# GIỚI THIỆU VỀ THUẬT GIẢI

- Thuật ngữ và khái niệm
- Độ phức tạp về thời gian của thuật giải
- Các ví dụ
- Một số công cụ toán học
- Kết luận và lưu ý

# THUẬT NGỮ VÀ KHÁI NIỆM

- Thuật giải là một thủ tục xác định bao gồm một dãy hữu hạn các bước cần thực hiện để thu được lời giải bài toán
- Một thuật giải luôn có một tập dữ liệu đầu vào (input) và một tập dữ liệu đầu ra (output) tương ứng với yêu cầu và lời giải bài toán

# THUẬT NGỮ VÀ KHÁI NIỆM

Có thể mô tả thuật giải bằng:

- Ngôn ngữ tự nhiên (Natural language)
- Mã giả (Pseudocode)
- Ngôn ngữ lập trình cấp cao (High programming languages) như Pascal, C/C++ vv

# THUẬT NGỮ VÀ KHÁI NIỆM

**Ví dụ**: Tìm K trong dãy A[1], A[2], ...., A[n]

Đầu vào: Số K, dãy n số A[1], A[2], ...., A[n]

Đầu ra: Một giá trị logic true hoặc false

```
SequentialSearch(A[1..n], K)
1   for i = 1 to n
2         do  if A[i] == K
3               then return true
4   return false
```

# THUẬT NGỮ VÀ KHÁI NIỆM

Độ phức tạp của thuật giải là chi phí về tài nguyên của hệ thống (chủ yếu là thời gian, bộ nhớ, CPU, đường truyền) cần thiết để thực hiện thuật giải

# THUẬT NGỮ VÀ KHÁI NIỆM

Phân tích thuật giải (Analyzing of Algorithm) là quá trình tìm ra những đánh giá về tài nguyên cần thiết để thực hiện thuật giải

# THUẬT NGỮ VÀ KHÁI NIỆM

Độ phức tạp về thời gian của thuật giải:

- Được qui về đếm số lệnh cần thực thi của thuật giải
- Đó là một hàm T(n) phụ thuộc vào kích thước n của input
- Coi như có một máy trừu tượng (abstract machine) để thực hiện thuật giải

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

- Thời gian tối thiểu để thực hiện thuật giải với kích thước đầu vào n gọi là thời gian chạy tốt nhất (best-case) của thuật giải
- Thời gian nhiều nhất để thực hiện thuật giải với kích thước đầu vào n được gọi là thời gian chạy xấu nhất (worst-case) của thuật giải
- Thời gian trung bình để thực hiện thuật giải với kích thước đầu vào n được gọi là thời gian chạy trung bình (average case) của thuật giải

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

**Ví dụ** Đánh giá độ phức tạp về thời gian của thuật giải

SequentialSearch(A[1..n], K)

```
1   for i = 1 to n
2         do  if A[i] == K
3               then return true
4   return false
```

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

- **Giải** Gọi $\alpha$, $\beta$ và $\gamma$ là thời gian thực hiện của phép gán, phép so sánh và trả về của thuật giải
  - Trường hợp tốt nhất: Nếu A[1] = K, thì $T(n) = \alpha + \beta + \gamma$
  - Trường hợp xấu nhất : Nếu $K \notin \{A[1], A[2], ...., A[n]\}$ thì $T(n) = (n+1)\alpha + n\beta + \gamma$

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

**Giải**

- Trường hợp trung bình: Gọi xác suất tìm kiếm thành công là p (xác suất không tìm thấy là 1-p)
  - Nếu tìm thành công, thì tồn tại i, A[i] = K, với Pr(i) = p/n và thời gian là f(i) = $i\alpha + i\beta + \gamma$
  - Nếu không tìm thấy thì thời gian là $(n+1)\alpha + n\beta + \gamma)$ và xác suất là 1-p

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

**Giải**

- Trường hợp trung bình: Gọi xác suất tìm kiếm thành công là p (xác không tìm thấy là 1-p) khi đó
- $T(n) = \Sigma_{i=1,..n} (i\alpha + i\beta + \gamma)p/n + ((n+1)\alpha + n\beta + \gamma))(1-p)$

  $= (\alpha + \beta)(n+1)p/2 + \gamma p + ((n+1)\alpha + n\beta + \gamma))(1-p)$

✓ Lưu ý: khi p=1 (luôn tìm thấy) $T(n) = (\alpha + \beta)(n+1)/2 + \gamma$

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

Giả sử f và g là các hàm không âm đối số nguyên dương n

- f(n) có độ tăng (bậc) không quá g(n) và viết $f(n) = O(g(n))$ nếu có hằng số $c > 0$ và số nguyên dương N sao cho $f(n) \leq cg(n)$, $\forall\ n \geq N$
- f(n) có độ tăng ít nhất là g(n) và viết $f(n) = \Omega(g(n))$ nếu có hằng số $c > 0$ và số nguyên dương N sao cho $f(n) \geq cg(n)$, $\forall\ n \geq N$
- f(n) có độ tăng là g(n) và viết $f(n) = \Theta(g(n))$ nếu $f(n) = O(g(n))$ và $f(n) = \Omega(g(n))$

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

**Ví dụ** Xét $T(n) = 20n^2 + 9n + 3$.

- $T(n) \leq 20n^2 + 9n^2 + 3n^2 = 32n^2, \forall\ n \geq 1$ vậy

  $T(n) = O(n^2)$, với, $c = 32, N = 1$

- $T(n) \geq 20n^2, \forall\ n \geq 1$ vậy

  $T(n) = \Omega(n^2)$, với, $c = 20, N = 1$

Suy ra $T(n) = \Theta(n^2)$

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

- Nếu $T_1(n) = O(f(n))$ và $T_2(n) = O(g(n))$ thì
  - $T_1(n)+T_2(n) = O(\max(f(n), g(n))$
  - $T_1(n).T_2(n) = O(f(n).g(n))$
  - $c.O(f(n)) = O(f(n))$
  - $O(c) \equiv O(1)$

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

- Nếu thuật giải có thời gian chạy tốt nhất (trung bình, xấu nhất) là T(n) và T(n) = O(g(n)) thì ta nói thời gian chạy tốt nhất (trung bình, xấu nhất) của thuật giải có bậc (độ tăng) không quá g(n) hay thời gian chạy tốt nhất (trung bình, xấu nhất) của thuật giải là O(g(n))

✓ Lưu ý : Phát biểu tương tự cho các ký hiệu $\Omega$ và $\Theta$

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

Lưu ý :

- Bậc (độ tăng) của thời gian chạy càng lớn thì thuật giải càng chậm (chẳng hạn, thuật giải có thời gian chạy $T(n) = O(n^3)$ kém hiệu quả (chậm) hơn thuật giải có thời gian chạy $T(n)=O(n^2)$
- Bỏ qua các lệnh gán chỉ số vòng lặp nếu đánh giá theo O-lớn

# ĐPT THỜI GIAN CỦA THUẬT GIẢI

- Quan hệ về độ tăng của hai hàm t(n) và g(n) thỏa mãn tính chất sau

$$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \begin{cases} 0, & \text{thì t(n) có độ tăng nhỏ hơn g(n)} \\ c, & \text{thì t(n) có độ tăng bằng g(n)} \\ \infty, & \text{thì t(n) có độ tăng lớn hơn g(n)} \end{cases}$$

# CÁC VÍ DỤ

- **Ví dụ 1** Viết và phân tích thuật giải tính gần đúng $e^x$ theo khai triển $e^x \approx 1 + x/1! + x^2/2! + ... + x^n/n!$
- **Giải** ?

# CÁC VÍ DỤ

```
Exp(x, n)
1    s = 1
2    for i = 1 to n
3        do  p = 1
4            for j = 1 to i
5                do p =  p*x/j
6            s = s + p
7    return s
```

# CÁC VÍ DỤ

Gọi $\alpha$, $\gamma$ là thời gian thực hiện lệnh gán và trả về, thì

- $T(n) = \alpha + n\alpha + (1+2+...+n)\alpha + n\alpha + \gamma$
- $T(n) = (2n + 1)\alpha + [n(n+1)/2]\alpha + \gamma$
- $T(n) = O(n^2)$

# CÁC VÍ DỤ

Exp(x, n) - Một thuật giải hiệu quả hơn

```
1   s = 1
2   p = 1
3   for i = 1 to n
4      do  p = p*x/i
5          s = s + p
6   return s
```

# CÁC VÍ DỤ

Phân tích độ phức tạp thuật giải thứ 2

- $T(n) = 2\alpha + 2n\alpha + \gamma$
- $T(n) = 2(n + 1)\alpha + \gamma$
- $T(n) = O(n)$

# CÁC VÍ DỤ

- **Ví dụ 2** Viết và phân tích thuật giải tính giai thừa của số tự nhiên n
- **Giải** ?

# CÁC VÍ DỤ

```
Factorial(n)
1   if n == 0 or n == 1
2         then  return 1
3         else  if n > 1
4                     then  return  n*Factorial(n-1)
```

# CÁC VÍ DỤ

Gọi T(n) là thời gian chạy của thuật giải

$$T(n)=\begin{cases} O(1), & n=0,1 \\ T(n-1)+O(1), & n>1 \end{cases}$$

# CÁC VÍ DỤ

$T(n) = T(n-1) + c$

$= T(n-2)+c +c = T(n-2) + 2c$

$= T(n-3)+c + 2c = T(n-3) + 3c$

....

$= T(n-(n-1)) + (n-1)c = T(1) + (n-1)c = c + (n-1)c = nc$

$T(n) = O(n)$ (tương đương với thuật giải không đệ qui)

# CÁC VÍ DỤ

Giả sử mỗi bước đòi hỏi 1 $\mu$s = $10^{-6}$sec thì với n = 100

- T(n) = O($n^3$) có thời gian chạy T(n) $\approx 100^3.10^{-6} = 1$ (sec)
- T(n) = O($2^n$) có thời gian chạy T(n) $\approx 2^{100}.10^{-6}$(sec) = $2^{100}.10^{-6}$(sec)/(60.60.24.365) $\approx$ 4.106(năm)

# CÁC VÍ DỤ

| Độ phức tạp (tăng dần) | Tên gọi |
|---|---|
| • O(1) | Hằng số |
| • O(lgn) | Logarithm |
| • O(n) | Tuyến tính |
| • O(nlgn) | n log n |

# CÁC VÍ DỤ

| Độ phức tạp (tăng dần) | Tên gọi |
|---|---|
| • $O(n^2)$ | Bậc hai |
| • $O(n^3)$ | Bậc ba |
| • $O(n^m)$ | Đa thức |
| • $O(m^n)$, $m \geq 2$ | Hàm mũ |
| • $O(n!)$ | Giai thừa |

# MỘT SỐ CÔNG CỤ TOÁN

- Phần nguyên sàn (floor), phần nguyên trần (ceiling) của số thực x, ký hiệu lần lượt $\lfloor x \rfloor$ và $\lceil x \rceil$, là các số nguyên thỏa mãn $x-1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x+1$

# MỘT SỐ CÔNG CỤ TOÁN

- Với mọi $x \neq 1$ thì $\Sigma_{k=0,n}\ x^k = (x^{n+1}-1)/(x-1)$
- Với mọi $|x|<1$ thì $\Sigma_{k=0,\infty}\ x^k = 1/(1-x)$

# KẾT LUẬN VÀ LƯU Ý

- Nếu bài toán có thuật giải với thời gian chạy xấu nhất là đa thức, $O(n^m)$, thì bài toán gọi là được giải tốt
- Nếu bài toán không có thuật giải với thời gian chạy xấu nhất là đa thức thì bài toán gọi là khó giải (intractable problem)
- Nếu bài toán khó đến mức không thể xây dựng được thuật giải thì nó được gọi là không giải được (unsolvable problem)

# KẾT LUẬN VÀ LƯU Ý

- Phân tích độ phức tạp thuật giải chủ yếu dựa trên kỹ thuật đếm và biểu diễn hệ thức truy hồi
- Có hai phương pháp chính để giải hệ thức truy hồi: Thay thế lặp và áp dụng công thức giải (chẳng hạn định lý Master)
- Phân tích trường hợp trung bình thường phức tạp hơn và cần thêm các công cụ toán học như lý thuyết xác suất, hàm sinh
- Trong nhiều trường hợp chỉ cần tính thời gian chạy xấu nhất

# ĐỌC VÀ TÌM HIỂU Ở NHÀ

- Đọc chương 1, 2, 3, 4 sách Introduction to Algorithms của Thomas H. Cormen, Charles E. Leiserson, Ronald D. Rivest
- Làm bài tập về nhà chương 1 đã cho trong DS bài tập