



# CHƯƠNG 1

# KỸ THUẬT

# PHÂN TÍCH THUẬT TOÁN

Khoa CÔNG NGHỆ PHẦN MỀM  
Trường Công nghệ Thông tin & Truyền thông  
ĐẠI HỌC CẦN THƠ

*Võ Huỳnh Trâm*



# MỤC TIÊU

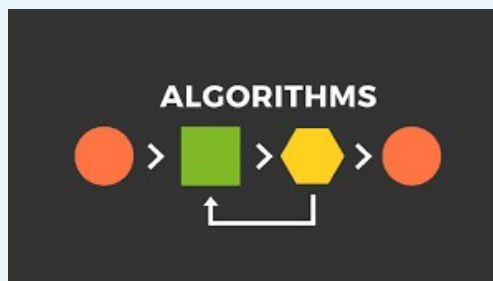
- **Sau khi học xong chương này, sinh viên cần:**
  - Hiểu sự cần thiết phải phân tích đánh giá thuật toán.
  - Biết các tiêu chuẩn để đánh giá một thuật toán.
  - Hiểu khái niệm độ phức tạp của thuật toán.
  - Vận dụng các quy tắc để tính độ phức tạp của chương trình không gọi chương trình con, chương trình có gọi các chương trình con không đệ quy.
  - Vận dụng các phương pháp thành lập phương trình đệ quy.
  - Vận dụng các phương pháp giải phương trình đệ quy



CANTHO UNIVERSITY

# Định nghĩa THUẬT TOÁN

- Khái niệm **thuật toán** (Algorithm)
  - Thuật toán là một dãy xác định các thao tác cơ bản áp dụng trên dữ liệu vào nhằm đạt được giải pháp cho một vấn đề.



- Thuật toán: Thủ tục tính toán nhận tập các *dữ liệu vào* (input) và tạo các *dữ liệu ra* (output)





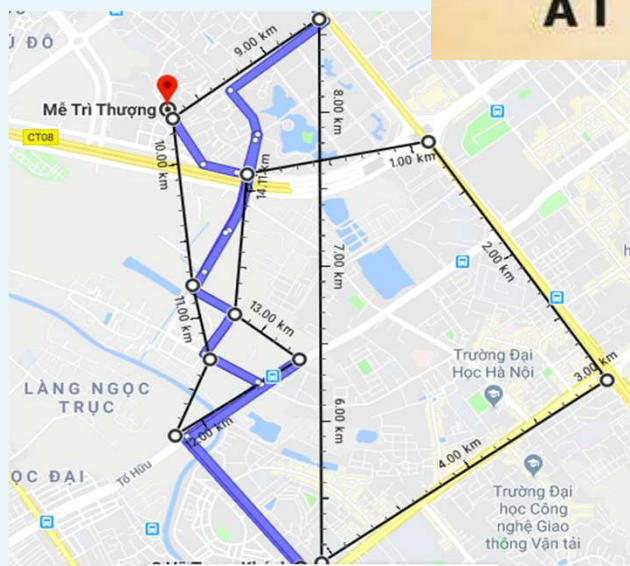
# THUẬT TOÁN

- Thuật toán hàng ngày trong cuộc sống: *Nấu cơm, Gọi điện thoại, ...*
- Thuật toán trong toán học/tin học: *Nhân hai ma trận, Tính tích phân, Giải hệ phương trình, ...*
- **Thuật toán** giữ vai trò trung tâm trong cuộc cách mạng công nghiệp 4.0 = Chìa khóa để **tăng năng suất của nhân lực**.



CANTHO UNIVERSITY

# (1) Thuật toán TÌM ĐƯỜNG NGẮN NHẤT



*Phần mềm chỉ đường  
giao thông  
(Google map,  
Grab, Uber,  
Giao hàng nhanh,...)*



CANTHO UNIVERSITY

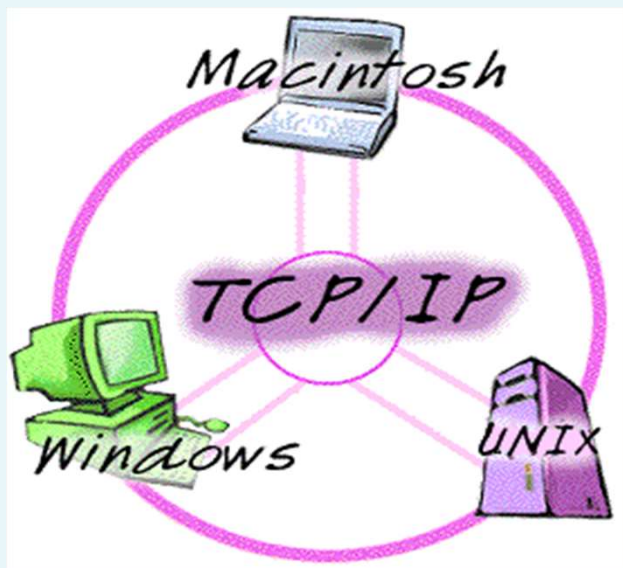
# (1) Thuật toán TÌM ĐƯỜNG NGẮN NHẤT

*Internet protocol suite*

hoặc *IP suite*

hoặc *TCP/IP protocol suite (1983)*

*Phần mềm định hướng đường truyền nhận tín hiệu cuộc gọi trong hệ thống mạng, viễn thông*







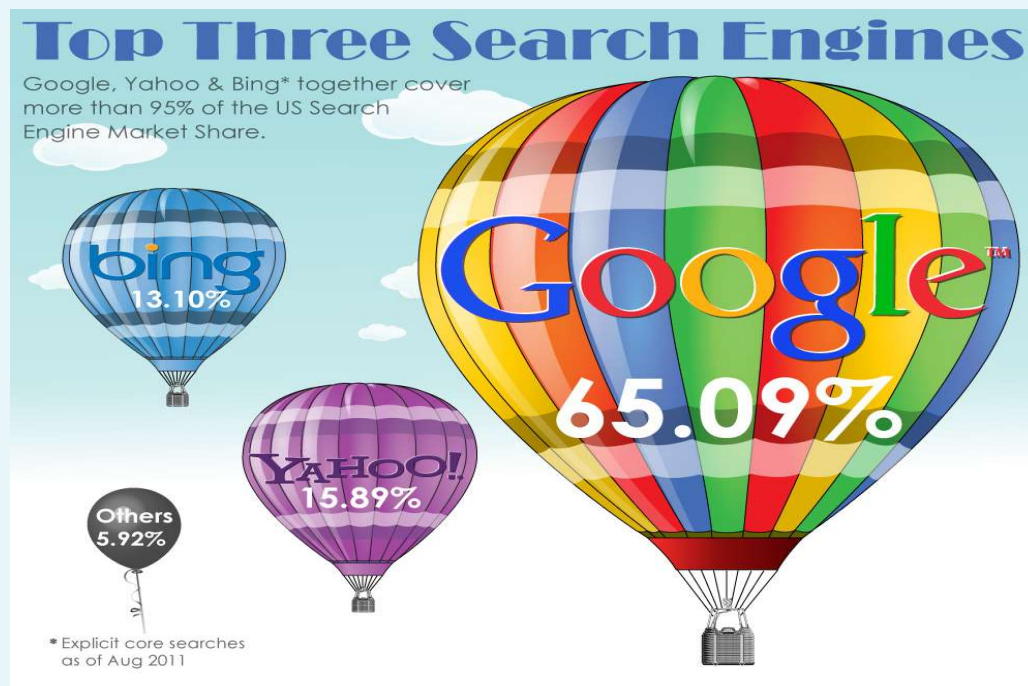
CANTHO UNIVERSITY

## (2) Thuật toán TÌM KIẾM

### Google Search



### PageRank (1996)





CANTHO UNIVERSITY

## (2) Thuật toán TÌM KIẾM

### Facebook News Feed



**EdgeRank**  
(2010)







CANTHO UNIVERSITY

### (3) Thuật toán NÉN / MÃ HÓA

Chương trình thu thập, diễn giải và mã hóa dữ liệu của cơ quan an ninh quốc gia Mỹ (NSA)

**DES**

**PRIM**

**Khufu**

**Khafre**

....



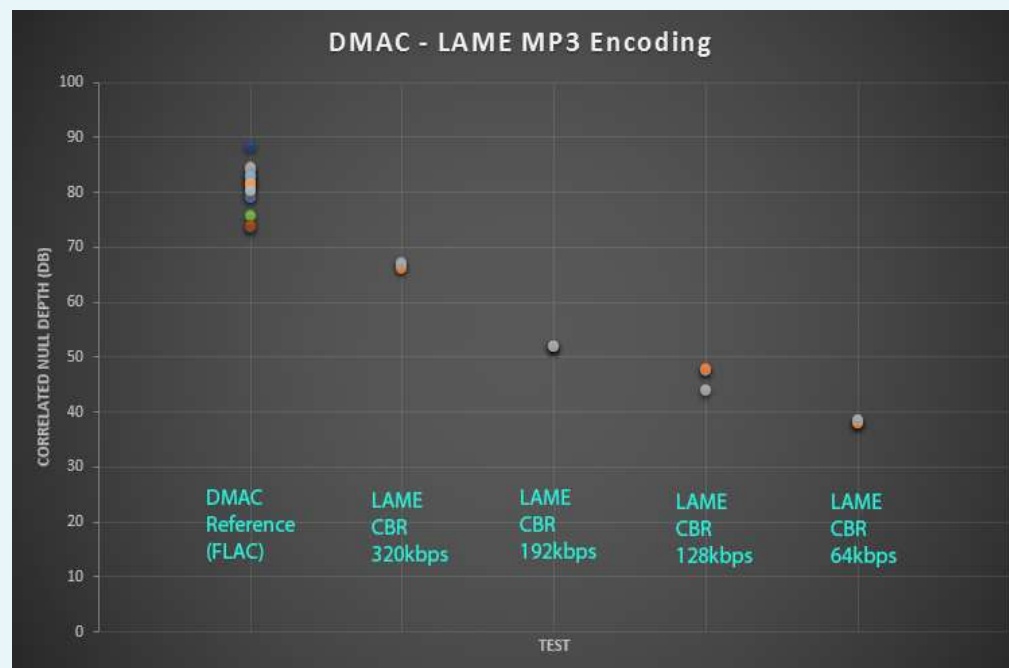


CANTHO UNIVERSITY

## (3) Thuật toán NÉN / MÃ HÓA

Chuẩn nén MP3

**Moving Picture  
Experts Group-1  
Audio Layer III**  
(1987)





## Đặc tả THUẬT TOÁN

– Có nhiều cách đặc tả thuật toán

- **Không hình thức** : *Ngôn ngữ tự nhiên*

**Ví dụ:** mô tả thuật toán tìm ước số chung lớn nhất của hai số nguyên.

Input: Hai số nguyên  $a, b$ .

Output: Ước số chung lớn nhất của  $a, b$ .

**Thuật toán:**

Bước 1: Nếu  $a=b$  thì  $USCLN(a, b)=a$ .

Bước 2: Nếu  $a > b$  thì tìm USCLN của  $a-b$  và  $b$ , quay lại bước 1;

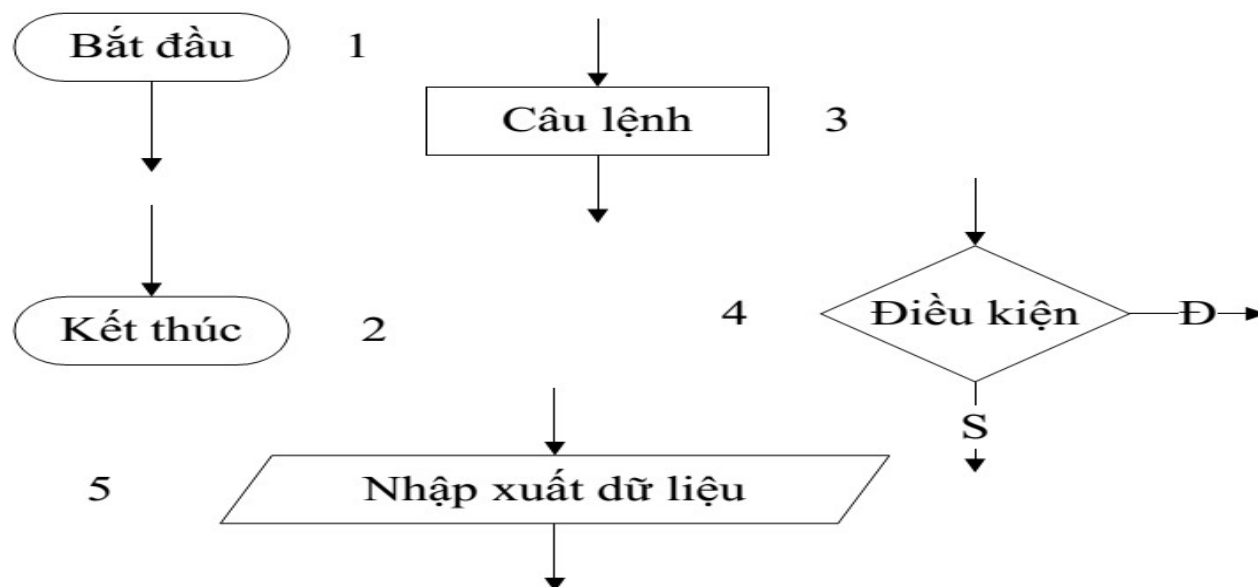
Bước 3: Nếu  $a < b$  thì tìm USCLN của  $a$  và  $b-a$ , quay lại bước 1;



## Đặc tả THUẬT TOÁN

- **Nửa hình thức** : Kết hợp ngôn ngữ tự nhiên và các kí hiệu toán học : *Lưu đồ*, *Sơ đồ khối*, ...

Các khối cơ bản của một sơ đồ thuật toán





## Đặc tả THUẬT TOÁN

- **Hình thức** : Ngôn ngữ giả (pseudocode).

*Ngôn ngữ Z, ngôn ngữ B, ...*

**if** Delta > 0 **then begin**

$x_1 = (-b - \sqrt{\text{delta}}) / (2 * a)$

$x_2 = (-b + \sqrt{\text{delta}}) / (2 * a)$

xuất kết quả : phương trình có hai nghiệm là  $x_1$  và  $x_2$

**end**

**else**

**if** delta = 0 **then**

xuất kết quả : phương trình có nghiệm kép là  $-b / (2 * a)$

**else** {trường hợp delta < 0 }

xuất kết quả : phương trình vô nghiệm



CANTHO UNIVERSITY

## Sự cần thiết phải phân tích, đánh giá thuật toán

- Cần phải phân tích, đánh giá thuật toán để:
  - Lựa chọn một **thuật toán tốt nhất** trong các thuật toán để cài đặt chương trình giải quyết bài toán đặt ra.
  - Cải tiến thuật toán hiện có để được một thuật toán tốt hơn.





## Tiêu chuẩn đánh giá thuật toán

- Một thuật toán được xem là tốt nếu đạt các tiêu chuẩn:

### (1) Tính đúng đắn

- Chạy trên dữ liệu thử: *Không khả thi*
- Chứng minh lý thuyết (bằng toán học): *Khó khăn*

### (2) Tính đơn giản

### (3) Tính nhanh chóng (thời gian thực thi)

- Rất quan trọng khi chương trình thực thi nhiều lần  
= *Hiệu quả thời gian thực thi*



## Thời gian thực hiện chương trình

- **Thời gian thực hiện chương trình** là một hàm của kích thước dữ liệu vào, ký hiệu  **$T(n)$** , trong đó  **$n$**  là *kích thước (độ lớn) của dữ liệu vào*.

**Ví dụ :** Chương trình tính **tổng của  $n$  số** có thời gian thực hiện là  **$T(n) = Cn$**  trong đó  $C$  là một hằng số.

- Thời gian thực hiện chương trình là một *hàm không âm*, tức là  **$T(n) \geq 0$** ,  $\forall n \geq 0$ .



## Đơn vị đo thời gian thực hiện

- Đơn vị của  **$T(n)$**  :
  - $T(n)$  không phải là đơn vị đo thời gian bình thường như giờ, phút, giây.
  - $T(n)$  xác định bởi *số các lệnh/chỉ thị* được thực hiện trong một *máy tính lý tưởng*.
- **VD**: Khi nói thời gian thực hiện của chương trình là  $T(n) = Cn$  thì có nghĩa là chương trình cần  **$Cn$**  lệnh/chỉ thị thực thi.



## Thời gian thực hiện: 3 trường hợp

- Thời gian thực hiện chương trình không chỉ phụ thuộc vào *kích thước* mà còn phụ thuộc vào *tính chất* của dữ liệu vào (*cùng kích thước dữ liệu vào nhưng thời gian thực hiện chương trình khác nhau*)
- Ví dụ :** *Tìm kiếm tuần tự*

MÔ PHỎNG VỚI  $N = 10$  VÀ DÃY A SAU:  
5, 7, 1, 4, 2, 9, 8, 11, 25, 51  
 $k = 2,$

A	5	7	1	4	2	9	8	11	25	51
i	1	2	3	4	5					



## Thời gian thực hiện trong trường hợp xấu nhất

- (1) Trường hợp *tốt nhất* : so sánh 1 lần
- (2) Trường hợp *trung bình*: so sánh  $n/2$  lần
- (3) Trường hợp *xấu nhất* : so sánh  $n$  lần

- Vì vậy, xem  $T(n)$  là thời gian thực hiện chương trình trong **trường hợp xấu nhất** trên dữ liệu vào có kích thước  $n$ .

Hay:  $T(n)$  là **thời gian lớn nhất** để thực hiện chương trình đối với mọi dữ liệu vào có cùng kích thước  $n$ .



## Tỷ suất tăng của hàm

- Ta nói hàm không âm  **$T(n)$**  có **tỷ suất tăng** (*growth rate*)  **$f(n)$**  nếu tồn tại các hằng số  $C$  và  $N_0$  sao cho
$$T(n) \leq Cf(n), \forall n \geq N_0$$

$\Rightarrow$  Tỷ suất tăng  $f(n)$  = tốc độ tăng của hàm khi  $n$  tăng

VD:  $\forall n \geq 0$ , hàm  $n^3$  có tốc độ tăng cao hơn  $n^2$  khi  $n$  tăng

- Ta có thể chứng minh được “*Cho một hàm không âm  $T(n)$  bất kỳ, luôn tìm được tỷ suất tăng  $f(n)$  của nó*”.





## Ví dụ về tỷ suất tăng

- **VD1:** Xét hàm  $T(n) = (n+1)^2$ . Đặt  $N_0 = 1$  và  $C = 4$  thì  $\forall n \geq 1$ , ta có  $T(n) = (n+1)^2 \leq 4n^2$ , tức *tỷ suất tăng của  $T(n)$  là  $f(n) = n^2$*
- **VD2:** Xét hàm  $T(n) = 3n^3 + 2n^2$ . Cho  $N_0 = 0$  và  $C = 5$ , ta có thể chứng minh  $\forall n \geq 0: 3n^3 + 2n^2 \leq 5n^3$  hay *tỷ suất tăng của  $T(n)$  là  $f(n) = n^3$*
- *Tuy nhiên, rất khó xác định tỷ suất tăng bằng cách như trên mà thường áp dụng quy tắc sau:*

**Quy tắc vận dụng:** Nếu  $T(n)$  là một đa thức của  $n$  thì tỷ suất tăng của  $T(n)$  là  *$n$  với số mũ cao nhất.*



## Khái niệm độ phức tạp của thuật toán

- Giả sử có 2 thuật toán P1 và P2 với *thời gian thực hiện*  $T_1(n) = 100n^2$  và  $T_2(n) = 5n^3$

- **Vấn đề** : P1 hay P2 nhanh hơn?

- **Cách giải quyết**: So sánh  $T_1(n)$  và  $T_2(n)$

- **Kết quả** : Phụ thuộc vào  $n$

$$P_1 : T_1(n) = 100n^2 \rightarrow f_1(n) = n^2$$

$$P_2 : T_2(n) = 5n^3 \rightarrow f_2(n) = n^3$$

$$n = 1: \quad T_1(n) = 100.1^2 > T_2(n) = 5n^3 = 5.1^3 \rightarrow P_2 \text{ nhanh hơn } P_1$$

$$n = 2: \quad T_1(n) = 100.2^2 > T_2(n) = 5n^3 = 5.2^3 \rightarrow P_2 \text{ nhanh hơn } P_1$$

.....

$$n = 19: \quad T_1(n) = 100.19^2 > T_2(n) = 5n^3 = 5.19^3 \rightarrow P_2 \text{ nhanh hơn } P_1$$

$$n = 20: \quad T_1(n) = 100.20^2 = T_2(n) = 5n^3 = 5.20^3 \rightarrow P_2 \text{ bằng } P_1$$

$$n = 21: \quad T_1(n) = 100.21^2 < T_2(n) = 5n^3 = 5.21^3 \rightarrow P_1 \text{ nhanh hơn } P_2$$

.....



## Khái niệm độ phức tạp của thuật toán

Giả sử có 2 thuật toán P1 và P2 với *thời gian thực hiện*  $T_1(n) = 100n^2$  và  $T_2(n) = 5n^3$

**Vấn đề :** P1 hay P2 nhanh hơn?

Khi  $n \leq 20$  :  $T_1(n) \geq T_2(n) \rightarrow P_2$  nhanh hơn  $P_1$

Khi  $n > 20$  :  $T_1(n) < T_2(n) \rightarrow P_1$  ( $n^2$ ) nhanh hơn ( $<$ )  $P_2$  ( $n^3$ )

- Như vậy, một cách hợp lý là nên xét *tỷ suất tăng của hàm thời gian thực hiện* chương trình thay vì xét *thời gian thực hiện*.

Khi đó: P1 thực hiện nhanh hơn P2 vì tỷ suất tăng  $n^2 < n^3, \forall n \geq 0$

**Tỷ suất tăng của hàm thời gian = Độ phức tạp của thuật toán**



## Khái niệm độ phức tạp của thuật toán

- **Ký pháp Ô lớn** (big-O notation ): Cho một thuật toán P có *thời gian thực hiện* là hàm  $T(n)$ , nếu  $T(n)$  có tỷ suất tăng là  $f(n)$  thì thuật toán P có độ phức tạp là  $f(n)$  và **ký hiệu thời gian thực hiện  $T(n)$  là  $O(f(n))$**  (đọc là “ô  $f(n)$ ”).
- **Tính chất: (1)  $O(C \cdot f(n)) = O(f(n))$**  với C là hằng số.

$$VD : O(100 n^2) = O(n^2)$$

$$(2) O(C) = O(1)$$

$$VD : O(100) = O(1)$$

- Lưu ý:* - Độ phức tạp của thuật toán là hàm chặn trên của hàm thời gian.  
- Hằng nhân tử C trong hàm chặn trên thường không có ý nghĩa.



CANTHO UNIVERSITY

# Khái niệm độ phức tạp của thuật toán

## Thời gian thực hiện **$T(n)$**

(- Thời gian thực thi chương trình  
- Đo bằng số chỉ thị/lệnh)

## Tỷ suất tăng **$f(n)$**

(- Hàm chặn trên của  $T(n)$ )

## Độ phức tạp **$O(f(n))$**

(Thang đo tỷ suất tăng)

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

$f(n) = n^2$

**$O(n^2)$**

2)  $T(n) = 3n^3 + 2n^2$

$f(n) = ?$

**$O(?)$**

3)  $T(n) = 7n^8 + 4n^7 + 10n^2 + 5$

$f(n) = ?$

**$O(?)$**



## Các hàm độ phức tạp thường gặp

Dạng O	Tên Phân loại	
$O(1)$	Hằng	
$O(\log_2(n))$	logarit	$\sim \log n$
$O(\sqrt{n})$	Căn thức	
$O(\sqrt[3]{n})$		
...		
$O(\sqrt[m]{n})$		
$O(n)$	Tuyến tính	
$O(n^2)$	Bình phương	Đa thức
$O(n^3)$	Bậc ba	
...		
$O(n^m)$	Đa thức	
$O(c^n)$ , với $c > 1$	Mũ	Độ phức tạp lớn
$O(n!)$	Giai thừa	

Phạm Thế Bảo

Có thể chấp  
nhận được

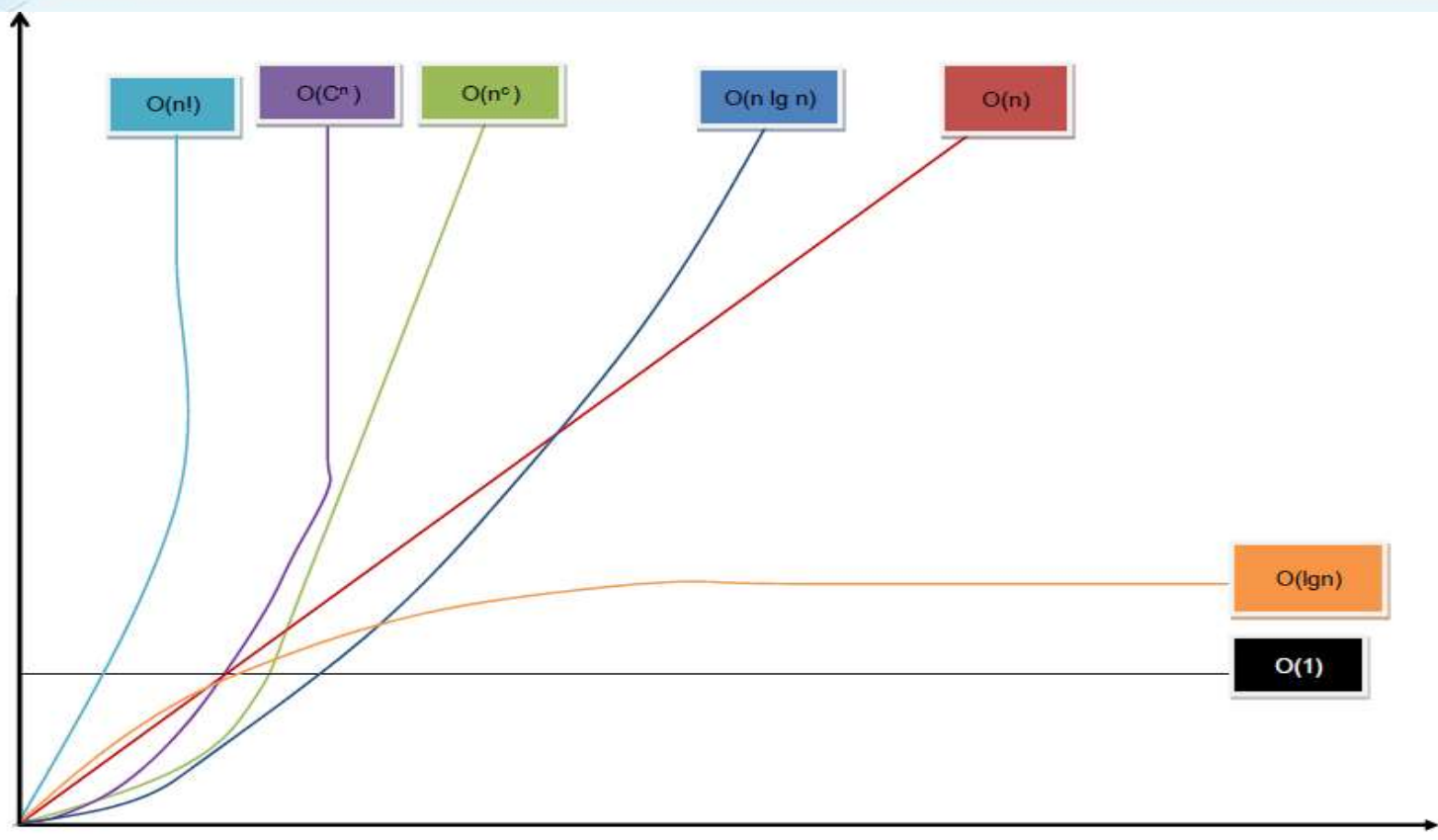
Cải tiến





CANTHO UNIVERSITY

## Đồ thị biến thiên các hàm độ phức tạp thường gặp

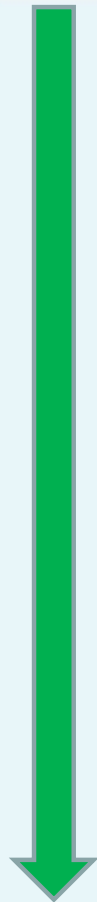




CANTHO UNIVERSITY

## Thang ưu tiên của Độ phức tạp

Thứ tự  
độ  
phức  
tạp  
tăng  
dần



Hàm hằng:  **$O(1)$**

Hàm logarit:  **$O(\log n)$**

Hàm tuyến tính:  **$O(n)$**

Hàm logarit tuyến tính:  **$O(n \log n)$**

Hàm đa thức:  **$O(n^c)$**

Hàm mũ:  **$O(C^n)$**

Hàm giai thừa:  **$O(n!)$**



## Giá trị biến thiên các hàm độ phức tạp thường gặp

$\lg n$	$n$	$n \lg n$	$n^2$	$n^3$	$2^n$
0	1	0	1	1	2
1	2	2	4	8	4
2	4	8	16	64	16
3	8	24	64	512	256
4	16	64	256	4096	65536
5	32	160	1024	32768	2147483648



## Bài tập: Độ phức tạp

1. Phân loại các hàm độ phức tạp sau theo 4 nhóm : *Hàm hằng, Hàm tuyến tính, Hàm đa thức, Hàm mũ*

$$2^n, (3/2)n, (3/2)^n, 2n^3, 3n^2, 1, 1000, 3n$$

2. Sắp xếp các hàm độ phức tạp trong Bài tập 1 theo thứ tự độ phức tạp tăng dần.

3. Sắp xếp các hàm sau theo thứ tự độ phức tạp tăng dần:

$$8n^2, 6n^3, 64, n\log_6 n, \log_8 n, 4n, n\log_2 n, 8^{2n}, \log_2 n$$



## Các nội dung cần nắm

- (1) Thuật toán là gì ?
- (2) Phân tích, đánh giá thuật toán để làm gì ?
- (3) Tiêu chí dùng đánh giá thuật toán ?
- (4) Định nghĩa độ phức tạp của thuật toán?
- (5) Thang mức độ ưu tiên của độ phức tạp ?