**Nhóm 6**

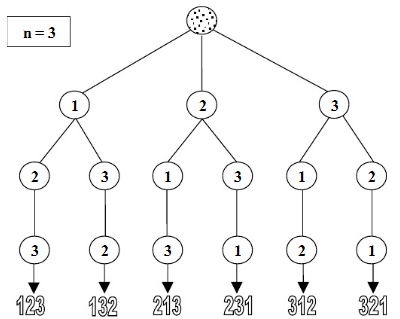
1. Lê Thị Ngọc Ánh - K54A2
2. Nguyễn Hồng Chương - K54A2
3. Lê Thị Hiến - K54A2
4. Trần Thị Thu Hằng - K54A2 (4/9/1991)

**Bài 2.** Phân tích bài toán, xây dựng thuật toán và viết chương trình cho các bài toán

* **2.1 : Thuật toán quay lui liệt kê hoán vị n phần tử**

Ý tưởng : Biểu diễn hoán vị duoi dạng P1, P2 …. Pn , trong đó Pi nhận giá trị từ 1 đến n và Pi # với Pj với i # j. Các giá trị từ 1 được đề cử cho Pi , trong đó giá trị j được chấp nhận nếu nó chưa được dùng trước đó. Do đó cần phải ghi nhớ xem giá trị j đã được sử dụng chưa . Ta thực hiện điều này bằng một mảng B ,trong đó Bj = true nếu j chưa được dùng và ngược lại .Đầu tiên các giá trị trong B này phải được khởi tạo là true , sau khi gán j cho xi thì nhận Bj = false , sau khi gọi xong thủ tục Try(i+1) thì thiết lập lại Bj = true , để đánh dấu nó chưa được dùng để cho bước thử tiếp theo .

**VD :** Với n = 3



Cây liệt kê , hoán vị 3 phần tử

* **Bài 2.2** : **Liệt kê các tổ hợp k phần tử của các số từ 1 đến n.**

Phân tích : Làm theo ý tưởng của phương pháp quay lui

Tổ hợp chập k của n phần tử (k <= n).

Dùng 1 mảng dd[n] đánh dấu xem phần tử đó đã dùng hay chưa.

Dùng mảng a[n] để lưu lại giá trị của từng phần tử.

Dùng giải thuật try(i) để sinh gía trị

Nếu i = k thì in giá trị nghiệm, ngược lại sinh tiếp +1 bằng try(i+1).

Ví dụ:

Input : k = 3, n = 4.

Output: In ra tổ hợp chập 3 của 4

1: 2 3

1: 2 4

2: 3 4

1: 3 4

Lược đồ:

void Try(i) {  
 for(j = cận dưới đến cận trên) {  
 if (chấp nhận j) {  
// *Lưu trạng thái cũ và chuyển sang trạng thái mới.*  
 *được tính theo trạng thái j*  
 if (.. là nghiệm)  
 In kết quả.  
 else  
 Try(i+1)  
// *Trả lại trạng thái cũ.*  
 }  
 }  
}

* **2.4: Bài toán xâu ABC**

Bài toán : Cho số nguyên dương N <100, tìm 1 xâu gồm toàn các kí tự A,B,C thoả mãn: Xâu có độ dài N, 2 đoạn con bất kì liền nhau đều khác nhau, xâu có ít kí tự C nhất.

Phân tích bài toán:

* Bài toán tìm tất cả các từ đẹp độ dài n cho trước yêu cầu tìm nghiệm là các vector x có n thành phần:

+ xi nhận giá trị trong tập {A,B,C}

+ x không có 2 đoạn con liên tiếp nào bằng nhau.

- Thuật giải:

+) Nếu dãy x[1…n] thỏa mãn điều kiện 2 dãy con liền kề bất kỳ khác nhau thì ký tự đầu tiên bao giờ cũng phải chứa một ký tự “C”.

+) Như vậy với mỗi dãy gồm k ký tự liên tiếp thì dãy đó bao giờ cũng phỉa có lớn hơn hoặc bằng (k div 4 ký tự “C” ).

+) Tại mỗi bước chọn x[i] nếu ta có T[i] ký tự “C” thì n-x[i] còn lại có ít nhất (n-x[i])div ký tự “C”.

+) Ta dùng điều kiện này để đánh giá có nên tiếp tục lựa chọn hay chuyển sang phương án khác hay không .

**Bài 3**: Tự đặt một số đề bài toán, phân tích bài toán, xây dựng giải thuật, đánh giá độ phức tạp và viết chương trình để minh họa phương pháp quay lui, phương pháp nhánh cận (mỗi phương pháp ít nhất 01 bài toán).

* **3.1 Phương pháp quay lui:**

*Bài toán 1* : Tô màu bản đồ (Map Coloring Problem)

Input : Đồ thị G=(V,E).

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

Các bước :

1. Lập danh sách các đỉnh của đồ thị E’ := [v1,v2,…,vn] được sắp xếp theo thứ tự bậc giảm dần : d(v1) >=d(v2) >= …>=d(vn).

Đặt i:=1;

1. Tô màu I cho đỉnh đầu tiên trong danh sách. Duyệt lần lượt các đỉnh tiếp theo và tô mà I cho đỉnh không kề đỉnh đã được tô màu i.
2. Nếu tất cả các đỉnh đã được to màu thì kết thúc, đồ thị được tô màu bằng I màu. Ngược lại, sang bước 4.
3. Loại khỏi E’ các đỉnh đã tô màu. Sắp xếp lại cá đỉnh trong E’ theo thứ tự bậc giảm dần. Đặt i:=i+1 và quay lại bước 2.

*Bài toán 2* : Tìm chu trình Hamilton của đồ thị (Hamiltonian Path Problem)

* *Chu trình Hamiltonian:* Là chu trình đi qua tất cả các đỉnh của đồ thị mỗi đỉnh đúng một lần và một cạnh trong đồ thị nối đỉnh đầu của dây chuyền với đỉnh cuối của nó.
* Ý tượng tìm 1 chu trình Hamilton :

Thuật toán đệ quy quay lùi thực hiện các bước sau:

B1: đưa đỉnh x vào đường đi

B2: kiểm tra chu trình đủ chiều dài chưa? (tức là đã đi hết tất cả các đỉnh chưa?) và đồng thời kiểm tra đình đầu và đỉnh cuối của chu trình có cạnh nối hay ko? Nếu true hết cả 2 điều kiện thì kết luận có chu trình Hamilton. Ngược lại gọi B3

B3: Duyệt hết tất cả các đỉnh kề với x

B4: Kiểm tra đỉnh kề đã đi qua chưa? Nếu true gọi lại hàm đệ qui tìm chu trình Hamilton

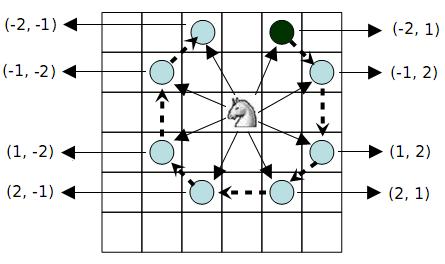
B5: Xóa đỉnh kề ra khỏi chu trình

*Bài toán 3*: Xếp hậu

* Liệt kê tất cả các cách sắp xếp những con hậu trên bàn cờ NxN sao cho chúng không ăn được nhau.
* Ta xếp n con hậu trên n dòng, Theo nguyên lý nhân ta có n\*n cách sắp xếp thoả mãn điều kiện đầu bải. Để làm điều đó ta dùng thủ tục đệ quy để gải bài toán.
* Ta đánh ghi số cột và dòng của bàn cờ từ 1 đến n, mỗi cách sắp xếp ứng với 1 bộ gồm a1,a2,.....,an với ai = j (j=1,2,...,n) có nghĩa là con hậu thứ i đặt vào cột j. Giả sử ta chọn được i-1 con hậu bằng cách duyệt tất cả các khả năng của nó.   
  Quan trọng nhất là ta tìm điều kiện chấp nhận j, một con hậu đứng ở một ô trong bàn cờ nó có nhiều nhất bốn hướng đi(đường dọc, đường ngang và hai đường chéo).  
  => Vậy điều kiện chấp nhận thứ i thoả mãn không nằm trên đường đi của tất cả i-1 con hậu đã xếp. Bởi vì n con hậu xếp ở hàng nên đường đi ngang của chúng là không chiến nhau, do đó khi chọn con hậu thư i chỉ cần kiểm tra xem trên 2 đường chéo và đường dọc của chúng có chiếu vào những con hậu đã xếp không? Để kiểm tra điều này mỗi đường ta dùng một biến trạng thái.
  + Đường dọc kiểm soát bằng biến b[j],(j=1,2,...,n).
  + Một đường chéo kiểm soát bằng biến c[i+j],i+j={2,....,2n}.
  + Còn đường chéo kia kiểm soát bằng biến d[i-j],i-j={1-n,....,n-1}.  
    Các biến trạng thái này khởi gán giá trị True trong thủ tục **Init**.
  + Như vậy con hậu thứ i được chấp nhận xếp vào cột j nếu nó thoả mãn cả ba biến b[j],c[i+j],d[i-j] đều có giá trị true. Các biến này gán giá trị False khi xếp xong con hậu thứ i, và trả lại giá trị true sau khi gọi Result hay Try(i+1).

*Bài toán 4*: Mã đi tuần (hay hành trình của quân mã) là bài toán về việc di chuyển một quân mã trên bàn cờ vua ( 8 x 8). Quân mã được đặt ở một ô trên một bàn cờ trống nó phải di chuyển theo quy tắc của cờ vua để đi qua mỗi ô trên bàn cờ đúng một lần.

* Ý tưởng cơ bản:  Dùng thuật toán quay lui; xuất phát từ 1 ô, gọi số nước đi là t=1, ta cho quân mã thử đi tiếp 1 ô (có 8 nước đi có thể), nếu ô đi tiếp này chưa đi qua thì chọn làm bước đi tiếp theo.



* Đi theo chiều kim đồng hồ.
* Tại mỗi nước đi kiểm tra xem tổng số nước đi bằng n\*n chưa, nếu bằng thì mã đã đi qua tất cả các ô ⇒ dừng (do chỉ cần tìm một giải pháp).
* Trường hợp ngược lại, gọi đệ quy để chọn nước đi tiếp theo.

Ngoài ra, nếu tại một bước tìm đường đi, nếu không tìm được đường đi tiếp thì thuật toán sẽ quay lui lại nước đi trước và tìm đường đi khác…

**Thuật giải đệ quy:**

Tại mỗi bước lần lượt cho quân mã thử tất cả các nước đi kế tiếp (tám nước đi kế tiếp). Với mỗi bước đi, kiểm tra xem nếu nước đi hợp lệ (chưa đi qua và ở trong bàn cờ) thì thử đi nước này. Nếu quân mã đã đi qua hết bàn cờ thì xuất kết quả. Ngược lại thì gọi đệ quy tiếp cho vị trí mới thử trên.

**Lưu ý** : là mỗi khi vị trí đã đi qua được đánh dấu chính bằng chính thứ tự nước đi trên bàn cờ. Sau khi không thử vị trí này thì phải bỏ đánh dấu để chọn giải pháp khác (trường hợp quay lui).

Nếu coi các ô của bàn cờ là các đỉnh của đồ thị và các cạnh là nối giữa hai đỉnh tương ứng với hai ô mã giao chân thì dễ thấy rằng hành trình của quân mã cần tìm sẽ là một đường đi Hamilton.

Ta có thể xây dựng hành trình bằng thuật toán quay lui kết hợp vớiphương pháp duyệt ưu tiên Warnsdorff: Nếu gọi deg(x, y) là số ô kề với ô (x, y) và chưa đi qua (kề ở đây theo nghĩa đỉnh kề chứ không phải là ô kề cạnh) thì từ một ô ta sẽ không thử xét lần lượt các hướng đi có thể, mà ta sẽ ưu tiên thử hướng đi tới ô có deg nhỏ nhất trước.

Trong trường hợp có tồn tại đường đi, phương pháp này hoạt động với tốc độ tuyệt vời: Với mọi n chẵn trong khoảng từ 6 tới 18, với mọi vị trí ô xuất phát, trung bình thời gian tính từ lúc bắt đầu tới lúc tìm ra một nghiệm < 1 giây. Tuy nhiên trong trường hợp n lẻ, có lúc không tồn tại đường đi, do phải duyệt hết mọi khả năng nên thời gian thực thi lại hết sức tồi tệ. (Có xét ưu tiên như trên hay xét thứ tự như trước kia thì cũng vậy thôi).