VUONG VAN THANG
Electronic Engineer - Ho Chi Minh City University of Technology

ĐỆ QUY - GIẢI THUẬT ĐỆ QUY

Điều đầu tiên: Để thực hiện được một bài toán đệ quy thì nó xuất phát từ một công thức toán học. vì vậy trước khi đi thực hiện một thuật toán Đệ Quy chúng ta phải tìm ra được một công thức cho thuật toán và điểm dừng cho nó.

Vậy làm sao để thực hiện được một công thức này??

Đơn giản là ta đi xét từng trường hợp cụ thể: ví dụ n = 0, n =1, n= 2, n =3...
từ đó nếu n> 3 thì ta có được công thức tính toán thông qua (n-1) hay (n-2)... như thế nào?

Chỉ cần suy luận cho ra được công thức toán => Bài toán về Đệ Quy được giải quyết.

Dưới đây ta sẽ đi vào một bài official hơn, cùng với đó là những bài toán điển hình cho thuật toán này:

Giới Thiệu

Đệ quy (Recursion) là một trong những giải thuật khá quen thuộc trong lập trình, mở rộng ra là trong toán học (thường được gọi với tên khác là “quy nạp”). Có một số bài toán, buộc phải sử dụng đệ quy mới giải quyết được, chẳng hạn như duyệt cây.

Tiền Đề Bài Viết

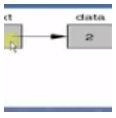
Trong những ngày đầu nghiên cứu về lập trình, tôi bắt gặp được khái niệm đệ quy và cảm thấy có hứng thú với nó. Sau một thời gian tìm hiểu, và kinh nghiệm bản thân, tôi xin chia sẻ cho các bạn bài viết này.

Đối Tượng Hướng Đến

Các lập trình viên muốn tìm hiểu về giải thuật đệ quy. Trong giới hạn bài viết, tôi sử dụng ngôn ngữ C/C++ làm ví dụ minh họa.

POPULAR POSTS

ĐỆ QUY - GIẢI THUẬT ĐỆ QUY
Điều đầu tiên: Để thực hiện được một bài toán đệ quy thì nó xuất phát từ một công thức toán học. vì vậy trước khi đi thực hiện một thuật toá...



Danh sách liên kết đơn
Danh sách liên kết đơn là gì? Tại sao phải dùng danh sách liên kết để lưu trữ dữ liệu? Kiến thứ Yêu cầu: Linked List = Pointer + Str...

QÚA TRÌNH BIÊN DỊCH MỘT CHƯƠNG TRÌNH C/C++

Đệ Quy Là Gì?

Một đối tượng được mô tả thông qua chính nó được gọi là mô tả đệ quy.
Một bài toán mang tính chất đệ quy khi nó có thể được phân rã thành các bài toán nhỏ hơn nhưng mang cùng tính chất với bài toán ban đầu, các bài toán nhỏ lại được phân rã thành các bài toán nhỏ hơn nữa.
Trong đời sống, ta cũng thường xuyên thấy một số hiện tượng đệ quy, ví dụ như hai chiếc gương đặt đối diện nhau, vòng xoắn ốc (trong vòng xoắn ốc có vòng xoắn ốc nhỏ hơn).

Hàm Đệ Quy

Trong lập trình, một hàm được gọi là đệ quy khi nó gọi chính nó trong thân hàm.
Ví dụ:

```
1. void Recusion()
2. {
3.     Recusion();
4. }
```

Hàm đệ quy không thể gọi tới nó mãi, cần phải có một điểm dừng (còn gọi là điểm neo) tại một trường hợp đặc biệt, gọi là trường hợp suy biến (degenerate case).
Chúng ta hiểu rằng khi một hàm được gọi, nó sẽ được đưa vào **Stack**, hàm đệ quy cũng vậy, mỗi lần gọi chính nó thì nó lại được đưa vào Stack, nếu như không có điểm dừng, hoặc gọi mãi mà chưa tới điểm dừng, sẽ dễ xảy ra tình trạng tràn bộ nhớ Stack.

THÀNH PHẦN CỦA MỘT HÀM ĐỆ QUY

Hàm đệ quy gồm 2 phần:

- Phần cơ sở: Điều kiện thoát khỏi đệ quy
- Phần đệ quy: Thân hàm có chứa lời gọi đệ quy

THIẾT KẾ GIẢI THUẬT ĐỆ QUY

Thực hiện 3 bước sau:

- Tham số hóa bài toán
- Phân tích trường hợp chung: Đưa bài toán về bài toán nhỏ hơn cùng loại, dần dần tiến tới t
  biến
- Tìm trường hợp suy biến

ƯU VÀ NHƯỢC ĐIỂM

Giải thuật đệ quy có ưu điểm là thuận lợi cho việc biểu diễn bài toán, đồng thời làm gọn chương trình. Tuy nhiên cũng có nhược điểm, đó là không tối ưu về mặt thời gian (so với sử dụng vòng lặp), gây tốn bộ nhớ.

Một Số Loại Đệ Quy

ĐỆ QUY TUYẾN TÍNH (LINEAR RECURSION)

Mỗi lần thực thi chỉ gọi đệ quy một lần
Một ví dụ rất quen thuộc là hàm tính giai thừa:

```
1. int Factorial(int n)
2. {
3.     if (n == 0)
4.     {
5.         return 1;
6.     }
7.     else
8.     {
9.         return n * Factorial(n - 1); // Linear Recursion
10.    }
11. }
```

ĐỆ QUY NHỊ PHÂN (BINARY RECURSION)

Mỗi lần thực thi có thể gọi đệ quy 2 lần
Ví dụ: Tính tổ hợp chập K của N bằng đệ quy:

```
1. int Combine(int n, int k)
2. {
3.     if (k == 0 || k == n)
4.     {
5.         return 1;
6.     }
7.     else
8.     {
9.         return (Combine(n - 1, k) + Combine(n - 1, k - 1)); // Bi
   nary Recursion
10.    }
11. }
```

ĐỆ QUY LỒNG (NESTED RECURSION)

Tham số trong lời gọi đệ quy là một lời gọi đệ quy. Đệ quy lồng chiếm bộ nhớ rất nhanh.
Ví dụ: Hàm Ackerman

```
1. int Ackerman(int m, int n)
2. {
3.     if (m == 0)
4.     {
5.         return (n + 1);
6.     }
7.     else
8.     {
9.         if (n == 0)
10.        {
11.            return Ackerman(m - 1, 1);
12.        }
13.        else
14.        {
15.            return Ackerman(m - 1, Ackerman(m, n - 1)); // Nested
    Recursion
16.        }
17.    }
18. }
```

BÀI TOÁN THÁP HÀ NỘI - TOWER OF HANOI

Có thể còn ít người Việt Nam biết Tháp Hà Nội, nhưng rất nhiều thanh niên sinh viên trên toàn thế giới lại biết đến nó. Đó là vì, Tháp Hà Nội là tên một bài toán rất nổi tiếng trong Chương trình khoa học tính toán (Computing Science) dành cho sinh viên những năm đầu tại các trường đại học ở nhiều nơi trên thế giới.

Tương truyền rằng ngày xửa ngày xưa, lâu lắm rồi, ở một vùng xa xôi viễn đông, thành phố Hà Nội của Việt Nam, vị quân sư của Hoàng đế vừa qua đời, Hoàng đế cần một vị quân sư mới thay thế. Bản thân Hoàng đế cũng là một nhà thông thái, nên ngài đặt ra một bài toán đố, tuyên bố ai giải được sẽ được phong làm quân sư. Bài toán của Hoàng đế là: cho n cái đĩa (ngài không nói chính xác là bao nhiêu) và ba cái trục: A là trục nguồn, B là trục đích, và C là trục trung chuyển. Những cái đĩa có kích cỡ khác nhau và có lỗ ở giữa để có thể lồng vào trục, theo quy định "nhỏ trên lớn dưới". Đầu tiên, những cái đĩa này được xếp tại trục A. Vậy làm thế nào để chuyển toàn bộ các đĩa sang trục B, với điều kiện chuyển từng cái một và luôn phải đảm bảo quy định "nhỏ trên lớn dưới", biết rằng trục C được phép sử dụng làm trục trung chuyển?

Vì địa vị quân sư được coi là vinh hiển nên có rất nhiều người dự thi. Từ vị học giả đến bác

nông phu, họ đua nhau trình lên Hoàng đế lời giải của mình. Nhiều lời giải dài tới hàng nghìn bước, và nhiều lời giải có chữ "chuyển sang bước tiếp theo" (go to). Nhưng hoàng đế thấy mệt mỏi vì những lời giải đó, nên cuối cùng hạ chiếu: "Ta không hiểu những lời giải này. Phải có một cách giải nào khác dễ hiểu và nhanh chóng hơn". May mắn thay, cuối cùng đã có một cách giải như thế.

Thật vậy, ngay sau khi chiếu vua ban ra, một vị cao tăng trông bề ngoài giống như một kỳ nhân hạ sơn tới xin yết kiến hoàng đế. Vị cao tăng nói: "Thưa Bệ hạ, bài toán đố đó dễ quá, hầu như nó tự giải cho nó". Quan trùm cấm vệ đứng hầu ngay bên cạnh vua quắc mắt nhìn gã kỳ nhân, muốn quẳng gã ra ngoài, nhưng Hoàng đế vẫn kiên nhẫn tiếp tục lắng nghe. "Nếu chỉ có 1 đĩa, thì...; nếu có nhiều hơn 1 đĩa (n>1), thì...", cứ thế vị cao tăng bình tĩnh giảng giải. Im lặng được một lát, cuối cùng Hoàng đế sốt ruột gắt: "Được, thế cao tăng có nói rõ cho ta lời giải hay không cơ chứ?". Thay vì giải thích tiếp, gã kỳ nhân mỉm cười thâm thúy rồi biến mất, bởi vì hoàng đế tuy giỏi giang nhưng rõ ràng là chưa hiểu ý nghĩa của phép truy hồi (recursion). Nhưng các bạn sinh viên ngày nay thì có thể thấy cách giải của vị cao tăng là hoàn toàn đúng.

Toàn bộ đoạn chữ nghiêng ở trên được trích nguyên văn từ cuốn sách giáo khoa dành cho sinh viên ngành thuật toán và lập trình - "giải toán nâng cao và cấu trúc dữ liệu" (intermediate problem solving and data structures) do Paul Henman và Robert Veroff, hai giáo sư Đại học New Mexico, cùng biên soạn với Frank Carrano, giáo sư Đại học Rhode Island (Mỹ).

Bạn nào chưa từng biết Tháp Hà Nội thì cũng nên "thử sức" một chút xem sao, vì đây là một trò chơi rất thú vị. Bạn có thể bắt đầu bằng bài toán 3 đĩa, rồi nâng lên 4 đĩa. Với 4 đĩa chắc bạn bắt đầu thấy rắc rối. Nâng tiếp lên 5 và cao hơn nữa, chẳng hạn n = 1 triệu, bài toán sẽ rắc rối đến mức không ai đủ kiên trì và đủ thì giờ để thử từng đĩa một. Vậy mà vị cao tăng dám nói là dễ quá! Xin tiết lộ, ấy là vì vị đó đã sử dụng phép truy hồi - một quy tắc toán học cho phép xác định số hạng thứ n từ số hạng đứng trước nó, tức số hạng thứ n-1. Cái giỏi của vị cao tăng là ông tìm ra một quy tắc chung, tức một thuật toán chung cho tất cả các bước chuyển đĩa.

Vậy thay vì mô tả toàn bộ quá trình chuyển đĩa từng cái một như những thí sinh trước đó đã làm, vị cao tăng chỉ mô tả một quy tắc chung. Cứ làm theo quy tắc đó, lặp đi lặp lại chẳng cần suy nghĩ gì, rồi cuối cùng tự nhiên sẽ tới đích. Vì thế vị cao tăng nói rằng bài toán này "tự nó giải nó".

Trong khoa học tính toán ngày nay, phép truy hồi là thuật toán cơ bản để lập trình. Ưu điểm của phương pháp truy hồi là ở chỗ nó dùng một công thức nhất định để diễn tả những phép tính lặp đi lặp lại bất chấp số lần lặp lại là bao nhiêu. Nếu số lần lặp lại lên đến con số hàng triệu hàng tỷ thì con người không đủ sức và thời gian để làm, nhưng máy tính thì có thể giải quyết trong chớp mắt. Điểm mạnh của computer là ở chỗ nó không hề biết e ngại và mệt mỏi trước những công việc lặp đi lặp lại lên đến hàng triệu hàng tỷ lần. Và vì thế, việc cộng tác giữa computer với con người là mô hình lý tưởng của lao động trí óc trong cuộc sống hiện đại.

Về mặt lịch sử, Tháp Hà Nội được E. Lucas phát hiện từ năm 1883, nhưng mãi đến gần đây người ta mới nhận ra ý nghĩa hiện đại của nó. Hiện vẫn chưa rõ vì sao Lucas lại gọi chồng đĩa trong bài toán là Tháp Hà Nội, mà không gọi là Tháp Bắc Kinh, hay Tháp Tokyo.

Tháp Hà Nội đã mở tung cánh cửa cho tương lai khi nhiều nghiên cứu lấy Tháp Hà Nội làm điểm xuất phát đã đạt được thành tựu mới:

(1) Nâng câu hỏi của Tháp Hà Nội lên một mức cao hơn, sao cho số lần chuyển đĩa là nhỏ nhất. Các nhà toán học đã phát hiện ra rằng Tháp Hà Nội có cùng bản chất với bài toán tìm Đường Hamilton (Hamilton Path) trên một hình giả phương cấp n (n-Hypercube), một bài toán cũng rất nổi tiếng.

(2) Nhà toán học D.G. Poole đã sáng tạo ra Lược Đồ Hà Nội - một tam giác có các đỉnh tương ứng với các cách sắp xếp đĩa trong Tháp Hà Nội, từ đó tìm ra những liên hệ lý thú giữa Tam giác Pascal với Lược đồ Hà Nội. Liên hệ này đã được công bố trong một công trình mang một cái tên đầy liên tưởng: Pascal biết Hà Nội (Pascal knows Hanoi).

Hiện nay, tại một số đại học ở Australia, uy tín của sinh viên Việt Nam trong lĩnh vực lập trình được đánh giá ngang với sinh viên Ấn Độ - một cường quốc lập trình của thế giới, làm cho Tháp Hà Nội vốn đã nổi tiếng lại càng nổi tiếng hơn.

Cách giải Bài toán:

Bài toán có 2 chiều, chiều nghịch dùng để phân tích bài toán và dùng để lập trình, chiều thuận để thực hiện trên mô hình. Ở đây nguyên lí thực hiện (tức là cách hiểu bài toán) nằm ở chiều nghịch, còn cách thực hiện nằm ở chiều thuận. Chiều nghịch dùng cho dân lập trình và chiều thuận dành cho dân toán.
Chiều nghịch thì dân lập trình hay gọi là bài toán đệ qui. Bài toán này có ví dụ đơn giản như sau: giaithừa(n)=giaithừa(n-1)*n. Muốn tính giai thừa của n thì đơn giản, lấy n nhân với giai thừa của (n-1). Còn giai thừa của (n-1) thì tính sao? Đơn giản, cứ lấy (n-1) nhân với giai thừa của (n-2) …

- **Chiều nghịch (nguyên lí):** Muốn đưa n khối tròn từ cột A (cột nguồn) sang cột C (cột đích) thông qua cột B (cột trung gian) thì chỉ cần đưa (n-1) khối tròn từ A qua B, rồi đưa 1 khối tròn từ A qua C và cuối cùng là đưa (n-1) khối tròn từ B qua C. Đã làm xong bài toán!!!
Còn (n-1) khối tròn từ A sang B thì làm sao mà đưa? Đơn giản, khi đó xem A là cột nguồn, B là cột đích và C là cột trung gian. Việc tiến hành tương tự, đưa (n-2) khối từ cột nguồn qua cột trung gian, 1 khối từ cột nguồn sang cột đích và cuối cùng là (n-2) khối từ cột trung gian sang cột đích. Còn về source code thì sao, xin xem phần sau cùng vì nếu viết ở đây sẽ có thể gây rối một vài bạn đọc không được học về lập trình.

- Chiều thuận: Nói đơn giản để hiệu thì như chiều nghịch đã nói. Còn thực hiện (chiều thuận) thì làm sao? Khối tròn đầu tiên từ cột nguồn A thì chuyển vào đâu? Cột trung gian B hay cột đích C?
Với khối tròn đầu tiên thì chuyển về cột đích (với n là số lẻ) và cột trung gian (với n là số chẵn). Xếp được 1 khối tròn (khối nhỏ nhất) theo đúng yêu cầu thì tiếp theo là xếp 2 khối tròn theo đúng yêu cầu (2 khối nhỏ nhất và nhỏ nhì). Cứ xếp 2 khối đó vào cột còn lại so với lần đầu tiên và làm tiếp vớ 3, 4, … khối tròn. Điều vừa nói được diễn đạt như sau:
Với n lẻ: cách xếp các khối như sau. Đầu tiên là xếp được 1 khối nhỏ nhất (bài toán với n=1). Sau đó xếp 2 khối nhỏ nhất (bài toán với n=2) ….

- 1 khối nhỏ nhất qua C (cột đích)
- 2 khối nhỏ nhất qua B (cột trung gian)
- 3 khối nhỏ nhất qua C (cột đích)
- 4 khối nhỏ nhất qua B (cột trung gian)

• Tiếp tục như trên

Với n chẵn: cách xếp các khối như sau. Đầu tiên là xếp được 1 khối nhỏ nhất (bài toán với n=1). Sau đó xếp 2 khối nhỏ nhất (bài toán với n=2) ….

• 1 khối nhỏ nhất qua B (cột trung gian)

• 2 khối nhỏ nhất qua C (cột đích)

• 3 khối nhỏ nhất qua B (cột trung gian)

• 4 khối nhỏ nhất qua C (cột đích)

• Tiếp tục như trên

Phát triển của cách làm trên: Muốn chuyển n khối tròn từ cột nguồn sang cột đích thì làm như sau (ở đây ta phải hiểu rõ cột nguồn không nhất thiết là cột A, cột đích là C hay cột trung gian là B mà phải hiểu tổng quát, có thể cột nguồn là B chẳng hạn (trong phép chuyển (n-1) cột từ cột B sang cột C như trong cách làm ở phần nghịch)):

Với n lẻ: cách xếp các khối như sau. Đầu tiên là xếp được 1 khối nhỏ nhất (bài toán với n=1). Sau đó xếp 2 khối nhỏ nhất (bài toán với n=2) ….

• 1 khối nhỏ nhất qua cột đích

• 2 khối nhỏ nhất qua cột trung gian

• 3 khối nhỏ nhất qua cột đích

• 4 khối nhỏ nhất qua cột trung gian

• Tiếp tục như trên

Với n chẵn: cách xếp các khối như sau. Đầu tiên là xếp được 1 khối nhỏ nhất (bài toán với n=1). Sau đó xếp 2 khối nhỏ nhất (bài toán với n=2) ….

• 1 khối nhỏ nhất qua cột trung gian

• 2 khối nhỏ nhất qua cột đích

• 3 khối nhỏ nhất qua cột trung gian

• 4 khối nhỏ nhất qua cột đích

• Tiếp tục như trên

- Như vậy thì số lần chuyển cho bài toán là bao nhiêu? Với bài toán tháp Hà Nội chuyển n khối tròn từ cột nguồn A sang cột đích C thông qua cột trung gian B thì cần có 2^0 + 2^1 + 2^2 + … + 2^n lần chuyển

Sưu tầm.

Đây là hình ảnh của cách chuyển với 3 và 4 đĩa. DR đã lấy nó trên internet, rồi đem chỉnh sửa, tăng thêm các điểm dừng để các bạn dễ hình dung cách giải hơn. OK

Với 4 đĩa:

Với 3 đĩa:

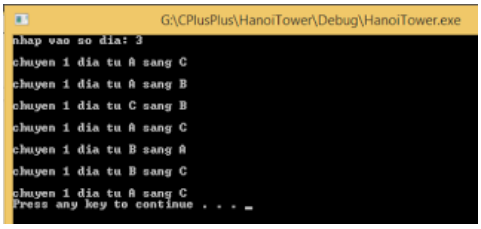
Source code:

C++ Code:

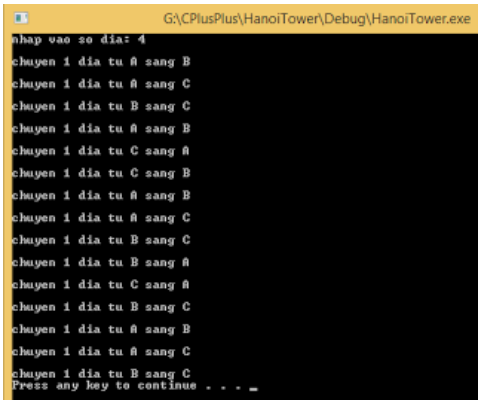
```
#include <conio.h>
#include <iostream>
#include <process.h>
using namespace std;
void hn(int n, char a, char b, char c)
{
 if (n == 1)
 {
  cout << "\nchuyen 1 dia tu " << a << " sang " << c;
  cout << "\n";
 }
 else
 {
  hn(n - 1, a, c, b);
  hn(1, a, b, c);
  hn(n - 1, b, a, c);
 }
}
int main()
{
 int n;
 //clrscr();
 cout << "nhap vao so dia:";
 cin >> n;
 hn(n, 'A', 'B', 'C');
 system("pause");
 return 0;
}
```

Kết quả như sau:

Chạy với n lẻ cho n = 3:



Chạy với n chẵn n = 4:



lewer Post Home Older Post

0 COMMENTS :

POST A COMMENT

Enter your comment...

Comment as: Tùng Phạm (Gc ▾ Sign out

Publish Preview Notify me

SIGN UP FOR OUR NEWSLETTER Enter your email address... SUBSCRIBE!

BLOGGER NEWS BLOGROLL BLOGROLL ABOUT