Đệ quy quay lui (Backtracking)

- Bài toán duyệt (liệt kê/vét cạn) tất cả các phương án (cấu hình tổ hợp) thỏa mãn các điều kiện (ràng buộc đặt ra)

Liệt kê tất cả các xâu nhị phân độ dài 3

000
001
010
011
100
101
110
111

Liệt kê tất cả các hoán vị của 1, 2, 3

1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

Liệt kê tất cả các nghiệm nguyên dương của phương trình: X1 + X2 + X3 = 5

1 1 3
1 2 2
1 3 1
2 1 2
2 2 1
3 1 1

Điền số Sudoku

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	1	4	3	6	5	8	9	7
3	6	5	8	9	7	2	1	4
8	9	7	2	1	4	3	6	5
5	3	1	6	4	2	9	7	8
6	4	2	9	7	8	5	3	1
9	7	8	5	3	1	6	4	2

Đệ quy quay lui (Backtracking)

- Bài toán duyệt (liệt kê/vét cạn) tất cả các phương án (cấu hình tổ hợp) thỏa mãn các điều kiện (ràng buộc đặt ra)
- Lời giải được biểu diễn bởi bộ các biến quyết định (X1, X2, . . ., Xn), mỗi biến Xk có thể nhận các giá trị từ tập Ak
- Xét lần lượt từng biến (giả sử lần lượt xét các biến theo thứ tự X1, X2, . . ., Xn)
- Với mỗi biến thì duyệt tất cả các giá trị có thể gán cho biến đó (kiểm soát ràng buộc)

```
1.  Try(k){  // thử giá trị cho Xk
2.    for each v in Ak do
3.        if check(v, k){ // kiểm tra xem v có thể gán cho Xk mà ko vi phạm ràng buộc
4.            Xk = v; // gán v cho Xk
5.            [Cập nhận một số cấu trúc dữ liệu liên quan]
6.            if k = n then Solution();
7.            else Try(k+1);
8.            [Khôi phục trạng thái cấu trúc dữ liệu được cập nhật ở dòng 5
9.        }
10.  }

Main(){
    Try(1);
}
```

Đệ quy quay lui (Backtracking)

- Liệt kê các xâu nhị phân độ dài n

```
public class Main {
    static int[] X;
    static int n;
    public static void solution(){
        for(int i = 1; i <= n; i++) System.out.print(X[i]);
        System.out.println();
    }
    public static void Try(int k){
        for(int v = 0; v <=1; v++){
            X[k] = v;
            if(k == n) solution();
            else Try(k+1);
        }
    }
    public static void main(String[] args) {
        n = 3;
        X = new int[n+1];
        Try(1);
    }
}
```

Đệ quy quay lui (Backtracking)

- Liệt kê các xâu nhị phân độ dài n

```
public class Main {
    static int[] X;
    static int n;
    public static void solution(){
        for(int i = 1; i <= n; i++) System.out.print(X[i]);
        System.out.println();
    }
    public static void Try(int k){
        for(int v = 0; v <=1; v++){
            X[k] = v;
            if(k == n) solution();
            else Try(k+1);
        }
    }
    public static void main(String[] args) {
        n = 3;
        X = new int[n+1];
        Try(1);
    }
}
```

```
000
001
010
011
100
101
110
111
```

Đệ quy quay lui (Backtracking)

- Liệt kê các hoán vị của 1, 2, . . ., n. Sử dụng mảng đánh dấu: mark[v] = true nếu v đã xuất hiện, và mark[v] = false, ngược lại

```
public class Main {
    static int[] X;
    static boolean[] mark;
    static int n;
    public static void solution(){
        for(int i = 1; i <= n; i++)   System.out.print(X[i] + " ");
        System.out.println();
    }
    public static void Try(int k){
        for(int v = 1; v <= n; v++){
            if(!mark[v]) {
                X[k] = v;
                mark[v] = true;
                if (k == n) solution();
                else Try(k + 1);
                mark[v] = false;
            }
        }
    }
    public static void main(String[] args) {
        n = 3;
        X = new int[n+1];   mark = new boolean[n+1];
        for(int v = 1; v <= n; v++) mark[v] = false;
        Try(1);
    }
}
```

Đệ quy quay lui (Backtracking)

- Liệt kê các hoán vị của 1, 2, . . ., n. Sử dụng mảng đánh dấu: mark[v] = true nếu v đã xuất hiện, và mark[v] = false, ngược lại

```
public class Main {
    static int[] X;
    static boolean[] mark;
    static int n;
    public static void solution(){
        for(int i = 1; i <= n; i++)   System.out.print(X[i] + " ");
        System.out.println();
    }
    public static void Try(int k){
        for(int v = 1; v <= n; v++){
            if(!mark[v]) {
                X[k] = v;
                mark[v] = true;
                if (k == n) solution();
                else Try(k + 1);
                mark[v] = false;
            }
        }
    }
    public static void main(String[] args) {
        n = 3;
        X = new int[n+1];   mark = new boolean[n+1];
        for(int v = 1; v <= n; v++) mark[v] = false;
        Try(1);
    }
}
```

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```