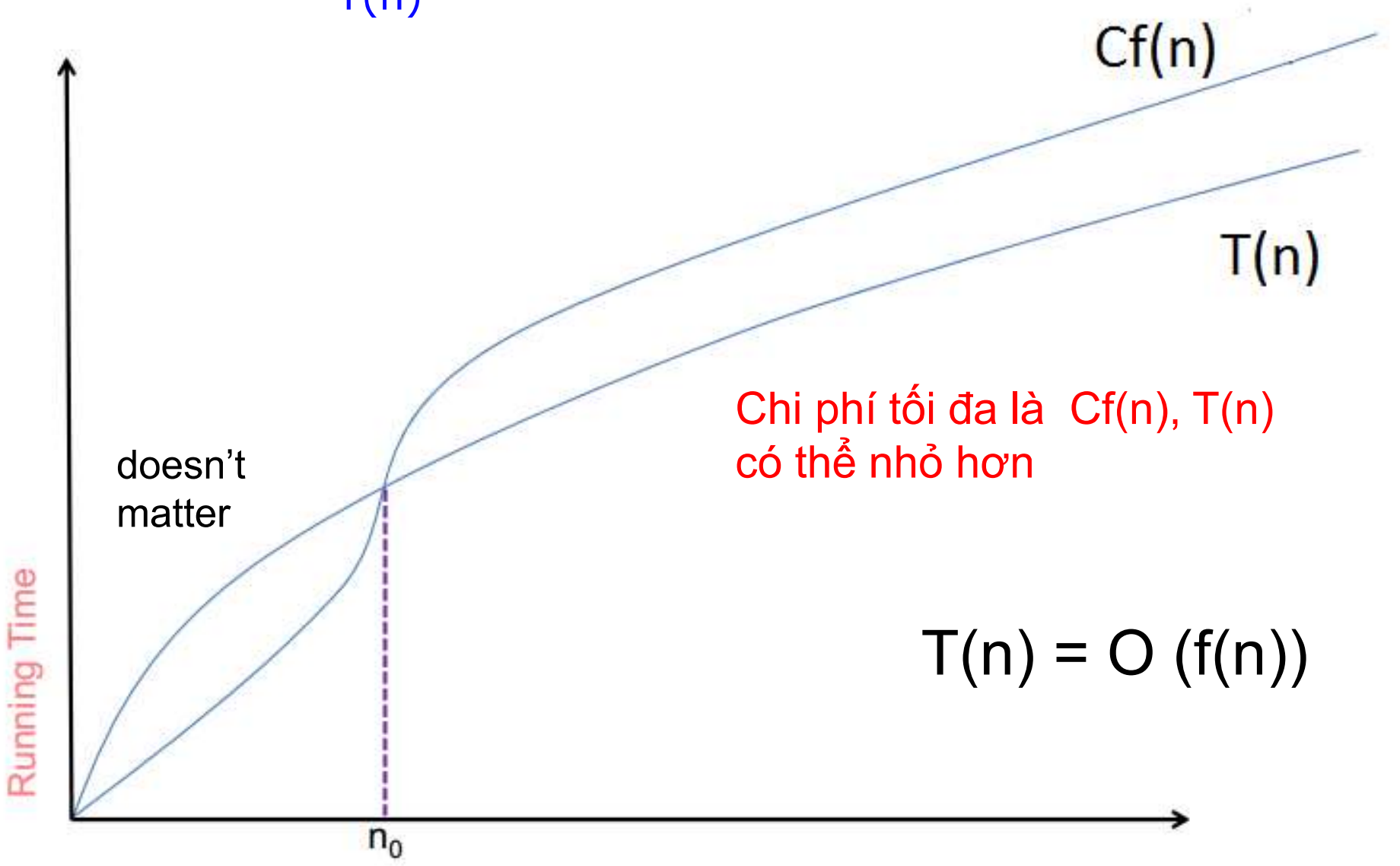
- Ý nghĩa chung:
  - Phân lớp "cấp độ lớn" của hàm T(n) khi n đủ lớn
  - GT nào có độ phức tạp ở phân lớp thấp hơn thì hiệu quả hơn



- Độ phức tạp của GT: được xác định thông qua các ký hiệu tiệm cận O, Ω, Θ --> có 3 loại độ phức tạp phổ biến

Big-O

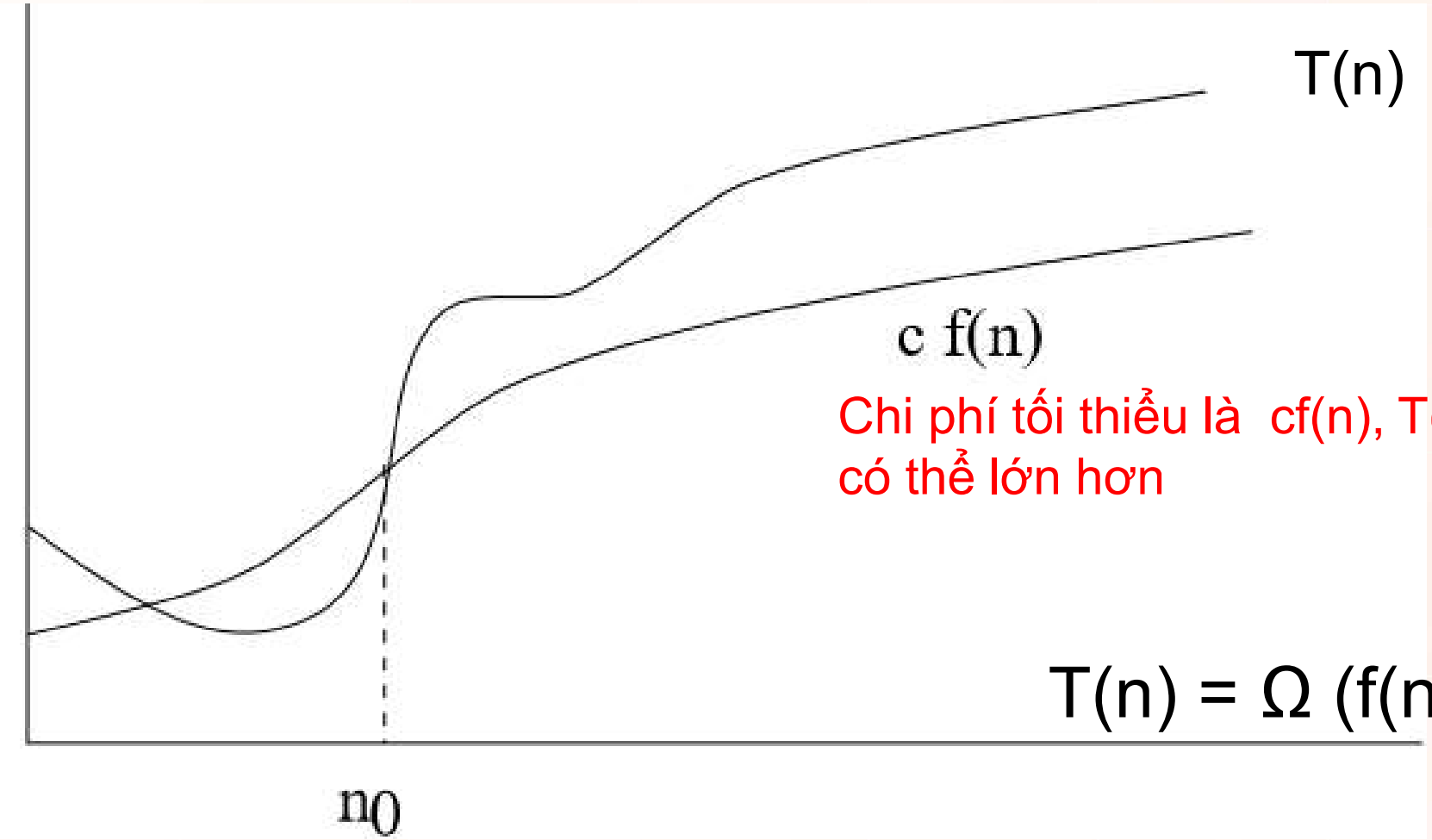
- Khi n tăng, T(n) tăng không nhanh hơn Cf(n), Cf(n) là chặn trên (asymptotic upper bound) của T(n)



Chi phí tối đa là Cf(n), T(n) có thể nhỏ hơn

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

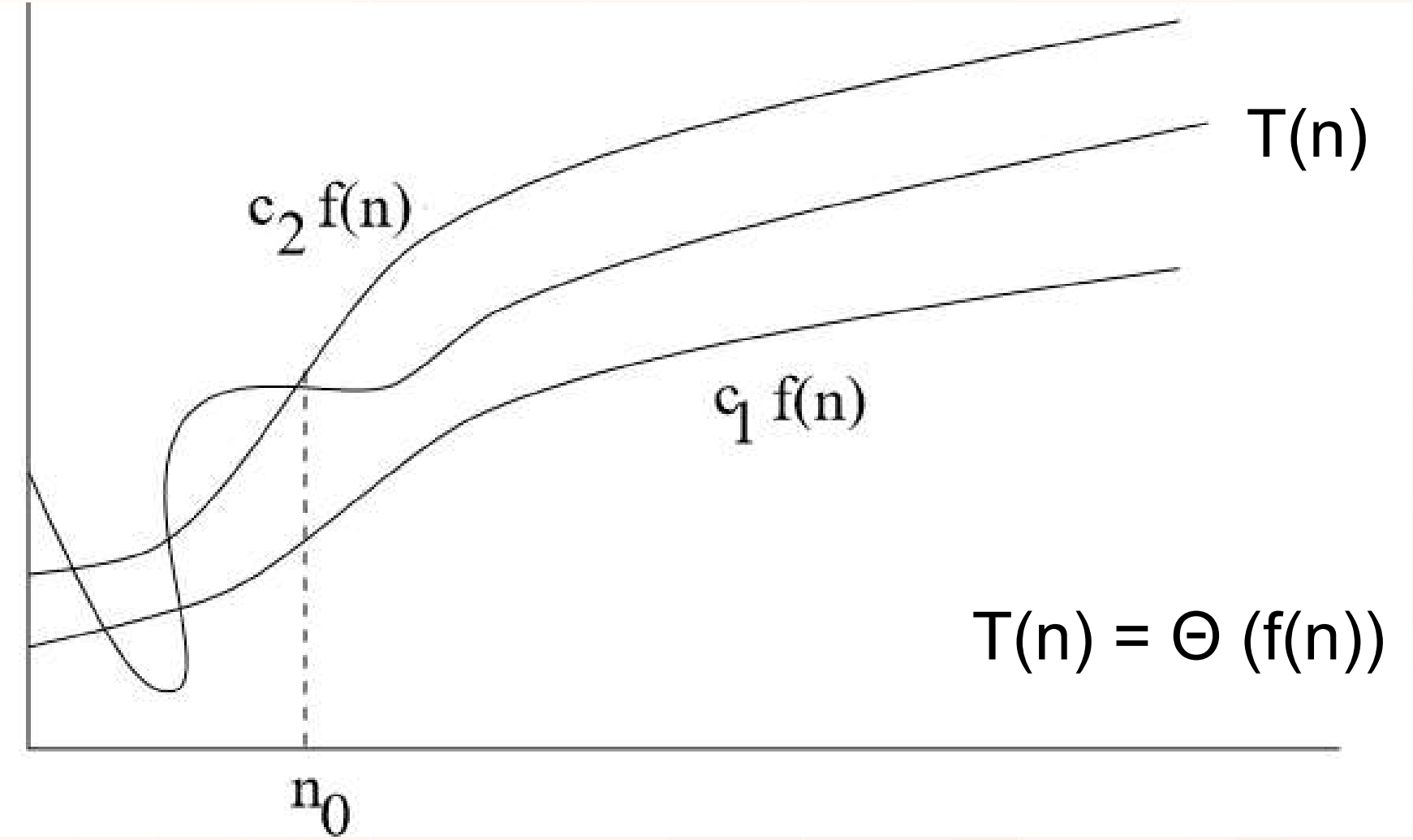
Big- Ω



Chi phí tối thiểu là cf(n), T(n) có thể lớn hơn

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

Big-Θ



Big-O notation (upper bound)

- Định nghĩa (phân tích GT):
  - T(n) = O(f(n)) nếu tồn tại hằng số dương $c \in \mathbb{R}^+$ và $n_0 \in \mathbb{N}$ sao cho:

$$T(n) \leq cf(n) \text{ với } \forall\ n \geq\ n_0$$

Ta nói: T(n) có bậc tăng trưởng là f(n) , T(n) có độ phức tạp là O của f(n)

Big-O notation

- Ký hiệu: T(n) = O(f(n)) . Bản chất: $T(n) \in O(f(n))$
- Chú ý:
  - Dấu = chỉ là ký hiệu hình thức
  - O(f(n)) là tập hợp

$$O(f(n)) = \{t : N^* \rightarrow N^* \mid \exists\, c \in R^+ , \exists n_0 \in N, \forall\, n \geq n_0 , t(n) \leq cf(n)\}$$

O(f(n)) = {t(n) : ∃ positive constants c and n_0, such that ∀n ≥ n_0, we have 0 ≤ t(n) ≤ cf(n) }

Big-O notation

- Lưu ý:
  - Có thể đánh giá ĐPT bằng ký hiệu tiệm cận khác như Ω, Θ → nên kèm theo ký hiệu khi nói đến ĐPT. VD: "GT có ĐPT $O(n^2)$"
  - f(n) chỉ là 1 hàm chặn trên của T(n), vẫn có thể có cách ước lượng chặt hơn
  - Luôn tìm được f(n) và cần tìm f(n) nhỏ nhất có thể

Big-O notation

- Ta thấy: Với $T(n) = 10n$
  - $T(n) = O(n)$, $T(n) = O(n^2)$, $T(n) = O(n^3)$
  - $10n \in O(n)$? $O(n^2)$? $O(n^3)$ (Hỏi: dùng ký hiệu gì thay cho ? là đúng)

- Ví dụ 1: Xét $T(n) = 3n^2 + 5n + 4$. Tìm f(n) để $T(n) = O(f(n))$?

Ta có:

$T(n) = 3n^2 + 5n + 4 \leq 3n^2 + 5n^2 + 4n^2$, $\forall\ n \geq 1$

$\Rightarrow T(n) \leq 12n^2\ \forall\ n \geq 1.$

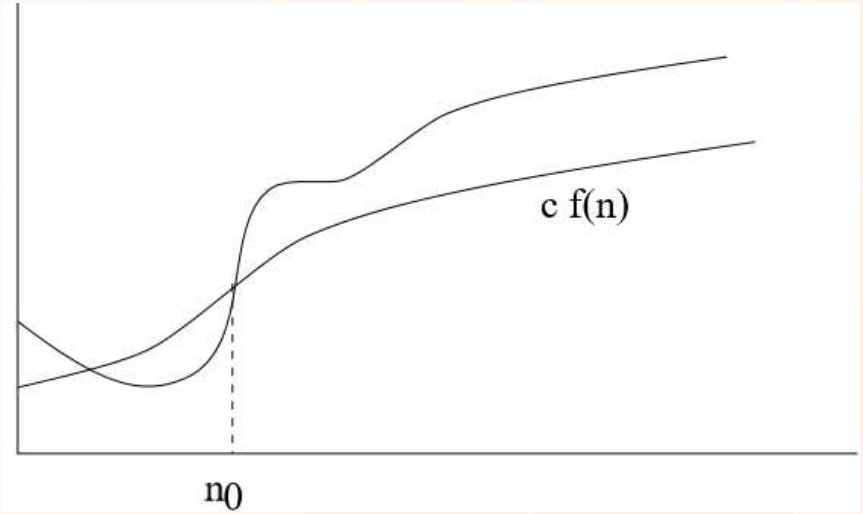
Chọn $c=12$, $n_0 = 1$, theo định nghĩa của Big-O, ta có dccm, $T(n) = O(n^2)$

Big-Ω notation

- Định nghĩa:

$T(n) = \Omega(f(n))$ nếu và chỉ nếu tồn tại các hằng số dương $c \in \mathbb{R}^+$, $n_0 \in \mathbb{N}$ sao cho:

$$T(n) \geq c.f(n) \text{ với } \forall\ n \geq n_0$$



Ký hiệu Tiệm cận Ω

Định nghĩa:

$$\Omega(f(n))=\{t: N \rightarrow N^* \mid \exists c \in R^+ , \exists n_0 \in N, \forall n \geq n_0 , t(n) \geq cf(n)\}$$

Tính chất:

Cho f, g : $N \rightarrow N^*$, Ta có:

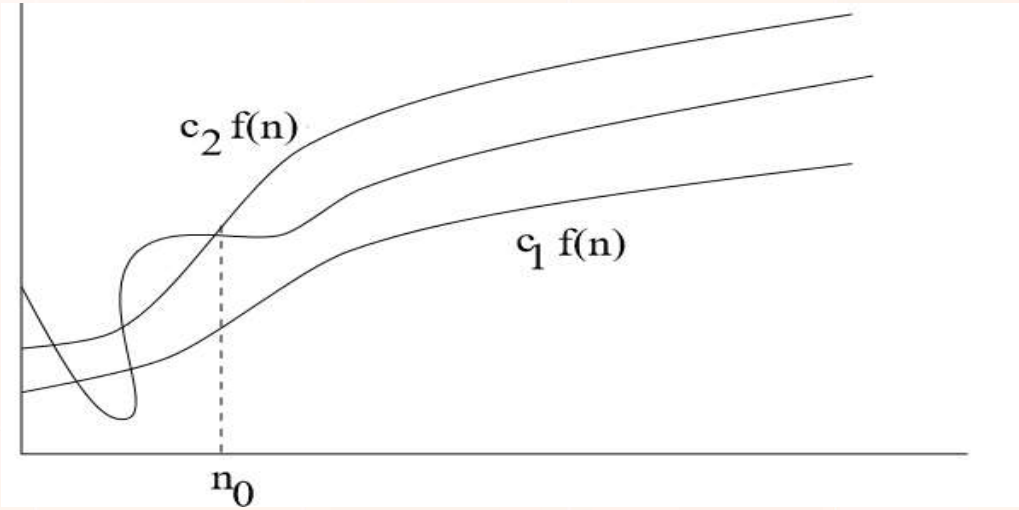
$f(n) \in O(g(n)) \Leftrightarrow g(n) \in \Omega (f(n))$

Big-Θ notation

- Định nghĩa:

$T(n) = \Theta(f(n))$ nếu và chỉ nếu tồn tại các hằng số dương $c_1, c_2 \in \mathbb{R}^+$ và $n_0 \in \mathbb{N}$ sao cho:

$$c_1 f(n) \le T(n) \le c_2 f(n) \text{ với } \forall\ n \ge n_0$$



Ký hiệu Tiệm cận Θ

Định nghĩa:

$f(n) = \Theta(g(n))$ nếu và chỉ nếu

$f(n) = O(g(n))$ và $g(n) = O(f(n))$

$\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$

Hướng dẫn tại lớp (đã sửa trên bảng)

Chứng minh:

$f(n) + g(n) = \Theta(\max(f(n), g(n)))$ Lưu ý: Θ là Big – Theta

Chứng minh: $\mathbf{\Theta(\alpha g(n)) = \Theta(g(n))}$ với α là hằng số thực dương

Bài tập trên lớp (lấy điểm quá trình): Inclass#04

Chọn làm 2 trong 3 câu (3.5 điểm), câu thứ 3 là bonus (tối đa bonus 1 điểm)

a) Phép suy ra bên dưới là đúng hay sai và vì sao?

$$\frac{1}{2}n^2 = O(n^2)$$

$$n^2 + 1 = O(n^2)$$

$$\Rightarrow \frac{1}{2}n^2 = n^2 + 1 \quad ???$$

b) Xét $f(n) = 7n^2$,
$g(n)=n^2 - 80n$,
$h(n)=n^3$

Chứng minh (dùng định nghĩa, không dùng lim):

$f(n) = O(g(n))$,
$g(n) = O(f(n))$,
$f(n) = O(h(n))$,
$h(n) \neq O(f(n))$

c. Nếu $t1(n) \in O(f(n))$ và $t2(n) \in O(g(n))$ thì $t1(n) + t2(n) \in O(\max\{f(n), g(n)\})$