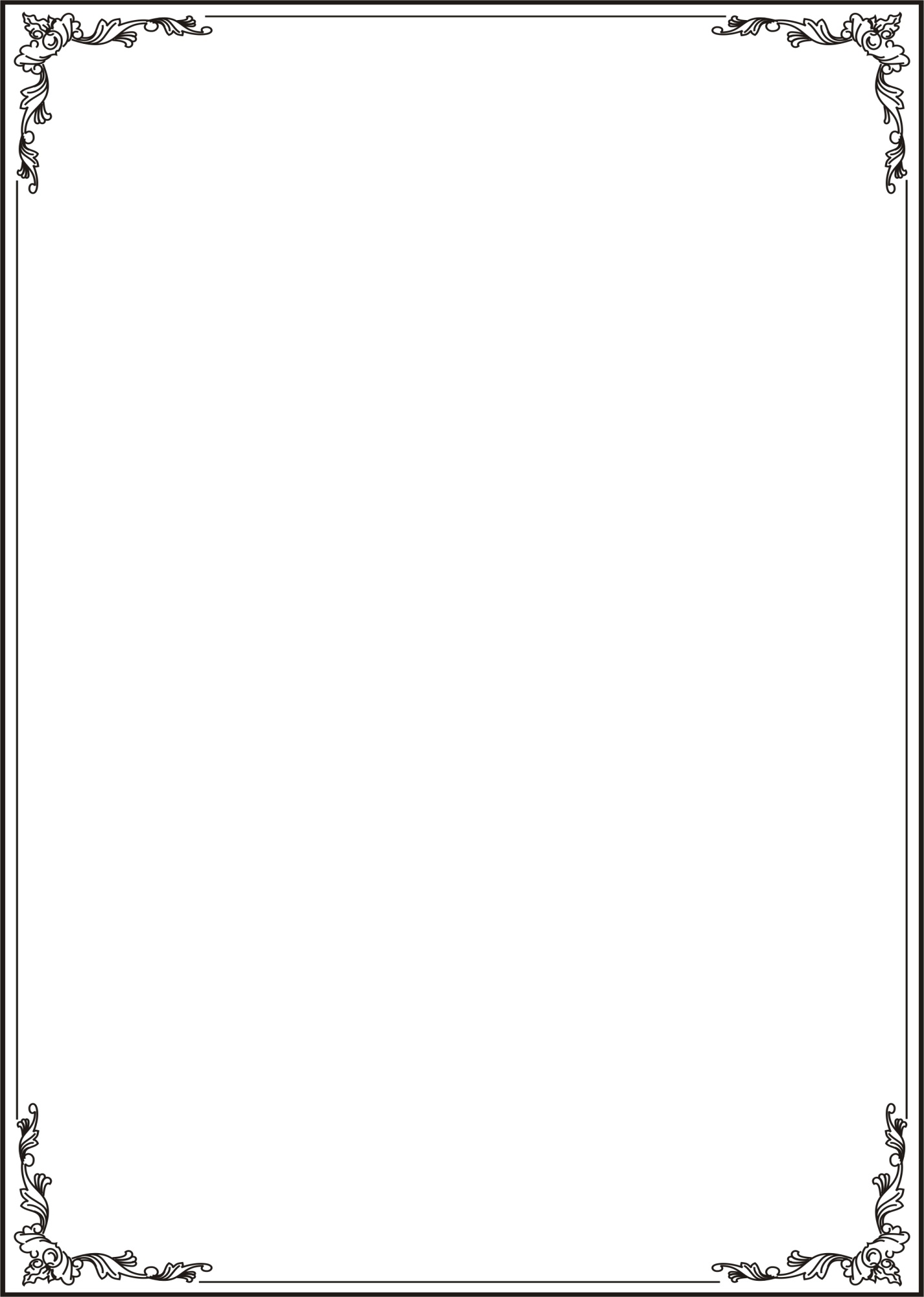
****

**ĐẠI HỌC QUỐC GIA HÀ NỘI**

TRƯỜNG ĐH KHOA HỌC TỰ NHIÊN

KHOA TOÁN – CƠ – TIN

-----🙞🙜🕮🙞🙜-----

**BÀI TIỂU LUẬN**

*Đề tài:* “Khử đệ quy: các dạng khử đệ quy và ví dụ”

Giảng viên: TS. Nguyễn Thị Hồng Minh

Nhóm 3:

1. Bùi Thị Thu Dung
2. Trần Đức Hào
3. Đào Thị Hằng
4. Bùi Thị Thanh Hương

1. **Khái quát**

***1. Phương pháp đệ quy***

P được gọi là một thủ tục đệ quy khi trong thân thủ tục có lời gọi đến chính nó.

Thực hiện một thủ tục đệ quy là thực hiện một dãy chương trình con lồng nhau.

mức 0 mức 1 mức 2 … suy biến

lời gọi P lời gọi P lời gọi P lời gọi P

Hình: quá trình thực hiện một thủ tục đệ quy

***2.Ưu điểm***: đơn giản, dễ hiểu, dễ triển khai thành chương trình.

***3.Nhược điểm:*** Chiếm dụng quá nhiều bộ nhớ, dễ gây lỗi tràn ngăn xếp, thường thực hiện chậm.

Hơn thế nữa, không phải ngôn ngữ lập trình nào cũng cho phép mã hóa thủ tục đệ quy.

* Chúng ta có thể khắc phục các nhược điểm trên bằng cách chuyển thủ tục đệ quy thành thủ tục lặp thông thường và sử dụng sự hỗ trợ của ngăn xếp.

**II.Phương pháp khử đệ quy**

Phương pháp **khử đệ quy** là gì?

***1.Định nghĩa:***

Khử đệ quy là phương pháp thay thế các lời gọi đệ quy bằng một thuật toán tương đương, bỏ đi các lời gọi đệ quy thay vào đó bằng việc sử dụng các vòng lặp và sự hỗ trợ của ngăn xếp dữ liệu.

Đệ quy và khử đệ quy là 2 khái niệm thường hay đi song hành cùng nhau, không có khử đệ quy khi không có đệ quy, nhưng ngược lại, có thể có đệ quy mà không có khử đệ quy.

*So sánh đệ quy và khử đệ quy*

|  |  |  |
| --- | --- | --- |
|  | Đệ quy | Không đệ quy |
| Cấu trúc lặp | Xây dựng một cấu trúc tùy ý (thường là hàm, có trường hợp dùng xây dựng trong lớp). | Sử dụng vòng lặp của ngôn ngữ lập trình. (có hai loại: vòng lặp xác định và không xác định số lần lặp) |
| Điểm dừng | Khi gặp trường hợp suy biến của quá trình đệ quy. | Khi điều kiện duy trì vòng lặp không còn được thỏa mãn. |
| Quá trình lặp | Đi từ trường hợp ban đầu đến trường hợp suy biến. Tại mỗi bước lặp độ phức tạp của bài toán giảm dần. | Là vòng lặp đơn thuần. Sau mỗi bước giá trị biến đếm thay đổi để đi đến điểm dừng. |
| Lặp vô hạn | Các bước đệ quy không thể làm đơn giản bài toán, chương trình không thể hội tụ về trường hợp suy biến. | Nếu điều kiện duy trì vòng lặp không bao giờ sai thì sẽ lặp vô hạn. |
| So sánh ưu nhược điểm | *Ưu:* Đệ quy có thể giải quyết tốt một số bài toán dạng vòng lặp 1 cách dễ dàng hơn. Chương trình thiết kế đơn giản. Dễ đưa ra phương án tối ưu, giải quyết được một số vấn đề vòng lặp thông thường không giải quyết được.  *Nhược:* Do mỗi lần gọi lại làm chương trình sinh vùng nhớ mới lên dùng đệ quy rất tốn bộ nhớ. Quản lý không tốt sẽ tràn bộ nhớ. | *Ưu:* ít tốn bộ nhớ, dễ dàng trong việc quản lý các vòng lặp. Quá trình thiết kế vòng lặp cũ đơn giản hơn rất nhiều so với đệ quy.  *Nhược:* nhiều bài toán với vòng lặp thông thường có khi không giải quyết được phải nhờ đến đệ quy. |

***2.Các dạng khử đệ quy thường gặp***

***2.1 Dạng khử ED[AP]***

*Dạng khử đệ quy ED[AP]:*

P(x) =

if E(x) then D(x)

Else {

A(x);

P(f(x));

}

Trong đó :

x là tập biến(một hoặc một bộ nhiều biến).

P(x) là thủ tục đệ quy phụ thuộc x

A(x), D(x) là các nhóm thao tác (lệnh) không đệ quy

f(x) là hàm biến đổi x

Xét quá trình thi hành P(x)

Gọi P0 là lần gọi P thứ 0 (đầu tiên) P(x)

P1 là lần gọi P thứ 1 (lần 2) P(f(x))

Pi là lần gọi P thứ i (lần i+1) P(f(f…f(x)…))

(P(fi(x)) là hợp i lần hàm f)

Trong lần gọi nếu E(x) đúng (TRUE) thì thi hành lệnh D(x)và kết thúc quá trình gọi.

Giả sử P được gọi đúng (n+1) lần. Khi đó ở trong lần gọi cuối cùng (thứ n) thì E(x) đúng, lệnh D(x) được thi hành và chấm dứt thao tác gọi thủ tục P(x).

Sơ đồ khối quá trình thực hiện lệnh gọi thủ tục P(x) có dạng sau:

P(x)≡

if E(x)

D(x)

else

A(x)

x=f(x)

if E(x)

D(x)

else

A(x)

x=f(x)

…

endif

endif

end.

Tương ứng với vòng lặp sau:

While(not E(x)) do

A(x);

x:=f(x);

End;

D(x);

Ví dụ 1

Tìm USCLN của 2 số nguyên dựa vào thuật toán Eucclide.

* Giải thuật đệ quy (dưới dạng thủ tục) tìm USCLN(m,n) bằng thuật toán Euclide:

USCLN(m,n,var us)=if(n=0) then us :=m

Else USCLN(n,m mod n,us);

* Trong trường hợp này thì:

x là (m,n,us)

B(x) là n=0;

D(x) là lệnh gán us :=m

A(x) là lệnh rỗng

f(x) là f(m,n,us)=(n,m mod n,us)

* Đoạn lệnh lặp tương ứng là:

While (n<>0) do

sd:=m mod n;

m := n;

n:= sd;

end;

us:=m;

* Thủ tục không đệ quy tương ứng trong pascal.

Procedure USCLN(m,n:integer;var us:integer);

Var sd : integer;

Begin

While(n<>0) do

sd:= m mod n;

m:=n;

n:=sd;

end;

* Hàm con không đệ quy tương ứng trong C++

Void USCLN(int m,int n,int &us)

{

While(n!=0){

int sd=m%n;

m=n;

n=sd;

}

Us=m;

}

*Ví dụ 2*

Phân tích 1 số thành thừa số các số nguyên tố

ví dụ bài toán phân tích số n thành thừa số nguyên tố bằng đệ quy :   
  
void phantich(int n)  
{

if(n==1) return ;  
else  
{

int d= 2;  
while(n%d!=0)  
d= d+ 1;  
printf("%d ",d);  
phantich(n/d);

}  
  
}

Khử đệ quy dạng ED[AP] cho bài toán phân tích thừa số nguyên tố :

Ta sẽ thay if(n==1) bằng while(n>1).

Phantich(n/d) bằng n=n/d;

Để khử lời gọi đệ quy.  
void phantich(int n)  
{

while(n>1)

{

int d=2;

while((n%d) != 0)

d = d+1;

printf("%d ",d);

n = n/d;

}

}  
Trong ví dụ trên có :   
E(x) : n =1   
D(x) : Exit  
A(x) : int d=2;  
while((n%d) != 0)  
d = d+1;  
printf("%d ",d);  
f(x) : n/d

***2.2. Dạng ED[APB]***

Lược đồ đệ quy:

P(x)

if (E (x) )

D ( x );

else

A ( x );

P ( f (x) );

B ( x );

endif

end.

**Trong đó:**

E( x): điều kiện suy biến

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

A(x), B(x): thao tác ngoài lời gọi đệ qui

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

Khử dạng ED[APB]:

P(x)

if (E (x) )

D ( x );

else

A ( x );

x = f ( x ); //f1

if ( E ( x ) )

D ( x );

else

A ( x );

x = f ( x ); //f2

…

endif

B ( x ) // x = f2 (x)

endif

B ( x ) // x = f1 (x)

End.

Giải thích cách khử trên:

* P kết thúc sau k+1 lần gọi:
* E(fk(x)) = true với
* fk(x) = f(f…(f(x))…)
* fo(x) = x
* Sau khi thực hiện D(fk(x)) sẽ phải thực hiện B(fk(x))
* Tiếp đó sẽ thực hiện B(fk-1(x))… và cuối cùng là B(x).
* Tóm lại, giá trị của x mỗi lần biến đổi x=f(x) được cất giữ thành chồng (stack) rồi dỡ ra theo chiều từ trên xuống để thực hiện B(x)

Sử dụng ngăn xếp *S*, kiểu phần tử phù hợp với *x* và hai vòng lặp để xây

dựng thuật toán khử đệ qui:

Ngoài ra:

P(x)≡

Start (S ); //Khởi tạo ngăn xếp S

While not E ( x ) do // Vòng lặp khử lời gọi đệ qui

A(x);

Push(x,S); //Cất x vào ngăn xếp

x := f(x);

endwhile;

D(x)

While not Is\_Empty (S) do//Vòng lặp gỡ ngăn xếp

Pick(S,x); *//Lấy x vào ngăn xếp*

B(x)

Endwhile;

end.

Phân tích ví dụ:

Ví dụ 3: Cho hai số tự nhiên a, n hãy tính an ?

+ Ý tưởng của bài toán:

Nếu n=1 thì trả ra kết quả là a.

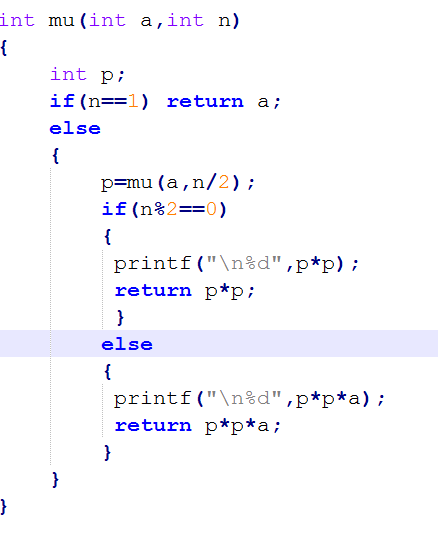
Ngược lại nếu n 1 thì cứ tính giá trị của p= a mũ n/2 lưu vào ngăn xếp.

Cho đến khi n=1 thì bắt đầu lấy các giá trị đó ra gọi đệ quy và tính toán.

Trong đó nếu n 2 thì an = p\*p

Ngược lại thì an = p\*p\*a

+ Lược đồ thuật toán:



Thành phần của giải thuật đệ quy dang ED[APB] của thuật toán tính an là:

E (x) n=1.

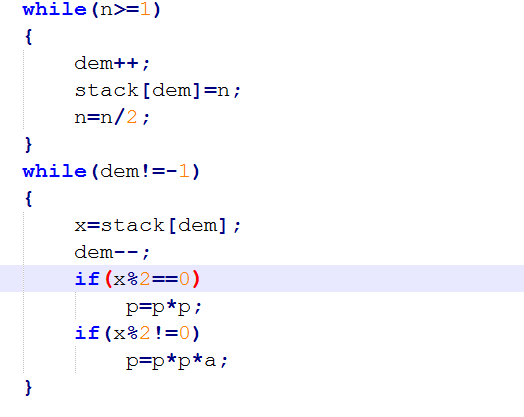
D(x) return a.

A(x) .

f(x) f ( a, n/2 ).

B(x)

+ Khử đệ quy cho bài toán trên bằng cách dùng xâu kí tự làm ngăn xếp như sau:



***2.3.Dạng 3: ED[APBP]***

*Dạng đệ quy:*

P(x) ≡

if (E(x))

D(x);

else

A(x);

P(f(x));

B(x);

P(g(x));

endif

end.

Trong đó: E(x): điều kiện suy biến

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

A(x), B(x): thao tác ngoài lời gọi đệ quy

f(x), g(x): hàm biến đổi tham số của 2 lời gọi đệ quy.

*Quá trình thực hiện thủ tục đệ quy P(x):*

* Nếu E(x) đúng thì thực hiện D(x).
* Còn nếu E(x) sai thì thực hiện A(x); gọi P(f(x)); thực hiện B(x); gọi P(g(x)), khi gọi P(g(x)) thì lại phát sinh lệnh A(g(x)). Như vậy, ngoài việc phải lưu vào stack các giá trị fi(x) thì ta còn phải lưu vào stack các giá trị gi(x) tương ứng khi ta lấy dữ liệu từ stack để thực hiện lệnh B(x) vào stack…. Điều kiện dừng là khi truy xuất tới phần tử lưu đầu tiên trong stack.

*Thuật toán khử đệ quy tương ứng với thủ tục P(x):*

P(x) ≡

Start (S);

Push ((x,1), S);

repeat

while not E(x) do

A(x);

Push ((x,2), S);

x := f(x);

endwhile;

D(x);

Pop (S, (x,k));

if(m = 2)

B(x);

x := g(x);

endif;

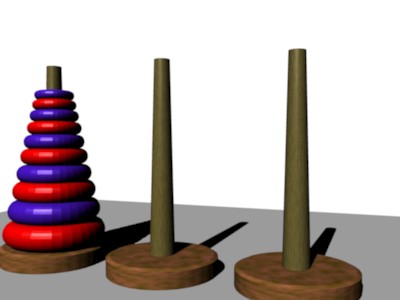
until (m = 1);

end.

*Ví dụ:* Bài toán Tháp Hà Nội.

Cho 3 cái cọc và một số đĩa có kích thước khác nhau để có thể cho vào cái cọc này. Ban đầu sắp xếp các đĩa theo trật tự kích thước vào một cọc sao cho đĩa nhỏ nhất nằm trên cùng. Yêu cầu di chuyển toàn bộ số đĩa sang một cọc khác tuân theo quy tắc:

* Mỗi lần chỉ được di chuyển một đĩa.
* Mỗi đĩa chỉ có thể được đặt lên một đĩa lớn hơn (không nhất thiết là 2 đĩa này phải có kích thước liền kề, tức là đĩa nhỏ nhất có thể nằm trên đĩa lớn nhất).



Dạng đệ quy thủ tục Tháp Hà Nội:

thapHN(n, A, B, C) ≡

if (n=1)

chuyen1tang(A, B);

else

{

thapHN(n-1, A, C, B);

chuyen1tang(A, B);

thapHN(n-1, C, B, A);

}

Với n là số đĩa, A là cột đầu, B là cột giữa, C là cột cuối, chuyen1tang(A, B) là thao tác chuyển đĩa từ cột A sang cột B.

Trong trường hợp này:

x là bộ (n, A, B, C)

E(x): n=1

D(x): chuyen1tang(A, B)

A(x):

f(x) = f(n, A, B, C) = (n-1, A, C, B)

nên n n – 1, A A, B C, C B

B(x): chuyen1tang(A, B)

g(x) = g(n, A, B, C) = (n – 1, C, B, A)

nên n n – 1, A C, B B, C A

**Tổ chức ngăn xếp để khử lời gọi đệ quy.**

* Đối với lời gọi đệ quy cuối cùng, sau nó không còn lệnh nào khác nữa, có thể khử mà không cần dùng ngăn xếp vì không lưu lại các biến cục bộ, đia chỉ quay ra cũng không cần lưu giữ, chỉ cần gán lại giá trị của các tham số như trong lời gọi đệ quy.
* Đối với các lời gọi đệ quy không phải là lời gọi đệ quy cuối cùng thì phải dùng đến ngăn xếp. Trước mỗi lần thực hiện lặp lại thủ tục, các biến cục bộ cần lưu lại là:

+ số đĩa (kích thước của bài toán).

+ chồng đĩa nào là nguồn, là trung gian, là đích trong bước lặp đó.

Thiết kế:

* Cần 1 ngăn xếp mỗi lần lưu lại 1 bộ (n,x,tg,d). Ta dùng hàm MA\_ASCII(n) để thay số nguyên n bằng 1 ký tự có mã ASCII chính bằng n. Mỗi phần tử của ngăn xếp là 1 ký tự.
* Quá trình đi vào ngăn xếp: nếu chưa phải là trường hợp suy biến thì đẩy vào ngăn xếp 4 ký tự ứng với số đĩa, cọc nguồn, cọc trung gian và cọc đích.
* Trường hợp suy biến thì giải bài toán trực tiếp.
* Quá trình đi ra: lấy dần từ ngăn xếp từng bộ 4 kí tự ứng với số đĩa, cột nguồn, trung gian và đích. Thực hiện lặp lại với các tham số mới lấy ra, lặp tới khi ngăn xếp rỗng.

*Giải thuật khử đệ quy tương ứng:*

{

Creat\_Stack(S);

Push ((n, A, B, C, 1), S);

While (n > 1) {

Push ((n, A, B, C, 2), S);

n --;

swap(B, C);

}

chuyen1tang(A, B);

pop (S, (n, A, B, C, m))

if(m=2)

{

chuyen1tang(A, B);

n --;

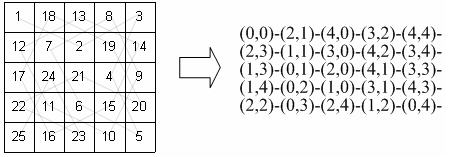
swap(A, C);

}

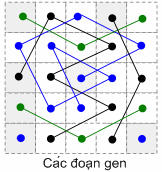
Until (k=1)

}

***3. Phương pháp mới :“xử lý những đoạn gen”***  
***3.1 Mô tả:***Ta để ý rằng đáp án của bài toán là mội chuỗi những bước nhảy liên tiếp qua hết tất cả các ô trên bàn cờ giống như một “đoạn gen” duy nhất gồm nhiền “nhiễm sắc thể” nối tiếp nhau. Các nhiễm sắc thể tức là các ô của bàn cờ. Hai nhiễm sắc thể liên tiếp được nối với nhau nếu thỏa mãn luật nhảy của quân mã . Như vậy ta có thể đưa bài toán về một bài toán khác là từ những “nhiễm sắc thể” đơn lẻ hãy tìm cách ghép nối chúng lại với nhau sao cho tạo thành một “đoạn gen” duy nhất.   
  
Ví dụ: Ta xem đáp án của bàn cờ 5x5 và “đoạn gen” hoàn chỉnh:

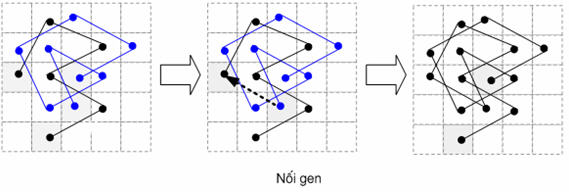


***3.2 Định nghĩa các thuật ngữ:***  
**1. Nhiễm sắc thể:**là một ô trên bàn cờ NxN   
  
**2. Đoạn gen:**là một chuỗi gồm một hoặc nhiều nhiễm sắc thể liên tiếp nhau.Hai nhiễm sắc thể liên tiếp nhau trong một đoạn gen thỏa mãn luật nhảy của quân mã. Mỗi đoạn gen sẽ có đầu và đuôi, đoạn gen gồm duy nhất một nhiễm sắc thể thì đầu và đuôi là một.

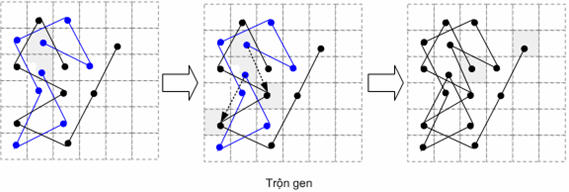


*(hình trên gồm 7 đọan gen có đầu là các ô tô đậm)*

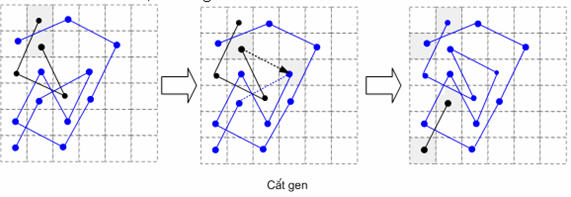
**3. Nối gen:** Ta nối đầu (hay đuôi) của đoạn gen A vào đầu (hay đuôi) của đoạn gen B tạo thành đoạn C thỏa mãn C là một đoạn gen có các nhiễm sắc thể là các nhiễm sắc thể từ đoạn gen A và đoạn gen B.



**4. Trộn gen:**Ta trộn gen A vào trong gen B như sau: Từ đầu và đuôi của đoạn gen A ta nối vào **tại một bước nhảy**nào đó của đoạn gen B tạo thành đoạn C thỏa mãn đoạn C là một đoạn gen có các nhiễm sắc thể là các nhiễm sắc thể từ A và B.



**5. Cắt gen:**Cắt hai đoạn gen A và đoạn B tạo thành hai đoạn gen C và đoạn gen D như sau: ta tìm trên đoạn gen B vị trí để có thể nối đầu (hay đuôi) của đoạn gen A vào và tạo thành đoạn gen C. Phần còn lại của đoạn gen B sẽ bị cắt ra ngoài tạo thành đoạn gen D.



***3.4 Chứng minh quy tắc:***  
Với ba công cụ biến đổi những đoạn gen như trên: nối gen, trộn gen và cắt gen ta chứng minh rằng sẽ luôn tìm ra lời giải cho bài toán đặt ra: nối các nhiễm sắc thể riêng lẻ lại và tạo thành một đoạn gen duy nhất   
  
Ta thấy rằng với phương pháp nối gen, từ hai đoạn gen A và đoạn gen B ban đầu sau khi nối tạo thành đoạn gen C chứa các nhiễm sắc thể có trong A và B. Như vậy số đoạn gen sau khi thực hiện một phép nối sẽ giảm đi một đoạn.   
  
Trộn đoạn gen A vào đoạn gen B tạo thành đoạn gen C chứa các nhiễm sắc thể có trong A và B. Nên sau khi thực hiêm một phép trộn gen số đoạn gen sẽ giảm đi một đoạn.   
  
Với phép cắt gen, đoạn gen A cắt vào đoạn gen B, tạo thành đoạn gen C và đoạn gen D trong đó đoạn gen C và đoạn gen D chứa các nhiễm sắc thể của đoạn gen A và đoạn gen B. Như vậy số đoạn gen sau khi thực hiện phép cắt gen sẽ không đổi (nhưng nó sẽ tạo ra những trường hợp mới để có thể thực hiện hai phép nối gen và trộn gen.   
  
Vì quân mã tại một vị trí bất kỳ trên bàn cờ có tối thiểu là 2 cách chọn cho bước nhảy kế tiếp nên ta luôn luôn tồn tại hai cặp gen nào đó mà chúng có thể cắt nhau được.   
  
Từ đó ta có thể kết luận rằng số đoạn gen ban đầu N\*N đoạn sau nhiều lần biến đổi gen (dùng lần lượt ba phép biến đổi trên) sẽ tạo thành một đoạn gen duy nhất chính là lời giải ta cần tìm.

***3.5. Tiến hành xây dựng chương trình***  
Bước 1: Chuyển N\*N ô bàn cờ vào các đoạn ghen ta sẽ được N\*N đoạn ghen. Mỗi đoạn ghen có chiều dài là 1.   
  
Bước 2: Thược hiện các phép “nối ghen”, “trộn gen” và “cắt gen” cho đến khi chỉ còn một đoạn gen duy nhất. Đoạn gen này có chiều dài là N\*N chính là đáp án của bài toán.   
  
Tuy nhiên để tiết kiệm bộ nhớ ta có thể thực hiện theo các sau : Ngay từ khi khởi tạo ta dùng phương pháp đệ quy truyền thống nhưng chỉ cho chương trình đệ quy tới giá trị N\*(N-1), sau đó mới áp dụng phương pháp xử lý các đoạn gen như sau.   
  
Bước 1. Ban đầu ta dùng cách đệ quy truyền thống cho mã nhảy đến một giá trị là N\*(N-1) thì dừng lại. (Còn lại N ô chưa điền)   
  
Bước 2. Lần lượt điền những ô chưa điền với những giá tri tăng dần N\*(N-1)+1, N\*(N-1)+2,... đến N\*N thì dừng lại (nhu vậy tất cả các ô đều được điền nhưng không đúng luật nhảy của quân mã).   
  
Bước 3. Biến đổi mảng mảng 2 chiều đã điền thành các đoạn gen   
  
Bước 4. Sau đó ta phân tích bàn cờ hai chiều (đã đánh số từ 1=>NxN) thành các đoạn gen.   
  
Bước 5. Tiến hành xử lý các đoạn gen dùng lần lượt 3 công cụ “nối gen”, “trộn gen” và “cắt gen” để nối các đọan gen thành một đoạn gen duy nhất.   
Bước 6. Xuất kết quả ra file.   
**☞Những lưu ý khi khi xây dựng chương trình**  
  
Khi “nối gen” và “trộn gen” không thể thực hiện được nữa chương trình thường chạy “cắt gen”. Sau khi đoạn gen A cắt đoạn gen B và sinh ra đoạn gen C và gen D thì ta phải đảo ngược đoạn gen C này để tránh trường hợp đoạn gen C lại cắt đoạn gen D sẽ sinh ra đoạn gen A và B thì chương trình sẽ bị lặp.   
  
Tuy nhiên ta không thể dự đoán trước được có những trường hợp bị lặp đặc biệt nào đó nên cách giải quyết tốt nhất là khi đoạn gen A cắt đoạn gen B ta tìm hết các vị trí cắt trên B mà A có thể cắt (tối đa là 8 vị trí), nếu có vị trí cắt ta lấy ngẫu nhiên một vị trí trên B trong các vị trí có thể đó để làm điểm cắt. Như thế chương trình sẽ tránh được những trường hợp lặp đặc biệt nào đó.

*Phân công công việc trong nhóm:*

|  |  |
| --- | --- |
| **Tên Thành Viên** | **Công Việc** |
| Bùi Thị Thu Dung (nhóm trưởng) | Phần mở rộng và ví dụ minh họa.  Làm powerpoint. |
| Trần Đức Hào | Dạng ED[AP], ví dụ minh họa |
| Đào Thị Hằng | Dạng ED[APB], ví dụ minh họa.  So sánh đệ quy và khử đệ quy. |
| Bùi Thị Thanh Hương | Dạng ED[APBP], ví dụ minh họa.  Soạn thảo word. |