ĐẠI HỌC THÁI NGUYÊN
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

LÊ ĐÌNH LONG

MỘT SỐ THUẬT TOÁN CHỌN LỌC VÀ ỨNG DỤNG TRONG TIN HỌC PHỔ THÔNG

Chuyên ngành: KHOA HỌC MÁY TÍNH
Mã số: 60 48 0101

LUẬN VĂN THẠC SĨ KHOA HỌC MÁY TÍNH

NGƯỜI HƯỚNG DẪN KHOA HỌC
TS. VŨ VINH QUANG

Thái Nguyên - 2015

MỞ ĐẦU

Thuật toán là một trong những khái niệm quan trọng nhất trong tin học. Thuật toán xuất phát từ nhà khoa học Arập Abu Ja'far Mohammed ibn Musa al Khowarizmi. Chúng ta có thể xem thuật toán là một công cụ dùng để giải bài toán được xác định trước. Việc nghiên cứu về thuật toán có vai trò rất quan trọng trong khoa học máy tính vì máy tính chỉ giải quyết được vấn đề khi đã có hướng dẫn giải rõ ràng và đúng đắn. Nếu hướng dẫn giải sai hoặc không rõ ràng thì máy tính không thể giải đúng được bài toán. Trong khoa học máy tính, thuật toán được định nghĩa là một dãy hữu hạn các thao tác được sắp xếp theo một trình tự nhất định sao cho sau khi thực hiện dãy thao tác ấy, từ input của bài toán, ta nhận được output cần tìm.

Ở Việt Nam môn Tin học được đưa vào giảng dạy chính thức ở trường phổ thông từ năm học 2006 - 2007 tuy nhiên trong thực tế môn Tin học đã được đưa vào tham gia thi học sinh giỏi cấp tỉnh, cấp quốc gia từ rất lâu: Hội thi Tin học trẻ không chuyên toàn quốc được tổ chức lần đầu vào năm 1995, kỳ thi học sinh giỏi Tin học quốc gia được tổ chức vào năm 1995 và đặc biệt kỳ thi Olympic Tin học quốc tế (IOI) tổ chức lần đầu vào năm 1989. Từ đó đến nay các kỳ thi học sinh giỏi, Olympic Tin học ngày một nhiều và đòi hỏi kiến thức rất cao.

Chúng ta biết rằng để có kết quả cao trong kỳ thi chọn học sinh giỏi môn Tin học nói chung thì học sinh phải có vốn kiến thức về thuật toán để giải được các bài toán khó (đặc biệt là các thuật toán nâng cao), sau đó học sinh sẽ sử dụng ngôn ngữ lập trình nào đó để lập trình dựa vào thuật toán đã tìm được và giải bài toán theo yêu cầu. Chương trình giảng dạy ở sách giáo khoa của môn Tin học hiện hành trong trường phổ thông có lượng kiến thức rất hạn chế và đơn giản, không đủ cơ sở để học sinh có thể dựa vào vốn kiến thức đó để tham gia một kỳ thi học sinh giỏi cấp thành phố hay cấp cao hơn. Câu hỏi đặt ra: ***"Làm thế nào để học sinh có thể đạt kết quả cao trong các kỳ thi học sinh giỏi môn Tin học trong trường phổ thông?"*** yêu cầu đặt ra là các giáo viên giảng dạy môn Tin học trong trường phổ thông phải suy nghĩ, tìm tòi tài liệu về một số thuật toán như: Thuật toán đệ quy**,** thuật toán

tham lam, thuật toán xấp xỉ và một số thuật toán trên đồ thị ... là những thuật toán sử dụng hiệu quả để giải nhiều bài toán Tin học.

Xuất phát từ thực tế đó, đề tài luận văn: **"MỘT SỐ THUẬT TOÁN CHỌN LỌC VÀ ỨNG DỤNG TRONG TIN HỌC PHỔ THÔNG"** với mục đích tìm hiểu, nghiên cứu một số thuật toán và cách ứng dụng vào giảng dạy, bồi dưỡng đội tuyển học sinh giỏi môn Tin học ở trường phổ thông.

Nội dung chính của luận văn gồm 3 chương, phần phụ lục với các nội dung chính như sau:

Chương 1: *Luận văn trình bày tổng quan về các khái niệm cơ bản về thuật toán và độ phức tạp của thuật toán, vấn đề phân lớp các bài toán trên cơ sở đánh giá độ phức tạp của thuật toán. Các kiến thức này sẽ là nền tảng về mặt lý thuyết tính toán để nghiên cứu các chương tiếp sau của luận văn.*

Chương 2: *Trong chương này luận văn trình bày tổng quan về thuật toán đệ quy, thuật toán tham lam, thuật toán xấp xỉ và một số thuật toán trên mô hình đồ thị.*

Chương 3: *Dựa vào cơ sở lý thuyết của thuật toán được trình bày ở chương 2, trong chương này luận văn sẽ cài đặt chương trình cho một số bài toán cụ thể.*

Phần phụ lục:

Toàn bộ các kết quả thực nghiệm giải các bài toán được cài đặt bằng ngôn ngữ Pascal version 7.0 trên máy tính PC.

Chương 1
CÁC KHÁI NIỆM VỀ THUẬT TOÁN VÀ ĐỘ PHỨC TẠP CỦA THUẬT TOÁN

1.1 Khái niệm cơ bản về thuật toán

1.1.1 Khái niệm bài toán Tin học

Trong phạm vi tin học, người ta quan niệm ***bài toán là một công việc nào đó muốn máy tính thực hiện*** [2].

Khi dùng máy tính để giải bài toán, ta cần quan tâm tới 2 vấn đề: Dữ liệu cần được đưa vào máy tính (Input) là gì? và cần lấy ra (Output) thông tin gì? nói một cách khác, cho một bài toán là việc mô tả rõ input và output của bài toán.

Vấn đề còn lại là: Làm thế nào để từ input ta có được output?

1.1.2 Khái niệm thuật toán

Khác với toán học (các yêu cầu của bài toán thường là chứng minh sự tồn tại đáp án chứ không yêu cầu tìm một cách chi tiết để tìm ra đáp số đó), giải một bài toán Tin học là việc đi tìm một lời giải cụ thể, tường minh để đưa ra output của bài toán dựa trên input đã cho. Việc chỉ ra một cách tìm output của bài toán gọi là một thuật toán. Có nhiều cách phát biểu khái niệm về thuật toán. Dưới đây là cách phát biểu được chọn để đưa vào sách giáo khoa Tin học phổ thông.

Khái niệm về thuật toán: ***Thuật toán là một dãy hữu hạn các thao tác được sắp xếp theo một trình tự nhất định để sau khi thực hiện dãy các thao tác đó, từ input ta có output cần tìm*** [2].

Trong lĩnh vực khoa học máy tính, cụm từ "thuật toán" đôi khi còn được gọi là: "giải thuật".

Ví dụ 1: Thuật toán tô màu đồ thị

- Input: đồ thị G = (V, E).
- Output: đồ thị G = (V, E) có các đỉnh đã được gán màu.

Thuật toán: Có nhiều cách để mô tả thuật toán khác nhau. Dưới đây là cách mô tả thuật toán dạng liệt kê các bước:

Bước 1: Lập danh sách các đỉnh của đồ thị $E' := [v_1, v_2, ..., v_n]$ được sắp xếp theo thứ tự bậc giảm dần: $d(v_1) \geq d(v2) \geq ... d(v_n)$

Đặt i := 1;

Bước 2: Tô màu i cho đỉnh đầu tiên trong danh sách. Duyệt lần lượt các đỉnh tiếp theo và tô màu i cho đỉnh không kề đỉnh đã được tô màu i.

Bước 3: Nếu tất cả các đỉnh đã được tô màu thì kết thúc, đồ thị được tô bằng i màu. Ngược lại, chuyển sang bước 4;

Bước 4: Loại khỏi E' các đỉnh đã tô màu. Sắp xếp lại các đỉnh trong E' theo thứ tự bậc giảm dần.

Đặt i := i +1 và quay lại bước 2.

1.2 Yêu cầu của thuật toán

Thuật toán phải đảm bảo được các yêu cầu sau đây [2], [4].

1. Tính xác định: Các bước của thuật toán phải được trình bày rõ ràng, mạch lạc, đảm bảo cho người đọc chỉ hiểu theo một nghĩa duy nhất.

2. Tính khả thi: Thuật toán phải thực hiện được, nghĩa là ta có thể sử dụng máy tính kết hợp giữa các ngôn ngữ lập trình để thể hiện thuật toán hay có thể kiểm tra thuật toán chỉ bằng giấy và bút (còn gọi là Test).

3. Tính dừng: Nếu dữ liệu vào thỏa mãn điều kiện đầu vào thì thuật toán phải kết thúc và cho ra kết quả sau một số hữu hạn bước.

4. Tính chính xác (tính đúng đắn): Thuật toán phải cho kết quả chính xác và thể hiện đúng đắn trên cơ sở toán học.

5. Tính tối ưu: Thuật toán phải có chi phí về không gian bộ nhớ ít nhất và chạy trong thời gian nhanh nhất.

1.3. Thể hiện thuật toán

Thuật toán được thể hiện bằng một trong các cách sau

- Sử dụng liệt kê các bước.
- Sử dụng lưu đồ (sơ đồ khối).
- Sử dụng ngôn ngữ lập trình.

1.4 Độ phức tạp của thuật toán

1.4.1 Chi phí phải trả cho một quá trình tính toán

Chi phí phải trả cho một quá trình tính toán bao gồm chi phí về không gian (bộ nhớ - số ô nhớ cần sử dụng trong quá trình tính toán) và chi phí về thời gian (thời gian cần sử dụng cho một quá trình tính toán).

Nếu cho một thuật toán A. Thuật toán này thực hiện trên bộ dữ liệu e.

$\Rightarrow$ Thuật toán này phải trả 2 giá: giá về không gian là $L_A(e)$, giá về thời gian là $T_A(e)$, e là bộ dữ liệu vào.

Ví dụ 2: Xét thuật toán A, "Tìm số lớn nhất trong một dãy số".

```
Begin
    Max := x1;
    For i := 2 to n do
    If max < xi then max := xi ;
End.
```

Thực hiện A trên hai bộ dữ liệu khác nhau:

+ Bộ dữ liệu $e_1 = \{0, 4, 9, 5, 7, 6\}$:

Khi đó $L_A(e_1) = 7$ (dữ liệu vào) + 2 (biến trung gian) = 9.

$T_A(e_1) = 8$ (thời gian để thực hiện tất cả các phép tính cơ bản).

+ Bộ dữ liệu $e_2 = \{3, 4, 6, 7, 9, 10, 12, 15\}$:

$L_A(e_2) = 11$.

$T_A(e_2) = 15$.

Khi đó ta có các khái niệm về chi phí phải trả trong các trường hợp như sau:

- Chi phí phải trả trong trường hợp xấu nhất:

  - Chi phí xấu nhất về bộ nhớ: $L_A(n) = \text{Max}\ \{L_A(e) \mid e \leq n\}$

  - Chi phí xấu nhất về thời gian: $T_A(n) = \text{Max}\ \{T_A(e) \mid e \leq n\}$

- Chi phí phải trả trung bình:

Là tổng số các chi phí khác nhau ứng với các bộ số liệu chia cho tổng số số bộ số liệu.

- Chi phí phải trả tiệm cận:

Đó là biểu thức biểu diễn tốc độ tăng của chi phí thực tế phải trả. Nó có gía trị tiệm cận với chi phí thực tế.

- Nhận xét: Ngày nay do sự phát triển không ngừng của khoa học công nghệ kỹ thuật điện tử nên chi phí về bộ nhớ không còn là vấn đề cần thiết phải bàn tới mà ta chỉ quan tâm tới chi phí phải trả về thời gian thực hiện giải thuật. Từ đây ta chỉ xét đến thời gian thực hiện giải thuật T(n), hay đó chính là độ phức tạp của thuật toán.

Sau đây là việc phân tích thời gian thực hiện giải thuật, một trong các tiêu chuẩn quan trọng để đánh giá hiệu lực của giải thuật vốn hay được đề cập tới.

1.4.2. Phân tích thời gian thực hiện giải thuật

Với một bài toán, không chỉ có một giải thuật. Chọn một giải thuật đưa tới kết quả nhanh là một đòi hỏi thực tế. Nhưng căn cứ vào đâu để có thể nói được giải thuật này nhanh hơn giải thuật kia?

Có thể thấy ngay: thời gian thực hiện một giải thuật, (hay chương trình thể hiện một giải thuật đó) phụ thuộc vào rất nhiều yếu tố. Một yếu tố cần chú ý trước tiên đó là *kích thước của dữ liệu đưa vào.* Chẳng hạn thời gian sắp xếp một dãy số phải chịu ảnh hưởng của số lượng các số thuộc dãy số đó. Nếu gọi n là số lượng này (kích thước của dữ liệu vào) thì thời gian thực hiện T của một giải thuật phải được biểu diễn như một hàm của n: T(n).

Các kiểu lệnh và tốc độ xử lý của máy tính, ngôn ngữ viết chương trình và chương trình dịch ngôn ngữ ấy đều ảnh hưởng tới thời gian thực hiện; nhưng những yếu tố này không đồng đều với mọi loại máy trên đó cài đặt giải thuật, vì vậy không thể đưa chúng vào khi xác lập T(n). Điều đó có nghĩa là T(n) không thể được biểu diễn thành đơn vị thời gian bằng giây, bằng phút được. Tuy nhiên, không phải vì thế mà không thể so sánh được các giải thuật về mặt tốc độ. Nếu như thời gian thực hiện của một giải thuật là $T_1(n) = c.n^2$ và thời gian thực hiện một giải thuật khác là

$T_2(n) = k.n$ (với c và k là một hằng số nào đó), thì khi n khá lớn, thời gian thực hiện giải thuật T_2 rõ ràng ít hơn so với thời gian thực hiện giải thuật T_1. Như vậy, nếu nói thời gian thực hiện giải thuật bằng T(n) tỉ lệ với n^2 hay tỉ lệ với n cũng cho ta ý niệm về tốc độ thực hiện giải thuật đó khi n khá lớn (với n nhỏ thì việc xét T(n) không có ý nghĩa). Cách đánh giá thời gian thực hiện giải thuật độc lập với máy tính và các yếu tố liên quan với máy như vậy sẽ dẫn tới khái niệm về cấp độ lớn của thời gian thực hiện giải thuật hay còn gọi là độ phức tạp tính toán của giải thuật.

1.4.3 Độ phức tạp của thuật toán

Nếu thời gian thực hiện một giải thuật là $T(n) = c.n^2$ (với c là hằng số) thì ta nói: Độ phức tạp tính toán của giải thuật này có cấp là n^2 (hay cấp độ lớn – tốc độ tăng – của thời gian thực hiện giải thuật là n^2) và ký hiệu là:

$T(n) = O(n^2)$ (kí hiệu chữ O lớn). Một cách tổng quát có thể định nghĩa:

Một hàm f(n) được xác định là O(g(n))

f(n) = O(g(n)) và được gọi là có cấp g(n) nếu tồn tại các hằng số c và n_0 sao cho $f(n) \leq c.g(n)$ khi $n \geq n_0$ nghĩa là f(n) bị chặn trên bởi một hằng số nhân với g(n), với mọi giá trị của n từ một điểm nào đó. Thông thường các hàm thể hiện độ phức tạp tính toán của giải thuật có dạng: $O(\log_2 n)$, O(n), $O(n\log_2 n)$, $O(n^2)$, $O(n^3)$, $O(2^n)$, O(n!), $O(n^n)$.

O(g(n)) còn gọi là độ phức tạp tiệm cận của hàm f(n).

Dưới đây là một số hàm số hay dùng để ký hiệu độ phức tạp tính toán và bảng giá trị của chúng để tiện theo dõi sự tăng của hàm theo đối số n.

$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n
0	1	0	1	1	2
1	2	2	4	8	4
2	4	8	16	64	16
3	8	24	64	512	256
4	16	64	256	4096	65536
5	32	160	1024	32768	2.147.483.648

Các hàm như 2^n, n^n được gọi là *hàm loại mũ*, ngoài ra còn có hàm n! và một số hàm khác có độ phức tạp lớn hơn các hàm mũ. Một giải thuật mà thời gian thực hiện của nó có cấp là các hàm loại mũ thì tốc độ rất chậm. Các như n^3, n^2, $n\log_2 n$, $\log_2 n$ được gọi là các *hàm loại đa thức*. Giải thuật với thời gian thực hiện có cấp hàm đa thức thì thường hiệu quả và chấp nhận được.

Ví dụ 3: Tính giá trị đa thức $P(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$ *với* $a_0, a_1, ..., a_n$, x nhập từ bàn phím.

Thuật toán 1:

```
1. Input n, a_o, a_1, a_2, ..., a_n, x;
2. S := a_o;
3. for i := 1 to n do
   begin
      p := 1;
      for j := 1 to i do  p := p*x;
      S:= S + a_i*p;
   end;
4. Output s;
```

Với mỗi giá trị i của vòng lặp 3, vòng lặp 3.2 thực hiện i vòng lặp nên khi n = i nó thực hiện đủ n vòng lặp. Vậy vòng lặp 3 thực hiện $\frac{n(n-1)}{2}$ lần câu lệnh sau do nên thời gian tính toán tỉ lệ thuận với n^2.

Vậy độ phức tạp tính toán của thuật toán trên là $O(n^2)$.

Thuật toán 2: Vì $x^n = x*x^{n-1}$ nên có thể tận dụng kết quả của lần tính trước cho lần tính sau:

```
1. Input n, a_o, a_1, a_2, ..., a_n, x;
2. S := a_o;  p:=1;
3. for i := 1 to n do
   begin
        p := p* x;
```

```
        S := s + p;
    end;
4.  Output S;
```

Hai lệnh 2 và 4 đều có độ phức tạp tính toán là O(1). Vòng lặp 3 cần thực hiện n lần hai thao tác tính s và p. Vậy số lần thực hiện lệnh 3 là 2n. Do vậy, độ phức tạp tính toán của thuật toán trên là O(n).

1.4.4. Các qui tắc xác định độ phức tạp tính toán của giải thuật

Xác định độ phức tạp tính toán của một giải thuật bất kì có thể dẫn tới những bài toán phức tạp. Tuy nhiên, trong thực tế, đối với một số giải thuật ta cũng có thể phân tích được bằng một số quy tắc đơn giản như [2], [4]:

- *Quy tắc tổng*

Giả sử $T_1(n)$ và $T_2(n)$ là thời gian thực hiện của 2 đoạn chương trình P_1 và P_2 mà $T_1(n) = O(f(n))$; $T_2(n) = O(g(n))$ thì thời gian thực hiện P_1 rồi P_2 tiếp theo sẽ là:

$T_1(n) + T_2(n) = O(\max(f(n), g(n)))$.

Ví dụ, trong một chương trình có 3 bước thực hiện mà thời gian thực hiện từng bước lần lượt là $O(n^2)$, $O(n^3)$ và $O(\log_2 n)$ thì thời gian thực hiện 2 bước đầu là $O(\max(n^2, n^3)) = O(n^3)$.

Một ứng dụng khác của quy tắc này là nếu $g(n) \leq f(n)$ với mọi $n \geq n_o$ thì $O(f(n) + g(n))$ cũng là $O(f(n))$. Chẳng hạn: $O(n^4+n^2) = O(n^4)$ và $O(n+\log_2 n) = O(n)$.

- *Quy tắc nhân*

Nếu tương ứng với P_1 và P_2 là $T_1(n) = O(f(n))$; $T_2(n) = O(g(n))$ thì thời gian thực hiện P_1 và P_2 lồng nhau sẽ là: $T_1(n).T_2(n) = O(f(n).g(n))$.

Ví dụ, câu lệnh gán: x := x+1 có thời gian thực hiện bằng c (hằng số) nên được đánh giá là O(1).

Câu lệnh: for i := 1 to n do x := x+1; có thời gian thực hiện O(n.1)=O(n).

Câu lệnh:

```
for i := 1 to n do
        for j := 1 to n do x := x+1;
```

có thời gian thực hiện được đánh giá là: $O(n.n) = O(n^2)$. Có thể suy ra $O(c.f(n)) = O(f(n))$ chẳng hạn $O(n^2/2) = O(n^2)$.