LAB 2: Phân tích thuật toán

1 Phân tích độ phức tạp thời gian của thuật toán

1.1

a. n^2

(a) Khi tăng gấp đôi kích thước đầu vào:

$$\frac{(2n)^2}{n^2} = 4$$

$\Rightarrow$ Thuật toán chạy chậm hơn **4 lần**.

(b) Khi tăng kích thước đầu vào thêm 1:

$$\frac{(n+1)^2}{n^2} = 1 + \frac{2}{n} + \frac{1}{n^2}$$

Với n lớn $\to 1 \Rightarrow$ Không đáng kể.

b. n^3

(a) Khi tăng gấp đôi kích thước đầu vào:

$$\frac{(2n)^3}{n^3} = 8$$

$\Rightarrow$ Thuật toán chạy chậm hơn **8 lần**.

(b) Khi tăng kích thước đầu vào thêm 1:

$$\frac{(n+1)^3}{n^3} = 1 + \frac{3}{n} + \frac{3}{n^2} + \frac{1}{n^3}$$

Với n lớn $\to 1 \Rightarrow$ Không đáng kể.

c. $100n^2$

(a) Khi tăng gấp đôi kích thước đầu vào:

$$\frac{(100(2n))^2}{100n^2} = 4$$

$\Rightarrow$ Thuật toán chạy chậm hơn **4 lần**.

(b) Khi tăng kích thước đầu vào thêm 1:

$$\frac{100(n+1)^2}{n^2} = 100 + \frac{200}{n} + \frac{100}{n^2}$$

Với n lớn $\to 1 \Rightarrow$ Không đáng kể.

d. $n \log n$

(a) Khi tăng gấp đôi kích thước đầu vào:

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

Với n lớn $\to 2 \Rightarrow$ Thuật toán chạy chậm hơn **gần 2 lần**.

(b) Khi tăng kích thước đầu vào thêm 1:

$$\frac{(n+1)\log(n+1)}{n\log n} = 1 + \frac{\log(n+1)}{\log n} + \frac{1}{n\log n}$$

Với n lớn $\to 1 \Rightarrow$ Không đáng kể.

e. 2^n

(a) Khi tăng gấp đôi kích thước đầu vào:

$$\frac{2^{2n}}{2^n} = 2^n$$

Với n lớn $\to$ rất lớn $\Rightarrow$ Thuật toán chạy chậm hơn **rất nhiều lần**.

(b) Khi tăng kích thước đầu vào thêm 1:

$$\frac{2^{n+1}}{2^n} = 2$$

$\Rightarrow$ Thuật toán chạy chậm hơn **2 lần**.

1.2

a. Phát biểu định nghĩa chính thức của ký pháp Big-O:

$$\exists\, c > 0,\, n_0 > 0 \text{ sao cho } T(n) \leq c \cdot f(n), \quad \forall n \geq n_0.$$

b. Cho $f(n)$ và $g(n)$ là các hàm số dương. Sử dụng định nghĩa cơ bản của ký pháp Big-O, chứng minh rằng:

$$\max(f(n), g(n)) = O(f(n) + g(n)).$$

Chứng minh: Với mọi n, ta có:

$$\max(f(n), g(n)) \leq f(n) + g(n).$$

Do đó, $\exists\ c > 0$ sao cho:

$$\max(f(n), g(n)) \leq c \cdot (f(n) + g(n)).$$

Theo định nghĩa:

$$\max(f(n), g(n)) = O(f(n) + g(n)).$$

c. Chứng minh rằng mệnh đề sau là vô nghĩa: *"Thời gian chạy của thuật toán A ít nhất là $O(n^2)$".*

O là ký pháp mô tả cận trên (upper bound). Do đó, mệnh đề trên không có ý nghĩa vì nó không xác định được cận dưới (lower bound) của thời gian chạy thuật toán A.

1.3

a. Chứng minh hoặc phản bác mệnh đề sau:

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

Chứng minh: Ta có:

$$2^{n+1} = 2 \cdot 2^n \leq c \cdot 2^n, \quad \text{với } c = 2,\, n_0 = 1.$$

$\Rightarrow$ **đpcm**.

b. Chứng minh hoặc phản bác mệnh đề sau:

$$2^{2n} = O(2^n)$$

Phản bác: Ta có:

$$2^{2n} = (2^n)^2 \gg c \cdot 2^n, \quad \text{với mọi } c > 0,\, n_0 > 0.$$

⇒ **Khác hoàn toàn**.

c. Chứng minh hoặc phản bác mệnh đề sau:
Nếu $f(n) = O(g(n))$ thì

$$\log f(n) = O(\log g(n)), \quad \text{với giả thiết } f(n) > 1, g(n) > 1.$$

Chứng minh: Theo định nghĩa Big-O, ta có:

$$f(n) \leq c \cdot g(n), \quad \forall n \geq n_0.$$

Lấy log hai vế, ta được:

$$\begin{aligned} &\log f(n) \leq \log\big(c \cdot g(n)\big) \\ \Leftrightarrow\ &\log f(n) = \log c + \log g(n) \\ \Leftrightarrow\ &\log f(n) \leq c' \cdot \log g(n) \end{aligned}$$

$$\text{với } c' = \log c + 1, \quad n > n_0.$$

⇒ **đpcm**.

d. Chứng minh hoặc phản bác mệnh đề sau:
Nếu $f(n) = O(g(n))$ và $g(n) \geq 1$ thì

$$2^{f(n)} = O(2^{g(n)}).$$

Phản bác: Giả sử $f(n) = 2n$, $g(n) = n$. Khi đó:

$$f(n) = O(g(n)) \quad \text{và} \quad g(n) \geq 1.$$

Mà:

$$2^{f(n)} = 2^n \ll 2^{2n} \quad \text{với mọi } c > 0,\, n > n_0.$$

⇒ **Khác hoàn toàn**.

1.4

Sắp xếp theo thứ tự tăng dần theo tốc độ tăng trưởng (growth rate) của các hàm sau:
Gợi ý: Sử dụng quy tắc L'Hôpital hoặc so sánh logarit để xử lý các hàm phức tạp.

Bài làm

a. $f_1(n) = n + 10; f_2(n) = \sqrt{2n}; f_3(n) = n^2 \log(n); f_4(n) = n^{2.5}; f_5(n) = 10^n; f_6(n) = 100^n$.

Sắp xếp:

$$\begin{aligned}
f_1(n) &= O(n) \\
f_2(n) &= O(\sqrt{n}) \\
f_3(n) &= O(n^2 \log(n)) \\
f_4(n) &= O(n^{2.5}) \\
f_5(n) &= O(10^n) \\
f_6(n) &= O(100^n)
\end{aligned}$$

$$\Rightarrow f_2(n) < f_1(n) < f_3(n) < f_4(n) < f_5(n) < f_6(n)$$

b. $g_1(n) = n(\log n)^3; g_2(n) = n^{4/3}; g_3(n) = 2^n; g_4(n) = n^{\log(n)}; g_5(n) = 2^{2n}$.

Phân tích:

- So sánh $g_1(n) = n(\log n)^3$ và $g_2(n) = n^{4/3}$:

$$\lim_{n\to\infty} \frac{n(\log n)^3}{n^{4/3}} \stackrel{L}{=} 0.$$

$\Rightarrow g_1(n) < g_2(n)$.

- So sánh $g_2(n) = n^{4/3}$ và $g_3(n) = 2^n$:

$$\lim_{n\to\infty} \frac{n^{4/3}}{2^n} = 0.$$

$\Rightarrow g_2(n) < g_3(n)$.

– So sánh $g_3(n) = 2^n$ và $g_4(n) = n^{\log n}$:

$$\lim_{n\to\infty} \frac{2^n}{n^{\log n}} = \lim_{n\to\infty} 2^{n-(\log n)^2} = \infty.$$

$\Rightarrow g_3(n) > g_4(n)$.

– So sánh $g_3(n) = n^{\log n}$ và $g_5(n) = 2^{2n}$:

$$\lim_{n\to\infty} \frac{n^{\log n}}{2^{2n}} = \lim_{n\to\infty} 2^{(\log n)^2-2n} = 0.$$

$\Rightarrow g_3(n) < g_5(n)$.

– So sánh $g_1(n) = n(\log n)^3$ và $g_4(n) = n^{\log n}$:

$$\lim_{n\to\infty} \frac{n(\log n)^3}{n^{\log n}} = \lim_{n\to\infty} n^{1-\log n}(\log n)^3 = 0.$$

$\Rightarrow g_1(n) < g_4(n)$.

– So sánh $g_4(n) = n^{\log n}$ và $g_2(n) = n^{4/3}$:

$$\lim_{n\to\infty} \frac{n^{\log n}}{n^{4/3}} = \lim_{n\to\infty} n^{\log n-4/3} = \infty.$$

$\Rightarrow g_4(n) > g_2(n)$.

Kết luận:

$$g_1(n) < g_2(n) < g_4(n) < g_3(n) < g_5(n)$$

1.5

Sử dụng Master Theorem để giải các phương trình đệ quy sau:

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

b. $T(n) = 3T\left(\frac{n}{2}\right) + n$

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

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

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

f. $T(n) = 3T\left(\frac{n}{3}\right) + n$

Gợi ý: Xác định rõ các tham số a, b, và $f(n)$ trước khi áp dụng Master Theorem. Nếu Master Theorem không áp dụng được, sử dụng phương pháp thế.
Master Theorem: Let $a \geq 1$ and $b > 1$ be constants, and let $f(n)$ be an asymptotically positive function. Consider the recurrence:

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

Bài làm

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

$$a = 2, \quad b = 2, \quad f(n) = n.$$

Ta có:

$$n^{\log_b a} = n^{\log_2 2} = n.$$

Do đó, theo trường hợp 2 của Master Theorem:

$$T(n) = \Theta(n \log n).$$

b. $T(n) = 3T\left(\frac{n}{2}\right) + n$

$$a = 3, \quad b = 2, \quad f(n) = n.$$

Ta có:

$$n^{\log_b a} = n^{\log_2 3} \approx n^{1.585}$$

Do đó, theo trường hợp 1 của Master Theorem:

$$\Rightarrow f(n) = O(n^{\log_b a - \epsilon}) \text{ với } \epsilon \approx 0.585.$$

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

$$a = 4, \quad b = 2, \quad f(n) = n^2.$$

Ta có:

$$n^{\log_b a} = n^{\log_2 4} = n^2.$$

Do đó, theo trường hợp 2 của Master Theorem:

$$T(n) = \Theta(n^2 \log n).$$

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

$$\begin{aligned}
T(n) &= T(n/4) + n/2 + n \\
T(n) &= T(n/8) + n/4 + n/2 + n \\
T(n) &= T(n/16) + n/8 + n/4 + n/2 + n \\
&\vdots \\
T(n) &= T(1) + n(1 + 1/2 + 1/4 + \ldots) \\
T(n) &= T(1) + 2n \\
\Rightarrow T(n) &= O(n).
\end{aligned}$$

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

$$a = 2, \quad b = 2, \quad f(n) = 1.$$

Ta có:

$$n^{\log_b a} = n^{\log_2 2} = n.$$

Do đó, theo trường hợp 1 của Master Theorem:

$$\Rightarrow f(n) = O(n^{\log_b a - \epsilon}) \text{ với } \epsilon = 1.$$

f. $T(n) = 3T\left(\frac{n}{3}\right) + n$

$$a = 3, \quad b = 3, \quad f(n) = n.$$

Ta có:

$$n^{\log_b a} = n^{\log_3 3} = n.$$

Do đó, theo trường hợp 2 của Master Theorem:

$$T(n) = \Theta(n \log n).$$

1.6

Phân tích độ phức tạp thời gian của các thuật toán sau trong các trường hợp (Worst case, Average case, Best case):

a. Tìm kiếm nhị phân (Binary Search)

b. Kiểm tra số nguyên tố

c. Thuật toán Euclid tính Ước số chung lớn nhất (GCD)

d. Đếm số bit 1 trong biểu diễn nhị phân của số nguyên

Bài làm

a. Tìm kiếm nhị phân (Binary Search)

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

Giải phương trình đệ quy trên ta được:

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

- **Best case:** Phần tử cần tìm nằm ngay ở giữa mảng $\Rightarrow$ chỉ cần 1 phép so sánh.

$$T_{\text{best}}(n) = O(1)$$

- **Worst case:** Phần tử cần tìm không tồn tại hoặc nằm ở biên, mỗi lần thu hẹp tìm kiếm còn một nửa, lặp cho đến khi còn 1 phần tử.

$$T_{\text{worst}}(n) = O(\log n)$$

- **Average case:** Trung bình cũng phải thực hiện số lần so sánh tương tự như worst case.

$$T_{\text{avg}}(n) = O(\frac{\log n}{2}) = O(\log n)$$

b. Kiểm tra số nguyên tố

$$T(n) = O(\sqrt{n})$$

- **Best case:** Số n là số chẵn lớn hơn 2 $\Rightarrow$ chỉ cần 1 phép chia.

$$T_{\text{best}}(n) = O(1)$$

- **Worst case:** Số n là số nguyên tố $\Rightarrow$ phải kiểm tra tất cả các số từ 2 đến $\sqrt{n}$.

$$T_{\text{worst}}(n) = O(\sqrt{n})$$

- **Average case:** Trung bình cũng phải thực hiện số phép chia tương tự như worst case.

$$T_{\text{avg}}(n) = O(\sqrt{n})$$

c. Thuật toán Euclid tính Ước số chung lớn nhất (GCD)

d. Đếm số bit 1 trong biểu diễn nhị phân của số nguyên