Giới thiệu về thuật toán

Nhập môn lập trình

Trình bày: …; Email: …@fit.hcmus.edu.vn

Nội dung

- Khái niệm về thuật toán
- Chương trình cài đặt thuật toán
- Độ phức tạp của thuật toán
- Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp
- Thuật ngữ và bài đọc thêm tiếng Anh

Khái niệm về thuật toán

Bài toán và thuật giải

- Máy tính là một công cụ đắc lực hỗ trợ con người trong việc tính toán và xử lý.
- Phát biểu bài toán bằng ngôn ngữ tự nhiên không thể là đầu vào cho máy tính.
- Con người phải mô hình hóa bài toán thông qua những cấu trúc dữ liệu, vốn được hỗ trợ bởi các ngôn ngữ lập trình, từ cơ sở đến nâng cao như mảng, cấu trúc, tập hợp, đồ thị, cây, …

Bài toán và thuật giải

- Trên cơ sở mô hình dữ liệu đã được xây dựng, con người phải chỉ ra cho máy tính một cách thức để giải quyết bài toán (gọi là thuật toán hay giải thuật).
- Thuật toán có thể hiểu là một qui trình xử lý bao gồm các bước cụ thể có thể thực hiện để giải quyết một bài toán.

Các tiêu chuẩn của thuật toán

- Mỗi thuật toán cần đáp ứng 6 tiêu chuẩn:
  - **Tính hữu hạn**: Thuật toán phải kết thúc thực thi sau một số lượng hữu hạn các bước xử lý.
  - **Tính xác định**: Mỗi bước xử lý phải được mô tả rõ ràng, chính xác, không nhập nhằng.
  - **Tồn tại dữ liệu đầu vào**: Thuật toán phải có dữ liệu đầu vào hợp lệ, được mô tả rõ ràng.
  - **Tính có kết quả**: Thuật toán phải cho ra kết quả đúng trên cơ sở dữ liệu đầu vào hợp lệ.
  - **Tín hiệu quả**: Mỗi bước xử lý phải đơn giản với thời gian thực thi hữu hạn. Trong thực tế điều này có nghĩa là phải thực thi trong khoảng thời gian có thể chấp nhận được.
  - **Tính phổ dụng**: Thuật toán có thể áp dụng để xử lý một họ các bài toán.

Mô tả thuật toán

- Chúng ta có thể sử dụng một trong bốn cách sau để mô tả thuật toán:
  - Ngôn ngữ tự nhiên: tiếng Việt, tiếng Anh, …
  - Lưu đồ.
  - Mã giả: thường dựa vào cú pháp của một số ngôn ngữ lập trình thông dụng như Pascal, C/C++, …
  - Ngôn ngữ lập trình cấp cao.
- Không nên đi quá sâu vào chi tiết kỹ thuật làm mất đi tính trừu tượng của thuật toán.

Mô tả thuật toán bằng NNTN

- Ví dụ tìm số lớn nhất trong dãy
  - **Bước 1**: Đặt ***số lớn nhất hiện tại*** bằng với số nguyên không dấu đầu tiên của dãy.
  - **Bước 2**: Nếu không còn số nguyên nào kế tiếp thì chuyển sang Bước 4.
  - **Bước 3**: Nếu số nguyên kế tiếp lớn hơn ***số lớn nhất hiện tại*** thì đặt số lớn nhất hiện tại bằng số nguyên kế tiếp này. Quay về Bước 2.
  - **Bước 4**: ***Số lớn nhất hiện tại*** chính là số lớn nhất của cả dãy, đây là kết quả của thuật toán.

Mô tả thuật toán bằng NNLT

- Ví dụ tìm số lớn nhất trong dãy

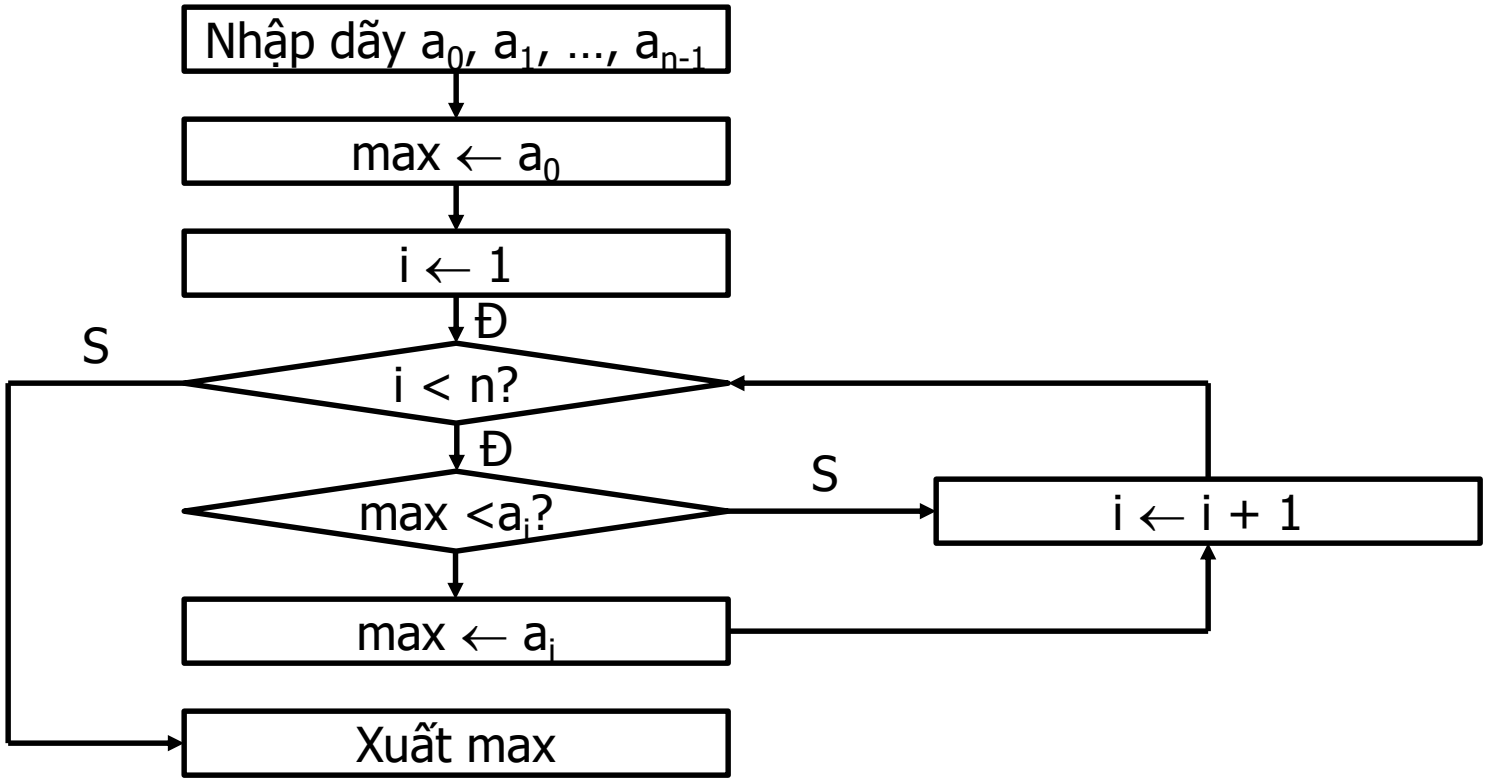
```
int timSoLonNhat(int a[], int n) {
    int i, max;
    for (i = 1; i < n; i++)
        if (max < a[i])
            max = a[i];
    return max;
}
```

Mô tả thuật toán bằng lưu đồ

- Sử dụng các khối hình sau:
  - Các hình chữ nhật biểu thị các chỉ chị tính toán hay xử lý (nhập, xuất, gán, thực hiện phép tính).
  - Các hình thoi thể hiện các quyết định rẽ nhánh tùy theo biểu thức trong hình thoi có giá trị đúng (Đ) hay sai (S).
  - Các mũi tên là hướng đi của luồng điều khiển, thể hiện thứ tự thực hiện của các khối xử lý.

Mô tả thuật toán bằng lưu đồ

- Ví dụ tìm số lớn nhất trong dãy



Mô tả thuật toán bằng mã giả

- Ví dụ tìm số lớn nhất trong dãy

max $\leftarrow a_0$
i $\leftarrow$ 1
while i < n do
    if max < a_i **then**
        max $\leftarrow a_i$
    i $\leftarrow$ i + 1
endwhile
write (max)

Chương trình cài đặt thuật toán

Xem giáo trình NMLT

- Tổ chức dữ liệu cho mỗi hàm chương trình
  - Dữ liệu nhập, xuất và tính toán trung gian
- Tổ chức các hàm cho chương trình
  - Hàm về nhập/xuất, xử lý (cài đặt các thuật toán) và chương trình chính, kết nối
- Chạy thử nghiệm thuật toán
  - Chuẩn bị các bộ dữ liệu kiểm thử (dữ liệu nhập và kết quả mong đợi)
  - Chạy thử, ghi nhận kết quả và đánh giá đúng sai

Độ phức tạp thuật toán

Khái niệm

- Đối với một vấn đề đặt ra có thể tồn tại rất nhiều thuật toán xử lý.
- Ngoài việc phải chỉ ra rằng một thuật toán có tính đúng đắn ta còn phải biết thuật toán nào tốt hơn khi giải quyết cùng một vấn đề (theo nghĩa chạy nhanh hơn hay độ phức tạp thấp hơn).

Độ phức tạp về thời gian

- Trong cùng một điều kiện hoạt động (dữ liệu đầu vào, tốc độ phần cứng, …) thì thuật toán nào cho kết quả sớm nhất sẽ là thuật toán tốt nhất.
- Tuy nhiên, để đảm bảo điều kiện hoạt động đồng nhất là rất khó vì trong hệ thống đa nhiệm thì CPU không dành 100% công suất để phục vụ riêng cho chương trình đang chạy thử nghiệm.

Độ phức tạp về thời gian

- Không phải bất cứ nhóm thuật toán nào cũng có thể được cân đo, đong đếm một cách tường minh.
- Ví dụ tìm số Fibonacci thứ n, biết

  $F_0 = F_1 = 1$

  $F_n = F_{n-1} + F_{n-2}$ nếu $n \geq 2$

Tìm số Fibonacci thứ n

- Thuật toán thứ nhất

	Mã giả	Cài đặt bằng C/C++
1	**Computing:** Fibonacci(n)	int Fibo1(int n) {
2	**if** n <= 1 **then**	int a, b, c, k;
3	Result = n	if (n <= 1)
4	**else**	return n;
5	a = 0, b = 1	a = 0; b = 1;
6	**for each** k = 2, …, n do	for (k = 2; k <= n; k++) {
7	c = a + b;	c = a + b;
8	a = b	a = b;
9	b = c	b = c;
10	**endfor**	}
11	Result = c	return c;
12	**endif**	}
13	**write** (Result)	

Tìm số Fibonacci thứ n

- Thuật toán thứ hai

	Mã giả	Cài đặt bằng C/C++
1	**Computing:** Fibonacci(n)	int Fibo2(int n) {
2	**if** n <= 1 **then**	if (n <= 1)
3	Result = n	return n;
4	**else**	
5	A = Fibonacci(n – 2)	int A = Fibo2(n – 1);
6	B = Fibonacci(n – 1)	int B = Fibo2(n – 2);
7	Result = A + B	return A + B;
8	**endif**	}
9	**write** (Result)	

Tìm số Fibonacci thứ n

- Đánh giá độ phức tạp thời gian (so sánh thời gian thực hiện của hai thuật toán)

n	Thuật toán 1	Thuật toán 2
40	41 ns	1048 μs
60	61 ns	1 giây
80	81 ns	18 phút
100	101 ns	13 ngày
120	121 ns	36 năm
160	161 ns	3.8×10^7 năm
200	201 ns	4×10^{13} năm

Độ phức tạp về không gian

- Dựa trên mức độ tiêu thụ tài nguyên của hệ thống (như bộ nhớ, đường truyền, …) để làm cơ sở đánh giá thuật toán.
- Dựa và cấu trúc dữ liệu được sử dụng trong biểu diễn dữ liệu cũng như trong quá trình xử lý của thuật toán.
- Độ phức tạp này thường không được chú ý nhiều.

Thuật toán hoán đổi giá trị

- Hoán đổi giá trị của a và b.

	Thuật toán thứ nhất	Thuật toán thứ 2
1	temp ← a	a ← a + b
2	a ← b	b ← a – b
3	b ← temp	a ← a – b

- Thuật toán thứ nhất tồi hơn thuật toán thứ hai (trên phương diện độ phức tạp về không gian) do cần thêm một biến temp để lưu trữ giá trị tạm thời.

Các vấn đề tìm hiểu mở rộng kiến thức nghề nghiệp

Xem giáo trình NMLT (152-156)

- Tham khảo một số ví dụ về **đánh giá độ phức tạp lý thuyết** của thuật toán tìm kiếm tuyến tính và nhị phân.

Thuật ngữ
và bài đọc thêm tiếng Anh

Thuật ngữ tiếng Anh

- ***algorithm***: cách thức hay qui trình để giải quyết bài toán
- ***algorithmic complexity***: độ phức tạp thuật toán
- ***algorithm implementation***: cài đặt thuật toán, chuyển thuật toán thành các chỉ thị được viết trong một NNLT cụ thể, cũng nghĩa với mã hóa thuật toán (algorithm coding)
- ***executable***: có thể chạy được
- ***flow chart***: lưu đồ, một phương tiện trực quan dùng để mô tả sự hoạt động của thuật toán
- ***natural languages***: ngôn ngữ tự nhiên (tiếng Việt, Anh, Pháp, …)
- ***pseudo-code***: mã giả, một phương tiện thông dụng được dùng để mô tả và trình bày thuật toán

Bài đọc thêm tiếng Anh

- **Thinking in C**, Bruce Eckel, E-book, 2006.
- **Theory and Problems of Fundamentals of Computing with C++**, John R.Hubbard, Schaum's Outlines Series, McGraw-Hill, 1998.