HW1: Data Structures and Algorithms

Nguyễn Tiến Đạt

24th September 2024

1 Sơ lược về độ phức tạp thuật toán:

Độ phức tạp thuật toán là thời gian để thực thi thuật toán

2 Một số ví dụ

Example 1. Find upper bound for:

a) $f(n) = 3n + 5$
Ta có: $f(n) = 3n + 5 \leq 8n \quad$ với mọi $n \geq 1$
$\Rightarrow$ Độ phức tạp thuật toán là $O(n)$.

b) $f(n) = 4n^2 + 3$
Ta có: $f(n) = 4n^2 + 3 \leq 5n^2 \quad$ với mọi $n \geq 1$
$\Rightarrow$ Độ phức tạp thuật toán là $O(n^2)$.

c) $f(n) = n^4 + 100n^2 + 80$
Ta có: $f(n) = n^4 + 100n^2 + 80 \leq 2n^4 \quad$ với mọi $n \geq 10$
$\Rightarrow$ Độ phức tạp thuật toán là $O(n^4)$.

d) $f(n) = 5n^3 - 5n^2$
Ta có: $f(n) = 5n^3 - 5n^2 \leq 5n^3 \quad$ với mọi $n \geq 1$
$\Rightarrow$ Độ phức tạp thuật toán là $O(n^3)$.

e) $f(n) = 502$
Ta có: $f(n) = 502 \leq 502 \quad$ với mọi $n \geq 1$
$\Rightarrow$ Độ phức tạp thuật toán là $O(1)$.

Example 2. Tính $S = \frac{n*(n-1)}{2}$

$S = \frac{n*(n-1)}{2} \leq n^2$ với mọi $n \geq 0$

$\Rightarrow$ S có độ phức tạp $O(n)$

Example 3.

```
for i in range(0, n):
    print('current number', i)
```

Chương trình chỉ chạy một vòng lặp n lần ⇒ độ phức tạp $O(n)$

Example 4.

```
for i in range(0, n):
    print('Currrent number', i)
    break
```

Chương trình chạy một lần và break ⇒ độ phức tạp $O(1)$

Example 5.

```
def function(n):
    i = 1
    while i <= n:
        i = i * 2
        print(i)
function(100)
```

Chương trình chạy vòng lặp while với điều kiện phụ thuộc vào i = 1 cho đến khi i <= n (n là đối số đầu vào), sau mỗi một lần lặp i tăng gấp đôi.
⇒ độ phức tạp $O(n)$

Example 6.

```
for i in range(0, n):
    for j in range(0, n):
        print("Value of i, j", i, j)
```

Chương trình chạy 2 vòng 'for' ⇒ độ phức tạp $O(n)$

Example 7.

```
public void function(int n){
    int i, j, k, count = 0;
    for(i = n/2; i <= n; i++){
        for(j = 1; j + n/2 <= n; j++){
            for(k = 1; k <= n; k = k * 2){
                count++;
            }
        }
    }
}
```

⇒ độ phức tạp $O(n^3)$

Example 8.

```
public void function(int n){
    int i,j,k count = 0;
    for(i = n/2; i <= n; i++)
        for(j = 1; j <= n; j = 2 * j)
            for(k = 1; k <= n; k = k * 2)
                count++;
}
```

⇒ độ phức tạp $O(n^3)$

3 Bài tập vận dụng

Exercise 1. Độ phức tạp của thuật toán là gì ?

a) $T(n) = nlogn + 3n + 2$
Ta có:
$T(n) = nlogn + 3n + 2 \leq 3nlogn$ với mọi n ≤ 10
$\Rightarrow$ độ phức tạp $O(nlogn)$

b) $T(n) = nlog(n!) + 5n^2 + 7$
Ta có:
$nlog(n!) = nlog(1.2.3...n) = nlog1 + nlog2 + ... + nlogn \leq n(nlogn) = n^2logn$
$\Rightarrow$ độ phức tạp $O(n^2logn)$

c) $T(n) = 1000n + 0.01n^2$
Ta có:
$1000n + 0.01n^2 \leq 2n^2$ với mọi n ≤ 600
$\Rightarrow$ độ phức tạp $O(n^2$

d) $T(n) = 100nlog + n^3 + 100n$
Ta có:
$100nlogn + n^3 + 100n \leq 100n^3$ với mọi n ≤ 1
$\Rightarrow$ độ phức tạp $O(n^3$

e) $T(n) = 0.01nlogn + n(logn)^2$
Ta có:
$0.01nlogn + n(logn)^2 \leq 2n(logn)^2$ với mọi n ≤ 1
$\Rightarrow$ độ phức tạp $O(n.(logn)^2$

Exercise 2. Xác định độ phức tạp thuật toán của chương trình:

a) Độ phức tạp $O(n)$

b) Độ phức tạp $O(n)$

c) Độ phức tạp $O(n^2)$

d) Độ phức tạp $O(n)$

e) Độ phức tạp $O(n^3)$

4 Bài tập

- **(Master Theorem)** Nếu n là lũy thừa của b, thì nghiệm của đệ quy

$$T(n) = \begin{cases} 1, & \text{if } n \leq 1 \\ aT\left(\frac{n}{b}\right) + n^d, & \text{if } n > 1, a \geq 1, b > 1, d \geq 0 \end{cases}$$

là:

$$T(n) = \begin{cases} \mathcal{O}(n^d), & \text{if } a < b^d \\ \mathcal{O}(n^d \log n), & \text{if } a = b^d \\ \mathcal{O}(n^{\log_b a}), & \text{if } a > b^d. \end{cases}$$

Example 9. Xác định độ phức tạp của thuật toán:

$$T(n) = \begin{cases} 2T(n-1) - 1, & \text{if } n > 0. \\ 1, & \text{otherwise.} \end{cases}$$

Lời giải:

Với $n > 0$, ta có:

$$T(n) = 2T(n-1) - 1$$

Thay giá trị của $T(n-1)$ vào $T(n)$:

$$\begin{aligned} T(n) &= 2\left(2T(n-2) - 1\right) - 1 \\ &= 2^2 T(n-2) - 2^1 - 1 \end{aligned}$$

Tiếp tục thay $T(n-2)$:

$$\begin{aligned} T(n) &= 2(2T(n-2) - 1) - 1 \\ &= 2\left(2\left(2T(n-3) - 1\right) - 1\right) - 1 \\ &= 2^3 T(n-3) - 2^2 - 2^1 - 1 \end{aligned}$$

Như vậy, công thức tổng quát là:

$$T(n) = 2^k T(n-k) - \sum_{i=0}^{k-1} 2^i$$

Dễ thấy $\sum_{i=0}^{k-1} 2^i$ là một tổng cấp số nhân có $u(1) = 1$ và $q = 2$. Suy ra:

$$T(n) = 2^k T(n-k) - (2^k - 1)$$

Khi $n - k = 0$ hay $k = n$, ta có:

$$T(n) = 2^n T(0) - (2^n - 1) = 2^n - (2^n - 1) = 1$$

Vậy độ phức tạp của hàm là:

$$T(n) = O(2^n)$$

Tại sao độ phức tạp là $O(2n)$?

- Mỗi lần giảm n xuống $n-1$, số lời gọi đệ quy nhân đôi, dẫn đến tăng trưởng theo hàm 2^n.
- Thuật toán này thực hiện một số lượng cấp số nhân lời gọi đệ quy, dù kết quả cuối cùng là 1.
- Vì thế, độ phức tạp thời gian chính là số lượng lời gọi đệ quy, chứ không phải giá trị trả về cuối cùng.

Example 10. Xác định độ phức tạp của thuật toán:

$$T(n) = \begin{cases} 3T(n-1), & \text{if } n > 0. \\ 1, & \text{otherwise.} \end{cases}$$

Lời giải:

Với $n > 0$, ta có:

$$T(n) = 3T(n-1)$$

Thay giá trị của $T(n-1)$ vào $T(n)$:

$$\begin{aligned} T(n) &= 3\left(3T(n-2)\right) \\ &= 3^2T(n-2) \end{aligned}$$

Tiếp tục thay $T(n-2)$:

$$\begin{aligned} T(n) &= 3\left(3T(n-2)\right) \\ &= 3\left(3\left(3T(n-3)\right)\right) \\ &= 3^3T(n-3) \end{aligned}$$

Như vậy, công thức tổng quát là:

$$T(n) = 3^kT(n-k)$$

Khi $n - k = 0$ hay $k = n$, ta có:

$$T(n) = 3^nT(0) = 3^n \cdot 1 = 3^n$$

Vậy độ phức tạp của hàm là:

$$T(n) = O(3^n)$$

Example 11. Giải công thức đệ quy để xác định độ phức tạp thuật toán.

1. $T(1) = 1$, and for all $n \geq 2$ là lũy thừa của 2, $T(n) = 2T\left(\frac{n}{2}\right) + 6n - 1$.
2. $T(1) = 2$, and for all $n \geq 2$ là lũy thừa của 3, $T(n) = 4T\left(\frac{n}{3}\right) + 3n - 5$.
3. $T(1) = 1$, and for all $n \geq 2$ là lũy thừa của 2, $T(n) = 3T\left(\frac{n}{2}\right) + n^2 - n$.

Bài làm

1.

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 6n - 1 & \text{if } n \geq 2, \\ 1 & \text{otherwise.} \end{cases}$$

Cách 1: Sử dụng định lý tổng quát với

$$a = 2, \quad b = 2, \quad d = 1.$$

Ta có:

$$\begin{aligned} & a = b^d \\ \Rightarrow\ & T(n) = O(n \log n) \end{aligned}$$

Cách 2: Sử dụng cây đệ quy

Ta có:

$$T(\frac{n}{2}) = 2T(\frac{n}{4}) + 6(\frac{n}{2}) - 1$$
$$T((\frac{n}{4}) = 2T(\frac{n}{8}) + 6(\frac{n}{4}) - 1$$

Khi thay $T(\frac{n}{2})$ vào $T(n)$:

$$T(n) = 2(2T(\frac{n}{4}) + 6(\frac{n}{2}) - 1) + 6n - 1$$
$$= 2^2T(\frac{n}{2^2}) + 2.(6n) - 2 - 1$$

Tiếp tục thay $T(\frac{n}{4})$:

$$T(n) = 2[2[2T(\frac{n}{8}) + 6(\frac{n}{4}) - 1] + 6(\frac{n}{2}) - 1] + 6n - 1$$
$$= 2^3T(\frac{n}{2^3}) + 3(6n) - 4 - 2 - 1$$

Dễ dàng thu được tổng quát:

$$T(n) = 2^kT(\frac{n}{2^k}) + k(6n) - (\sum_{i=0}^{k-1} 2^i)$$
$$= 2^kT(\frac{n}{2^k}) + k(6n) - 2^k + 1$$

2.

$$T(n) = \begin{cases} 4T\left(\frac{n}{3}\right) + 3n - 5 & \text{if } n \geq 3, \\ 2 & \text{otherwise.} \end{cases}$$

Cách 1: Sử dụng định lý tổng quát với

$$a = 4, \quad b = 3, \quad d = 1.$$

Ta có:

$$a > b^d \quad (\log_3 4 \approx 1.2619)$$
$$\Rightarrow T(n) = O(n^{\log_3 4}) = O(n^{1.2619})$$

Cách 2: Sử dụng cây đệ quy (Tương tự câu 1.)

3.

$$T(n) = \begin{cases} 3T\left(\frac{n}{2}\right) + n^2 - n & \text{if } n \geq 2, \\ 1 & \text{otherwise.} \end{cases}$$

Cách 1: Sử dụng định lý tổng quát với

$$a = 3, \quad b = 2, \quad d = 2.$$

Ta có:

$$a < b^d \quad (\log_2 3 \approx 1.585)$$
$$\Rightarrow T(n) = O(n^2)$$

Cách 2: Sử dụng cây đệ quy (Tương tự câu 1.)

Exercise 3: Xác định độ phức tạp thuật toán

1.

$$T(n) = \begin{cases} 3T(n-1) + 2 & \text{if } n \geq 2, \\ 1 & \text{otherwise.} \end{cases}$$

Cách 1: Sử dụng phương pháp cây đệ quy.

Ta phân tích cây đệ quy cho công thức $T(n) = 3T(n-1) + 2$. Ở mỗi bước, độ phức tạp giảm dần về $T(1)$, với mỗi lần chia làm ba nhánh và cộng thêm hằng số. Tổng hợp độ phức tạp qua các mức của cây, ta có:

$$T(n) = \mathcal{O}(3^n).$$

2.

$$T(n) = \begin{cases} T(n-1) + 2n - 3 & \text{if } n \geq 2, \\ 3 & \text{otherwise.} \end{cases}$$

Cách 1: Sử dụng phương pháp cây đệ quy.

Mỗi bước đệ quy chỉ thêm vào các bậc tuyến tính $2n - 3$, tổng độ phức tạp là tổng của các bậc này khi đệ quy giảm dần về $T(1)$. Suy ra:

$$T(n) = \mathcal{O}(n^2).$$

3.

$$T(n) = \begin{cases} 2T(n-1) + n - 1 & \text{if } n \geq 2, \\ 1 & \text{otherwise.} \end{cases}$$

Cách 1: Sử dụng phương pháp cây đệ quy.

Mỗi lần đệ quy, hàm được nhân đôi và cộng thêm các giá trị tuyến tính. Tổng hợp qua các mức, ta có:

$$T(n) = \mathcal{O}(2^n).$$

4.

$$T(n) = \begin{cases} 2T(n-1) + 3n + 1 & \text{if } n \geq 2, \\ 5 & \text{otherwise.} \end{cases}$$

Cách 1: Sử dụng phương pháp cây đệ quy.

Độ phức tạp của công thức này là tổng của các bước tăng tuyến tính qua các mức nhân đôi trong cây đệ quy, suy ra:

$$T(n) = \mathcal{O}(2^n).$$

5.

$$T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 3n + 2 & \text{if } n \geq 2, \\ 4 & \text{otherwise.} \end{cases}$$

Cách 1: Sử dụng định lý tổng quát với:

$$a = 2, \quad b = 2, \quad d = 1.$$

Ta có:

$$a = b^d$$

$$\Rightarrow T(n) = \mathcal{O}(n \log n).$$