

Cơ sở phân tích độ phức tạp giải thuật

Giới thiệu chung

Phân tích giải thuật là sự khảo sát tính *hiệu quả* (theo nghĩa định lượng) của một giải thuật dưới hai tiêu chuẩn: thời gian (thực hiện) và không gian (tài nguyên chiếm dụng).

Kích thước dữ liệu nhập

Gọi n là kích thước dữ liệu thì độ phức tạp là một hàm theo n : $f(n)$.

Đơn vị trong phép đo thời gian thực thi

Sử dụng thời gian vật lý (giờ, phút, giây, ...) không phải là lựa chọn tốt.

Độ hiệu quả về mặt thời gian của một giải thuật sẽ được xác định dựa trên việc đếm số lần thực hiện của thao tác cơ sở với dữ liệu đầu vào có kích thước n .

Để ước lượng thời gian vật lý $T(n)$ cho một giải thuật, ta có thể sử dụng công thức:

$$T(n) \approx t \times f(n)$$

với t là thời gian thao tác cơ sở thực thi trên một phần cứng cụ thể, $f(n)$ là số lần thao tác cơ sở phải thực hiện và n là kích thước dữ liệu nhập.

Bậc tăng trưởng

Khi n đủ lớn, *các hằng số cũng như những toán hạng có bậc nhỏ không còn mang nhiều ý nghĩa nên được loại bỏ.*

Ví dụ: Xét hai hàm $0.1n^2 + n + 100$ và $0.1n^2$.

n	$0.1n^2$	$0.1n^2 + n + 100$
10	10	120
100	1000	1200
1000	100000	101100

Nếu $f(n) = \frac{1}{2}n(n-1)$ thì khi n đủ lớn, $f(n) = \frac{1}{2}n^2 - \frac{1}{2}n \approx n^2$.

Trường hợp tốt nhất, trung bình và xấu nhất

Với nhiều giải thuật, tính hiệu quả của chúng không chỉ phụ thuộc vào kích thước dữ liệu nhập mà còn là đặc trưng/phân bố của mỗi dữ liệu đầu vào.

Ví dụ: Giải thuật tìm tuần tự giá trị k trong dãy n giá trị.

Trường hợp xấu nhất (worst-case): $\mathcal{W}(n)$ là hàm đo số lần thực thi nhiều nhất của thao tác cơ sở trên thể hiện dữ liệu bất kỳ kích thước n .

Trường hợp trung bình (average-case): $\mathcal{A}(n)$ là hàm đo số lần thực thi trung bình của thao tác cơ sở trên thể hiện dữ liệu bất kỳ kích thước n .

Trường hợp tốt nhất (best-case): $\mathcal{B}(n)$ là hàm đo số lần thực thi ít nhất của thao tác cơ sở trên thể hiện dữ liệu bất kỳ kích thước n .

Ví dụ: Tìm tuần tự giá trị k trong dãy n giá trị.

```
seqsearch(a[1 .. n], k) {
  for (idx = 1; idx ≤ n; idx++)
    if (a[idx] == k)
      return 1;
  return 0;
}
```

$$\mathcal{B}(n) = 1$$

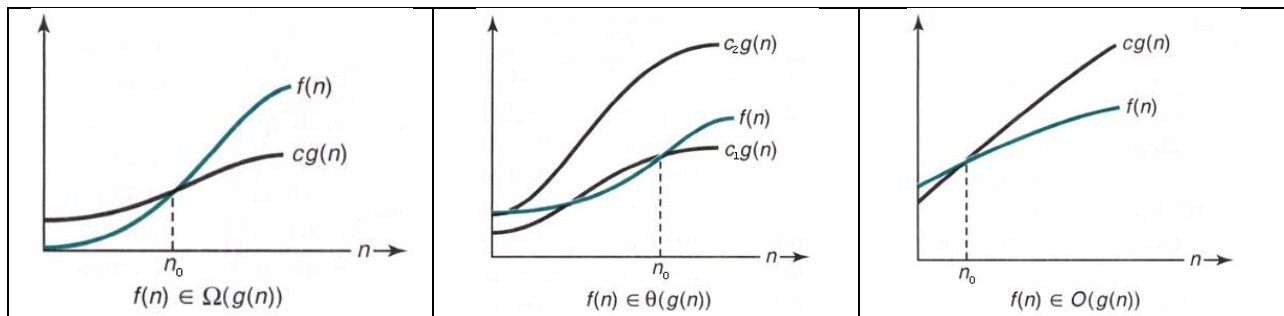
$$\mathcal{W}(n) = n$$

$$\mathcal{A}(n) = (n + 1) / 2 \text{ nếu } k \text{ có trong dãy và } \mathcal{A}(n) = n \text{ nếu } k \text{ không có trong dãy}$$

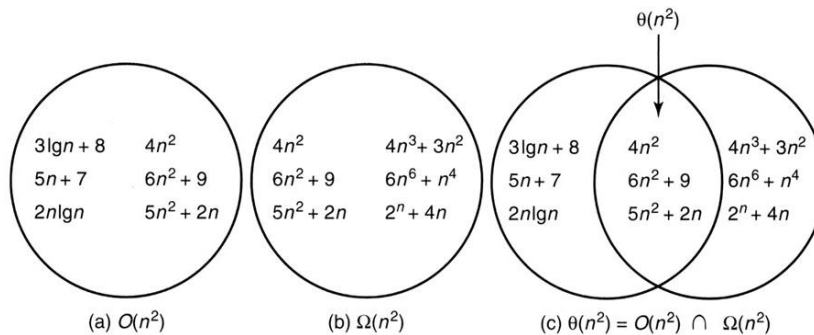
Các ký hiệu tiệm cận và các hàm chuẩn biểu diễn độ phức tạp

Gọi $f(n)$ và $g(n)$ là hai hàm không âm, định nghĩa trên \mathbb{N} . Hàm $f(n)$ để chỉ “thời gian” thực thi của giải thuật, còn $g(n)$ là một hàm đơn giản nào đó dùng để so sánh.

- $O(g(n))$ là tập hợp của các hàm có bậc tăng trưởng nhỏ hơn hoặc bằng $g(n)$.
- $\Omega(g(n))$ là tập hợp của các hàm có bậc tăng trưởng lớn hơn hoặc bằng $g(n)$.
- $\Theta(g(n))$ là tập hợp của các hàm có cùng bậc tăng trưởng như $g(n)$.



Định lý 1: Với hai hàm bất kỳ $f(n)$ và $g(n)$, $f(n) \in \Theta(g(n))$ nếu và chỉ nếu $f(n) \in O(g(n))$ và $f(n) \in \Omega(g(n))$.



Định lý 2: Nếu $f_1(n) \in O(g_1(n))$ và $f_2(n) \in O(g_2(n))$. Khi đó, $f_1(n) + f_2(n) \in O(\max\{g_1(n), g_2(n)\})$

Phân tích giải thuật không đệ qui

Sự phân tích thường trải qua những bước sau:

1. Xác định kích thước dữ liệu nhập.
2. Nhận diện thao tác cơ sở.
3. Độ phức tạp phụ thuộc phân bố dữ liệu?
4. Xây dựng đa thức $f(n)$ mô tả tổng số lần thao tác cơ sở được thực thi.
5. Qui đa thức $f(n)$ về hàm chuẩn biểu diễn bậc tăng trưởng tương ứng.

Ví dụ: Nhân hai ma trận $n \times n$.

```
c[1 .. n][1 .. n] = 0;
for (i = 1; i ≤ n; i++)
    for (j = 1; j ≤ n; j++)
        for (k = 1; k ≤ n; k++)
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
```

Ví dụ: Sắp xếp nổi bọt

```
BubbleSort(a[1 .. n]) {
    for (i = 2; i < n; i++)
        for (j = n; j ≥ i; j--)
            if (a[j - 1] > a[j])
                a[j - 1] ↔ a[j];
}
```

Gọi $C(n)$ là số lần thực hiện phép so sánh với kích thước dữ liệu n :

$$C(n) = \frac{n(n-1)}{2} \in \Theta(n^2)$$

Cải tiến

```
BubbleSort(a[1 .. n]) {
    flag = true;
    m = 1;
    while (flag) {
        flag = false;
        m++;
        for (j = n; j ≥ m; j--)
            if (a[j - 1] > a[j]) {
                a[j - 1] ↔ a[j];
                flag = true;
            }
    }
}
```

Trường hợp tốt nhất: $\mathcal{B}(n) = (n-1)$

Trường hợp xấu nhất: $W(n) = \frac{n(n-1)}{2} \in \Theta(n^2)$

Trường hợp trung bình:

$$A(n) = \frac{1}{n-1} \sum_{i=1}^{n-1} C(i) = \frac{4n^2 - 2n}{12} \in \Theta(n^2)$$

với $C(i)$ là tổng số lần so sánh sau lượt duyệt thứ i :

$$C(i) = \sum_{j=n-i}^{n-1} j$$

Ví dụ: Sắp xếp Chèn

```
InsertionSort(a[1 .. n]) {
  for (i = 2; i ≤ n; i++) {
    v = a[i];
    j = i - 1;
    while (j ≥ 1) && (a[j] > v) {
      a[j + 1] = a[j];
      j--;
    }
    a[j + 1] = v;
  }
}
```

Trường hợp tốt nhất: $B_{comp}(n) = n - 1 \in \Theta(n)$

Trường hợp xấu nhất: $W_{comp} = \sum_{i=2}^n (i - 1) = \frac{n(n-1)}{2} \in \Theta(n^2)$

Trường hợp trung bình:

$$A_{comp}(n) = \sum_{i=2}^n C(i) = \frac{n^2 - n}{4} + n - \ln n - \gamma \approx \frac{n^2}{4} \in \Theta(n^2)$$

với $\gamma = 0.5772\dots$ là hằng số Euler và $C(i)$ là chi phí trung bình cho lượt chèn phần tử a_i :

$$C(i) = \sum_{j=1}^{i-1} \frac{1}{i} \times j + \frac{1}{i} \times (i - 1)$$

Ví dụ: Số lượng bit trong biểu diễn nhị phân của số nguyên dương n ?

```
Binary(n) {
  count = 1;
  while (n > 1) {
    count++;
    n = ⌊n / 2⌋;
  }
  return count;
}
```






























$$\frac{n}{2^k} = 1 \Rightarrow 2^k = n \Leftrightarrow k = \log_2 n$$

Hệ thức truy hồi

Ví dụ: Họ nhà thỏ và dãy số Fibonacci (Fibonacci, 1202)

Một cặp thỏ mới sinh (1 đực, 1 cái) được thả lên hoang đảo. Cho rằng chúng trưởng thọ và độ tuổi sinh sản của thỏ là sau 2 tháng tuổi. Khi đạt đến tuổi sinh sản thì hàng tháng, thỏ cái sẽ sinh ra một cặp đủ nếp lần tẻ. Cho biết số **cặp** thỏ trên đảo ở tháng thứ n ?

Đầu tiên, vì là hoang đảo nên có 0 cặp thỏ (F_0).

Tuổi > 2	Tuổi ≤ 2	(trong) Tháng	Tổng số cặp
	 ① 	1	1
	 ② 	2	1
 ③ 	 ① 	3	2
 ④ 	 ②  ① 	4	3
 ⑤  ③  ④	 ②  ①  ① 	5	5
 ⑥  ④  ③	 ②  ②   ①  ①  ①	6	8

Tháng thứ n (F_n): trên đảo có $F_{n-1} + F_{n-2}$ cặp, gồm F_{n-2} cặp mới sinh và F_{n-1} cặp ở độ tuổi sinh sản. Công thức tổng quát là:

$$F_n = F_{n-1} + F_{n-2} \text{ với } F_0 = 0, F_1 = 1.$$

Giải hệ thức truy hồi

Tìm lời giải cho một hệ thức truy hồi với ràng buộc của điều kiện đầu nghĩa là tìm công thức tường minh biểu diễn toán hạng chung.

Phương pháp thay thế tiến

Xây dựng công thức để tính toán hạng chung (dựa vào một số bước tính toán tường minh) và kiểm chứng tính đúng.

Ví dụ: Xét hệ thức truy hồi và điều kiện đầu sau

$$x_n = 2x_{n-1} + 1 \text{ với } n > 1 \text{ và } x_1 = 1$$

Công thức đề nghị là

$$x_n = 2^n - 1 \text{ với } n > 0$$

Để kiểm chứng, ta thay thế trực tiếp vào hệ thức hoặc sử dụng chứng minh qui nạp.

Phương pháp thay thế lùi

Biến đổi x_n như là một hàm của x_{n-i} với $i = 1, 2, \dots$

Để dừng tiến trình này, chọn giá trị i sao cho $n - i$ đạt đến điều kiện đầu và từ đây suy ra lời giải của hệ thức truy hồi.

Ví dụ: $x_n = x_{n-1} + n$ với $n > 0$ và $x(0) = 0$

Hệ thức truy hồi tuyến tính thuần nhất bậc k với các hệ số hằng

Hệ thức truy hồi dưới đây không thể giải được bằng hai phương pháp trên:

$$x_n = c_1 x_{n-1} + c_2 x_{n-2} + \dots + c_k x_{n-k}$$

hay

$$f(n) = x_n - c_1 x_{n-1} - c_2 x_{n-2} - \dots - c_k x_{n-k} = 0$$

với c_1, c_2, \dots, c_k là các số thực, $c_k \neq 0$ và k điều kiện đầu của hệ thức là $x_0 = C_0, x_1 = C_1, \dots, x_{k-1} = C_{k-1}$.

Lời giải của hệ thức sẽ có dạng $x_n = r^n$ với r là một hằng số. Từ đây:

$$r^n = c_1 r^{n-1} + c_2 r^{n-2} + \dots + c_k r^{n-k}$$

hay

$$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \dots - c_{k-1} r - c_k = 0$$

Đây được gọi là **phương trình đặc trưng** của hệ thức truy hồi.

Hệ thức truy hồi tuyến tính thuần nhất bậc 2 với các hệ số hằng

Phương trình đặc trưng tương ứng có dạng:

$$ar^2 + br + c = 0$$

Gọi r_1 và r_2 là nghiệm của phương trình này.

Trường hợp 1: Nếu r_1 và r_2 là số thực, $r_1 \neq r_2$. Lời giải tổng quát là:

$$x_n = \alpha(r_1)^n + \beta(r_2)^n, \forall \alpha, \beta \in \mathbb{R}$$

Trường hợp 2: Nếu $r_1 = r_2$ là số thực. Lời giải tổng quát là:

$$x_n = \alpha r^n + \beta n r^n, \forall \alpha, \beta \in \mathbb{R}$$

Trường hợp 3: Nếu $r_{1,2} = u \pm iv$ thì lời giải là:

$$x_n = \gamma^n (\alpha \cos n\theta + \beta \sin n\theta)$$

$$\forall \alpha, \beta \in \mathbb{R}, \gamma = \sqrt{u^2 + v^2}, \theta = \arctan(v/u).$$

Ví dụ: Xét hệ thức truy hồi $x_n = x_{n-1} + 2x_{n-2}$ với $x_0 = 2, x_1 = 7$.

Lời giải là:

$$x_n = 3 \times 2^n - (-1)^n$$

Ví dụ: Xét hệ thức truy hồi $x_n - 6x_{n-1} + 9x_{n-2} = 0$ với $x_0 = 0, x_1 = 3$.

Lời giải là:

$$x_n = n3^n$$

Phân tích giải thuật đệ qui

Đối với các giải thuật đệ qui, quá trình phân tích thường diễn ra như sau:

1. Xác định kích thước dữ liệu nhập.
2. Nhận diện (các) thao tác cơ sở.
3. Độ phức tạp phụ thuộc phân bố dữ liệu?
4. Xây dựng *hệ thức truy hồi cùng điều kiện đầu* nhằm mô tả số lần thực hiện của thao tác cơ sở.
5. Giải hệ thức truy hồi và quy về hàm chuẩn biểu diễn bậc tăng trưởng tương ứng.

Ví dụ: Tính $n!$

```
Factorial(n) {
    if (n == 0)
        return 1;
    return Factorial(n - 1) * n;
}
```

Với bài này, kích thước dữ liệu là n . Thao tác cơ sở là phép nhân (tính quan trọng). Gọi số lượng phép nhân phải thực hiện là $\mathcal{M}(n)$. Hệ thức truy hồi là:

$$\mathcal{M}(n) = \mathcal{M}(n - 1) + 1 \text{ với } n > 0 \text{ và } \mathcal{M}(0) = 0.$$

Dùng *phương pháp thay thế lùi*, lời giải là:

$$\mathcal{M}(n) = n \in \Theta(n).$$

Nếu cho rằng phép so sánh là thao tác cơ sở (số lần thực hiện) thì hệ thức truy hồi được mô tả như sau:

$$\mathcal{C}(n) = \mathcal{C}(n - 1) + 1 \text{ với } n > 0 \text{ và } \mathcal{C}(0) = 1$$

Bằng phương pháp thay thế lùi: $\mathcal{C}(n) = n + 1 \in \Theta(n)$.

Ví dụ: Tháp Hà nội

```
HNTower(n, left, middle, right) {
    if (n) {
        HNTower(n - 1, left, right, middle);
        Movedisk(1, left, right);
        HNTower(n - 1, middle, left, right);
    }
}
```

Kích thước dữ liệu là n . Gọi số lần chuyển dời n đĩa là $\mathcal{M}(n)$.

Hệ thức truy hồi là:

$$\mathcal{M}(n) = \mathcal{M}(n - 1) + 1 + \mathcal{M}(n - 1) = 2\mathcal{M}(n - 1) + 1 \text{ với } n > 0 \text{ và } \mathcal{M}(1) = 1.$$

Bằng phép thay thế lùi, lời giải là:

$$\mathcal{M}(n) = 2^n - 1 \in \Theta(2^n)$$

Ví dụ: Số lượng bit trong biểu diễn nhị phân của số nguyên dương n ?

```
BitCount(n) {
  if (n == 1)
    return 1;
  return BitCount( $\lfloor n / 2 \rfloor$ ) + 1;
}
```

Thao tác cơ sở là phép cộng. Gọi $\mathcal{A}(n)$ là số lượng phép cộng cần phải thực thi với kích thước dữ liệu n . Hệ thức truy hồi như sau:

$$\mathcal{A}(n) = \mathcal{A}(\lfloor n / 2 \rfloor) + 1 \text{ với } n > 1 \text{ và } \mathcal{A}(1) = 0.$$

Giả sử $n = 2^k$, hệ thức truy hồi trở thành:

$$\mathcal{A}(2^k) = \mathcal{A}(2^{k-1}) + 1 \text{ với } n > 0 \text{ và } \mathcal{A}(2^0) = 0.$$

Tiến trình thay thế lùi chỉ ra:

$$\mathcal{A}(n) = \log_2 n \in \Theta(\log n)$$

Ví dụ: Họ nhà thỏ và dãy số Fibonacci.

Hệ thức truy hồi được viết lại là:

$$F_n - F_{n-1} - F_{n-2} = 0$$

Phương trình đặc trưng là:

$$r^2 - r - 1 = 0$$

với nghiệm:

$$r_{1,2} = \frac{1 \pm \sqrt{5}}{2}$$

Ta có:

$$F_n = \alpha \left(\frac{1 + \sqrt{5}}{2} \right)^n + \beta \left(\frac{1 - \sqrt{5}}{2} \right)^n$$

Dựa vào cặp điều kiện đầu, xác định được:

$$\alpha = 1/\sqrt{5} \text{ và } \beta = -1/\sqrt{5}.$$

Gọi $\phi = (1 + \sqrt{5})/2 \approx 1.61803$, $\hat{\phi} = (1 - \sqrt{5})/2 = -1/\phi \approx -0.61803$:

$$F_n = \frac{1}{\sqrt{5}} (\phi^n - \hat{\phi}^n)$$

Vì $-1 < \hat{\phi} < 0$ nên $\hat{\phi}^n$ sẽ rất nhỏ khi $n \rightarrow \infty$. Từ đây:

$$F_n \in (\phi^n)$$

Để đơn giản, tính:

$$F_n = \frac{1}{\sqrt{5}} \phi^n$$

rồi làm tròn về số nguyên gần nhất.

Một số giải thuật tính F_n

Đệ qui

```
Fibonacci(n) {
    if (n ≤ 1)
        return n;
    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
```

Gọi $\mathcal{A}(n)$ là số phép cộng phải thực hiện. Hệ thức truy hồi là:

$$\mathcal{A}(n) = \mathcal{A}(n-1) + \mathcal{A}(n-2) + 1 \text{ với } n > 1$$

và điều kiện đầu là $\mathcal{A}(0) = 0, \mathcal{A}(1) = 0$.

Đây là “Hệ thức truy hồi tuyến tính không thuần nhất bậc hai với các hệ số hằng”.

Gọi $\mathcal{B}(n) = \mathcal{A}(n) + 1$. Suy ra:

$$\mathcal{B}(n) - \mathcal{B}(n-1) - \mathcal{B}(n-2) = 0 \text{ với } n > 1$$

và điều kiện đầu là $\mathcal{B}(0) = 1, \mathcal{B}(1) = 1$. Giải hệ thức để có được kết quả:

$$\mathcal{A}(n) = \frac{1}{\sqrt{5}} (\phi^{n+1} - \hat{\phi}^{n+1}) - 1 \in \Theta(\phi^n)$$

Cài đặt không đệ qui

Dựa vào công thức $F_n = \frac{1}{\sqrt{5}} (\phi^n - \hat{\phi}^n)$, chi phí giải thuật là tuyến tính.

Dựa vào công thức $F_n = \text{round} \left(\frac{1}{\sqrt{5}} \phi^n \right)$, với tiếp cận giảm để trị, chi phí giải thuật là $\Theta(\log n)$.

Tiếp cận qui hoạch động (chi phí $\Theta(n)$):

```
Fibonacci(n) {
    if (n ≤ 1)
        return n;
    f0 = 0;
    f1 = 1;
    for (i = 2; i ≤ n; i++) {
        fn = f0 + f1;
        f0 = f1;
        f1 = fn;
    }
    return fn;
}
```

Lũy thừa ma trận

$$\begin{bmatrix} F(n+1) & F(n) \\ F(n) & F(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \text{ với } n \geq 1.$$

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n = \begin{cases} \left(\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n/2} \right)^2 & \text{nếu } n \text{ chẵn} \\ \left(\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{\lfloor n/2 \rfloor} \right)^2 \times \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} & \text{nếu } n \text{ lẻ} \end{cases}$$

Chương trình

```
void multiply(F[2][2], T[2][2]) {
    t1 = F[0][0] * T[0][0] + F[0][1] * T[1][0];
    t2 = F[0][0] * T[0][1] + F[0][1] * T[1][1];
    t3 = F[1][0] * T[0][0] + F[1][1] * T[1][0];
    t4 = F[1][0] * T[0][1] + F[1][1] * T[1][1];
    F[0][0] = t1;
    F[0][1] = t2;
    F[1][0] = t3;
    F[1][1] = t4;
}

void power(int F[2][2], int n) {
    if (n == 0 || n == 1)
        return;
    T[2][2] = {{1, 1}, {1, 0}};
    power(F, n / 2);
    multiply(F, T);
    if (n % 2 != 0) multiply(F, T);
}

int fib(int n) {
    F[2][2] = {{1, 1}, {1, 0}};
    if (n == 0)
        return 0;
    power(F, n - 1);
    return F[0][0];
}
```

Chương trình (cải tiến)

```
int Fibonacci(int n) {
    i = 1;    j = 0;
    k = 0;    h = 1;
    while (n) {
        if (n % 2) {
            t = j * h;
            j = i * h + j * k + t;
            i = i * k + t;
        }
        t = h * h;
        h = 2 * k * h + t;
        k = k * k + t;
        n = n / 2;
    }
    return j;
}
```