**TRƯỜNG ĐẠI HỌC QUỐC GIA HÀ NỘI**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA : TOÁN – CƠ – TIN HỌC**

**--------------- 🙟🕮🙝 ---------------**



**TIỂU LUẬN GIỮA KỲ**

***Bộ Môn*: Thiết Kế Và Đánh Giá Thuật Toán**

***Đề Tài :***  Phương Pháp Đệ Quy

|  |  |
| --- | --- |
| Sinh viên thực hiên : | Nguyễn Trọng Anh  Bùi Thị Cúc  Lộc Thị Thúy Linh  Bùi Đắc Tiến |
| Lớp : | K54A2 Toán Tin |

*Hà Nội : 3 / 2012*

1. **Phần mở đầu**

Giới thiệu nội dung các phần chính:

* Giải thích được phương pháp đệ quy là gì.
* Biết cách nhận dạng được bài toán đệ quy.
* Biết cách hiện thực hàm đệ quy.
* Giải thích được cách chạy một hàm đệ quy.
* Đánh giá thời gian thực hiện của phương pháp đệ quy.
* Phân loại được các dạng đệ quy.
* Xây dựng chương trình demo cho phương pháp đệ quy.
* Ưu nhược điểm của đệ quy
* Ứng dụng đệ quy trong hình học Fractal.

1. **Phần nội dung chi tiết.**
2. *Khái niệm*

* Khái niệm được định nghĩa là đệ quy khi khái niệm đó được định nghĩa hay xác định bởi chính nó.
* Giải thuật đệ quy là phương pháp giải bài toán bằng cách rút gọn bài toán thành một (hoặc một số) bài toán con tương tự như vậy dưng dữ liệu cần xử lí nhỏ hơn với trạng thái dừng tồn tại.
* Trong thực tế việc nhìn nhận các bài toán, giải thuật, các sự kiện, sự vật, quá trình… dưới góc độ của đệ quy sẽ cho ta một thuật giải đơn giản và hiệu quả hơn.
* Việc phân tích đối tượng thành nhiều thành phần mà trong số các thành phần có chứa thành phần của đối tượng được mô tả được gọi là mô tả mang tính đệ quy hay mô tả chính đối tượng qua chính nó.
* Ví dụ:
  + Búp bê Nga hay Búp bê babushka(Búp bê lồng nhau, Búp bê làm tổ, ...) là một loại búp bê đặc trưng của Nga. Nó là một bộ gồm những búp bê rỗng ruột có kích thước từ lớn đến nhỏ. Con búp bê nhỏ nhất sẽ được chứa đựng trong lòng con búp bê lớn hơn nó một chút, đến lượt mình con búp bê lớn được chứa trong một con búp bê khác lớn hơn, và cứ thế cho đến con lớn nhất sẽ chứa tất cả những con búp bê còn lại trong bộ.
  + Mô tả đệ quy tập số tự nhiên
    - Số 0 là số tự nhiên (1 )
    - Số tự nhiên bằng số tự nhiên cộng 1

n => (n+1)

1. *Nhận dạng bài toán đệ quy*
   1. Đặc trưng của các bài toán có thể giải bằng đệ quy (trích dẫn bài giảng của cô)

* Các bài toán phụ thuộc tham số
* Ứng với giá trị đặc biệt nào đó của tham số thì bài toán có giải thuật để giải (trường hợp suy biến)
* Trong trường hợp tổng quát bài toán có thể qui về dạng tương tự với một bộ giá trị mới của tham số và sau một số hữu hạn lần thì có thể dẫn tới trường hợp suy biến.
  1. Cấu trúc bài toán đệ quy

Mỗi thuật toán đệ quy gồm 2 phần chính: phần neo (phần cơ sở) và phần đệ quy (phần tổng quát).

* Phần neo: là phần có nghiệm tầm thường của bài toán.
* Phần đệ quy: thuật toán giải bài toán được giải thông qua chính nó.

Ví dụ: tính n!

* Phần neo: 0! = 1
* Phần đệ quy: n! = n(n-1)!

* 1. Các loại đệ quy

Thông thường đệ quy được phân thành 2 loại: Đệ quy trực tiếp và đệ quy gián tiếp.

* + - Đệ quy trực tiếp là loại đệ quy mà đối tượng được mô tả trực tiếp qua nó
    - Đệ quy gián tiếp là loại đệ quy mà đối tượng được mô tả gián tiếp qua nó

1. *Phương pháp tìm giải thuật đệ quy và lược đồ phương án*
   1. Phương pháp tìm giải thuật đệ quy (trích dẫn bài giảng của cô)

Để xây dựng giải thuật một bài toán có tính đệ quy bằng phương pháp đệ quy ta cần thực hiện tuần tự 3 nội dung sau:

* Thông số bài toán.
* Tìm các trường hợp neo (suy biến) cùng giải thuật tương ứng: đây là các trường hợp suy biến điều khiển (trường hợp kích thước bài toán nhỏ nhất), mà giải thuật không đệ quy.
* Tìm giải thuật trong trường hợp tổng quát bằng phân rã bài toán theo kiểu đệ quy hoặc là bài toán trên nhưng có kích thước nhỏ hơn

Ví du: bài toán tính n!

* Phần neo GiaiThua(0) = 1
* Phân rã bài toán: Giaithua = n\* GiaiThua(n-1).
  1. Lược đồ phương án
     1. Lược đồ

Recursive\_Algorithm(pn) ≡

**if** (pn = p0) *//TH suy biến*

<*Thực hiện giải thuật trường hợp suy biến>;*

**else** //*TH tổng quát*

<Lệnh;>

Recursive\_Algorithm(pn-1);

<Lệnh;>

**endif**

**End.**

pn **←** pn-1 **←** pn-2 **←**  … **←**  po

* + 1. Ví dụ

Tính n!

**GiaiThua(n):int ≡**

**if** (n= =1)

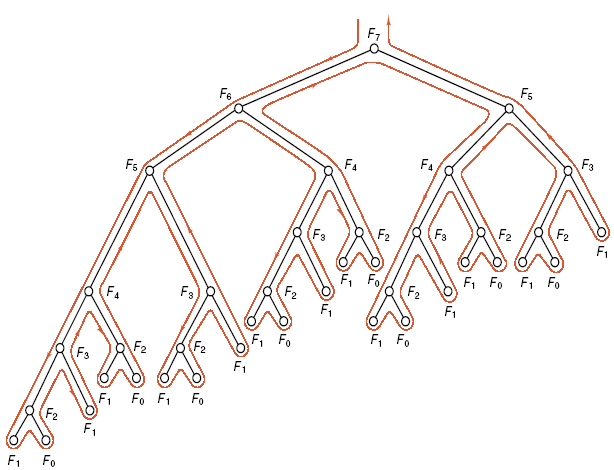
GiaiThua = 1;

**else**

// gọi lại hàm giai thừa với tham số n-1

GiaiThua = n \* GiaiThua(n-1)

**End.**

**Tính Fibonaci(n)** 

Fibonaci(n):int ≡

**if** (n==1|| n = = 2)

Fibonaci = 1;

**else**

// số Fibonaci hiện tại được tính dựa trên 2 số trước nó

Fibonaci = Fibonaci(n-1) + Fibonaci(n-2);

**End.**

**Tháp Hà Nội**

ThapHaNoi(n,xp,tg,dich) ≡

**if** (n==1)

Chuyen(A,B);

**else{**

ThapHaNoi(n-1,xp,dich,tg);

Chuyen(A,B);

ThapHaNoi(n-1,dich,tg,xp);

}

**End.**

Chuyen(xp,dich) ≡

print(xp,”→”,dich)

**End.**

1. *Đánh giá độ phức tạp của phương pháp đệ quy*
   1. Độ phức tạp

Thời gian tính toán của một giải thuật thường là một hàm của kích thước dữ liệu nhập.

Chúng ta quan tâm đến:

* *Trường hợp trung bình* (*average case*): thời gian tính toán mà một giải thuật cần đối với một “dữ liệu nhâp thông thường” (typical input data).
* *Trường hợp xấu nhất* (*worst case*): thời gian tính toán mà một giải thuật cần đối với một “dữ liệu nhâp xấu nhất”

//trich dẫn bài giảng của cô

* + - * Xác định quan hệ truy hồi trong phép đệ qui
        + Gọi *T*(*n*) là độ phức tạp của giải thuật đệ qui với kích thước bài toán *n*
        + c = const là độ phức tạp thuật toán trong trường hợp suy biến khi *n*=*n*0

*T*(*n0*) = *c*

* + - * Với *f*(*n*) là hàm biến đổi tham số *n,* nếu giải thuật được thực hiện *a* lần bài toán con với tham số *f*(*n*)

*T*(*n*) = *a*.*T*(*f*(*n*)) + *g*(*n*) với *g*(*n*) là độ phức tạp các thao tác ngoài lời gọi đệ qui

* + - * Công thức truy hồi xác định độ phức tạp giải thuật đệ qui

trường hợp ngược lại

Ví dụ:

* Tính n! có thời gian thực hiện
  + *T(1) = 1*
  + *T(n) = T (n-1) +1 n>1*

*= T (n-1) + 1 + 1*

*= T (n-3) + 1 + 1 + 1*

*= …*

*= T (n – (n-1)) + 1 + ...+1*

*= 1 + 1 + …+ 1 = n*

* T(n) = O (n)
* Tính số Fibonaci số n >2
  + *T(1) = 1, T(2) = 1*
  + *T(n) =*

*= (T (n – 2) + T (n – 3))*

*+ (T (n – 3) + T (n – 4))*

*= T(n – 2) + 2T(n –3) + T(n -4)*

*= …*

*Người ta c/m được số Fibonaci thứ n là:*

*F(n) =*

* *T(n) = O*
* Tháp Hà Nội
  + T(1) = 1
  + T(n) = 2T(n – 1) + 1 khi n>1

=

=

= …

=

* T(n) =

1. *Các dạng đệ quy đơn giản thường gặp* 
   1. Đệ quy tuyến tính (dạng ED[AP] – trích dẫn bài giảng của cô)

P(x)

**if** (E(x))

D(x);

**else**

A(x);

P(f(x));

**endif**

**End.**

E(x): điều kiện suy biến

D(x): lời giải trong trường hợp suy biến

A(x): thao tác trước lời gọi đệ qui

f(x): hàm biến đổi tham số lời gọi đệ qui

Ví dụ: Hàm tính n!

* GiaiThua(n)

**if** ( n=0)

GiaiThua = 1;

**else**

GiaiThua = n\*GiaiThua(n – 1);

**endif**

**End**

* 1. Đệ quy nhi phân

P(x) ≡

**if** (E(x))

D(x);

**else**

A(x);

P(f(x));

P(f(x));

**endif**

**End.**

E(x): điều kiện suy biến

D(x): lời giải trong trường hợp suy biến

A(x) thao tác ngoài lời gọi đệ qui

f(x): hàm biến đổi tham số lời gọi đệ qui

Ví dụ: tính số Fibonaci thứ n

Fibonaci ≡

**if** (n<2)

F = 1;

**else**

F = Fibonaci (n-1) + Fibonaci (n-2);

**End**

**End**

* 1. Đệ quy phi tuyến tính

P(x)

**for** giá trị đầu **to** giá trị cuối **do**

D(x);

**If** (E(x))

A(x);

**Else**

P(f(x));

**endif**

**endfor**

**End.**

E(x): điều kiện suy biến.

D(x): lời giải trong trường hợp suy biến

A(x): thao tác trước lời gọi đệ qui

f(x): hàm biến đổi tham số lời gọi đệ qui

**Ví dụ:** Cho dãy {Xn} xác định công thức truy hồi:

X0  = 1;

Xn = X0  + X1 + X2  +…. + Xn-1

X(n)

if (n = = 0)

X = 1;

else

tg = 0;

for ( i = 0 to n-1)

tg =tg + X[i];

X = tg;

Endif

End

1. *Ưu, nhược điểm của đệ quy*
   1. Ưu điểm

* Một chương trình viết theo thuật giải đệ quy sẽ: sáng sủa, dễ hiểu, nêu bật được bản chất vấn đề.
* Bản chất của đệ quy là ý tưởng quy nạp hay hạ bậc trong toán học nghĩa là đưa một số vấn đề phức tạp về một số vấn đề đơn giản hơn.
* Cần lưu ý rằng đệ quy không những là phương pháp lập trình quan trọng mà còn là phương pháp suy nghĩ để giải quyết vấn đề một cách tổng quát dựa trên ý tưởng:
  + - Đơn giản hóa công việc
    - Phân vùng để xử lí
  + Lập trình đệ quy là một phương tiện hiệu quả để tổ chức một thao tác tìm kiếm phức tạp trong một tập hợp các khả năng có thể có.
  1. Nhược điểm
* Tuy nhiên khi thực hiện chương trình, chương trình dịch cần thu xếp một vùng nhớ gọi là stack để lưu giữ các giá trị trung gian trong quá trình tính toán, nếu số lượng các trí trị đó quá nhiều, có thể tràn stack.
* Việc sử dụng đệ quy cho các bài toán có khối lượng tính toán lớn sẽ dẫn đến tốn thời gian.
* Lời gọi hàm phức tạp, đôi khi khó hình dung

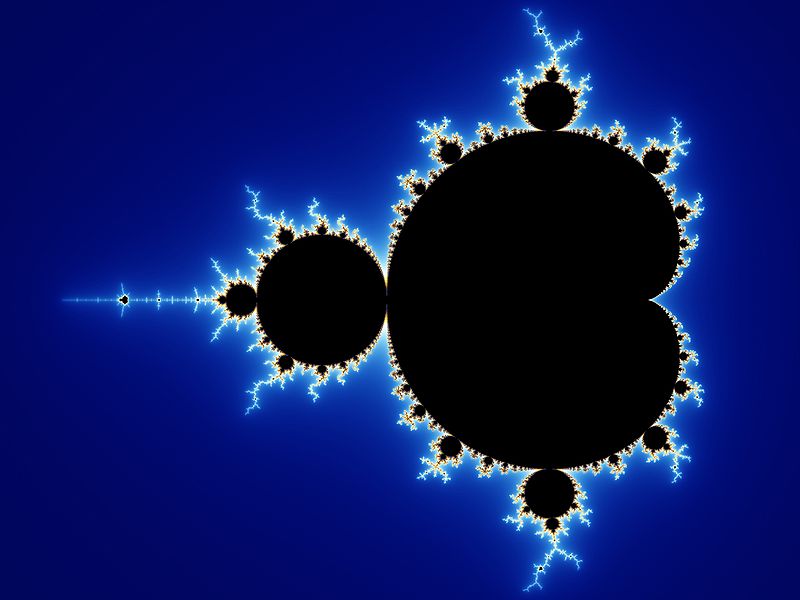
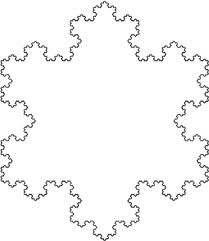
1. *Ứng dụng đệ quy trong Fractal*

*7.1: Giới thiệu về hình học Fractal*

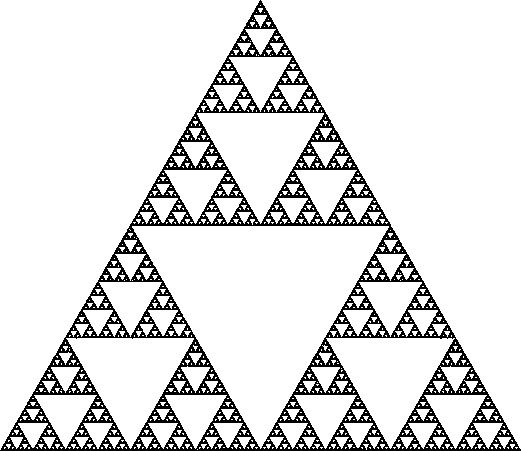
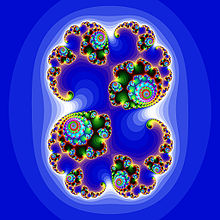
Nhiều đường cong và hình ảnh có một thuộc tính là tính tự tương tự. Nói một cách trực quan chúng giống nhau ở mỗi tỷ lệ: cho dù có phóng to hình ảnh của đường cong đến cỡ nào, thì nó vẫn có cùng mức độ chi tiết. Một số đường cong có tính tự tương tự chính xác, tức là nếu một vùng được phóng to, thì hình ảnh sau khi phóng to giống hệt như hình ảnh ban đầu (giả sử hình ảnh không quay hoặc tịnh tiến). Trong khi đó một số đường cong khác chỉ có tính tự tương tự thống kê, tức là mức độ bất quy tắc và lượn sóng của đường cong chỉ giống nhau một cách “trung bình” cho dù hình ảnh được phóng to như thế nào(Ví dụ: đường bờ biển, cành cây, bề mặt bọt biển, hệ thống tĩnh, động mạch trong cơ thể, đám mây………).

*Benoit Mandelbrot gọi những hình thức khác nhau của sự tự tương tự là fractal.*

Ví dụ về 1 số hình ảnh Fractal:

*Tập Mandelbrot Bông Tuyết Von Koch*

*Đệm Sierpinsky Tập Julia*

7.2: Ứng dụng đệ quy trong hình học Fractal

Đệ quy thường làm cho việc vẽ hình ảnh hình học phức tạp trở nên hết sức đơn giản. Ngoài ra nó cho phép chúng ta phân giải hoặc tinh chỉnh đối tượng ở những mức độ chi tiết khác nhau. Thuật toán đệ quy tạo nên những hình ảnh hấp dẫn, có ứng dụng hữu ích trong khoa học, kỹ thuật.

*Ví dụ: Vẽ bông tuyết Von Koch*

Bông tuyết Koch bậc 0 (K0) chỉ là một đoạn thẳng nằm ngang có chiều dài đơn vị. Chúng ta tạo K1 bằng cách chia K thành 3 phần bằng nhau, và thay thế phần giữa bằng một tam giác với cạnh có chiều dài là 1/3

*K*1

*K*2

Bông tuyết Koch bậc k được tạo ra bởi các bông tuyết Koch bậc k-1. Chúng ta có cách làm đệ quy khá tự nhiên để vẽ bông tuyết Koch bậc k bất kỳ như đoạn mã giả định sau:

Vẽ Kn:

If(n=0) vẽ một đoạn thẳng;

Else

{

Vẽ Kn-1

Quay trái 600

Vẽ Kn-1

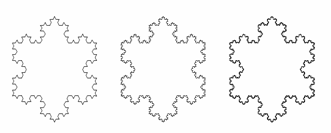
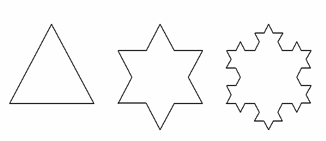
Quay phải 1200

Vẽ Kn-1

Quay trái 600

Vẽ Kn-1

}



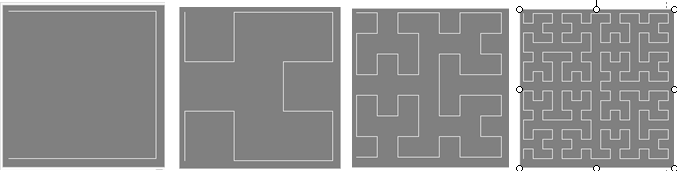
*Ví Dụ 2: Sinh chuỗi và tạo đường cong Peano*

Dùng chuỗi để điều khiển “con rùa”

'F' có nghĩa là forward(L, l) (đi theo hướng hiện hành một khoảng L, có vẽ đoạn thẳng)

'+' có nghĩa là turn(A) (quay phải một góc *A* độ)

'-' có nghĩa là turn(-A) (quay trái một góc *A* độ)



* Cho phép rẽ nhánh:

'[' : saveTurtle()🡪 lưu trạng thái hiện tại của con rùa

']' : restoreTurtle()🡪 khôi phục trạng thái của con rùa vế trạng thái trước đó.

* Trạng thái của con rùa = {CP, CD}
* thêm các lệnh sau vào produceString():

'[': saveTurtle();break; 🡨 đẩy trạng thái hiện hành của con rùa vào ngăn xếp.

']': restoreTurtle();break; 🡨 lấy trạng thái hiện hành từ đỉnh ngăn xếp

1. Khử đệ quy

* Là quá trình chuyển đổi một giải thuật đệ quy thành giải thuật không đệ quy.
* Chưa có giải pháp cho việc chuyển đổi này một cách tổng quát.
* Cách tiếp cận:
* Dùng quan điểm đệ quy để tìm thuật giải cho bài toán.
* Mã hóa giải thuật đệ quy.
* Khử đệ quy để có giải thuật không-đệ-quy.
* 8.1 Khử đệ quy bằng vòng lặp
  + Ý tưởng: Lưu lại các trị của các lần tính toán trước làm dữ liệu cho việc tính toán của lần sau.
  + Đi từ điều kiện biên đi tới điều kiện kết thúc.
* 8.2 Khử đệ quy bằng stack
  + Khởi tạo stack với số phần tử phù hợp.
  + Đưa bộ tham số đầu vào stack.
  + Khi stack không trống
    - Lấy bộ tham số ra khỏi stack;
    - Xử lý các tác vụ cơ bản ứng với tham số này.Nếu gặp một tác vụ đệ quy thì lại đưa bộ tham số của tác vụ đệ quy tương ứng vào stack.

|  |  |
| --- | --- |
| **Phân công** | |
| Nguyễn Trọng Anh (nhóm trưởng) | Phân công công việc cho nhóm, thiết kế power point. |
| Bùi Thị Cúc | Khái niệm, nhận dạng bài toán, phương pháp và lược đồ giải thuật đệ quy. |
| Lộc Thị Thúy Linh | Độ phức tạp của giải thuật đệ quy, ví dụ và chương trình demo |
| Bùi Thị Cúc | Phân loại đệ quy, ưu nhược điểm, khử đệ quy, ứng dụng. |