CS112.O11.KHTN - Homework Solutions

Group 2 - Hoàng Long, Cẩm Nguyên

October 2023

1 Bài 1:

Sau buổi thảo luận "Phân tích độ phức tạp của thuật toán không đệ quy", Hoàng Long và Cẩm Nguyên đã áp dụng các kiến thức thu được để phân tích một thuật toán bí mật. Chỉ biết rằng, trong trường hợp trung bình, thuật toán đó có độ phức tạp thời gian là

$$\Theta(NlogN)$$

Với mỗi câu dưới đây, hãy nhận xét nó "Đúng"hay "Sai", hay "Có thể đúng hoặc sai". Giải thích lí do.

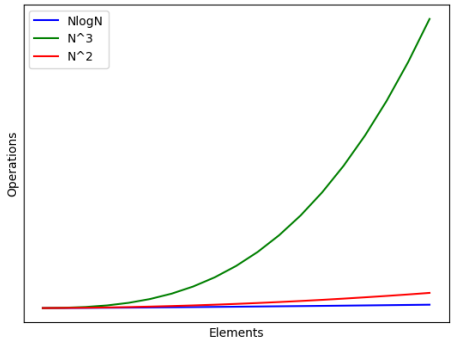
*** Nhận xét:**

- Độ phức tạp thời gian trong trường hợp trung bình cung cấp một giới hạn trên cho trường hợp tốt nhất. Do đó, trường hợp tốt nhất không thể tệ hơn trường hợp trung bình, mà phải là $O(NlogN)$

- Độ phức tạp thời gian trong trường hợp trung bình cung cấp một giới hạn dưới cho trường hợp xấu nhất. Do đó, trường hợp xấu nhất không thể tốt hơn trường hợp trung bình, mà phải là $\Omega(NlogN)$

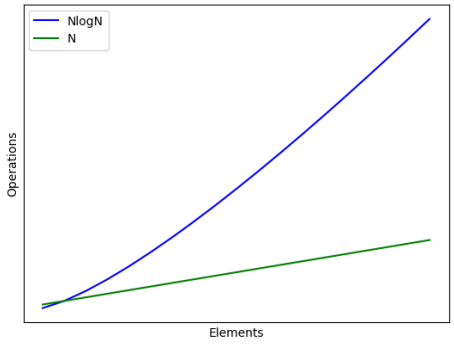
a. Trong trường hợp xấu nhất, độ phức tạp thời gian của thuật toán là $\Omega(N^3)$

Đáp án: Có thể đúng hoặc sai.
Như nhận xét ban đầu, trong trường hợp xấu nhất, độ phức tạp là $\Omega(NlogN)$. Mà N^3 có giới hạn dưới nghiêm ngặt là $NlogN$. Do đó, có thể trong trường hợp xấu nhất, độ phức tạp là $\Omega(N^3)$.
Mặt khác, còn có các hàm khác xếp chặt giữa N^3 và $NlogN$, như N^2. Cũng có thể trong trường hợp xấu nhất, độ phức tạp không phải $\Omega(N^3)$ mà là một đáp án khác, ví dụ $\Omega(N^2)$.



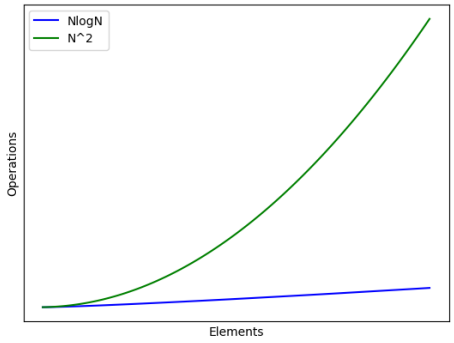
b. Trong trường hợp xấu nhất, độ phức tạp thời gian của thuật toán là $O(N)$

Đáp án: Sai
Như nhận xét ban đầu, trong trường hợp xấu nhất, độ phức tạp là $\Omega(NlogN)$, do đó, không thể nào là $O(N)$ được.



c. Trong trường hợp tốt nhất, độ phức tạp thời gian của thuật toán là $\Omega(N^2)$

Đáp án: Sai
Như nhận xét ban đầu, trong trường hợp tốt nhất, độ phức tạp là $O(NlogN)$, do đó, không thể nào là $\Omega(N^2)$ được.



2 Bài 2:

Cho đoạn code Python sau, tính độ phức tạp thời gian và giải thích cách tính.

```
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
```

Đáp án: $O(nlog^2(n))$

Đoạn code gồm 3 vòng lặp lồng nhau:

- Vòng lặp (1) cho i chạy từ n//2 đến n, chạy khoảng $\frac{n}{2}$ lần. Độ phức tạp của (1) là $O(\frac{1}{2}n)$, áp dụng quy tắc bỏ hằng số, độ phức tạp là $O(n)$.
- Vòng lặp (2) cho j chạy từ 1 đến n, nhưng giá trị của j ở mỗi lần sẽ tăng gấp đôi, nên (2) chạy $log(n)$ lần. Độ phức tạp của (2) là $O(log(n))$
- Vòng lặp (3) cho k chạy từ 1 đến n, nhưng giá trị của k ở mỗi lần sẽ tăng gấp đôi, nên (3) chạy $log(n)$ lần. Độ phức tạp của (3) là $O(log(n))$

Áp dụng quy tắc nhân, độ phức tạp của đoạn code là $O(n) * O(log(n)) * O(log(n)) = O(nlog^2(n))$.