CHƯƠNG 1

KỸ THUẬT
PHÂN TÍCH THUẬT TOÁN

Tuần 3: **Tính độ phức tạp của Chương trình đệ quy**

(Phương pháp lời giải tổng quát)

Võ Huỳnh Trâm

Dạng phương trình đệ quy

- Dạng tổng quát của một phương trình đệ quy sẽ là:

$$T(n) = \begin{cases} C(n) \\ F(T(k)) + d(n) \end{cases}$$

- **C(n)**: thời gian thực hiện chương trình ứng với trường hợp đệ quy dừng.
- **F(T(k)):** hàm xác định thời gian theo T(k).
- **d(n)**: thời gian phân chia bài toán và tổng hợp các kết quả.

Giải phương trình đệ quy

- Có 3 phương pháp giải phương trình đệ quy:

(1) Phương pháp truy hồi.

- Triển khai T(n) theo T(n-1), rồi T(n-2), … cho đến T(1) hoặc T(0)

- Suy ra nghiệm

(2) Phương pháp đoán nghiệm.

- Dự đoán nghiệm f(n)

- Áp dụng định nghĩa tỷ suất tăng và chứng minh f(n) là tỷ suất tăng của T(n)

(3) Phương pháp lời giải tổng quát.

Phương pháp Lời giải tổng quát

- Trong mục này, ta sẽ nghiên cứu các phần sau:
  - Bài toán đệ quy tổng quát.
  - Thành lập phương trình đệ quy tổng quát.
  - Giải phương trình đệ quy tổng quát.
  - Các khái niệm về *nghiệm thuần nhất, nghiệm riêng* và *hàm nhân*.
  - Nghiệm của phương trình đệ quy tổng quát khi **d(n) là hàm nhân.**
  - Nghiệm của phương trình đệ quy tổng quát khi **d(n) không phải là hàm nhân.**

Bài toán đệ quy tổng quát

- **Giải thuật Chia để trị**:

  - Phân rã bài toán kích thước n thành a bài toán con có kích thước n/b.

  - Giải các bài toán con và tổng hợp kết quả.

- Với các bài toán con, tiếp tục *Chia để trị* đến khi các bài toán con kích thước 1 $\rightarrow$ *Thuật toán đệ quy*.
- **Giả thiết :**

  - Bài toán con kích thước 1 lấy **một** đơn vị thời gian

  - Thời gian chia bài toán và tổng hợp kết quả để được lời giải của bài toán ban đầu là **d(n)**.

Thành lập phương trình đệ quy tổng quát

- Gọi T(n) là thời gian để giải bài toán kích thước n, thì T(n/b) là thời gian để giải bài toán con kích thước n/b.
- *Khi n = 1*: T(1) = **1**.
- *Khi n > 1*: giải a bài toán con kích thước n/b tốn **aT(n/b)** + thời gian để phân chia bài toán và tổng hợp là **d(n)**.
- Vậy ta có phương trình đệ quy tổng quát có dạng:

$$T(1) = 1$$
$$T(n) = aT(n/b) + d(n)$$

Ví dụ MergeSort

```
void mergeSort(int arr[], int l, int r) {
    if (l < r)   {
        int m = l+(r-l)/2;
        mergeSort(arr, l, m);
        mergeSort(arr, m+1, r);
        merge(arr, l, m, r); }
    }
```

- *Khi n = 1*: tốn C1
- *Khi n > 1*: a = b= 2 (Giải đệ quy 2 bài toán con kích thước n/2, nên cần 2T(n/2)).
- Thời gian phân chia bài toán và tổng hợp kết quả: d(n) = nC2.
- Vậy phương trình đệ quy:

$$T(n) = \begin{cases} C_1 & \text{nếu } n=1 \\ 2T(\frac{n}{2}) + nC_2 & \text{nếu } n>1 \end{cases}$$

Giải phương trình đệ quy tổng quát

$$T(n) = \begin{cases} 1 & \text{neu } n = 1 \\ aT\left(\frac{n}{b}\right) + d(n) & \text{neu } n > 1 \end{cases}$$

Dùng phương pháp truy hồi với giả thiết $n=b^k$

$$T(n) = aT\left(\frac{n}{b}\right) + d(n)$$

$$T(n) = a\left[aT\left(\frac{n}{b^2}\right) + d\left(\frac{n}{b}\right)\right] + d(n) \qquad = a^2T\left(\frac{n}{b^2}\right) + ad\left(\frac{n}{b}\right) + d(n)$$

$$T(n) = a^2\left[aT\left(\frac{n}{b^3}\right) + d\left(\frac{n}{b^2}\right)\right] + ad\left(\frac{n}{b}\right) + d(n)$$

$$= a^3T\left(\frac{n}{b^3}\right) + a^2d\left(\frac{n}{b^2}\right) + ad\left(\frac{n}{b}\right) + d(n)$$

....

$$T(n) = a^iT\left(\frac{n}{b^i}\right) + \sum_{j=0}^{i-1} a^j d\left(\frac{n}{b^j}\right)$$

Giải phương trình đệ quy tổng quát

$$T(n) = a^i T\left(\frac{n}{b^i}\right) + \sum_{j=0}^{i-1} a^j d\left(\frac{n}{b^j}\right)$$

Giả sử $n = b^k$, quá trình suy rộng trên sẽ kết thúc khi i = k.
Khi đó ta được:

$$T\left(\frac{n}{b^i}\right) = T\left(\frac{n}{b^k}\right) = T\left(\frac{b^k}{b^k}\right) = T(1) = 1$$

Thay vào trên ta có:

$$T(n) = a^k + \sum_{j=0}^{k-1} a^j d\left(b^{k-j}\right)$$

Nghiệm thuần nhất và nghiệm riêng

$$T(n) = a^k + \sum_{j=0}^{k-1} a^j d\left(b^{k-j}\right)$$

Nghiệm thuần nhất

Nghiệm riêng

Nghiệm thuần nhất là nghiệm chính xác khi d(n)=0, ∀n : biểu diễn thời gian giải tất cả bài toán con.

Nghiệm riêng phụ thuộc hàm tiến triển, số lượng và kích thước bài toán con: biểu diễn thời gian tạo bài toán con và tổng hợp kết quả.

Nghiệm thuần nhất và nghiệm riêng

$$T(n) = a^k + \sum_{j=0}^{k-1} a^j d\left(b^{k-j}\right)$$

Nghiệm thuần nhất
$a^k = n^{\log_b a}$

Nghiệm riêng

NTN: Do $n=b^k$ nên $n^{\log_b a} = (b^k)^{\log_b a} = b^{\log_b a^k} = a^k$

Nghiệm của phương trình là: MAX(NTN,NR).

Hàm nhân

- Hàm f(n) là **hàm nhân** (multiplicative function) nếu: **$f(m.n) = f(m).f(n)$** $\forall$m,n nguyên dương
- **Ví dụ:**
  - Hàm **f(n) = 1** là hàm nhân vì: $f(m.n)=1.1=1=f(m).f(n)$
  - Hàm **f(n) = n** là hàm nhân vì: $f(m.n)=m.n=f(m).f(n)$
  - Hàm **f(n) = n^k** là hàm nhân vì:

    $$f(m.n) = (m.n)^k = m^k.n^k = f(m).f(n).$$
  - Hàm **f(n) = logn** *không phải* là hàm nhân vì:

  $$f(m.n) = \log(m.n) = \log m+\log n \neq \log m.\log n = f(m).f(n)$$

Tính nghiệm riêng khi d(n) là hàm nhân

- Khi d(n) là **hàm nhân**, ta có:

$d(b^{k-j}) = d(b.b.b\ldots b) = d(b).d(b)\ldots d(b) = [d(b)]^{k-j}$

$$NR = \sum_{j=0}^{k-1} a^j d\left(b^{k-j}\right) = \sum_{j=0}^{k-1} a^j [d(b)]^{k-j} = [d(b)]^k \sum_{j=0}^{k-1} \left[\frac{a}{d(b)}\right]^j = [d(b)]^k \frac{\left[\frac{a}{d(b)}\right]^k - 1}{\frac{a}{d(b)} - 1}$$

$$\text{Hay } NR = \frac{a^k - [d(b)]^k}{\frac{a}{d(b)} - 1}$$

$$NR = \frac{a^k - [d(b)]^k}{\frac{a}{d(b)} - 1}$$

Ba trường hợp

- **Trường hợp 1**: **a > d(b)**

Trong công thức trên: $a^k > [d(b)]^k$

Theo quy tắc tính độ phức tạp:

$$NR \text{ là } O(a^k) = O(n^{\log_b a}) = NTN.$$

Do đó **$T(n) = O(n^{\log_b a})$**

<u>Nhận xét</u> : T(n) chỉ phụ thuộc vào a, b mà không phụ thuộc hàm tiến triển d(n) → Cải tiến thuật toán = Giảm a : giảm số bài toán con.

Ba trường hợp

$$NR = \frac{a^k - [d(b)]^k}{\frac{a}{d(b)} - 1}$$

- **Trường hợp 2**: **$a < d(b)$**

  Trong công thức trên: $[d(b)]^k > a^k$

  Theo quy tắc tính độ phức tạp :

  NR là $O([d(b)]^k) = O(n^{\log_b d(b)}) >$ NTN.

  Do đó **$T(n) = O(n^{\log_b d(b)})$**

  Nhận xét : T(n) phụ thuộc vào b và hàm tiến triển d(b) $\rightarrow$ Cải tiến thuật toán = Giảm d(b) : cải tiến việc phân chia bài toán và tổng hợp kết quả.

Ba trường hợp

$$NR = \frac{a^k - [d(b)]^k}{\frac{a}{d(b)} - 1}$$

Trường hợp 3: **a = d(b)**

Công thức trên không xác định nên phải tính trực tiếp nghiệm riêng:

$$NR = [d(b)]^k \sum_{j=0}^{k-1} \left[\frac{a}{d(b)}\right]^j = a^k \sum_{j=0}^{k-1} 1 = a^k k \qquad (\text{do } a = d(b))$$

Do $n = b^k$ nên $k = \log_b n$ và $a^k = n^{\log_b a}$.

Vậy NR là $n^{\log_b a}\log_b n$ > NTN.

Do đó **$T(n) = O(n^{\log_b a}\log_b n)$**

Tính nghiệm riêng khi d(n) không phải là hàm nhân

- Trong trường hợp hàm tiến triển d(n) không phải là hàm nhân thì không thể áp dụng các công thức ứng với ba trường hợp nói trên mà chúng ta phải **tính trực tiếp** NR, sau đó so sánh với NTN và lấy **MAX(NR,NTN).**

Quy tắc chung để giải phương trình đệ quy

- **Lưu ý khi giải một phương trình đệ quy cụ thể**:

(1) Xem phương trình có thuộc **dạng phương trình tổng quát** không ?

(2) Nếu **có**, xem hàm tiến triển d(n) có **dạng hàm nhân** không?

a) Nếu **<u>có</u>**: xác định a, d(b); so sánh a, d(b) để vận dụng một trong ba trường hợp trên.

b) Nếu **<u>không</u>** : tính trực tiếp nghiệm để so sánh.

(3) Suy ra nghiệm của phương trình = Max(NTN,NR)

Ví dụ 1. GPT với T(1) = 1 và

$$1/\ \ T(n) = 4T\left(\frac{n}{2}\right) + n$$

- Phương trình đã cho có dạng phương trình tổng quát.
- d(n)=n là hàm nhân.
- a = 4 và b = 2.
- d(b) = b = 2 < a (Trường hợp 1)

$\rightarrow$ $T(n) = O(n^{\log_b a}) = O(n^{\log 4}) = O(n^2)$.

Ví dụ 2. GPT với T(1) = 1 và

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

- Phương trình đã cho có dạng phương trình tổng quát.
- $d(n)=n^2$ là hàm nhân.
- a = 4 và b = 2.
- $d(b) = b^2 = 4 = a$ (Trường hợp 3)

$\rightarrow T(n) = O(n^{\log_b a}\log_b n) = O(n^{\log 4}\log n) =(n^2\log n).$

Ví dụ 3. GPT với T(1) = 1 và

$$3/\ \ T(n) = 4T\left(\frac{n}{2}\right) + n^3$$

- Phương trình đã cho có dạng phương trình tổng quát.
- $d(n)=n^3$ là hàm nhân.
- $a = 4$ và $b = 2$.
- $d(b) = b^3 = 8 > a$ (Trường hợp 2)

$\rightarrow T(n) = O(n^{\log_b d(b)}) = O(n^{\log 8}) = O(n^3)$.

Ví dụ 4. GPT với T(1) = 1 và

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

- PT thuộc dạng phương trình tổng quát nhưng
  d(n) = nlogn *không phải là một hàm nhân*.
- a = 2 và b = 2
- NTN = $n^{\log_b a} = n^{\log 2} = n$
- Do d(n) = nlogn không phải là hàm nhân nên ta phải tính nghiệm riêng bằng cách xét trực tiếp

Ví dụ 4. (tt)

$$NR = \sum_{j=0}^{k-1} a^j d\left(b^{k-j}\right) = \sum_{j=0}^{k-1} 2^j 2^{k-j} \log 2^{k-j}$$

$$= 2^k \sum_{j=0}^{k-1} (k - j) = 2^k \frac{k(k+1)}{2} = O(2^k k^2)$$

- Theo cách giải phương trình đệ quy tổng quát thì $n = b^k$ nên $k = \log_b n$, ở đây do $b = 2$ nên $2^k = n$ và $k = \log n$,
- $NR = O(2^{\log n} \log^2 n) = O(n\log^2 n) > NTN = n$

$\rightarrow$ $T(n) = O(n\log^2 n)$.

Một số tổng thông dụng

$S = 1 + 2 + 3 + \ldots + n = n(n+1)/2 \approx n^2/2$

$S = 1 + 2^2 + 3^2 + \ldots + n^2 = n(n+1)(2n+1)/6 \approx n^3/3$

$S = 1 + a + a^2 + a^3 + \ldots + a^n = (a^{n+1} - 1)/(a-1)$

Nếu $0 < a < 1$ thì

$S \leq 1/(1-a)$

và khi $n \to \infty$ thì

S tiến về $1/(1-a)$

$S = 1 + 1/2 + 1/3 + \ldots + 1/n = \ln(n) + \gamma$

Hằng số Euler $\gamma \approx 0.577215665$

$S = 1 + 1/2 + 1/4 + 1/8 + \ldots + 1/2^n + \ldots \approx 2$

Bài tập 4-1. GPT với T(1) = 1 và

$$T(n) = T\left(\frac{n}{2}\right) + 1$$

- Phương trình đã cho có dạng phương trình tổng quát.
- d(n)=1 là hàm nhân.
- a = 1 và b = 2.
- d(b) = 1 = a.

$\rightarrow$ $T(n) = O(n^{\log_b a}\log_b n) = O(n^{\log 1}\log n) = O(\log n)$.

Bài tập 4-2. GPT với T(1) = 1 và

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

- Phương trình đã cho có dạng phương trình tổng quát.
- d(n)=logn không phải là hàm nhân.
- a = 2 và b = 2
- NTN = $O(n^{\log_b a}) = O(n^{\log 2}) = O(n)$.
- Tính trực tiếp nghiệm riêng.

Bài tập 4-2. (tt)

$$NR = \sum_{j=0}^{k-1} a^j d(b^{k-j}) = \sum_{j=0}^{k-1} 2^j \log 2^{k-j}$$

$$NR = \sum_{j=0}^{k-1} 2^j (k-j) = \sum_{j=0}^{k-1} k2^j - \sum_{j=0}^{k-1} j2^j$$

$$NR = O(k\sum_{j=0}^{k-1} 2^j) = O(k\frac{2^k - 1}{2-1})$$

$$NR = O(k2^k) = O(n\log n) > n = NTN$$

$$T(n) = O(n\log n)$$

CANTHO UNIVERSITY

Bài tập 8.

$$C^k_n = \begin{cases} 1 & \text{neu } k = 0 \text{ hoac } k = n \\ C^{k-1}_{n-1} + C^k_{n-1} & \end{cases}$$

- Gọi T(n) là thời gian để tính C^k_n
- Thì thời gian để tính C^k_{n-1} là T(n-1)
- Khi n=1 thì k=0 hoặc k=1 => chương trình trả về giá trị 1, tốn O(1) = C_1
- Khi n>1, trong trường hợp xấu nhất, chương trình phải làm các việc:
  - Tính C^k_{n-1} và C^{k-1}_{n-1} : tốn 2T(n-1).
  - Thực hiện phép cộng và trả kết quả: tốn C_2

a) Ta có phương trình: T(1)=C_1 và T(n)=2T(n-1) +C_2

CANTHO UNIVERSITY

Bài tập 8. (tt)

b) Giải phương trình $T(n) = 2T(n-1) + C_2$

- $T(n) = 2[2T(n-2)+C_2] + C_2$

  $= 4T(n-2) + 3C_2$

  $= 4[2T(n-3) + C_2] + 3C_2$

  $= 8T(n-3) + 7C_2$

  ……

  $= 2^iT(n-i) + (2^i-1)\, C_2$

- $T(n) = 2^{n-1}C_1 + (2^{n-1}-1)\, C_2$

  $= (C_1 + C_2)2^{n-1} - C_2 = O(2^n)$