PHÂN TÍCH ĐỘ PHỨC TẠP THUẬT TOÁN KHÔNG ĐỆ QUY

Nhóm 8: Cẩm Giang và Ngọc Quân

October 2023

Bài 1: Trong trường hợp trung bình, thuật toán đó có độ phức tạp thời gian là $\theta(NlogN)$:
Nhận xét:

- Trường hợp tốt nhất: bé hơn hoặc bằng tiệm cận trên (O)
- Trường hợp xấu nhất: Lớn hơn hoặc bằng tiệm cận dưới (Ω)

a. Trong trường hợp xấu nhất, độ phức tạp thời gian của thuật toán là $\Omega(N^3)$
Độ phức tạp trong trường hợp xấu nhất không thể tốt hơn trường hợp trung bình nên nó là $\Omega(NlogN)$. Mà $\Omega(N^3)$ nằm trong $\Omega(NlogN)$. Do đó, trong trường hợp xấu nhất, độ phức tạp có thể là $\Omega(N^3)$.
Mặt khác, còn có các hàm khác nằm giữa N^3 và $NlogN$, như N^2. Cũng có thể trong trường hợp xấu nhất, độ phức tạp có thể là $\Omega(N^2)$.
=> Nhận định trên có thể **đúng hoặc sai**.

b. Trong trường hợp xấu nhất, độ phức tạp thời gian của thuật toán là O(N)
Độ phức tạp trong trường hợp xấu nhất không thể tốt hơn trường hợp trung bình nên nó là $\Omega(NlogN)$. Mà $O(N)$ không nằm trong $\Omega(NlogN)$.
=> Nhận định trên **sai**.

c. Trong trường hợp tốt nhất, độ phức tạp thời gian của thuật toán là $\Omega(N^2)$
Độ phức tạp trong trường hợp tốt nhất không thể tệ hơn trường hợp trung bình nên nó là $O(NlogN)$. Mà $\Omega(N^2)$ Không nằm trong $O(NlogN)$
=> Nhận định trên **sai**.

Bài 2: Tính độ phức tạp của đoạn code sau:

```
for i in range ( n // 2 , n + 1 ) :          (1)
    j = 1
    while j <= n :                           (2)
        k = 1
        while k <= n :                       (3)
            count += 1
            k = k * 2
        j = j * 2
```

Các bước tính độ phức tạp thuật toán:
- Input size: n
- Phép toán trọng tâm:

```
count += 1
k = k * 2
```

- Ta thấy đoạn code trên có **3 vòng lặp** lồng nhau:

- **Vòng while (3)**: vòng lặp này bắt đầu với k = 1 và tiếp tục cho đến khi k không còn nhỏ hơn hoặc bằng n nữa. Mỗi lần lặp, k được nhân đôi => có $log_2(n)$ lần lặp.
- **Vòng while (2)**: Vòng lặp này bắt đầu với j = 1 và tiếp tục cho đến khi j không còn nhỏ hơn hoặc bằng n nữa. Mỗi lần lặp, j được nhân đôi => có $log_2(n)$ lần lặp.
- **Vòng for (1)**: Vòng lặp này chạy từ $\frac{n}{2}$ đến n, bao gồm cả $\frac{n}{2}$ và n, vì vậy có tổng cộng $\frac{n}{2} + 1$ lần lặp.

=> Tổng số lần phép tính trọng tâm thực thi: $(\frac{n}{2} + 1) * (log_2(n)) * (log_2(n))$

=> Áp dụng một số qui tắc ước lượng ta được độ phức tạp thuật toán là:

$$O(n(log_2(n))^2)$$

——— HẾT ———