ĐÁNH GIÁ THUẬT TOÁN

vietanh@vnu.edu.vn

Mục tiêu

- Hiểu được khái niệm độ phức tạp thuật toán
- Hiểu được một số qui tắc xác định độ phức tạp
- Biết một độ phức tạp thường gặp phụ thuộc kích thước dữ liệu vào
- Biết đánh giá độ phực tạp của các lệnh

Nội dung chính

- Độ phức tạp của thuật toán
- Ký hiệu ô lớn và biểu diễn thời gian chạy thuật toán
- Một số tính chất
- Một số qui tắc xác định độ phức tạp
- Đánh giá thời gian thực hiện các lệnh
- Độ phức tạp thuật toán phụ thuộc dữ liệu vào

Độ phức tạp thuật toán

- Thuật toán được đánh giá mức độ Hiệu quả thông qua
  - Không gian bộ nhớ sử dụng
  - Thời gian thực thi
    - Chương trình biên dịch
    - Tốc độ xử lý máy tính
    - Kỹ năng lập trình
    - **Dữ liệu vào**
  - Dễ dàng cài đặt

Độ phức tạp thuật toán …

- Phân tích độ phức tạp theo hai cách tiếp cận
  - Thực nghiệm
  - Toán học

Độ phức tạp thuật toán …

Phân tích thực nghiệm

- Đo thời gian, vẽ đồ thị, nội suy
- Dễ thực hiện
- Dùng để dự đoán

Phân tích toán học

- Ước lượng số phép toán như là hàm kích thước dữ liệu vào
- Dùng để dự đoán và giải thích

Ký hiệu O lớn, biểu diễn thời gian thuật toán

- Giả sử T(n) là thời gian thực hiện một thuật toán nào đó và f(n), g(n), h(n) là các hàm xác định dương với mọi n.
- Hàm O lớn: O(g(n)) nếu tồn tại các hàng số c và n_0 sao cho T(n) <= c.g(n) với mọi n>= n_0 và gọi là kí pháp chữ O lớn, hàm g(n) được gọi là giới hạn trên của hàm T(n).
- Ví dụ, nếu T(n) = n^2 + 1 thì T(n) = O(n^2).

chọn c=2 và n_0 =1, khi đó với mọi n>=1, ta có T(n)= n^2+1 <= 2n^2 =2g(n).

Ký hiệu O lớn, biểu diễn thời gian thuật toán

- Ta sẽ lấy cận trên chặt (tight bound) để biểu diễn
- thời gian chạy của thuật toán.
- Ta nói f(n) là cận trên chặt của T(n) nếu
  - T(n) = O(f(n)), và
  - Nếu T(n) = O(g(n)) thì f(n) = O(g(n)).
- Tổng quát nếu f(n) là một đa thức bậc k của n:
  $f(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$ thì $f(n)=O(n^k)$

Một số tính chất

Tính bắc cầu: Tất cả các kí pháp trên đều có tính bắc cầu, chẳng hạn: nếu f(n)= O(g(n)) và g(n)= O(h(n)) thì f(n)= O(h(n))

(ii) Tính phản xạ: Tất cả các kí pháp trên đều có tính phản xạ, chẳng hạn: f(n)=O(f(n))

Xác định độ phức tạp thuật toán

- Quy tắc hằng số
  - Nếu P có thời gian thực hiện T(n)= O(c_1f(n)) với c_1 là một hằng số dương thì có thể coi P có độ phức tạp tính toán là O(f(n))
- Quy tắc nhân
  - Nếu P có thời gian thực hiện T(n)= O(f(n)). Khi đó nếu thực hiện k(n) lần P với k(n)=O(g(n)) thì độ phức tạp tính toán sẽ là: O(f(n) g(n)).

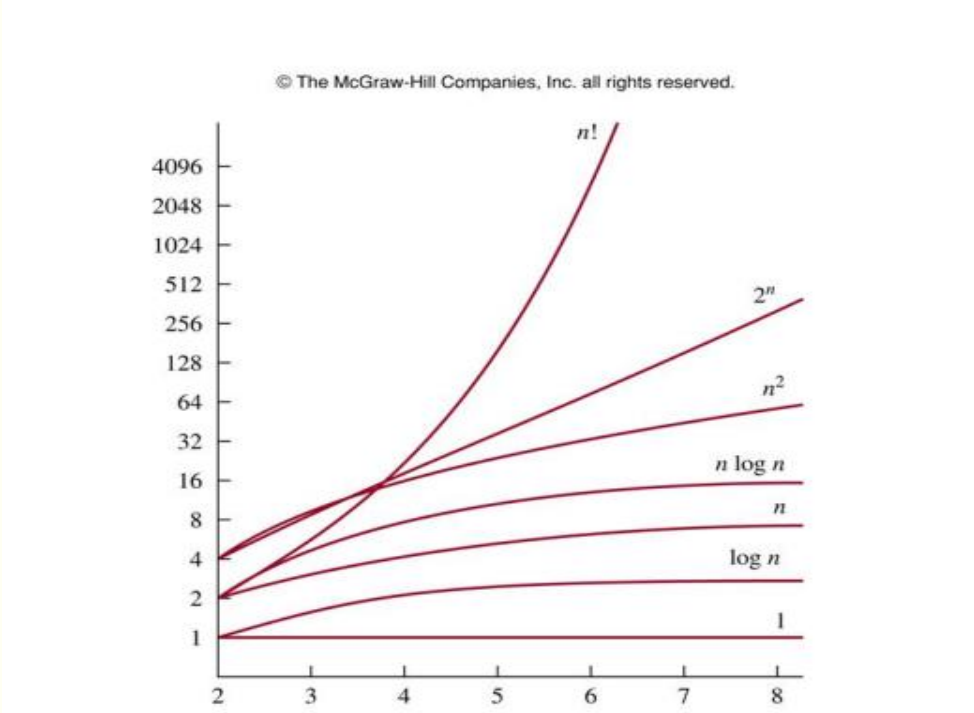
Đánh giá thời gian thực hiện các lệnh

- Một lệnh đơn là O(1).
- Một câu lệnh hợp thành: theo quy tắc cộng.
- Câu lệnh điều kiện IF: Giả sử thời gian thực hiện 02 câu lệnh thành phần dạng đủ là f(n) và g(n) thì O(Max(f(n),g(n)).
- Câu lệnh lặp tắc nhân O(k(n).f(n)).
- Có thể xem hàm và thủ tục như là một chương trình độc lập và áp dụng các quy tắc trên để đánh giá.

Đánh giá thời gian thực hiện các lệnh ...

- Một số cấp độ thời gian

Ký hiệu ô lớn	Tên gọi
O(1)	hằng
O(logn)	logarit
O(n)	tuyến tính
O(nlogn)	nlogn
$O(n^2)$	bình phương
$O(n^3)$	lập phương
$O(2^n)$	mũ



Một số ví dụ

```
for (i= 1;i<=n;i++)
        for (j= 1;j<=n;j++)
                //Lệnh
```

Dùng quy tắc nhân ta có O(n^2)

Một số ví dụ

```
for (i= 1;i<=n;i++)
        for (j= 1;j<=m;j++)
                //Lệnh
```

Dùng quy tắc nhân ta có O(n*m)

Một số ví dụ

```
for (i= 1;i<=n;i++)
        //lệnh1
for (j= 1;j<=m;j++)
        //lệnh 2
```

=> Dùng quy tắc cộng và quy tắc lấy max, sẽ có

O(max (n,m))

Một số ví dụ

```
for (i= 1;i<=n;i++) {
        for (u= 1;u<=m;u++)
                for (v= 1;v<=n;v++)
                        //lệnh
        for j:= 1 to x do
                for k:= 1 to z do
                        //lệnh
}
```

=> O(n*max (n*m,x*z))

Một số ví dụ

```
for (i= 1;i<=n;i++) {
        for (u= 1;u<=m;u++)
                for (v= 1;v<=n;v++)
                        //lệnh
        for j:= 1 to x do
                for k:= 1 to z do
                        //lệnh
}
```

=> O(n*max (n*m,x*z))

Một số ví dụ

```
for (i= 1;i<=n;i++)
        for (j= 1;j<=m;j++) {
                for (k= 1;k<=x;k++)
                        //lệnh
                for (h= 1;h<=y;h++)
                        //lệnh
        }
```

=> O(n*m* max (x,y))

Một số ví dụ

```
for (i= 1;i<=n;i++)
        for (u= 1;u<= m;u++)
                for (v= 1;v<=n;v++)
                        //lệnh
for (j= 1;j<=x;j++)
                for (k= 1;k<=z;k++)
                        //lệnh
```

=> O(max (m*n^2,x*z))

Độ phức tạp tính toán và dữ liệu vào

- Phụ thuộc kích thước dữ liệu và tình trạng thực tế của dữ liệu đó.
- Trường hợp tốt nhất: T(n) là thời gian ít nhất
- Trường hợp xấu nhất: T(n) là thời gian lớn nhất
- Trường hợp trung bình: dữ liệu vào tuân theo một phân bố xác suất nào đó

Tổng kết

- Độ phức tạp đánh giá dựa vào thời gian thực hiện
- Chú ý cách đánh giá làm già thêm O
- Độ phức tạp trong các trường hợp: Tốt nhất, Xấu nhất, Trung bình
- Phân tích đánh giá thuật toán => Lựa chọn được thuật toán hiệu quả

Tiếp theo …

- Kỹ thuật đệ qui