Cấu trúc dữ liệu và Giải thuật

Chương I: Các kiến thức cơ bản

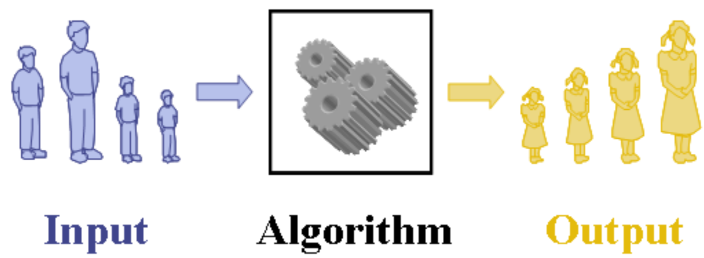
Các kiến thức cơ bản

Nội dung

- Các khái niệm
  - Giải thuật
  - Cấu trúc dữ liệu
- Phân tích giải thuật
  - Giả ngôn ngữ
  - Thời gian thực hiện giải thuật
  - Đánh giá độ phức tạp sử dụng tiệm cận

Giải thuật

- Một thủ tục bao gồm một dãy hữu hạn các bước cần thực hiện để thu được đầu ra cho đầu vào cho trước của một bài toán



Giải thuật

- Đặc trưng của giải thuật
  - Đầu vào
  - Đầu ra
  - Tính hữu hạn
  - Tính hiệu quả
  - Tính xác định

Giải thuật và Chương trình

Chương trình là một thể hiện của Giải thuật trong một ngôn ngữ lập trình nào đó

Cấu trúc dữ liệu

- Kiểu dữ liệu trừu tượng (Abstract Data Type)
  - Là mô hình toán học và những phép toán thực hiện trên mô hình toán học này
  - Ví dụ: ADT List
    - Dữ liệu: Các nút
    - Các phép toán:
      - Bổ sung một nút mới
      - Loại bỏ một nút
      - Tìm kiếm một nút có giá trị cho trước
      - …

Cấu trúc dữ liệu

- Cấu trúc dữ liệu
  - Sử dụng để biểu diễn mô hình toán học trong ADT
  - Việc cài đặt các kiểu dữ liệu trừu tượng đòi hỏi phải chọn các cấu trúc dữ liệu để biểu diễn
  - Liên quan đến cách thức tổ chức và truy nhập các phần tử dữ liệu
  - Ví dụ: ADT List
    - Cài đặt sử dụng cấu trúc mảng đơn giản
    - Cài đặt sử dụng cấu trúc con trỏ

Xây dựng chương trình giải bài toán

- Lời giải một bài toán bao gồm
  - Cấu trúc dữ liệu
  - Thuật toán
- Xây dựng chương trình giải bài toán
  - Tương tự như vòng đời của phần mềm
  - Gồm các bước
    - Thu thập yêu cầu: Hiểu rõ đầu vào và kết quả đầu ra
    - Thiết kế : Xây dựng giải thuật, bỏ qua các chi tiết về cách thức cài đặt dữ liệu hay các phương thức, tập trung vào các bước xử lý
    - Phân tích : Tìm, so sánh với giải thuật khác
    - Cài đặt: Xây dựng chương trình, quan tâm đến cách thức tổ chức, biểu diễn và cài đặt các phương thức
    - Kiểm thử : Bao gồm chứng minh tính đúng đắn của chương trình, kiểm thử các trường hợp , tìm, sửa lỗi

Thuật toán và độ phức tạp

- Đánh giá lượng tài nguyên các loại mà một giải thuật đã sử dụng.
  - Giải thuật này thực hiện trong thời gian thế nào → Phân tích về thời gian thực hiện giải thuật
  - Giải thuật này sử dụng bao nhiêu bộ nhớ → Phân tích độ không gian nhớ mà giải thuật (chương trình) cần có.

Phân tích thời gian thực hiện giải thuật

- Mục tiêu của việc xác định thời gian thực hiện một giải thuật:
  - Để ước lượng một chương trình sẽ thực hiện trong bao lâu
  - Để ước lượng kích thước dữ liệu đầu vào lớn nhất có thể cho một giải thuật
  - Để so sánh hiệu quả của các giải thuật khác nhau, từ đó lựa chọn ra một giải thuật thích hợp cho một bài toán
  - Để giúp tập trung vào đoạn giải thuật được thực hiện với thời gian lớn nhất

Phân tích thời gian thực hiện giải thuật

- Cách thức
  - Xác định độ phụ thuộc của thời gian tính của thuật toán vào kích thước của dữ liệu đầu vào
  - Các phương pháp thực hiện
    - Phương pháp thực nghiệm
    - Phương pháp phân tích dựa trên mô hình lý thuyết

Phân tích thời gian thực hiện giải thuật

- Phương pháp thực nghiệm
  - Cài đặt giải thuật bằng ngôn ngữ lập trình
  - Chạy chương trình với các dữ liệu đầu vào khác nhau
  - Đo thời gian thực thi chương trình và đánh giá độ tăng trưởng so với kích thước của dữ liệu đầu vào
- Hạn chế:
  - Sự hạn chế về số lượng và chất lượng của mẫu thử
  - Đòi hỏi môi trường kiểm thử (phần cứng và phần mềm) thống nhất , ổn định

Phân tích thời gian thực hiện giải thuật

- Phương pháp lý thuyết
  - Có khả năng xem xét dữ liệu đầu vào bất kỳ
  - Sử dụng để đánh giá các giải thuật mà không phụ thuộc vào môi trường kiểm thử
  - Sử dụng với những mô tả ở mức cao của giải thuật
- Thực hiện phương pháp này cần quan tâm
  - Ngôn ngữ mô tả giải thuật
  - Xác định độ đo thời gian tính
  - Một cách tiếp cận để khái quát hóa độ phức tạp về thời gian

Mô tả giải thuật – Giả ngôn ngữ

- Giả ngôn ngữ (Pseudo-code)
  - Mô tả mức khái quát cao được sử dụng trong diễn tả giải thuật

Giả ngôn ngữ = Cấu trúc lập trình + Ngôn ngữ tự nhiên

```
Algorithm arrayMax(A,n)
    Input: Mảng chứa n phần tử là số nguyên
    Output: Phần tử lớn nhất trong mảng
Begin
currentMax = A[0]
for i = 1 to n-1 do
    if currentMax < A[i] then currentMax = A[i]

return currentMax
End.
```

Giả ngôn ngữ

- Các cấu trúc lập trình trong giả ngôn ngữ
  - Câu lệnh gán: V = E hoặc V ← E
  - Cấu trúc điều khiển:
    - **if** B **then** S_1 [**else** S_2]
    - **Case**
      B_1 : S_1 ;
      B_2 : S_2 ;
      …
      B_n : S_n
      else *S_{n+1}*
      end case;

Giả ngôn ngữ

- Câu lệnh lặp
  - Vòng lặp với số lần lặp biết trước
    for i = m **to** n **do** S hoặc **for** i = n **down to** m **do** S
  - Với số lần lặp không biết trước
    while B **do** S hoặc **repeat** S **until** B

- Câu lệnh vào ra
  - Đọc dữ liệu vào
    read (<danh sách biến>);
  - Ghi dữ liệu
    write (<danh sách biến hoặc dòng ký tự>);

Giả ngôn ngữ

- Khai báo hàm

  Function <tên hàm> (<danh sách tham số>)
  Begin
  <các câu lệnh>
  return **(***giá trị)*
  End

- Gọi hàm*: Hàm được gọi bằng tên hàm cùng danh sách giá trị tham số thực sự, nằm trong biểu thức*

Giả ngôn ngữ

Function AVERAGE(A,n)
Begin
{A là một mảng gồm n phần tử là số nguyên. Giải thuật trả ra giá trị trung bình của các giá trị trong mảng}
1. sum = 0;
2. {Duyệt mảng} **for** I = 1 **to** n **do**
   sum = sum + A[i];
3. average = sum/n
4. return(average)

End.

Giả ngôn ngữ

- Khai báo thủ tục

  Procedure <tên thủ tục> (<danh sách tham số>)
  Begin
  <các câu lệnh>
  End

- Thủ tục được gọi bằng cách sử dụng câu lệnh

Call <tên thủ tục> (<danh sách giá trị tham số>)

Phân tích thời gian thực hiện giải thuật

- Độ đo thời gian tính sử dụng trong phương pháp phân tích lý thuyết
  - Phép toán cơ bản là phép toán có thể được thực hiện với thời gian bị chặn bởi một hằng số không phụ thuộc vào kích thước dữ liệu
- Thời gian tính của giải thuật được xác định bằng cách đếm số phép toán cơ bản mà giải thuật thực hiện

$$T(n) \approx c_{op*}C(n)$$

Phân tích thời gian thực hiện giải thuật

- Các phép toán cơ bản thường dùng
  - Gán giá trị cho biến số
  - Gọi hàm hay thủ tục
  - Thực hiện các phép toán số học
  - Tham chiếu vào mảng
  - Trả kết quả
  - Thực hiện các phép so sánh

Phân tích thời gian thực hiện giải thuật

- Dòng 1: 2 phép toán cơ bản
- Dòng 2: phép gán giá trị đầu cho i, phép so sánh i < n được thực hiện n lần
- Thân vòng lặp thực hiện n-1 lần, trong thân, tối thiểu phải thực hiện phép so sánh (2 phép toán cơ bản) , tăng i lên 1 (2 phép toán cơ bản) tối đa phải có thêm phép gán (2 phép toán cơ bản)
- Dòng 3: 1 phép toán cơ bản

Tổng số phép toán cơ bản trong Trường hợp xấu nhất : **7n-2**

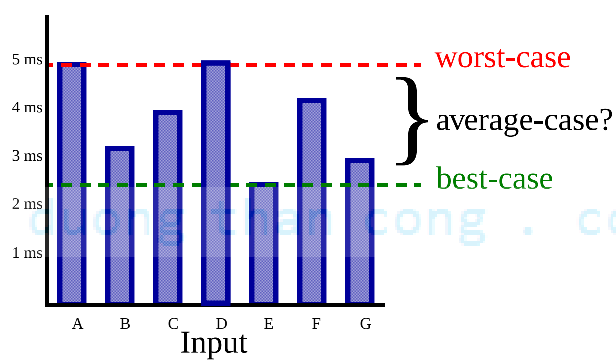
```
Function ARRAY-MAX(A,n)
Đầu vào : mảng A gồm n phần tử.
Đầu ra: phần tử lớn nhất trong
mảng
Begin
1. currentMax = A[0]
2. for i = 1 to n-1 do
     if currentMax < A[i] then
     currentMax = A[i]

3. return currentMax
End.
```

Phân tích thời gian thực hiện giải thuật

- Thời gian tính tồi nhất (Worst-case)
  - Thời gian nhiều nhất để thực hiện thuật toán với một bộ dữ liệu vào kích thước n
- Thời gian tính tốt nhất (Best-case)
  - Thời gian ít nhất để thực hiện thuật toán với một bộ dữ liệu cũng với kích thước n
- Thời gian tính trung bình (Average case)
  - Thời gian trung bình cần thiết để thực hiện thuật toán trên tập hữu hạn các đầu vào kích thước n

Phân tích thời gian thực hiện giải thuật



Phân tích thời gian thực hiện giải thuật

- Ví dụ : Tìm kiếm tuần tự một giá trị trên một mảng

a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]	a[11]	a[12]
4	8	7	10	21	14	22	36	62	91	77	81

- Thời gian xấu nhất : n
- Thời gian tốt nhất : 1
- Thời gian trung bình: T(n) = ∑i.pi

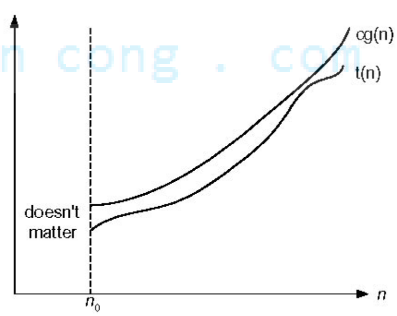
trong đó pi là xác suất giá trị cần tìm xuất hiện tại a[i]. pi = 1/n thì thời gian sẽ là (n+1)/2

Ký hiệu tiệm cận

- Khái niệm Big-O
  - Cho hàm số t(n) và g(n), ta nói rằng t(n) là O(g(n)) nếu tồn tại 2 hằng số nguyên dương c và n_0 sao cho

$t(n) \leq cg(n)$ for $n \geq n_0$

t(n) thuộc O(g(n))



Ký hiệu tiệm cận Big - O

- 7n -2
  - 7n-2 là O(n)

    tìm c > 0 và n0 ≥ 1 sao cho 7n-2 ≤ c*n với n ≥ n0

    điều này đúng với c = 7 và n0 = 1
- 3n3 + 20n2 + 5
  - 3n3 + 20 n2 + 5 là O(n3)

    tìm c > 0 và n0 ≥ 1 sao cho 3n3 + 20n2 + 5 ≤ c*n3 vơi n ≥ n0

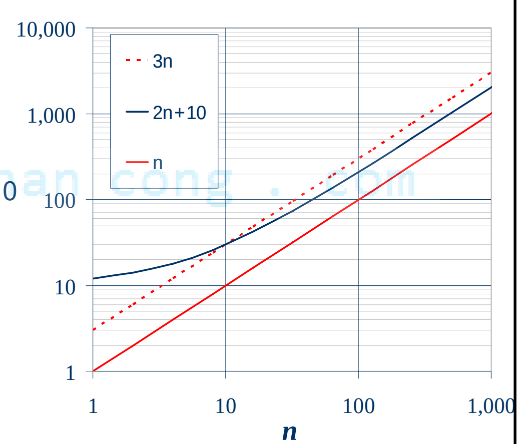
    điều này đúng với c = 4 và n0 = 21
- 3 log n + 5
  - 3 log n + 5 là O(log n)

    cần c > 0 và n0 ≥ 1 sao cho 3 log n + 5 ≤ c*log n với n ≥ n0

    ta xác định được c = 8 và n0 = 2

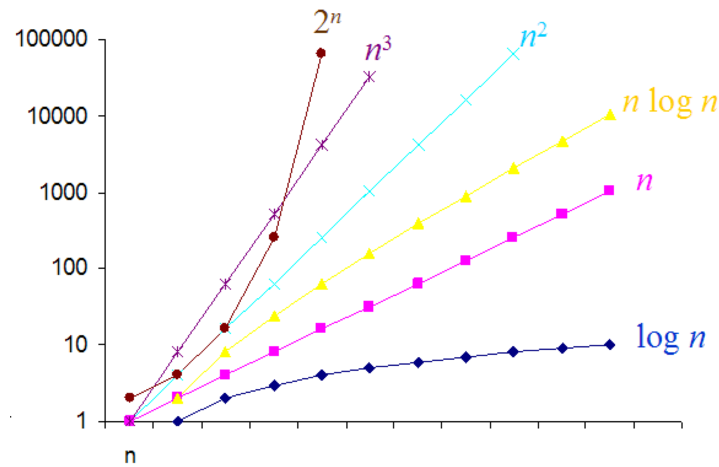
Ký hiệu tiệm cận Big - O

- Ví dụ: Giải thuật có T(n) = 2***n*** + 10 thì có độ phức tạp là ***O***(***n***)
  - $2n + 10 \le cn$
  - $(c - 2)\, n \ge 10$
  - $n \ge 10/(c - 2)$
  - Lấy $c = 3$ và $n_0 = 10$



Ký hiệu tiệm cận Big - O

– Đồ thị một số hàm cơ bản



Ký hiệu tiệm cận Big - O

– Big-O và độ tăng trưởng

- Big-O là **ký hiệu tiệm cận trên** của một hàm
- Nếu ta có T(n) là O(g(n)) thì độ tăng trưởng của T(n) không vượt quá độ tăng trưởng của g(n)

Ký hiệu tiệm cận Big - O

- Qui tắc xác định độ phức tạp về thời gian
  - Hàm thời gian T(n) của một đoạn của thuật toán là đa thức bậc k thì T(n) là $O(n^k)$
  - $n^x = O(a^n)$, với bất kỳ $x > 0$ và $a > 1$
  - $\log n^x = O(\log n)$, với $x > 0$

Ký hiệu tiệm cận Big - O

- Qui tắc xác định độ phức tạp
  - Cấu trúc tuần tự - Qui tắc tổng
    - Cho 2 đoạn của thuật toán P_1 và P_2 với thời gian thực hiện tương ứng là $T_1(n)$ và $T_2(n)$. Thời gian thực hiện P_1 và P_2 kế tiếp nhau là: $T_1(n) + T_2(n)$
    - Độ phức tạp của hai đoạn chương trình P_1 và P_2 liên tục nhau có thể xác định là $O(\max(f(n), g(n)))$ nếu $T_1(n) = O(f(n))$ và $T_2(n) = O(g(n))$.

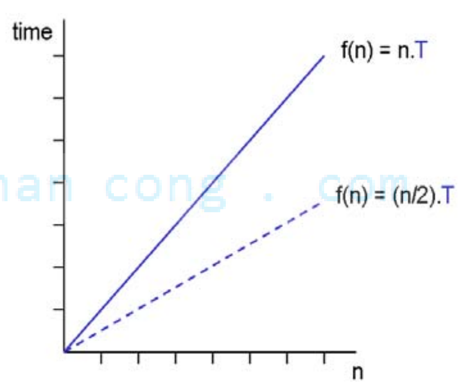
Ký hiệu tiệm cận Big - O

– Qui tắc xác định độ phức tạp
  - Cấu trúc lồng - Quy tắc nhân
    - Cho 2 đoạn chương trình P_1 và P_2 với thời gian thực hiện tương ứng là $T_1(n)$ và $T_2(n)$. Thời gian thực hiện P_1 và P_2 lồng vào nhau là: $T_1(n)T_2(n)$
    - Độ phức tạp của hai đoạn chương trình P_1 và P_2 liên tục nhau có thể xác định là O(f(n)*g(n)) nếu $T_1(n) = O(f(n))$ và $T_2(n) = O(g(n))$.

Ký hiệu tiệm cận Big - O

```
for i = 1 to n
begin
    P; {đoạn giải thuật với thời
    gian thực hiện T}
end
```

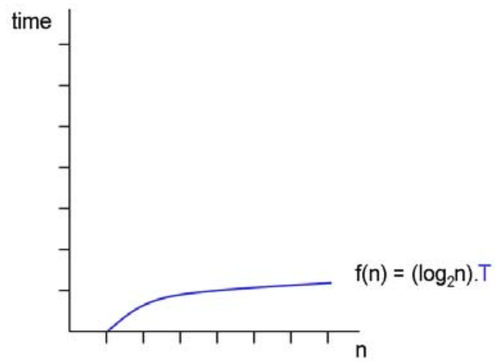
```
i: = 1
while (i <= n) do
begin
    P; {đoạn giải thuật với thời
    gian thực hiện T}
    i: = i+2;
end
```



Ký hiệu tiệm cận Big - O

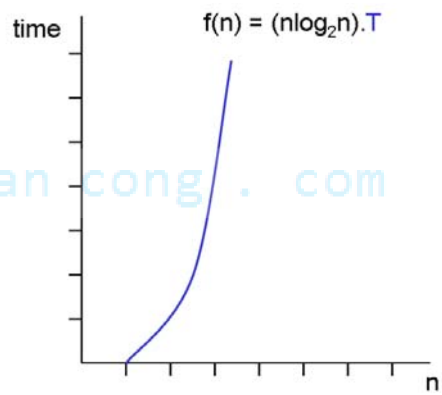
```
i := 1
while (i<= n) do
begin
    P; {đoạn giải thuật với thời
    gian thực hiện T}
    i:= i *2;
end
```

```
i :=n
while (i >= 1) do
begin
    P; {đoạn giải thuật với thời
    gian thực hiện T}
    i:= i/2
end
```



Ký hiệu tiệm cận Big - O

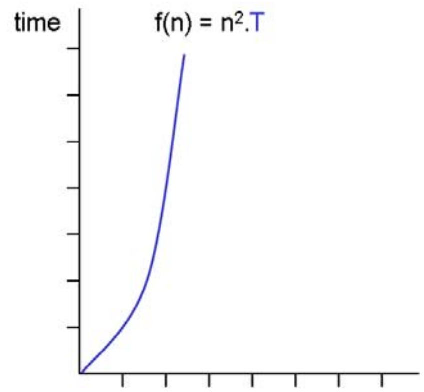
```
i = 1
while  (i <= n) do
begin
    j := 1 ;
    while (j <= n) do
    begin
            P ; {đoạn giải thuật với
    thời gian thực hiện T}
            j := j × 2;
    end
    i =: i + 1;
end
```



Ký hiệu tiệm cận Big - O

- Ví dụ

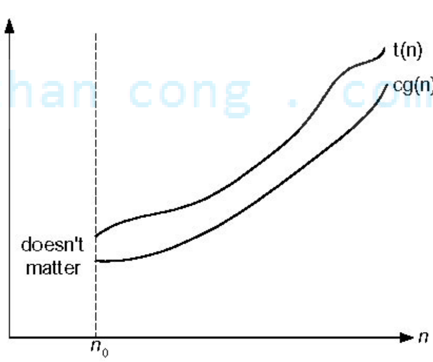
```
i = 1
while  (i <= n) do
begin
     j := 1 ;
     while (j <= n) do
     begin
              P;
              j := j +1;
     end
     i =: i + 1;
end
```



Các khái niệm tiệm cận khác

- Big- Omega
  - t(n) được coi là Ω(g(n)) nếu tồn tại một hằng số c >0 và một số nguyên $n_0 \geq 1$ sao cho $T(n) \geq c*g(n)$ với mọi $n \geq n_0$

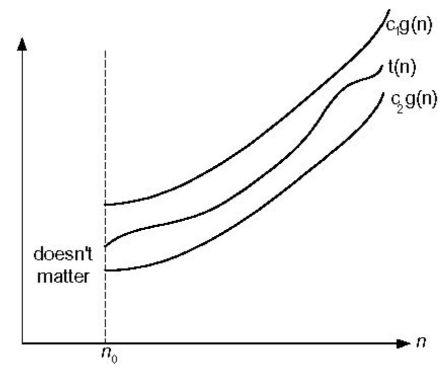
$t(n) \in \Omega(g(n))$



Các khái niệm tiệm cận khác

- Big-Theta
  - t(n) đựoc coi là $\Theta(g(n))$ nếu tồn tại hai hằng số c' > 0 và c'' > 0 và một số nguyên $n_0 \geq 1$ sao cho $c'*g(n) \leq T(n) \leq c''*g(n)$ với mọi $n \geq n_0$

$t(n) \in \Theta(g(n))$



Các khái niệm tiệm cận khác

- $5n^2 = \Omega(n^2)$ với c = 5 và $n_0 = 1$
- $5n^2 = \Omega(n)$ với c = 1 và $n_0 = 1$
- $5n^2 = \Theta(n^2)$ với c = 5 và $n_0 = 1$
- $3 \log(n) + \log(\log n) = \Omega(\log n)$ với c = 3 và $n_0 = 2$