

Phân tích và Thiết kế THUẬT TOÁN

Hà Đại Dương

duonghd@mta.edu.vn

Web: fit.mta.edu.vn/~duonghd

Bài 11 - Phương pháp quay lui Back tracking method

PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN

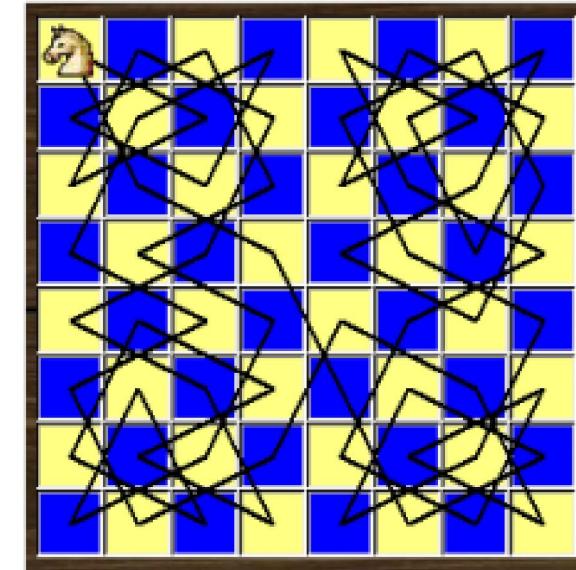
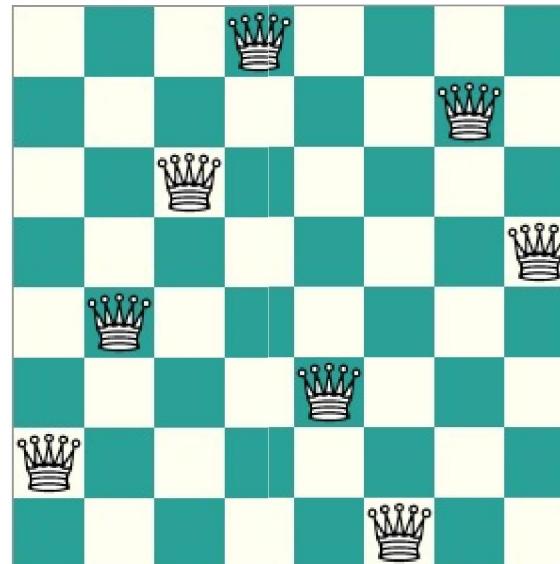
NỘI DUNG

- I. Giới thiệu
- II. Lược đồ chung
- III. Bài toán áp dụng
- IV. Bài tập

I. Giới thiệu

- Phương pháp quay lui dùng để giải các bài toán mà lời giải của nó X là một tập các phần tử x_1, x_2, \dots, x_n .
- Ví dụ: Bài toán 8 hậu
Mã đi tuần

...



I. Giới thiệu

- Nét đặc trưng của phương pháp quay lui là các bước hướng tới lời giải cuối cùng của bài toán hoàn toàn được làm thử.
- Tại mỗi bước, nếu có một lựa chọn được chấp nhận thì ghi nhận lại lựa chọn này và tiến hành các bước thử tiếp theo. Còn ngược lại không có lựa chọn nào thích hợp thì làm lại bước trước, xoá bỏ sự ghi nhận và quay về chu trình thử các lựa chọn còn lại.
- Hành động này được gọi là quay lui, thuật toán thể hiện phương pháp này gọi là quay lui.
- Điểm quan trọng của thuật toán là phải ghi nhớ tại mỗi bước đi qua để tránh trùng lặp khi quay lui. Để thấy là các thông tin này cần được lưu trữ vào một ngăn xếp, nên thuật toán thể hiện ý thiết kế một cách đệ quy.

II. Lược đồ chung

Lời giải của bài toán thường biểu diễn bằng một vec tơ gồm n thành phần $x = (x_1, \dots, x_n)$ phải thỏa mãn các điều kiện nào đó. Để chỉ ra lời giải x , ta phải xây dựng dần các thành phần lời giải x_i .

Tại mỗi bước i :

- Đã xây dựng xong các thành phần x_1, \dots, x_{i-1}
- Xây dựng thành phần x_i bằng cách lần lượt thử tất cả các khả năng mà x_i có thể chọn :
 - Nếu một khả năng j nào đó phù hợp cho x_i thì xác định x_i theo khả năng j . Thường phải có thêm thao tác ghi nhận trạng thái mới của bài toán để hỗ trợ cho bước quay lui. Nếu $i = n$ thì ta có được một lời giải, ngược lại thì tiến hành bước $i+1$ để xác định x_{i+1} .
 - Nếu không có một khả năng nào chấp nhận được cho x_i thì ta lùi lại bước trước (bước 1-1) để xác định lại thành phần x_{i-1} .

II. Lược đồ chung

Mô hình của phương pháp quay lui có thể viết bằng thủ tục sau, với n là số bước cần phải thực hiện, k là số khả năng mà xi có thể chọn lựa.

Try(i) ≡

for (j = 1 → k)

If (x_i chấp nhận được khả năng j)

{

Xác định x_i theo khả năng j;

Ghi nhận trạng thái mới;

if(i < n)

 Try(i+1);

else

 Ghi nhận nghiệm;

 Trả lại trạng thái cũ cho bài toán;

}

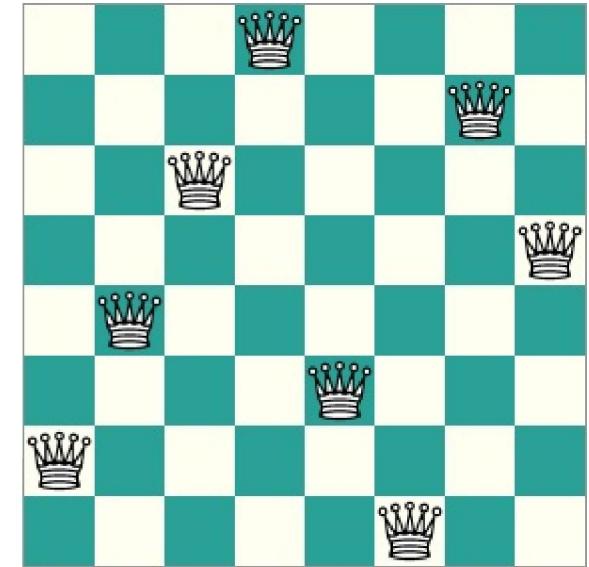
III. Bài toán áp dụng

I. Bài toán 8 hậu

1. Phát biểu bài toán

TÁM QUÂN HẬU ĐƯỢC ĐẶT LÊN BÀN CỜ VUA
sao cho chúng không ăn được nhau .

Bài toán này là một ví dụ nổi tiếng về việc dùng các phương pháp thử và sai
và phương pháp quay lui.



III. Bài toán áp dụng

I. Bài toán 8 hậu

2. Thiết kế thuật toán

Mấu chốt của thuật toán rõ ràng là xét xem có thể đặt quân hậu tiếp theo như thế nào. Theo luật cờ vua, một quân hậu có thể ăn các quân khác nếu nằm trên cùng 1 đường, đường này có thể là :

- Hàng,
- Cột,
- Các đường chéo (đi qua tọa độ vị trí của hậu).

Suy ra rằng mỗi hàng chỉ có thể chứa 1 và chỉ 1 quân hậu. Nên việc chọn vị trí cho quân hậu thứ i có thể giới hạn được ở hàng thứ i . Như thế tham số i trở thành chỉ hàng, và quá trình chọn vị trí cho quân hậu tiến hành trên toàn giá trị có thể có của các cột j .

III. Bài toán áp dụng

I. Bài toán 8 hậu

2. Thiết kế thuật toán

quy ước :

$x[i]$ // Chỉ quân hậu thứ i : nằm ở hàng i .

$x[i] = j$ // quân hậu thứ i đặt ở cột j ;

Để quân hậu i (trên hàng i) chấp nhận cột j thì cột j và 2 đường chéo qua ô $\langle i,j \rangle$ phải còn trống (tức là không có quân hậu khác chiếm lĩnh)

Lưu ý rằng trong 2 đường chéo :

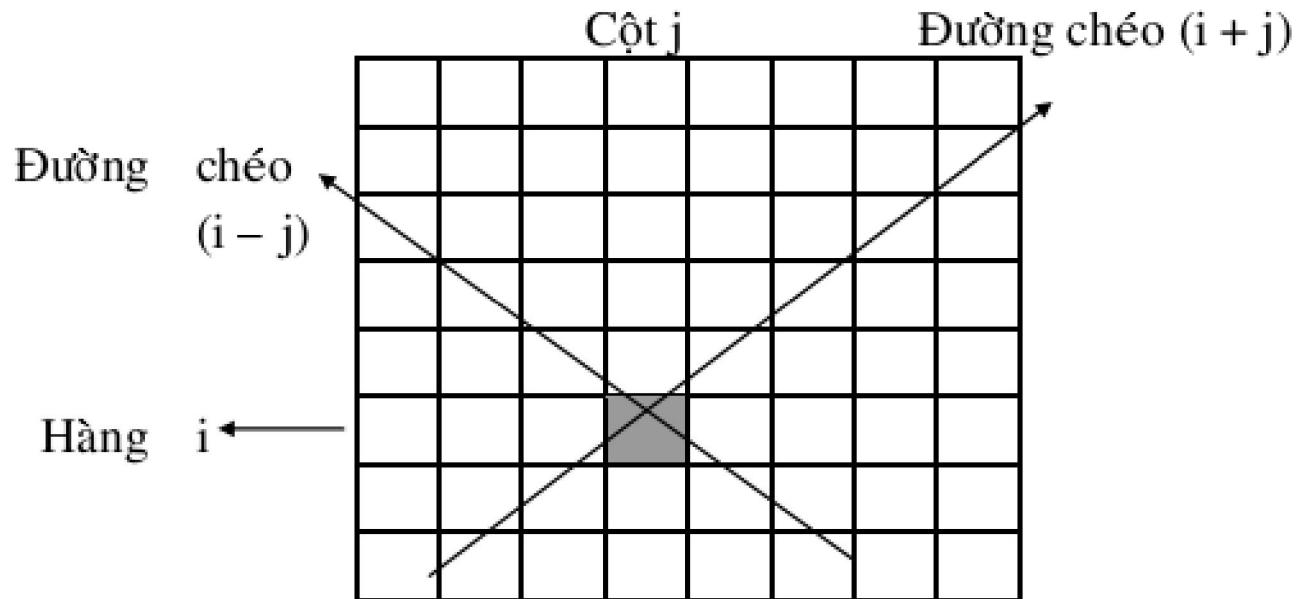
- Đường chéo ngược (vuông góc với đường chéo chính) : tất cả các ô đều có tổng 2 tọa độ i và j là hằng;

- Đường chéo thuận (song song với đường chéo chính) : gồm tất cả các ô (i,j) mà có hiệu các tọa độ $(i-j)$ là hằng số.

III. Bài toán áp dụng

I. Bài toán 8 hậu

2. Thiết kế thuật toán



Do đó ta sẽ chọn các mảng Boolean 1 chiều để biểu diễn các trạng thái này :

$a[j] = 1$: Có nghĩa là không có quân hậu nào ở cột.

$b[i+j] = 1$: Có nghĩa là không có quân hậu nào ở đường chéo ngược ($i+j$) .

$c[i-j] = 1$: Có nghĩa là không có quân hậu nào ở đường chéo thuận ($i-j$) .

III. Bài toán áp dụng

I. Bài toán 8 hậu

2. Thiết kế thuật toán

Vì :

$$1 \leq i, j \leq 8 \Rightarrow 2 \leq i+j \leq 16 \quad \text{Và} \quad -7 \leq i - j \leq 7.$$

Nên ta có thể khai báo :

```
int x[8],  
    a[8],  
    b[15],  
    c[15];
```

Với các dữ liệu đã cho, thì lệnh đặt quân hậu sẽ thể hiện bởi :

`x[i] = j; // đặt quân hậu thứ i trên cột j.`

`a[j] = 0; // Khi đặt hậu tại cột j , thì cột j không còn trống nữa`

`b[i + j] = 0; // Các đường chéo tương ứng cũng không còn`

`c[i - j] = 0; // trống nữa .`

III. Bài toán áp dụng

I. Bài toán 8 hậu

2. Thiết kế thuật toán

lệnh Dời quân hậu là :

//Làm cho hàng i và các đường chéo tương ứng trở thành trống

$$a[j] = 1;$$

$$b[i+j] = 1;$$

$$c[i-j] = 1;$$

Còn điều kiện an toàn là ô có tọa độ (i, j) nằm ở hàng và các đường chéo chưa bị chiếm (được thể hiện bằng trị True). Do đó, có thể được thể hiện bởi biểu thức logic :

$$(a[j] \&\& b[i+j] \&\& c[i-j])$$

Bài toán 8 hậu - Cài đặt

```
#include <conio.h>           int init()           int thu(int i)
#include <stdio.h>            {                   {
int x[9], a[9];             for (int i=1; i<=8; i++)   for (int j=1 ; j<=8; j++)
int b[16], c[16];           {                   if (a[j] && b[i+j-1] && c[i-j+8])
int thu(int i);             x[i]=0;           {                   x[i] = j;
int init();                 a[i]=1;           a[j] = 0;
int printkq();              }                   b[i+j-1] = 0;
int main()                  for (int i=1; i<=15; i++)   c[i-j+8] = 0;
{                           {                   if (i < 8 )
    init();               b[i]=1;           thu(i+1);
    thu(1);               c[i]=1;           else
    return 0;               }                   {
}                           return 0;           printkq();
}                           }                   }
                           }                   a[j] = 1;
                           }                   b[i+j-1] = 1;
                           }                   c[i-j+8] = 1;
                           }                   }
```

III. Bài toán áp dụng

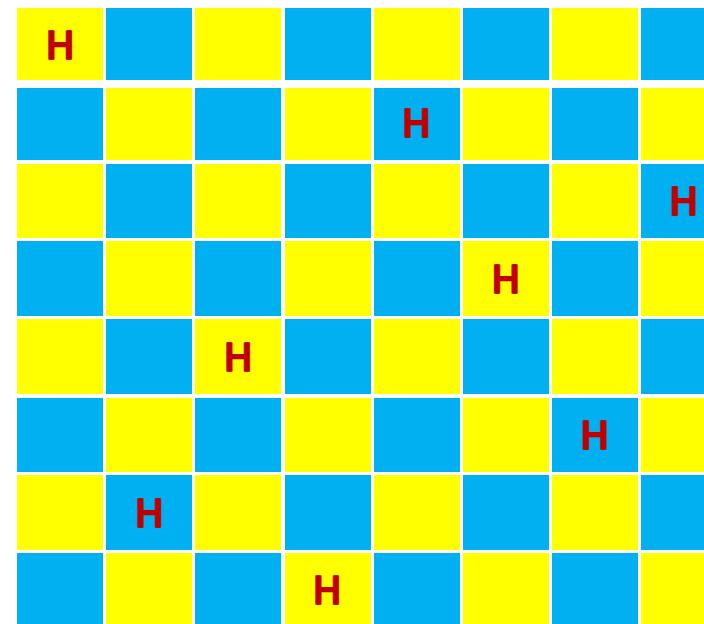
I. Bài toán 8 hậu

Kết quả	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	1	5	8	6	3	7	2	4
	1	6	8	3	7	4	2	5
	1	7	4	6	8	2	5	3
	1	7	5	8	2	4	6	3
	2	4	6	8	3	1	7	5
	2	5	7	1	3	8	6	4
	2	5	7	4	1	8	6	3
	2	6	1	7	4	8	3	5
	2	6	8	3	1	4	7	5
	2	7	3	6	8	5	1	4
	2	7	5	8	1	4	6	3
	2	8	6	1	3	5	7	4

III. Bài toán áp dụng

I. Bài toán 8 hậu

Kết quả



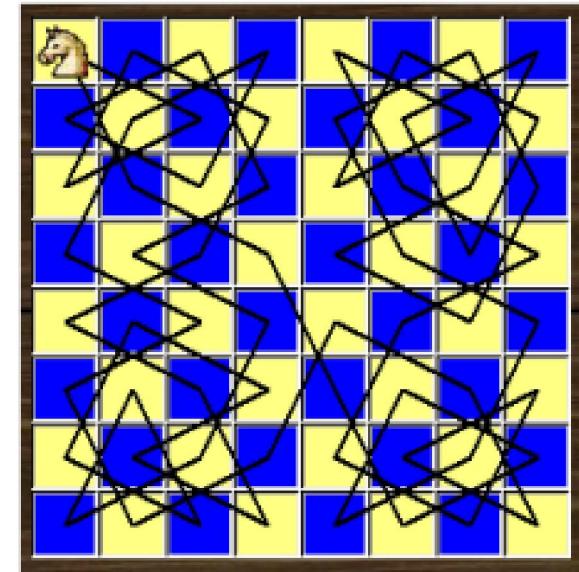
III. Bài toán áp dụng

II. Bài toán ngựa đi tuần

1. Bài toán

Cho bàn cờ có $n \times n$ ô. Một con ngựa được phép đi theo luật cờ vua, đầu tiên được đặt ở ô có tọa độ x_0, y_0 .

Vấn đề là hãy chỉ ra các hành trình (nếu có) của ngựa
Đó là ngựa đi qua
tất cả các ô của bàn cờ, mỗi ô đi qua đúng một lần



III. Bài toán áp dụng

II. Bài toán ngựa đi tuần

2. Thiết kế thuật toán

Cách giải quyết rõ ràng là xét xem có thể thực hiện một nước đi kếp nữa hay không. Sơ đồ đầu tiên có thể phát thảo như sau :

```
Try(i) ≡  
    for ( j = 1 → k )  
        If ( xi chấp nhận được khả năng k )  
            {  
                Xác định xi theo khả năng k;  
                Ghi nhận trạng thái mới;  
                if( i < n2 )  
                    Try(i+1);  
                else  
                    Ghi nhận nghiệm;  
                    Trả lại trạng thái cũ cho bài toán;  
            }  
    }
```

Để mô tả chi tiết thuật toán, ta phải qui định cách mô tả dữ liệu và các thao tác, đó là :

- Biểu diễn bàn cờ .
 - Các khả năng chọn lựa cho x_i ?
 - Cách thức xác định x_i theo j .
 - Cách thức ghi nhận trạng thái mới, trả về trạng thái cũ.
 - Ghi nhận nghiệm.
- . . .

* Ta sẽ biểu diễn bàn cờ bằng 1 ma trận vuông cấp n : int h[n][n];

Sở dĩ thể hiện mỗi ô cờ bằng 1 số nguyên thay cho giá trị boole (để đánh dấu ô đã được đi qua chưa) là vì ta muốn lần dò theo quá trình di chuyển của con ngựa.

Ta qui ước như sau :

$h[x][y] = 0 \equiv \hat{O} \langle x,y \rangle$ ngựa chưa đi qua;

$h[x][y] = i \equiv \hat{O} \langle x,y \rangle$ ngựa đã đi qua ở bước thứ i ($1 \leq i \leq n^2$).

* Các khả năng chọn lựa cho x_i ? Đó chính là các nước đi của ngựa mà x_i có thể chấp nhận được.

Với cặp tọa độ bắt đầu $\langle x,y \rangle$ như trong hình vẽ, có tất cả 8 ô $\langle u,v \rangle$ mà con ngựa có thể đi đến. Giả sử chúng được đánh số từ 0 đến 7 như hình sau :

Tọa độ (a,b)
(-2,-1)
(-1,-2)
(1,-2)
(2,-1)

hàng

x
→

1	2	3 t	4	5	1
4			3		2
5				2	3
					3
6				1	4
	7		0		5



cột
y

(8 bước đi có thể có của con ngựa)

Tọa độ (a,b)
(-2,1)
(-1,2)
(1,2)
(2,1)

Một phương pháp đơn giản để có được u, v từ x, y là ta dùng 2 mảng a và b lưu trữ các sai biệt về tọa độ .Nếu ta ~~đặt~~^{gán} một chỉ số k để đánh số “bước đi kế” thì chi tiết đó được thể hiện bởi : $u = x + a[k]; v = y + b[k]; k = \overline{0,7}$.

Điều kiện “chấp nhận được” có thể được biểu diễn như kết hợp của các điều kiện :

Ô mới phải thuộc bàn cờ ($1 \leq u \leq n$ và $1 \leq v \leq n$) và chưa đi qua ô đó, nghĩa là $h[u,v] = 0$;

* Để ghi nhận nước đi hợp lệ ở bước i , ta gán $h[u][v] = i$; và để hủy một nước đi thì ta gán $h[u][v] = 0$.

* Ma trận h ghi nhận kết quả nghiêm. Nếu có $\langle x,y \rangle$ sao cho $h\langle x,y \rangle = 0$ thì đó không phải là lời giải của bài toán , còn ngược là h chứa đường đi của ngựa.

II. Bài toán ngựa đi tuần

3. Cài đặt

Vậy thuật toán có thể mô tả như sau :

Input n, //Kích thước bàn cờ

x, y;//Toạ độ xuất phát ở bước i

Output h;

Mô tả :

Try(i, x, y) ≡

for(k = 0; k <= 7; k++)

{

 u = x + a[k];

 v = y + b[k];

 if (1 <= u ,v <= n &&h[u][v] == 0)

{

 h[u][v] = i;

 if (i < n*n)

 Try(i+1,u,v);

 else

 xuat_h(); // In ma trận h

}

 h[u][v] = 0;

}

II. Bài toán ngựa đi tuần

4. Thủ nghiệm

	n=5	x=1	y=1	
1	6	15	10	21
14	9	20	5	16
19	2	7	22	11
8	13	24	17	4
25	18	3	12	23

Vậy thuật toán có thể mô tả như sau :

Input n, //Kích thước bàn cờ
x, y;//Toạ độ xuất phát ở bước i

Output h;

Mô tả :

```

Try(i, x, y) ≡
    for(k = 0; k <= 7; k++)
    {
        u = x + a[k];
        v = y + b[k];
        if (1 <= u ,v <= n &&h[u][v] == 0)
        {
            h[u][v] = i;
            if (i < n*n)
                Try(i+1,u,v);
            else
                xuat_h(); // In ma trận h
        }
        h[u][v] = 0;
    }
}

```

II. Bài toán ngựa đi tuần

4. Thủ nghiệm

	n=6	x=2	y=3		
36	17	6	29	8	11
19	30	1	10	5	28
16	35	18	7	12	9
23	20	31	2	27	4
34	15	22	25	32	13
21	24	33	14	3	26

Vậy thuật toán có thể mô tả như sau :

Input n, //Kích thước bàn cờ
x, y;//Toạ độ xuất phát ở bước i

Output h;

Mô tả :

```

Try(i, x, y) ≡
    for(k = 0; k <= 7; k++)
    {
        u = x + a[k];
        v = y + b[k];
        if (1 <= u ,v <= n &&h[u][v] == 0)
        {
            h[u][v] = i;
            if (i < n*n)
                Try(i+1,u,v);
            else
                xuat_h(); // In ma trận h
        }
        h[u][v] = 0;
    }
}

```

III. Bài toán áp dụng

III. Bài toán liệt kê dãy nhị phân độ dài N

1. Phát biểu bài toán

Liệt kê các dãy có chiều dài n dưới dạng $x_1x_2\dots x_n$, trong đó $x_i \in \{ 0,1 \}$.

III. Bài toán áp dụng

III. Bài toán liệt kê dãy nhị phân độ dài N

2. Thiết kế thuật toán

Ta có thể sử dụng sơ đồ tìm tất cả các lời giải của bài toán. Hàm Try(i) xác định x_i , trong đó x_i chỉ có 1 trong 2 giá trị là 0 hay 1. Các giá trị này mặc nhiên được chấp nhận mà không cần phải thỏa mãn điều kiện gì.

III. Bài toán áp dụng

III. Bài toán liệt kê dãy nhị phân độ dài N

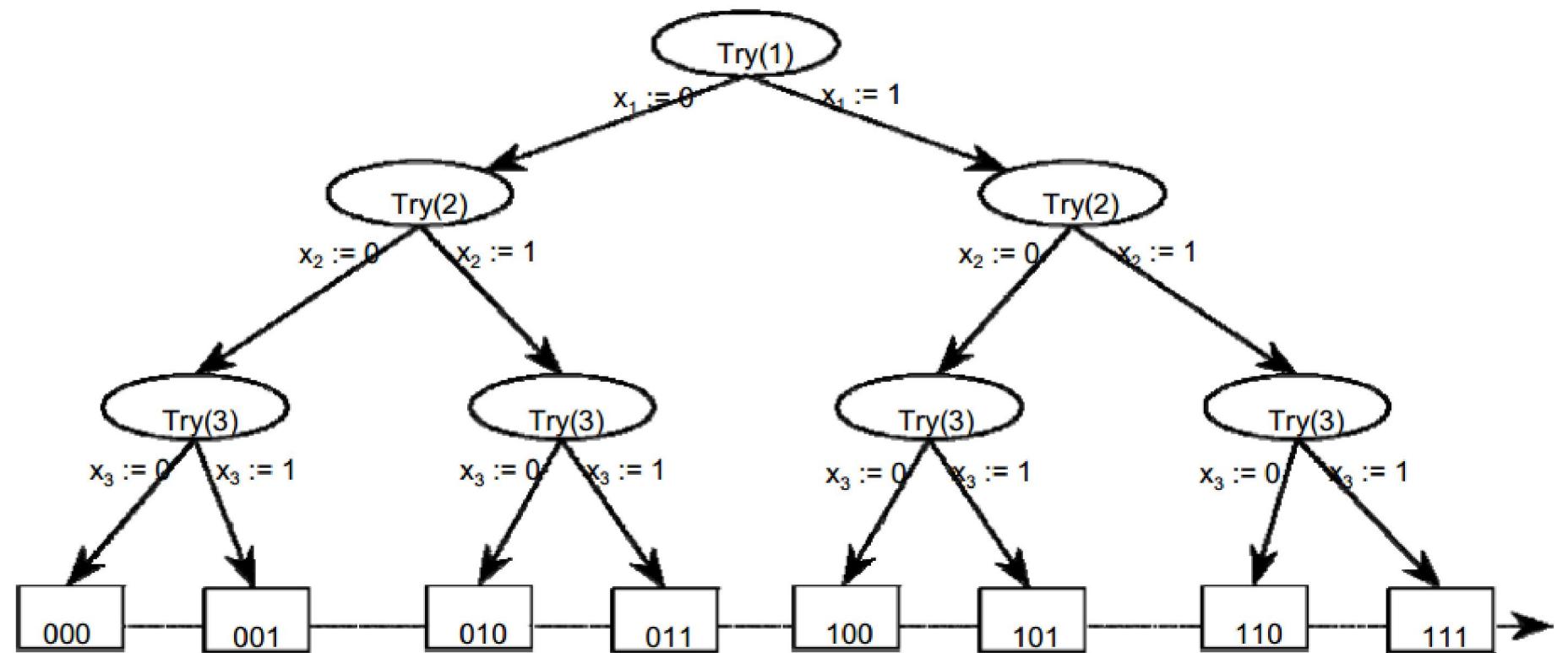
3. Cài đặt

```
Try ( i) ≡  
    for (j = 0; j <= 1; j++)  
    {  
        x[i] = j;  
        if (i < n )  
            Try (i+1);  
        else  
            Xuất(x);  
    }
```

III. Bài toán áp dụng

III. Bài toán liệt kê dãy nhị phân độ dài N

4. Kết quả



IV. Bài tập

1. Hoàn thiện cài đặt bài toán mã đi tuần
2. Giải bài toán cái túi theo giải thuật quay lui.

NỘI DUNG

- I. Giới thiệu
- II. Lược đồ chung
- III. Bài toán áp dụng
- IV. Bài tập