Đánh giá thủ tục (hoặc hàm) đệ qui.

Bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Ví dụ: Đánh giá thời gian thực hiện của hàm đệ qui sau:

(hàm tính n!)

```
Function fact(n:integer):integer;

Begin

If n<=1 then fact :=1

Else fact:=n*fact(n-1);

End;
```

Trong hàm này cỡ của dữ liệu vào là n. Giả sử thời gian thực hiện hàm là T(n) .Với n=1 chỉ cần thực hiện lệnh gán fact:=1; do đó T(1)= O(1). Với n>1 ,cần thực hiện lệnh gán fact:=n*(fact(n-1)). Do đó , thời gian T(n) sẽ là O(1) để thực hiện phép nhân (*) và phép gán(:=) cộng với thời gian T(n-1) để thực hiện lời gọi đệ qui fact(n-1). Tóm lại ta có quan hệ đệ qui như sau:

T(1) = O(1);

T(n) = O(1) + T(n-1);

Thay các O(1) bởi các hằng nào đó ta có quan hệ đệ qui như sau:

T(1) = C1 ;

T(n) = C2 + T(n-1);

Để giải phương trình đệ qui, tìm T(n), chúng ta áp dụng phương pháp thế lặp. Ta có phương trình đệ qui sau:

Đánh giá thủ tục (hoặc hàm) đệ qui.

T(m) = C2 + T(m-1); với m >1

Thay m lần lượt bởi các 2, 3,..,n-1,n ta được hệ các quan hệ sau:

T(2) = C2 + T(1);

T(3) = C2 + T(2);

.....

T(n-1) = C2 + T(n-2);

T(n) = C2 + (n-1) ;

Bằng các phép thế liên tiếp ta nhận được

T(n) = (n-1).C2 + T(1)

Hay T(n) = (n-1) C2 + C1;, trong đó C1, C2 là các hằng nào đó .

Do đó T(n) = O(n) ;

Từ đó ta có phương pháp tổng quát sau đây để đánh giá thời gian thực hiện các thủ tục (hàm) đệ qui. Ta giả thiết rằng các thủ tục (hàm)là đệ qui trực tiếp. Điều đó có nghĩa là các thủ tục (hàm) chỉ chứa các lời gọi đệ qui đến chính nó (không qua một thủ tục hoặc hàm nào khác cả). Giả sử thời gian thực hiện thủ tục (hàm) là T(n), với n là cỡ dữ liệu vào. Khi đó thời gian thực hiện các lời gọi đệ qui thủ tục sẽ là T(m), với m<n. Đánh giá thời gian $T(n_o)$ với n_0 là cỡ dữ liệu vào nhỏ nhất có thể được (trong ví dụ trên đó là T(1)). Sau đó đánh giá thân của thủ tục theo các qui tắc đã nêu trên ,ta sẽ nhận được quan hệ đệ qui như sau:

$T(n) = F(T(m_1),T(m_2),..,T(m_k))$

Trong đó m_1, m_2, ... , m_k < n. Giải phương trình đệ qui này ta sẽ nhận được sự đánh giá của T(n).

Bây giờ ta sử dụng những kiến thức trên để đánh giá hai ví dụ quen thuộc sau đây:

Ví dụ. Xác định độ phức tạp thuật toán của hàm tính dãy số Fibonaci:

Function Fibo (n : integer) : integer;

Var i , j , k : integer ;

Đánh giá thủ tục (hoặc hàm) đệ qui.

```
Begin
i := 1;
j := 0 ;
for k := 1 to n do
Begin
j := i + j ;
i := j - i ;
End;
Fibo := j;
End;
```

Thời gian thực hiện của lệnh trên là O(n) .

Một bài toán thường có nhiều cách giải, hay nhiều thuật toán để giải, với mỗi thuật toán khác nhau có thể sẽ có độ phức tạp khác nhau. Đánh giá độ phức tạp thuật toán là một trong những cách phân tích, so sánh và tìm ra trong những thuật toán đó một thuật toán tối ưu.

Ví dụ. Xét bài toán : Tính giá trị đa thức :

$P(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$, với $x = x_0$

Xét thuật toán 1 :

Tính giá trị từng hạng tử của đa thức :

1. Với i = 1 đến n tính $a_i x_0^i$.

2. Tính Đa thức P(x) có thể viết dưới dạng :

$P(x) = (...((a_n x + a_{n-1})x...)x + a_0$.

Xét Thuật toán 2:

1. P := a_n .

Đánh giá thủ tục (hoặc hàm) đệ qui.

2. Với i=1 đến n : P := $Px_0 + a_{n-i}$

3. Gán kết quả $P(x_0)$= P.

Chẳng hạn với n = 3.

$P(x) = a_3x^3 + a_2x^2 + a_1x + a_0 = (((a_3x + a_2)x + a_1) + a_0)$

1. Tính P := a_3

2. i = 1 : P = $(a_3x_0 + a_2)$

i = 2 : P = $(a_3x_0 + a_2)x + a_1$

i = 3 : P = $((a_3x_0 + a_2)x + a_1)x + a_0$

3. $P(x_0) = ((a_3x_0 + a_2)x + a_1)x + a_0$

Xét độ phức tạp của hai thuật toán trên :

Thuật toán 1: ở bước 1 ta có :

i = 1 : ta phải thực hiện 1 phép nhân.

i = 2 : ta phải thực hiện 2 phép nhân.

.....

i = n : ta phải thực hiện n phép nhân.

Như vậy ở bước 1 ta sẽ phải thực hiện 1 + 2 + …+ n =

ở bước 2 ta có : ta phải thực hiện n phép cộng.

Vậy ở thuật toán 1 cần $\frac{n(n+1)}{2} + n = \frac{n(n+3)}{2}$

Thuật toán 2: ta phải thực hiện n lần mỗi lần đòi hỏi 2 phép tính (một phép tính cộng và một phép tính nhân). Như vậy thuật toán 2 cần tất cả 2n phép tính. Nếu ta coi thời gian thực hiện mỗi phép tính nhân và tính cộng là như nhau và là một đơn vị thời gian thì thời gian thực hiện thuật toán 1 là , còn thời gian thực hiện thuật toán 2 là 2n .

Như vậy theo phân tích ở trên ta thấy rằng thời gian thực hiện thuật toán 2 ít hơn so với thời gian thực hiện thuật toán một. Thuật toán 2 có độ phức tạp là O(n), độ phức tạp tuyến tính, còn thuật toán 1 thì độ phức tạp là $O(n^2)$ độ phức tạp bậc hai.

Ta nhận thấy rằng việc đánh giá độ phức tạp của một thuật toán nào đó là việc không hề đơn giản nó đòi hỏi phải có phương pháp và cách tiếp cận riêng.

Ví dụ. Phân tích thuật toán Euclide tìm ước số chung lớn nhất của hai số nguyên dương a, b.

Thuật toán Euclide :

Input : a, b là hai số nguyên dương.

Output : ước số chung lớn nhất của hai số a, b.

```
Function USCLN(a, b)

Begin

x := a;

y := b;

While y <> 0

begin

r := x mod y

x := y;

y := r;

end;

USCLN := x;

End;
```

Để đánh giá độ phức tạp của thuật toán trên, ta đếm số phép chia thực hiện theo thuật toán.