Độ phức tạp thuật toán

TOÁN RỜI RẠC

NỘI DUNG

- Thuật toán
- Độ phức tạp thuật toán

THUẬT TOÁN

ĐỊNH NGHĨA

- Định nghĩa: *Thuật toán là một dãy các quy tắc, sao cho sau một số hữu hạn bước thực hiện các quy tắc này ta đạt được mục tiêu cần làm*
- Ví dụ: Thuật toán tìm số lớn nhất trong dãy $\{a_1, \ldots, a_n\}$
  - Bước 1: đặt max = a_1
  - Bước 2: nếu max < a_2 thì max = a_2
  - Bước 3: nếu max < a_3 thì max = a_3
  - …
  - Bước n: nếu max < a_n thì max = a_n

ĐẶC TRƯNG CỦA THUẬT TOÁN

- Đầu vào/đầu ra: dữ liệu vào và kết quả thu được
- Chính xác: kết quả đầu ra, trung gian đúng, nhất quán, phụ thuộc dữ liệu vào
- Dừng: thuật toán sẽ dừng sau một số hữu hạn bước (không giới hạn độ lớn) với bất kỳ bộ dữ liệu nào
- Xác định: từng bước phải được xác định rõ ràng, không gây ra nhập nhằng
- Tổng quát: có thể giải bài toán bất kỳ trong lớp bài toán được thiết kế để sử dụng

CÁCH MÔ TẢ THUẬT TOÁN

- Ngôn ngữ tự nhiên
  - Tên thuật toán + chức năng
  - Đầu vào
  - Đầu ra
  - Biến phụ (nếu có)
  - Chuỗi quy tắc thực hiện, các quy tắc được đánh số bằng các số tự nhiên

CÁCH MÔ TẢ THUẬT TOÁN

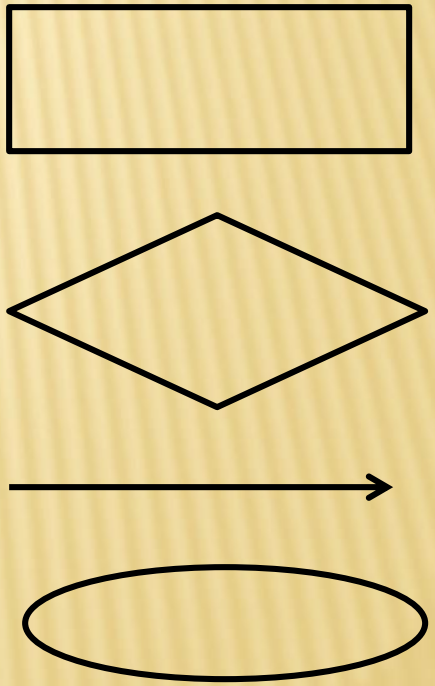
- Ngôn ngữ tự nhiên– ví dụ:
  - Giải thuật tìm UCLN của 2 số tự nhiên a và b
  - Đầu vào: 2 số tự nhiên a, b
  - Đầu ra: ước chung lớn nhất của 2 số đó
  - Thực hiện:
    - Bước 0: xác định a và b
    - Bước 1: nếu a > b thì đặt a = a-b
    - Bước 2: nếu a < b thì đặt b = b-a
    - Bước 3: nếu a != b thì quay lại bước 1
    - Bước 4: nếu a = b thông báo a là ước chung lớn nhất
    - Bước 5: kết thúc thuật toán

CÁCH MÔ TẢ THUẬT TOÁN

- Sơ đồ khối

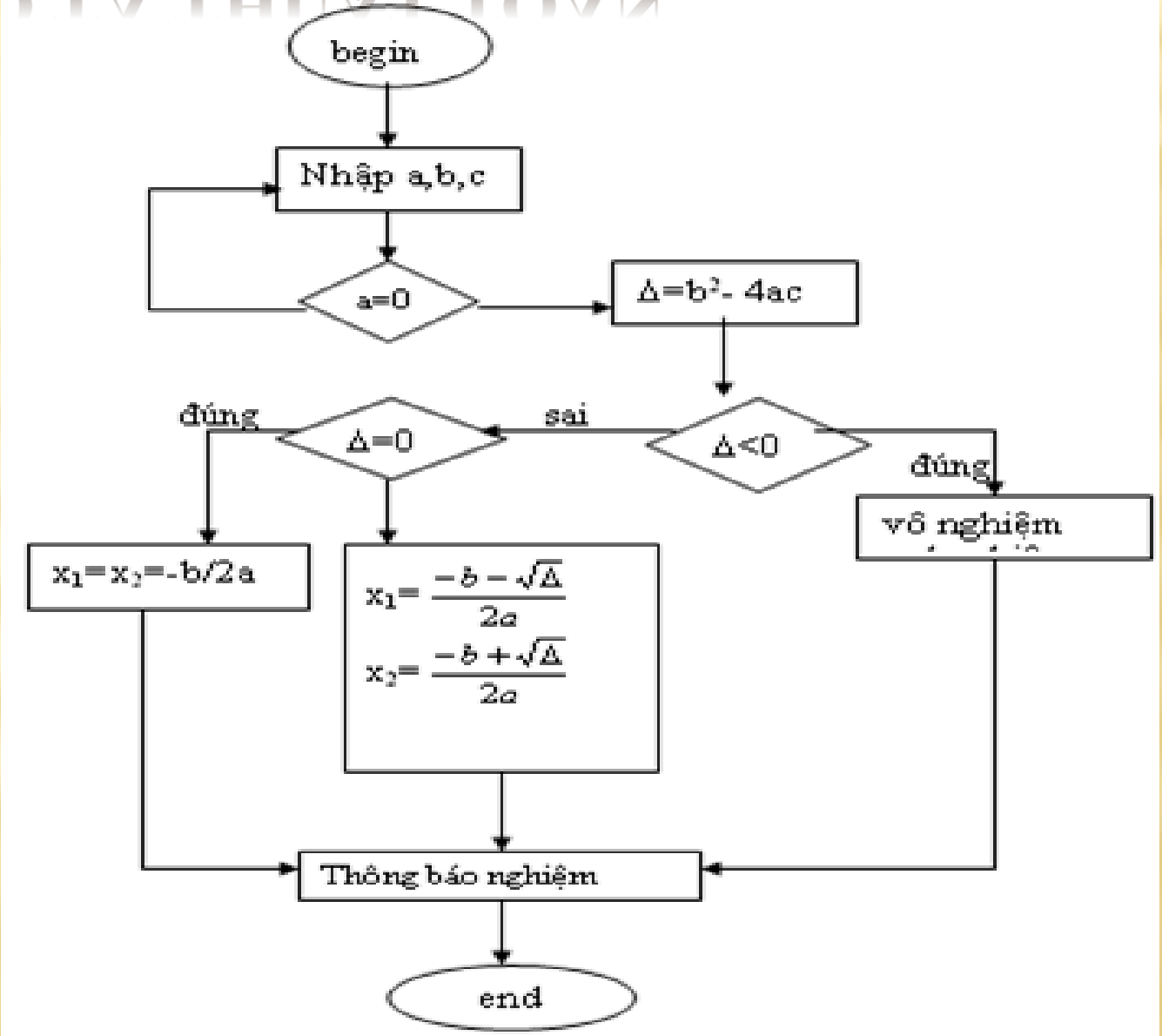
Sử dụng các loại khối:

- Khối thao tác xử lý: hình chữ nhật, chứa một hay nhiều câu lệnh cần thực hiện
- Khối chọn lựa: biểu diễn bằng hình thoi, chứa biểu thức điều kiện cần kiểm tra
- Đường đi: biểu diễn bằng đường nối có mũi tên chỉ trình tự thực hiện của các khối
- Bắt đầu, kết thúc: hình elip.



CÁCH MÔ TẢ THUẬT TOÁN

- Ví dụ:



CÁCH MÔ TẢ THUẬT TOÁN

- Giả mã / ngôn ngữ lập trình: chương trình là một dãy hữu hạn các câu lệnh viết theo quy tắc nhất định trên một ngôn ngữ lập trình nào đó (Pascal, C,…)
- Ví dụ: giải phương trình bậc 2

CÁCH MÔ TẢ THUẬT TOÁN

```
#include <math.h>
void GiaiPTBacHai(float a, float b, float c)
{
    float D = b*b - 4*a*c;
    if(D<0)
        printf("Vo nghiem");
    else if(D == 0)
        printf("%f", -b/(2*a));
    else
    {
        printf("%f", (-b + sqrt(D))/(2*a));
        printf("%f", (-b - sqrt(D))/(2*a));
    }
}
```

VẤN ĐỀ LIÊN QUAN

- Một bài toán có thể giải bằng nhiều thuật toán
- Đối với một thuật toán:
  - Độ phức tạp về không gian (dung lượng bộ nhớ sử dụng)
  - Độ phức tạp về thời gian
- Độ phức tạp về thời gian chạy:
  - Thiết kế thuật toán
  - Chương trình dịch
  - Tốc độ thực hiện các phép toán của máy
  - Bộ dữ liệu vào

ĐỘ PHỨC TẠP THUẬT TOÁN

GIỚI THIỆU

- Thời gian chạy của thuật toán phụ thuộc vào cỡ của dữ liệu vào
  - Ví dụ:Tìm giá trị lớn nhất, nhỏ nhất của dãy số, sắp xếp dãy số tăng dần, tìm số đường đi giữa hai đỉnh của đồ thị
- Với cùng cỡ thì thời gian chạy với những bộ dữ liệu khác nhau là khác nhau
  - Ví dụ: Tìm xem trong một dãy n phần tử có xuất hiện giá trị x không?

ĐỘ PHỨC TẠP THUẬT TOÁN

Được đánh giá dựa trên thời gian chạy của thuật toán

Các loại thời gian chạy được quan tâm:

- Thời gian chạy tốt nhất: thời gian tối thiểu cần thiết để thực hiện thuật toán với mọi bộ dữ liệu cùng cỡ
- Thời gian chạy tồi nhất: thời gian tối đa cần thiết để thực hiện thuật toán với mọi bộ dữ liệu cùng cỡ
- Thời gian chạy trung bình: là trung bình cộng thời gian chạy trên tất cả các bộ dữ liệu cùng cỡ

ĐỘ PHỨC TẠP THUẬT TOÁN

- Đánh giá thời gian chạy: bằng số lượng **phép toán sơ cấp** phải thực hiện (phép toán số học, lôgic, so sánh).
- Kết quả: hàm t(n)
- Độ phức tạp dựa vào tốc độ tăng của hàm t(n)
- Ví dụ:

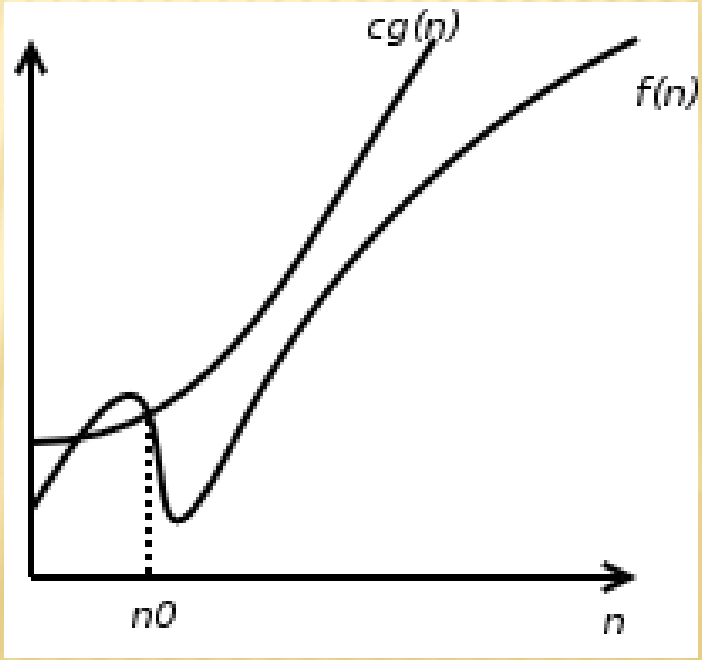
  $t(n) = 2n^2 + n - 15$

KÝ HIỆU O

- Định nghĩa: *Cho hàm f,g : N → R, ta nói f là O(g) nếu tồn tại các hằng số C và n_0 thỏa mãn*

$$\forall n > n_0.\ |f(n)| \leq C|g(n)|$$

(đọc là: Ô lớn)



KÝ HIỆU O

- Ví dụ: $t(n) = 2n^2 + n - 15$

  $\forall n > 15: |t(n)| < 2n^2 + n^2 = 3n^2$

  $t(n) = O(3n^2)$

  $t(n) = O(n^3), O(n^4)\ldots$

  $t(n) = O(n^{3/2})$?

- Để biểu diễn thời gian chạy, ta dùng cận trên chặt

BIỂU DIỄN THỜI GIAN CHẠY BỞI KÍ HIỆU O

- f(n) là cận trên chặt của t(n) nếu:

$$\begin{cases} t(n) = O(f(n)) \\ t(n) = O(g(n)) \end{cases} \Rightarrow f(n) = O(g(n))$$

Nói cách khác: ta không thể tìm được hàm g(n) là cận trên của t(n) mà g(n) tăng chậm hơn f(n).

- $t(n) = 2n^2 + n - 15 \Rightarrow t(n) = O(n^2)$
- Tổng quát: Nếu t(n) là 1 đa thức bậc k theo biến n

$t(n) = a_0 + a_1 x + a_2 x^2 + \cdots + a_k n^k$ thì $t(n) = O(n^k)$

CÁC QUY TẮC CƠ BẢN

- Loại bỏ hằng số:

Nếu f(n) = O(c.g(n)) thì f(n) = O(g(n))

Ví dụ: $f(n) = 3n^2 + 4n \Rightarrow f(n) = O(3n^2) = O(n^2)$

- Loại bỏ phần phụ:

Nếu ta có t(n) = O(f(n) + g(n)) thì ta cũng có

t(n) = O(max{f(n), g(n)})

Ví dụ: nếu $t(n) = O(n^3 + n^2 + 9)$ thì cũng có $t(n) = O(n^3)$

CÁC QUY TẮC CƠ BẢN

- Quy tắc cộng:

Nếu có k đoạn chương trình liên tiếp, và có độ phức tạp lần lượt là $t_1 = O(g_1)$, $t_2 = O(g_2)$, ..., $t_k = O(g_k)$ thì độ phức tạp của toàn bộ chương trình là:

$$t(n) = t_1(n) + t_2(n) + ... + t_k(n) = O(\max\{g_1, g_2, ... g_k\})$$

- Quy tắc nhân:

Nếu có 2 đoạn chương trình lồng nhau, độ phức tạp lần lượt là $t_1(n) = O(g_1)$ và $t_2(n) = O(g_2)$ thì độ phức tạp của cả chương trình là

$$t(n) = t_1(n).t_2(n) = O(g_1.g_2)$$

VÍ DỤ 1

```
Sum = 0;
Prod = 1;
for (i = 0; i <n; i++)
        sum = sum + a[i]*i
for (i = 0; i <n; i++)
        prod = prod*a[i]*i
```

Độ phức tạp = ?

VÍ DỤ 2

```
for (i = 0; i <n; i++)
        for (j = 0; j < n; j++)
                a[i][j] = 0;
for (i = 0; i <n; i++)
        a[i][i] = 1;
```

Độ phức tạp = ?

VÍ DỤ 3

```
res = n;
for (i = 0; i < n-1; i++)
        for (j = i+1; j<n; j++)
                for (k= 0, k<10, k++)
                        res = res + i*j
```

Độ phức tạp = ?

CÁC ĐÁNH GIÁ THÔNG DỤNG

Ký kiệu Ô lớn	Tên gọi
O(1)	Hằng số
O(lg n)	Logarit
O(n)	Tuyến tính
O(n lg n)	N log n (linearithmic)
$O(n^2)$	Bậc 2
$O(n^a)$	Đa thức
$O(m^n)$, m > 2	Hàm mũ
O(n!)	Giai thừa