

# Cấu Trúc Dữ Liệu & Giải Thuật

## Phân Tích Thuật Toán

# Thuật Toán

---

- ▶ Một chuỗi các bước tính toán được định nghĩa rõ ràng để giải quyết một vấn đề
- ▶ Nhận vào một tập các giá trị đầu vào
- ▶ Trả về một tập các giá trị đầu ra
- ▶ Biểu diễn bằng: mã nguồn, mã giả
- ▶ Tính đúng đắn (cần thiết): kết quả trả về phản ánh đúng mong muốn của thông tin nhận vào
- ▶ Tính hiệu quả (quan trọng): độ tin cậy, tốc độ xử lý, tài nguyên sử dụng

# Thuật Toán – Đánh Giá

---

- ▶ Một vấn đề giải quyết bởi thuật toán khác nhau
- ▶ Với mỗi thuật toán
  - ▶ Thời gian: thời gian chạy của thuật toán
  - ▶ Không gian: dung lượng bộ nhớ sử dụng
- ▶ Thời gian chạy
  - ▶ Dữ liệu đầu vào
  - ▶ Kỹ năng lập trình
  - ▶ Chương trình dịch, hệ điều hành
  - ▶ Tốc độ phép toán trên máy tính

# Thuật Toán – Thời Gian Chạy

---

- ▶ Phụ thuộc vào độ lớn dữ liệu đầu vào
  - ▶ Tìm một phần tử trong danh sách  $n$  phần tử
  - ▶ Sắp xếp danh sách  $n$  phần tử
  - ▶ Bài toán người giao hàng cần thăm  $n$  điểm
- ▶ Với dữ liệu cùng độ lớn, thời gian thay đổi
  - ▶ Tìm một phần tử trong danh sách  $n$  phần tử
    - Phần tử nằm ở đầu danh sách
    - Phần tử nằm ở cuối danh sách
    - Phần tử nằm ở giữa danh sách

# Thuật Toán – Thời Gian Chạy

---

- ▶ Phân tích thực nghiệm
  - ▶ Đo thời gian chạy thực, vẽ đồ thị, ...
  - ▶ Có kịch bản tiến hành thực nghiệm
  - ▶ Cần cài đặt thuật toán (khó, lâu)
  - ▶ Đôi khi không thể chạy hết các bộ dữ liệu
  - ▶ Để so sánh các thuật toán, phải sử dụng cùng môi trường (phần cứng & phần mềm)
  - ▶ Phù hợp cho dự đoán

# Thuật Toán – Thời Gian Chạy

---

- ▶ Phân tích toán học
  - ▶ Ước lượng số phép toán là hàm của độ lớn dữ liệu đầu vào
  - ▶ Thông thường thông qua sử dụng mã giả
  - ▶ Xét tất cả các bộ dữ liệu đầu vào
  - ▶ Không phụ thuộc môi trường tính toán
  - ▶ Phù hợp cho cả dự đoán và giải thích
  - ▶ Thường được sử dụng để đánh giá

# Thuật Toán – Thời Gian Chạy

---

- ▶ Trường hợp xấu nhất (thông thường)
  - ▶ Thời gian chạy lớn nhất của thuật toán trên tất cả các dữ liệu có cùng độ lớn
- ▶ Trường hợp trung bình (đôi khi)
  - ▶ Thời gian chạy trung bình của thuật toán trên tất cả các dữ liệu có cùng độ lớn
    - Khó: do phải biết phân phối xác suất của dữ liệu
- ▶ Trường hợp tốt nhất (hiếm)
  - ▶ Thời gian chạy ít nhất của thuật toán trên tất cả các dữ liệu có cùng độ lớn

# Mã Giả (pseudo-code)

---

- ▶ Mô tả bậc cao của một thuật toán
- ▶ Cấu trúc rõ ràng hơn văn xuôi
- ▶ Không chi tiết như mã nguồn
- ▶ Được ưa thích trong biểu diễn giải thuật
- ▶ Ẩn đi các khía cạnh thiết kế chương trình

**Algorithm** arraySum( $A, n$ ):

**Input:** mảng  $A$  ( $n$  số nguyên)

**Output:** tổng các phần tử của  $A$

$sum \leftarrow 0$ ;

**for**  $i \leftarrow 0$  **to**  $n-1$  **do**

$sum \leftarrow sum + A[i]$ ;

**return**  $currentMax$ ;



# Phân Tích Độ Phức Tạp Thời Gian

---

- ▶ Đánh giá thời gian chạy thuật toán
  - ▶  $T(n)$  = số lượng phép toán sơ cấp cần thực hiện (số học, lô-gic, so sánh)
  - ▶ Mỗi phép toán sơ cấp thực hiện trong khoảng thời gian cố định
  - ▶ Chỉ quan tâm tới tốc độ tăng của hàm  $T(n)$
  - ▶ Ví dụ:  $T(n) = 2n^2 + 3n + 10$

# Phân Tích Độ Phức Tạp Thời Gian

---

**Algorithm** *arraySum*( $A, n$ ):

**Input:** mảng  $A$  ( $n$  số nguyên)

**Output:** tổng các phần tử của  $A$

$sum \leftarrow 0$ ;

**for**  $i \leftarrow 0$  **to**  $n-1$  **do**

$sum \leftarrow sum + A[i]$ ;

**return**  $sum$ ;

# Độ Tăng Của Hàm

---

- ▶ Với  $n$  là độ lớn dữ liệu đầu vào
- ▶ Tỷ lệ tăng trưởng (chính xác):
  - ▶  $an^2 + bn + c$
  - ▶  $an + b$
  - ▶  $an \log n + bn + c$
- ▶ Bậc tăng trưởng (xấp xỉ):
  - ▶  $an^2 + bn + c \Rightarrow$  bậc  $n^2$
  - ▶  $an + b \Rightarrow$  bậc  $n$
  - ▶  $an \log n + bn + c \Rightarrow$  bậc  $n \log n$

# Ký Hiệu Tiệm Cận ( $O$ - Ô-lớn)

## ▶ $O$ - – Ô-lớn (chặn trên):

Ta nói rằng  $f(n)$  là ô lớn của  $g(n)$  nếu tồn tại các hằng số  $c > 0$ , và  $n_0 > 0$  sao cho:

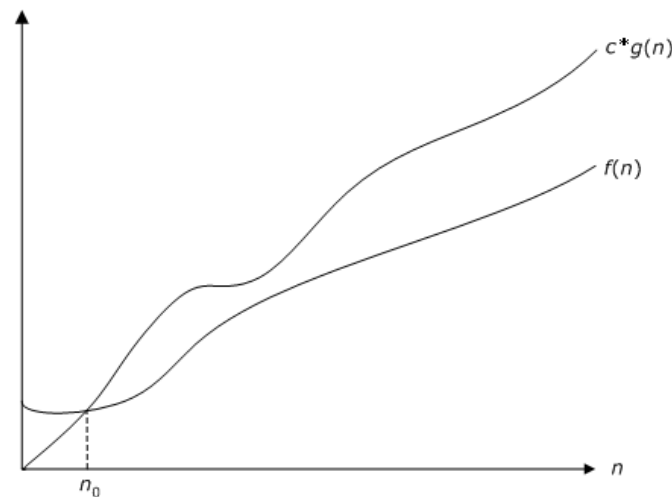
$$0 \leq f(n) \leq cg(n)$$

với mọi  $n \geq n_0$

Ký hiệu:  $f(n) \in O(g(n))$

▶ Ví dụ:  $2n^2 \in O(n^3)$

*Hàm không  
phải giá trị*



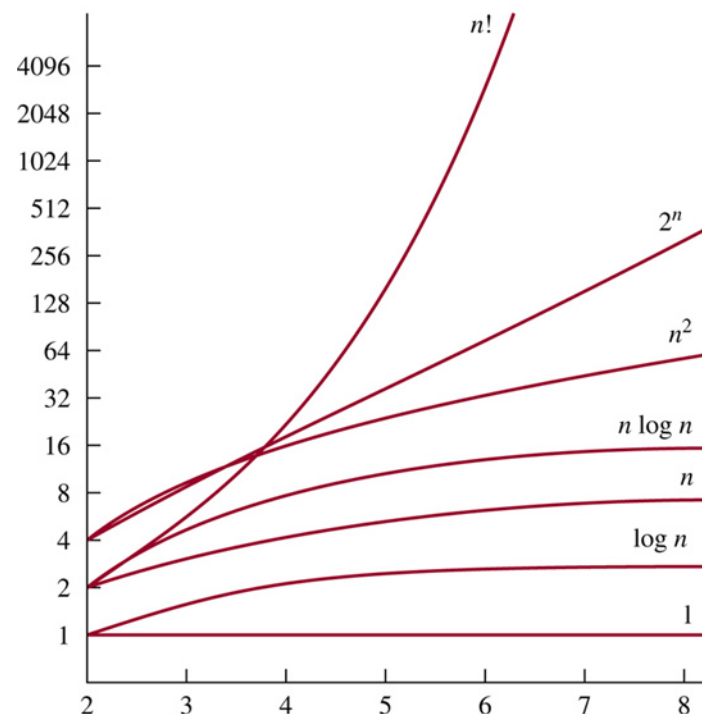
# Ký Hiệu $O$ - – Biểu Diễn Thời Gian Chạy

---

- ▶ Lấy cận trên chặt biểu diễn thời gian chạy của thuật toán
- ▶  $f(n)$  là cận trên chặt của  $T(n)$  nếu
  - ▶  $T(n) \in O(f(n))$ , và
  - ▶ Nếu  $T(n) \in O(g(n))$  thì  $f(n) \in O(g(n))$
- ▶ Nói cách khác
  - ▶ Không thể tìm được một hàm  $g(n)$  là cận trên của  $T(n)$  mà tăng chậm hơn hàm  $f(n)$



© The McGraw-Hill Companies, Inc. all rights reserved.



# Phân Tích Tiệm Cận Thuật Toán

---

- ▶ Xác định thời gian chạy sử dụng ký hiệu  $O$ -
  - ▶ Tìm số lần các phép toán sơ cấp thực hiện nhiều nhất là hàm của độ lớn dữ liệu đầu vào
  - ▶ Miêu tả hàm này theo ký hiệu  $O$ -
- ▶ Ví dụ:
  - ▶ Thuật toán  $arraySum(A, n)$  chạy nhiều nhất  $(4n + 2)$  phép toán sơ cấp
  - ▶ Ta nói thuật toán  $arraySum(A, n)$  chạy trong thời gian  $O(n)$

# Kỹ Thuật Đánh Giá Thời Gian Chạy

---

- ▶ Thời gian chạy các lệnh
  - ▶ Gán
  - ▶ Lựa chọn
  - ▶ Lặp
- ▶ Không đệ quy so với đệ quy



# Thời Gian Chạy Của Các Lệnh

---

- ▶ Lệnh gán

$X = \langle \text{biểu thức} \rangle$

Thời gian chạy của lệnh gán bằng thời gian thực hiện biểu thức

- ▶ Lệnh lựa chọn

if (điều kiện)  $\rightarrow T_0(n)$

    lệnh 1  $\rightarrow T_1(n)$

else

    lệnh 2  $\rightarrow T_2(n)$

Thời gian:  $T_0(n) + \min(T_1(n), T_2(n))$

# Thời Gian Chạy Của Các Lệnh

---

- ▶ Lệnh lặp: for, while, do-while

$$\sum_{i=1}^{X(n)} (T_0(n) + T_i(n))$$

$X(n)$ : số vòng lặp

$T_0(n)$ : điều kiện lặp

$T_i(n)$ : thời gian thực hiện vòng lặp thứ  $i$

# Phân Tích Hàm Độ Quy

---

- ▶ Định nghĩa đệ quy có 2 phần
  - ▶ Phần cơ sở: định nghĩa một số phần tử đầu tiên trong chuỗi
  - ▶ Phần đệ quy
- ▶ Ví dụ: thời gian tính hàm giai thừa
  - ▶  $T(1) \in O(1)$
  - ▶  $T(n) = T(n - 1) + O(1)$  với  $n > 1$

# Bài Tập

```
1 for (i = 0; i < n; i++)
2     for (j = 0; j < n; j++)
3         a[i][j] = 0;
4 for (i = 0; i < n; i++)
5     a[i][j] = 1;
```

$$T_3 = O(1)$$

$$T_2 = O(n)$$

$$T_{23} = O(n) \times O(1) = O(n)$$

$$T_1 = O(n)$$

$$T_{123} = O(n) \times O(n) = O(n^2)$$

$$T_5 = O(1)$$

$$T_4 = O(n)$$

$$T_{45} = O(n) \times O(1) = O(n)$$

$$T_{12345} = T_{123} + T_{45} = O(n^2) + O(n) = O(n^2 + n) = O(n^2)$$

# Bài Tập

---

$$T_4 = O(1)$$

$$T_6 = O(1)$$

$$T_3 = O(1)$$

$$T_{3456} = O(1)$$

$$T_2 = O(n)$$

$$T_{23456} = O(n) \times O(1) = O(n)$$

$$T_1 = O(n)$$

$$T_{12345} = O(n) \times O(n) = O(n^2)$$

```
1 for (i = 0; i < n; i++)
2     for (j = 0; j < n; j++)
3         if (i == j)
4             a[i][j] = 1;
5         else
6             a[i][j] = 0;
```

# Bài Tập

---

```
1 for (i = 0; i < n; i++)
2     for (j = 0; j < n; j++)
3         a[i][j] = 0;
4 for (i = 0; i < n; i++)
5     a[i][j] = 1;
```

$$T_3 = O(1)$$

$$T_2 = O(n)$$

$$T_{23} = O(n) \times O(1) = O(n)$$

$$T_1 = O(n)$$

$$T_{123} = O(n) \times O(n) = O(n^2)$$

$$T_5 = O(1)$$

$$T_4 = O(n)$$

$$T_{45} = O(n) \times O(1) = O(n)$$

$$T_{12345} = T_{123} + T_{45} = O(n^2) + O(n) = O(n^2 + n) = O(n^2)$$

# Bài Tập

---

```
1 sum = 0
2 for (i = 0; i < n; i++)
3     for (j = i + 1; j <= n; j++)
4         for (k = 1; k < 10; k++)
5             sum = sum + i * j * k;
```

# Bài Tập

---

```
1 sum = 0
2 for (i = 0; i < n; i++)
3     for (j = i + 1; j <= n; j++)
4         for (k = 1; k < m; k++) {
5             x = 2 * y
6             sum = sum + i * j * k;
7         }
```



# Bài Tập

---

```
1 sum = 0
2 thisSum = 0
3 for (i = 0; i < n; i++) {
4     thisSum += a[i];
5     if (thisSum > sum)
6         sum = thisSum;
7     else
8         thisSum = sum;
9 }
```