

	VIETTEL AI RACE	TD052
	THUẬT TOÁN QUAY LUI	Lần ban hành: 1

1. Mô hình thuật toán quay lui (Backtrack Algorithm)

Giả sử ta cần xác định bộ $X = (x_1, x_2, \dots, x_n)$ thỏa mãn một số ràng buộc nào đó. Ứng với mỗi thành phần x_i ta có n_i khả năng cần lựa chọn. Ứng với mỗi khả năng $j \in n_i$ dành cho thành phần x_i ta cần thực hiện:

- Kiểm tra xem khả năng j có được chấp thuận cho thành phần x_i hay không? Nếu khả năng j được chấp thuận thì ta xác định thành phần x_i theo khả năng j . Nếu i là thành phần cuối cùng ($i=n$) ta ghi nhận nghiệm của bài toán. Nếu i chưa phải cuối cùng ta xác định thành phần thứ $i+1$.
- Nếu không có khả năng j nào được chấp thuận cho thành phần x_i thì ta quay lại bước trước đó ($i-1$) để thử lại các khả năng còn lại.

Thuật toán quay lui được mô tả như sau:

```
Thuật toán Back-Track ( int i ) {
    for ( j =<Khả năng 1>; j <=n; j++
        ){ if (<chấp thuận khả năng
            j>)
            X[i] = <khả năng
            j>; if ( i ==n)
            Result(); else Back-
            Track(i+1);
        }
}
```

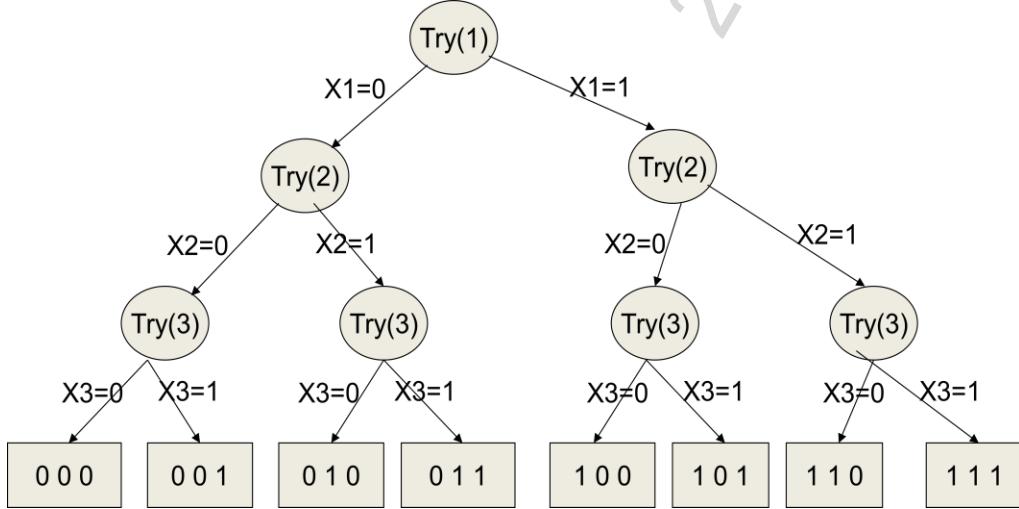
Ví dụ 2.7. Duyệt các xâu nhị phân có độ dài n .

Lời giải. Xâu nhị phân $X = (x_1, x_2, \dots, x_n) | x_i = 0, 1$. Mỗi $x_i \in X$ có hai lựa chọn $x_i = 0, 1$. Cả hai giá trị này đều được chấp thuận mà không cần có thêm bất kỳ điều kiện gì. Thuật toán được mô tả như sau:

```
void Try ( int i ) {
    for (int j =0; j<=1;
        j++){ X[i] = j;
        if (i ==n)
        Result(); else
        Try (i+1);
    }
}
```

	VIETTEL AI RACE	TD052
	THUẬT TOÁN QUAY LUI	Lần ban hành: 1

Khi đó, việc duyệt các xâu nhị phân có độ dài n ta chỉ cần gọi đến thủ tục Try(1). Cây quay lui được mô tả như Hình 2.1 dưới đây.



Hình 2.1. Duyệt các xâu nhị phân độ dài 3

Chương trình duyệt các xâu nhị phân có độ dài n bằng thuật toán quay lui được thể hiện như dưới đây.

```

#include <iostream>
#include <iomanip>
#define MAX 100
using namespace std;
int X[MAX], n,
dem=0;
void Init() { //thiết lập độ dài xâu nhị phân
    cout<<"\n Nhập n="; cin>>n;
}
void Result(void){ //In ra xâu nhị phân X[] = x1, x2,.., xn
    cout<<"\n Kết quả
    <<++dem<<":";
    for(int i =1; i<=n;
    i++)
        cout<<X[i]<<setw(3);
}
void Try(int i){ //thuật toán quay lui
    for (int j=0; j<=1; j++){//duyệt các khả năng j dành cho xi
        X[i]=j; //thiết lập thành phần xi là j
    }
}
  
```

	VIETTEL AI RACE THUẬT TOÁN QUAY LUI	TD052
		Lần ban hành: 1

```

if(i==n) //nếu i là thành phần cuối cùng
    Result(); // ta đưa ra kết quả
else //trong trường hợp khác
    Try(i+1); //ta xác định tiếp thành phần  $x_{i+1}$ 
}
}

int main(void){    Init(); Try(1);}


```

Ví dụ 2.8. Duyệt các tập con K phần tử của 1, 2, .., N.

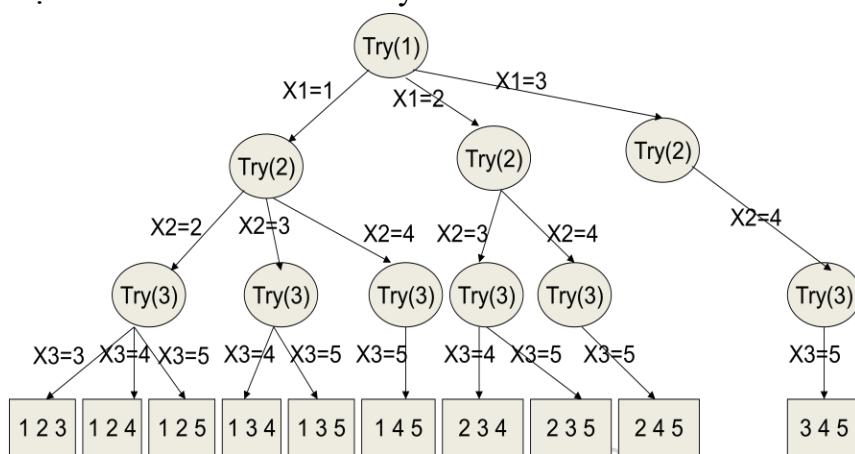
Lời giải. Mỗi tập con K phần tử $X = (x_1, x_2, \dots, x_K)$ là bộ không tính đến thứ tự K phần tử của 1, 2, .., N. Mỗi $x_i \in X$ có $N-K+i$ lựa chọn. Các giá trị này đều được chấp thuận mà không cần có thêm bất kỳ điều kiện gì. Thuật toán được mô tả như sau:

```

void Try ( int i ) {
    for (int j = X[i-1]+1; j<=N-K+ i;
        j++){ X[i] = j;
        if (i ==K)
            Result(); else Try
            (i+1);
    }
}


```

Khi đó, việc duyệt các tập con K phần tử của 1, 2, .., N ta chỉ cần gọi đến thủ tục Try(1). Cây quay lui được mô tả như hình dưới đây.



Hình 2.2. Duyệt các tập con 3 phần tử của 1, 2, 3, 4, 5.

Chương trình liệt kê các tập con k phần tử của 1, 2, .., n được thể hiện như sau.

```

#include <iostream>
#include <iomanip>
#define MAX 100

```

	VIETTEL AI RACE THUẬT TOÁN QUAY LUI	TD052
		Lần ban hành: 1

```

using namespace std;
int X[MAX], n, k, dem=0;
void Init(){//thiết lập giá trị cho n, k
    cout<<"\n Nhập n, k: "; cin>>n>>k;
}
void Result(void){ cout<<"\n Kết quả "<<++dem<<":";//đưa ra kết quả
    for(int i =1; i<=k; i++) cout<<X[i]<<setw(3);
}
void Try(int i){//thuật toán quay lui
    for (int j=X[i-1]+1; j<=n-k+i; j++){ //duyệt trên tập khả năng dành cho x_i
        X[i]=j; //thiết lập thành phần x_i là j
        if(i==k) //nếu x_i đã là thành phần cuối
            Result(); //ta đưa ra kết quả
        else //trong trường hợp khác
            Try(i+1); //ta đi xác định thành phần thứ x_{i+1}
    }
}
int main(void){
    Init(); X[0]=0 ; Try(1);
}

```

Ví dụ 2.9. Duyệt các hoán vị của 1, 2, .., N.

Lời giải. Mỗi hoán vị $X = (x_1, x_2, \dots, x_N)$ là bộ có tính đến thứ tự của 1, 2, .., N. Mỗi $x_i \in X$ có N lựa chọn. Khi $x_i = j$ được lựa chọn thì giá trị này sẽ không được chấp thuận cho các thành phần còn lại. Để ghi nhận điều này, ta sử dụng mảng chuaxet[] gồm N phần tử. Nếu $chuaxet[i] = \text{True}$ điều đó có nghĩa giá trị i được chấp thuận và $chuaxet[i] = \text{False}$ tương ứng với giá trị i không được phép sử dụng. Thuật toán được mô tả như sau:

```

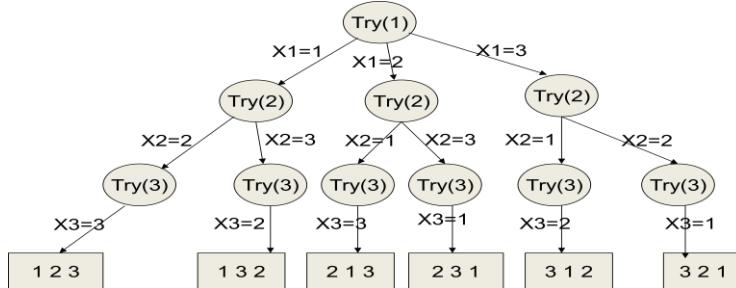
void Try ( int i ) {
    for ( int j=1; j<=N;
          j++){ if
            (chuaxet[j] ) {
                X[i] = j;chuaxet[j] =
                False; if ( i ==N)
                    Result();
                else Try (i+1);
                Chuaxet[j] = True;
            }
}

```



}

Khi đó, việc duyệt các hoán vị của 1, 2, .., N ta chỉ cần gọi đến thủ tục Try(1). Cây quay lui được mô tả như hình dưới đây.



Hình 2.3. Duyệt các hoán vị của 1, 2, 3.

Chương trình liệt kê tất cả các hoán vị của 1, 2, .., n được thể hiện như sau:

```
#include <iostream>
#include <iomanip>
#define MAX 100
using namespace std;
int X[MAX], n,
dem=0; bool
chuaxet[MAX];
void Init() { // thiết lập giá trị cho n
    cout<<"\n Nhập n="; cin>>n;
    for(int i=1; i<=n; i++) // thiết lập giá trị cho mảng chuaxet[]
        chuaxet[i]=true;
}
void Result(void){ // Đưa ra hoán vị hiện tại
    cout<<"\n Kết quả "<<++dem<<":";
    for(int i =1; i<=n; i++) cout<<X[i]<<setw(3);
}
void Try(int i){ // thuật toán quay lui duyệt các hoán vị của 1, 2, .., n.
    for (int j=1; j<=n; j++){ // duyệt các khả năng j cho thành phần xi
        if(chuaxet[j]){ // nếu khả năng j đúng chưa được dùng
            X[i]=j; // thiết lập thành phần xi là j
            chuaxet[j]=false; // thiết lập chuaxet[j] đã được
            // dùng if(i==n) // nếu xi đã là thành phần cuối cùng
            Result(); // ta đưa ra kết quả
        }
    }
}
```

	VIETTEL AI RACE THUẬT TOÁN QUAY LUI	TD052
		Lần ban hành: 1

```

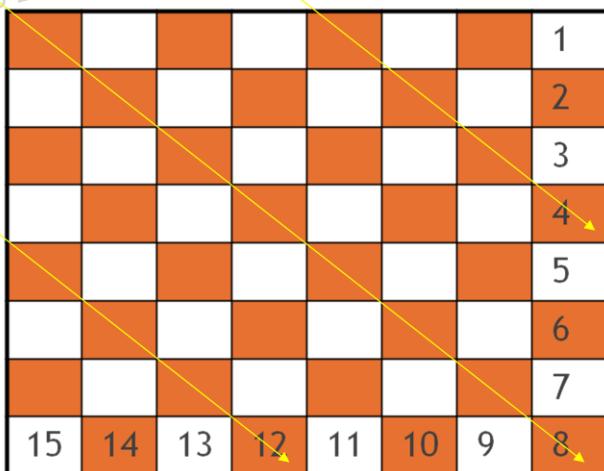
        else //trong trường hợp khác
            Try(i+1); //ta xác định tiếp thành phần thứ i+1
            chuaxet[j]=true; //nhớ hoàn trả lại giá trị cho chuaxet[j]
        }
    }
int main(void){
    Init(); Try(1);
}

```

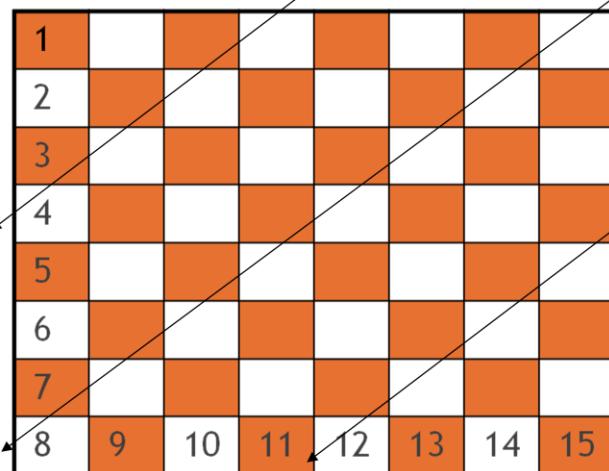
Ví dụ 2.10. Bài toán N quân hậu. Trên bàn cờ kích cỡ $N \times N$, hãy đặt N quân hậu mỗi quân trên 1 hàng sao cho tất cả các quân hậu đều không ăn được lẫn nhau.

Lời giải. Gọi $X = (x_1, x_2, \dots, x_n)$ là một hoán vị của $1, 2, \dots, n$. Khi đó, $x_i = j$ được hiểu là quân hậu hàng thứ i đặt ở cột j . Để các quân hậu khác không thể ăn được, quân hậu thứ i cần không được lấy trùng với bất kỳ cột nào, không được cùng đường chéo xuôi, không được cùng trên đường chéo ngược. Ta có n cột $Cot = (c_1, \dots, c_n)$, có $Xuoi[2*n-1]$ đường chéo xuôi, $Nguoc[2*n-1]$ đường chéo ngược. Quân hậu ở hàng i được đặt vào cột j nếu $A[j] = True$ (chưa có quân hậu nào ánh ngửi cột j), $Xuoi[i-j+n] = True$ (chưa có quân hậu nào ánh ngửi đường chéo $i-j+n$), $Nguoc[i + j - 1] = True$ (chưa có quân hậu nào ánh ngửi đường chéo ngược $i + j - 1$).

Đường chéo xuôi $Xuoi[i-j+n]$



Đường chéo ngược $Nguoc[i+j-1]$



Hình 2.4. Mô tả các đường chéo, xuôi đường chéo ngược

Thuật toán quay lui giải bài toán n quân hậu được mô tả như dưới đây. void Try (int i){

```

        for(int j=1; j<=n; j++){
            if( Cot[j] && Xuoi[ i - j + n ] && Nguoc[i + j - 1]){

```

	VIETTEL AI RACE	TD052
	THUẬT TOÁN QUAY LUI	Lần ban hành: 1

```

X[i] =j; Cot[j]=FALSE;
Xuoi[ i - j + n]=FALSE;
Nguoc[ i + j -
1]=FALSE; if(i==n)
Result();
else Try(i+1);
Cot[j] =
TRUE;
Xuoi[ i - j + n] = TRUE;
Nguoc[ i + j - 1] =
TRUE;
}
}

```

Chương trình giải bài toán n quân hậu được thể hiện như dưới đây.

```

#include <iostream>
#include <iomanip>
#define MAX 100
using namespace std;
int X[MAX], n,
dem=0;
bool COT[MAX], DCXUOI[MAX], DCNGUOC[MAX];;
void Init(){ //thiết lập kích cỡ bàn cờ
    cout<<"\n Nhập n="; cin>>n;
    for(int i=1; i<=n; i++){ //thiết lập tất cả các cột đều chưa bị án
        ngữ
        COT[i]=true;
    }
    for(int i=1; i<2*n; i++){ //thiết lập các đường chéo
        DCXUOI[i]=true; // đường chéo xuôi chưa bị án ngữ
        DCNGUOC[i]=true; // đường chéo ngược chưa bị án ngữ
    }
}
void Result(void){ //đưa ra một phương
    án      cout<<"\n Kết     quả

```

	VIETTEL AI RACE	TD052
	THUẬT TOÁN QUAY LUI	Lần ban hành: 1

```

"<<<++dem<<":";   for(int i =1;
i<=n; i++)
    cout<<X[i]<<setw(3);
}

void Try(int i){ //đây là thuật toán quay lui
    for (int j=1; j<=n; j++){ //duyệt các khả năng j đặt quân hậu vào hàng i
        if( COT[j] && DCXUOI[i-j+n]&& DCNGUOC[i+j-1]){
            //nếu đúng cột j, đường chéo xuôi i-j +n, đường chéo ngược i+j-1
            //chưa bị án ngữ
            X[i]=j; //ta đặt được quân hậu hàng i vào vốt j
            COT[j] = false; // cột j đã bị án ngữ
            DCXUOI[i-j+n]=false; // đường chéo xuôi i-j+n bị án ngữ
            DCNGUOC[i+j-1]=false; //đường chéo ngược i+j-1 bị án
            //ngữ if(i==n) // nếu đây là quân hậu hàng n
            Result();// ta đưa ra phương án hiện tại
        } else //trong trường hợp khác
            Try(i+1); // ta đặt tiếp quân hậu hàng i+1
        COT[j] = true; // nhớ trả lại giá trị cột j
        DCXUOI[i-j+n]=true; //trả lại giá trị đường chéo xuôi
        DCNGUOC[i+j-1]=true; // trả lại giá trị đường chéo ngược
    }
}

int main(void){ Init(); Try(1); }

```