Kỹ thuật lập trình

Programming Techniques

Ts. Nguyễn Đức Thuần
BM Hệ thống Thông Tin

Giới thiệu môn học

Nội dung môn hoc

- Chương 1: Kỹ thuật tổ chức chương trình
- Chương 2: Lập trình có cấu trúc – Hàm nâng cao
- Chương 3: Đệ qui
- Chương 4: Thử sai quay lui – Nhánh cận
- Chương 5:
- Chương 6:
- Chương 7:

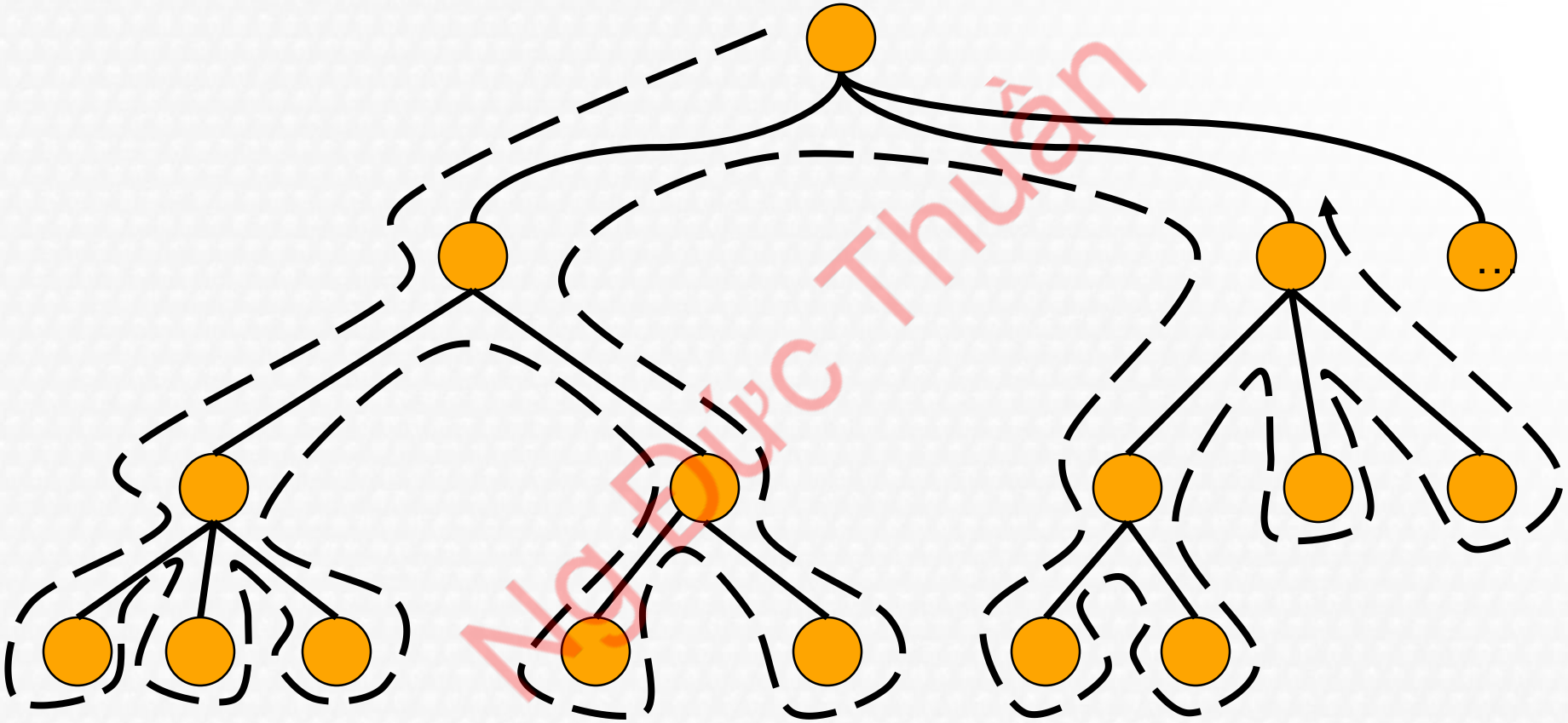
THỬ SAI & QUAY LUI

Chương 4: THỬ SAI & QUAY LUI

I. NỘI DUNG

- Giới thiệu
- Phương pháp
- Sơ đồ cài đặt
- Các ví dụ
- Ưu điểm và khuyết điểm

Chương 4: THỬ SAI & QUAY LUI



Chương 4: THỬ SAI & QUAY LUI

- Định nghĩa [Quay lui – Backtracking]:
  - Quay lui là một phương pháp thiết kế thuật toán để tìm nghiệm của bài toán bằng cách vét cạn (xét tất cả các phương án).
  - Phương pháp quay lui duyệt theo chiều sâu tất cả các nhánh.
  - Trong quá trình xây dựng thành phần thứ i (tìm nghiệm cho thành phần thứ i), nếu không thể xây dựng được thì quay lại chọn nghiệm khác cho thành phần thứ (i-1)

Phát biểu bài toán:

Giả sử nghiệm của bài toán cần tìm có dạng $X=(x_1, x_2, \ldots, x_k, \ldots)$, trong đó

- x_i là 1 thành phần nghiệm của bài toán
- x_i có một miền giá trị D_i nào đó ($x_i \in D_i$).
- Số lượng thành phần x_i có thể xác định hay không xác định
- Bài toán có tập ràng buộc là F

Yêu cầu:

Hãy xây dựng 1 nghiệm hay tất cả các nghiệm của bài toán thỏa điều kiện F

Chương 4: THỬ SAI & QUAY LUI

- Phương pháp Quay lui
  - Phương pháp Quay lui xây dựng dần nghiệm X của bài toán: Bắt đầu từ x_1 được chọn ra từ tập D_1, rồi đến x_2 được chọn ra từ tập D_2, ... bằng cách thử mọi khả năng có thể xảy ra.
  - Một cách tổng quát: Nếu chúng ta đã xác định được lời giải bộ phận gồm (i-1) thành phần $X(i-1) = (x_1, x_2, ..., x_{i-1})$, bây giờ chúng ta tìm giá trị cho thành phần x_i bằng cách xét mọi khả năng có thể có của x_i trong tập D_i. Với mỗi khả năng j ($j \in D_i$), chúng ta kiểm tra xem có thể thỏa điều kiện là nghiệm thành phần của bài toán không

Chương 4: THỬ SAI & QUAY LUI

```
void try(int i)   // tại bước thứ I để tìm giá trị cho x_i
        {         Duyệt   không gian trạng thái D_i  (của x_i)
            {   Ghi nhận 1 trạng thái chấp nhận được
                Nếu  Chưa hết trạng thái  hoặc  chưa trạng thái kết thúc
                        try(i+1);   //    thử bước tiếp
                ngược lại
                        Kiểm_tra_in_nghiệm;
                        Xóa trạng thái đã ghi nhận; // Quay lui
            }
}
```

Chương 4: THỬ SAI & QUAY LUI

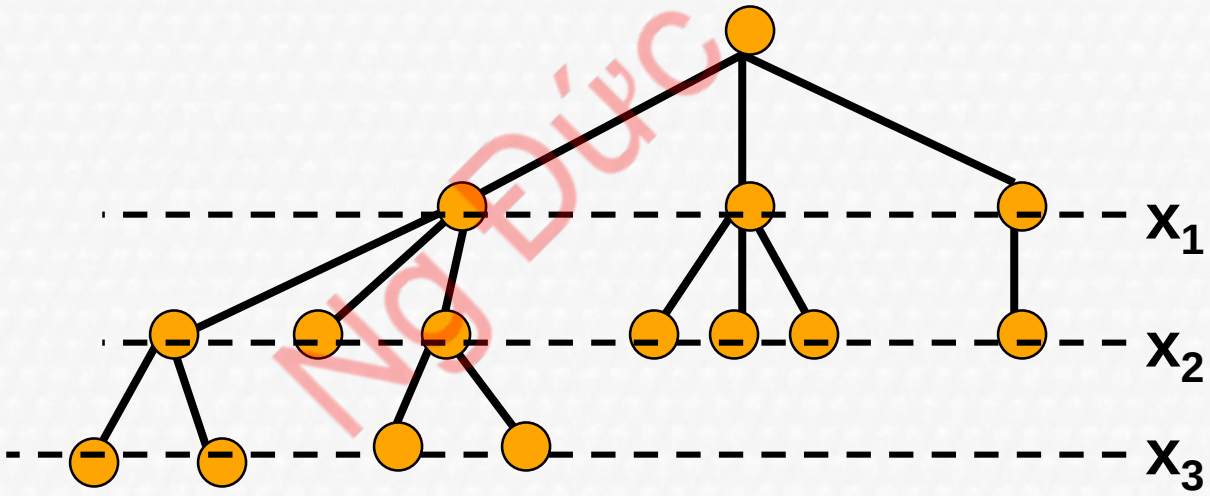
- *Các bước sử dụng phương pháp Quay lui*
  - Bước 1 [Biểu diễn nghiệm]: Biểu diễn nghiệm bài toán dưới dạng một vector

    $$X=(x_1, x_2, x_3, \ldots, x_k, \ldots)$$

  - Bước 2 [Tìm miền giá trị thô]: Xác định các miền giá trị cơ bản D_i cho các x_i ($D_i=[min_i, max_i]$)
  - Bước 3 [Ràng buộc]: Tìm những ràng buộc của x_i và giữa x_i và x_j. Từ đó có thể xác định lại các D_i
  - Bước 4: Xác định những điều kiện F khác để X là nghiệm của bài toán

Chương 4: THỬ SAI & QUAY LUI

- Cây tìm kiếm (Cây không gian tìm kiếm): Quá trình tìm kiếm lời giải theo phương pháp Quay lui sẽ sinh ra 1 cây tìm kiếm



Chương 4: THỬ SAI & QUAY LUI

- Đặc điểm của phương pháp Quay lui
  - Xây dựng dần từng thành phần trong 1 phương án
  - Trong quá trình xây dựng phương án nó thực hiện:
    - Tiến: Để mở rộng các thành phần của phương án
    - Lui: Nếu không thể mở rộng phương án
  - Xét mọi khả năng có thể xảy ra

- Phương pháp quay lui còn được gọi với những tên khác như: Vét cạn (Exhaustion), Duyệt, thử và sai (Trial and Error), …

Ví dụ:

Liệt kê dãy nhị phân độ dài N

- Yêu cầu đề bài: Nhập vào N, in ra dãy nhị phân có độ dài N

```
void try(int i)
int j;
for(j=0 ;j < BINARY; j++){
Data[i]=j; // thử các giá trị của data[i] có thể nhận
if(i == N){
print(<in dãy nhị phân>);
return;
}
try(i+1); //thử tiếp giá trị x[i+1]
}
```

Chương 4: THỬ SAI & QUAY LUI

BÀI TẬP

- **Bài 1:** (Tổng bằng P)

Cho 1 dãy số có n số nguyên và một số P. In ra các số thuộc dãy có tổng bằng P.

Ví dụ: a[]={5,7,10,9,2,12,15}; P=12

Kết quả: 12

10 2

5 7

- **Bài 2:** (Robot)Có 1 Robot mỗi bước có thể bước được 1m hoặc 2m. Hỏi:

Đoạn đường n mét có bao nhiêu cách bước? In ra cách bước của Robot.

Ví du n=4 có các cách bước

1 1 1 1

2 1 1

1 2 1

1 1 2

2 2

Chương 4: THỬ SAI & QUAY LUI

- **Bài 3:** (Hoán vị)
- In ra tất cả hoán vị của dãy số từ 1 đến n
- **Bài 4:** (Cân bàn)

Có n quả cân có trọng lượng lần lượt là $a_1, a_2, a_3,..,a_n$. Hãy trình bày cách bố trí các quả cân lên 2 đĩa cân bàn cho cân thăng bằng.

Ví dụ: các quả cân: 3 5 2 1 1

Một số cách bố trí:

3	2		5	
3	1	1	5	
2	1		3	

- Chú ý: Hai cách bố trí sau xem là khác nhau:

3	2	5	
5		3	2

- **Bài 5:** (Chuột đi du lịch)

Có một con chuột vượt qua một cánh đồng hình vuông kích thước nxn từ cạnh trái nhất sang cạnh phải nhất. Trên cánh đồng những ô mang giá trị 1 chuột có thể đi qua, những ô mang giá trị 0 chuột không thể đi qua. Tại 1 vị trí chuột chỉ có thể di chuyển đến các ô lân cận theo chiều dọc, chiều ngang. In ra các cách đi của chuột.

Ví dụ: Với cánh đồng

1,0,1,1,0

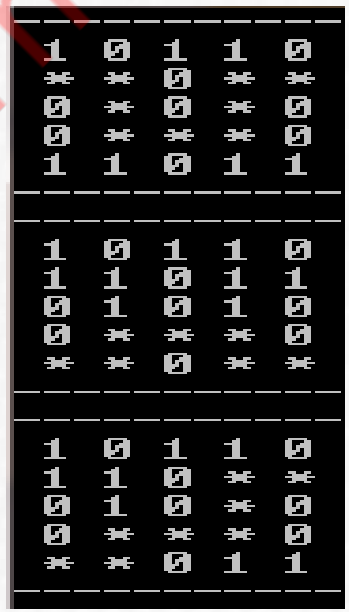
1,1,0,1,1

0,1,0,1,0

0,1,1,1,0

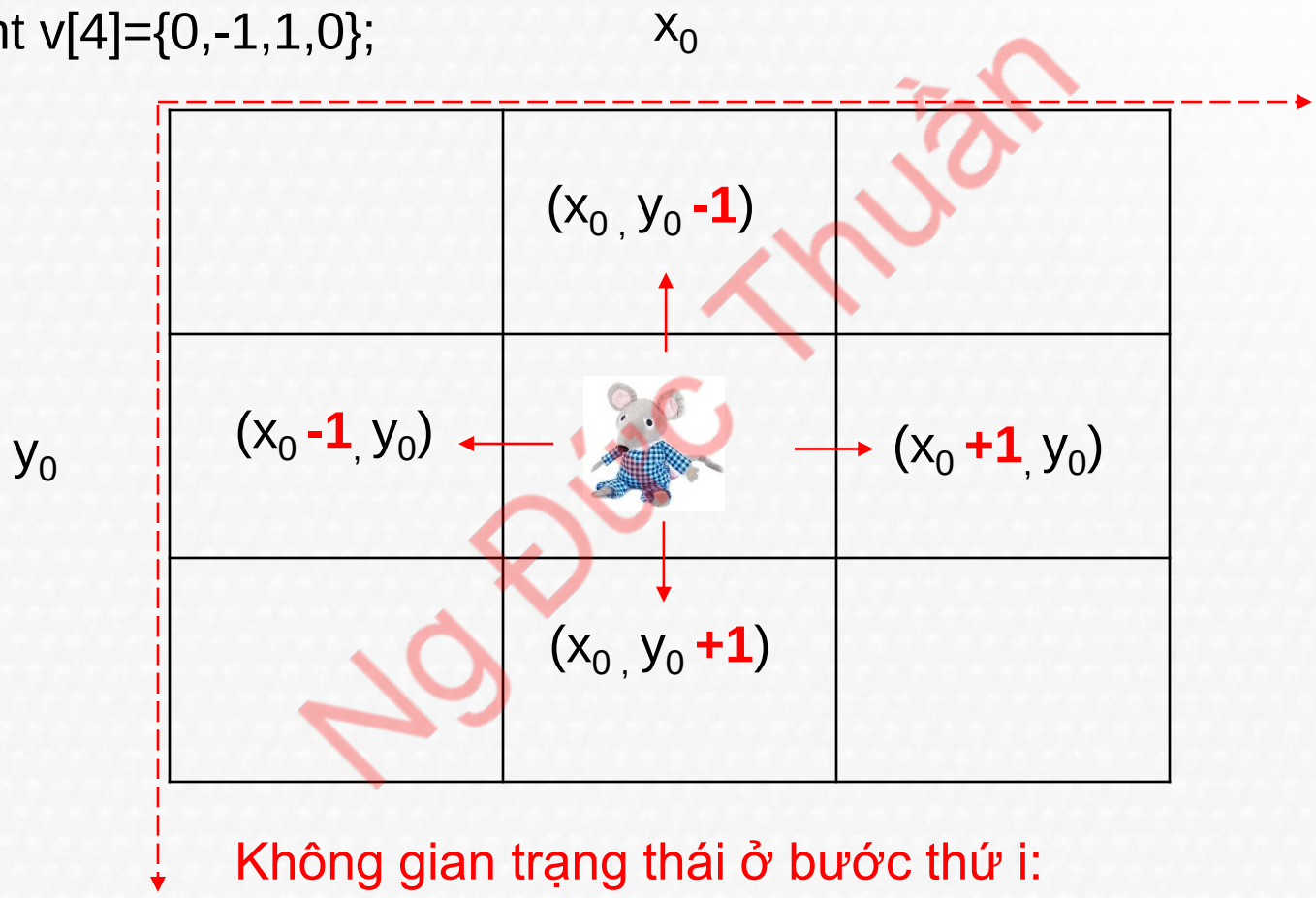
1,1,0,1,1

Chương 4: THỬ SAI & QUAY LUI



Chương 4: THỬ SAI & QUAY LUI

```
int h[4]={1,0,0,-1};
int v[4]={0,-1,1,0};
```



Không gian trạng thái ở bước thứ i:

(x_0 , y_0) + (h[j],v[j]) , j=0,..,3

Chương 4: THỬ SAI & QUAY LUI

- *Duyệt không gian trạng thái ở bước thứ i*:

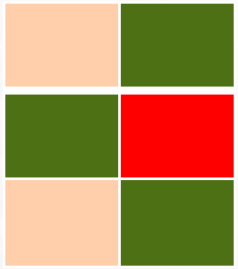
```
for(j=0;j<=3;j++)
{ p=x +h[j];
  q=y +v[j];
     if (mt[p][q]==1) // Nếu ô trạng thái là có thể đến được
          { x=p;y=q; // ghi nhận (đến)
               mt[p][q]=2; // xác đinh là ô đã đến
```

Chương 4: THỬ SAI & QUAY LUI

- Phần tử lính canh (Sentinal):

  Để khỏi kiểm tra đến các ô thuộc biên,

  - do ô ở biên chỉ có thể có 3 cách đi



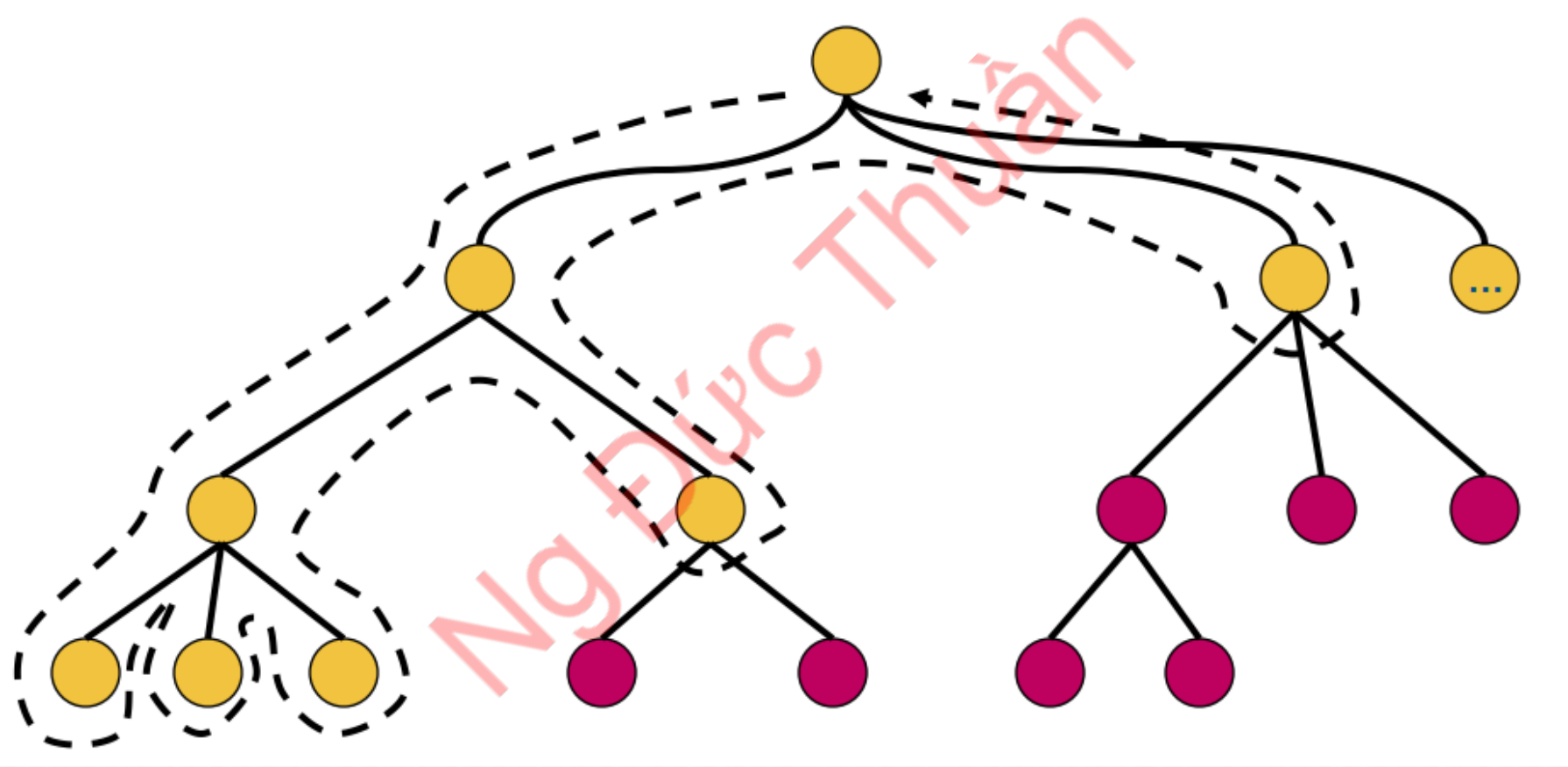
Ta bổ sung thêm các ô mới (các ô cạnh biên) mang giá trị 0, để khỏi kiểm tra các ô này khi xủ lý (các bước đi) gọi là các phần tử lính canh

0	0	0	0
0			0
0			0
0	0	0	0

Chương 4: NHÁNH CẬN

- PHƯƠNG PHÁP NHÁNH CẬN

Chương 4: NHÁNH CẬN



- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)
- 1. Bài toán tối ưu:
- Tìm một phương án thỏa mãn một số điều kiện (ràng buộc), phương án đó là tốt nhất theo một tiêu chí yêu cầu.

*Cho một hàm mục tiêu **f** hay còn được gọi là hàm đánh giá và các hàm ràng buộc logic $g_1, g_2, \ldots g_n$. Yêu cầu tìm một cấu hình **x** thỏa mãn tất cả các ràng buộc $g_1, g_2, \ldots g_n$ và **f(x)** là tốt nhất. Nghĩa là không tồn tại một cấu hình y thỏa mãn các điều kiện mà f(y) tốt hơn f(x).*

- Ví dụ: Tìm (x,y) để x+y=max và thỏa mãn điều kiện $x^2+y^2 \le 1$

- Như vậy hàm mục tiêu f ở đây là x+y -> max

- Hàm ràng buộc g là $x^2+y^2 \le 1$

$$X^*=(x,y)^* = \arg\max_{x^2+y^2 \le 1} (x+y)$$

- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)
- Thường hàm mục tiêu là hàm max() hoặc hàm min(), nếu ký hiệu điệu kiện ràng buộc là g(X) thì bài toán tối ưu có các dạng:

$$X^* = \arg\max_{g(X)} f(X) \quad \text{hoặc} \quad X^* = \arg\min_{g(X)} f(X)$$

Có nhiều bài toán tối ưu không có thuật toán nào thực sự hữu hiệu để giải quyết. Một giải pháp tìm nghiệm tối ưu là xét toàn bộ các phương án, rồi đánh giá để chọn phương án tốt nhất. (*Phương pháp vét cạn*)

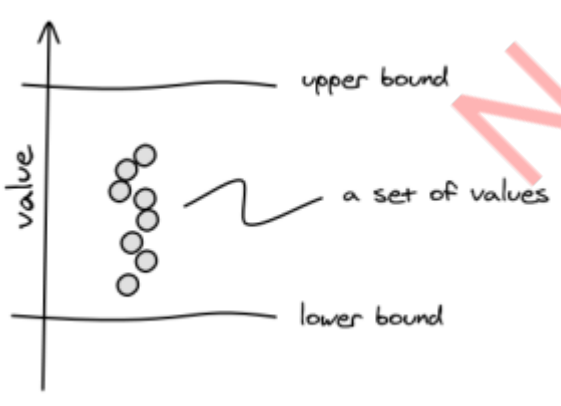
Phương pháp nhánh cận là 1 dạng cải tiến của phương pháp quay lui để tìm nghiệm tối ưu.

Chương 4: NHÁNH CẬN

- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)
  - Ý tưởng phương pháp nhánh cận

Giả sử đã xây dựng được k thành phần (x_1,x_2,..x_k) của nghiệm và khi mở rộng nghiệm (x_1,x_2,..x_k ,x_{k+1}), nếu biết rằng các thành phần mở rộng của nó đều không làm hàm mục tiêu tối ưu hơn thì không cần mở rộng nữa.

- Việc bỏ đi các phương án (nhánh) không cần thiết làm cho việc tìm nghiệm tối ưu sẽ nhanh hơn.
- Cận của bài toán ?



Cận trên *là một giá trị lớn hơn hoặc bằng giá trị lớn nhất trong một tập hợp.*

Cận dưới *là giá trị nhỏ hơn hoặc bằng giá trị nhỏ nhất trong một tập hợp.*

Chương 4: NHÁNH CẬN

- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)
- Cận: giá trị hàm mục tiêu ứng với giải pháp cục bộ (cây con) bắt nguồn từ nút đang xem xét.
- Cận trên là giá trị lớn hơn hoặc bằng giá trị lớn nhất và Cận dưới là giá trị bé hơn hoặc bằng so với nhỏ nhất so với các giá trị trong tập hợp.

Nhận xét: Việc đánh các cận để xây dựng các thành phần mở rộng là vấn đề cốt lõi của phương pháp.

- Ví dụ: Cận trên và cận dưới lương hàng năm của cầu thủ bóng đá chuyên nghiệp được trả lương cao nhất?

Chương 4: NHÁNH CẬN

- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)
- Ví dụ: Cho một tập hợp X các số nguyên dương, chia tập hợp đã cho thành hai tập hợp con sao cho sự khác biệt giữa các tổng của các phần tử của chúng là bé nhất?

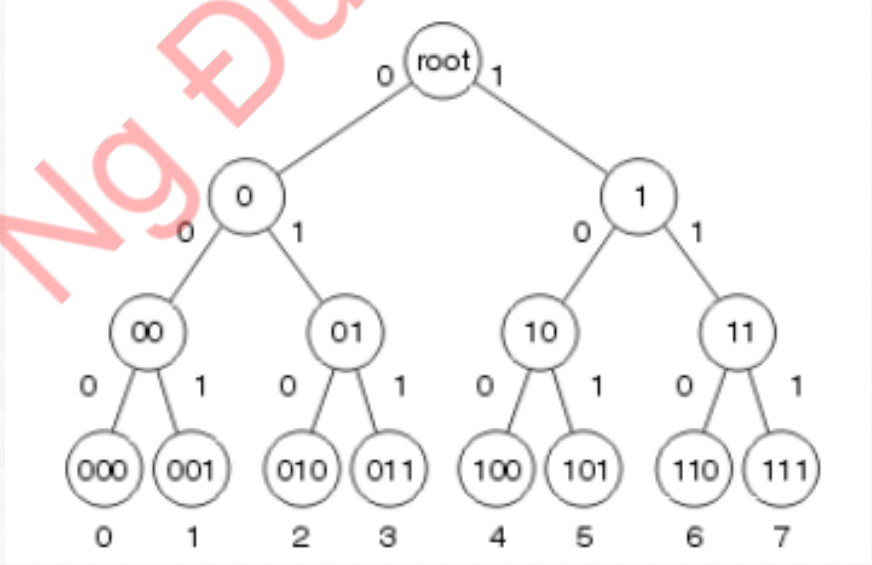
Xét $X = \{11; 7; 6; 4; 9; 14\}$ $\Sigma=51$

- *Làm sao tính được cận dưới ?*

- Nghiệm sau đây có phải là nghiệm tối ưu?

{11; 14}; {7; 6; 4; 9}

- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)
- Phương pháp Nhánh cận phân vùng không gian giải pháp thành các tập hợp con
- Thông thường các tập hợp con giải pháp tạo thành một cấu trúc cây
- Nút Lá trong cây là giải pháp. Các nút trong là giải pháp cục bộ
- Các giải pháp cục bộ cho phép lập luận tạo giải pháp cục bộ mở rộng hơn trong không gian tìm kiếm.



Chương 4: NHÁNH CẬN

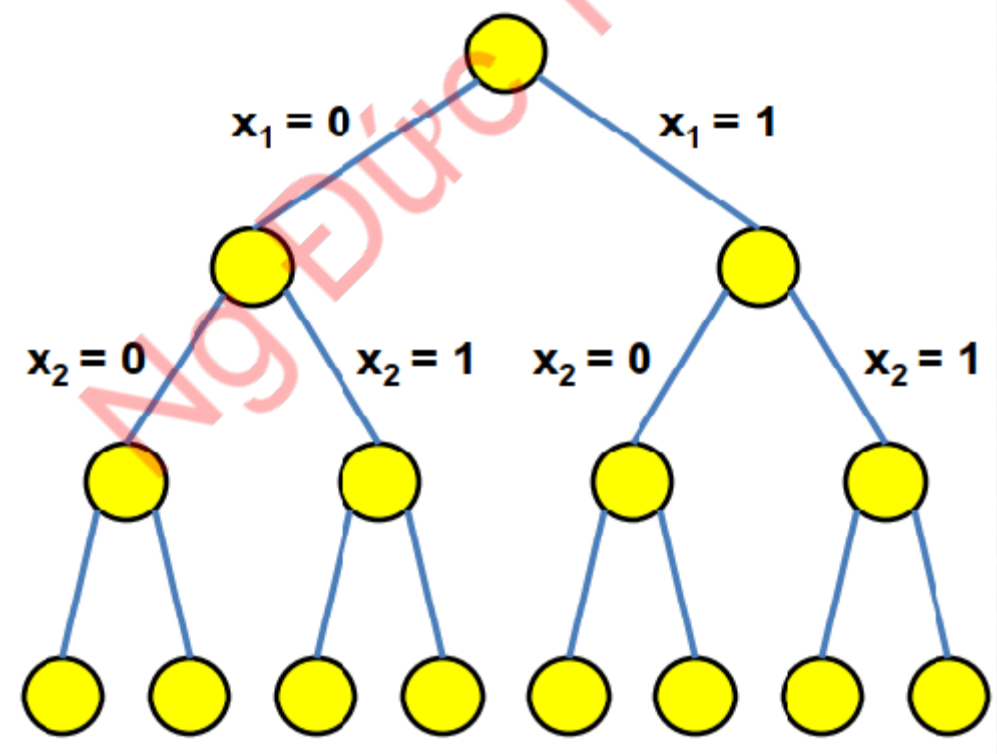
- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)
- Thuật toán nhánh cận có thể biểu diễn bằng mô hình đệ qui

```
Void thu(i)
{  Duyệt không gian trang thái cho bước i
     {   Ghi nhận trạng thái chấp nhận được;
         if (còn có thể ghi nhận giải pháp (mở rộng nghiệm) tốt hơn)
            {  if (giải pháp thỏa yêu cầu)  In nghiệm;
               else thu(i+1);
                 Hủy trạng thái đã ghi nhận;}
             }
            }
```

- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)
- Ví dụ: Tìm giá trị lớn nhất của biểu thức:

$$\max \leftarrow 15x_1+12x_2+4x_3+2x_4$$

Với $8x_1+5x_2+3x_3+2x_4 \leq 10$, $x_k=0,1$ với $k=1,2,..,4$



- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)
- MỘT SỐ VÍ DỤ MINH HỌA

1. Cho 1 dãy số nguyên $a_1, a_2, .., a_n$ và 1 số nguyên P. Hãy tìm tất cả các dãy con có tổng các phần tử bằng P.

Cách dùng nhánh cận

```
void thu(int i,int Tong)
 {int j;
   for (j=0;j<=1;j++)
    { b[i]=j; if (j==1) Tong=Tong+a[i];
      if ((i<n-1) && (Tong <P)) thu(i+1,Tong);
      else
      inketqua();
                    b[i]=0;
   if (j==1)Tong=Tong-a[i];
                    }
                    }
```

Chương 4: NHÁNH CẬN

- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)

2. Có 1 máy đổi tiền có thể nhận biết và đổi được các đồng tiền thành tổng các đồng tiền nhỏ hơn trong số các đồng tiền K_1, K_2, ..K_n ($K_1 < K_2 < .. < K_n$). Cho 1 đồng tiền có giá trị M liệt kê tất cả các cách đổi đồng tiền này.

```
void thu(int i,int sotien)
{ int k,l;
    l=sotien/a[i];
  for(k=0;k<=l;k++)
   { b[i]=k;
     if (((sotien-k*a[i])>0) && (i<n-1))       thu(i+1,sotien-k*a[i]);
       else
       Inkq();   b[i]=0;
  } }
```

- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)

3. (*n chàng trai & m cô gái*) Cho n chàng trai và m cô gái, mỗi chàng trai có mức độ tình cảm khác nhau cho mỗi cô gái. Hãy đưa ra cách kết các đôi sao cho mức độ tình cảm của các cặp cao nhất có thể được.

- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)

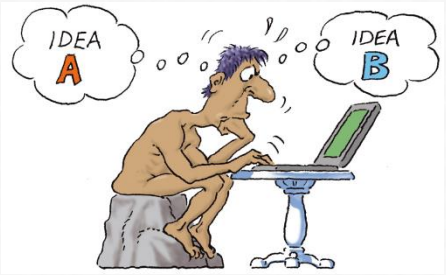
4. Bài toán cái túi: (Knapsack)

Cho trọng lượng và giá trị của n vật. Hãy chọn từ n các vật này các vật thích hợp để xếp vào một chiếc ba lô có thể tích W, sao cho tổng giá trị các vật được xếp vào ba lô là lớn nhất. Nói cách khác, đã cho hai mảng số nguyên val [0..n-1] và wt [0..n-1] đại diện cho các giá trị và trọng số tương ứng với n vật. Một số nguyên W đại diện cho dung lượng của ba lô, hãy tìm ra các mục sao cho tổng trọng số lớn nhất của các mục con được chọn nhỏ hơn hoặc bằng W.

https://www.geeksforgeeks.org/implementation-of-0-1-knapsack-using-branch-and-bound/

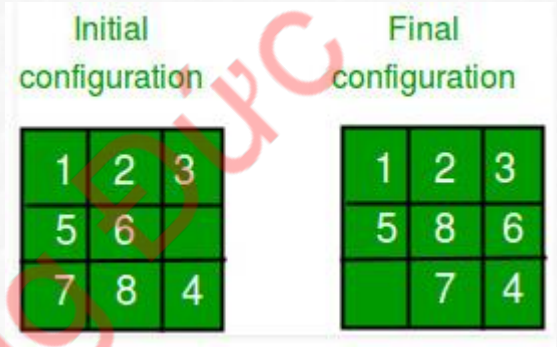
Chú ý: Bài toán này có thể giải bằng nhiều cách,

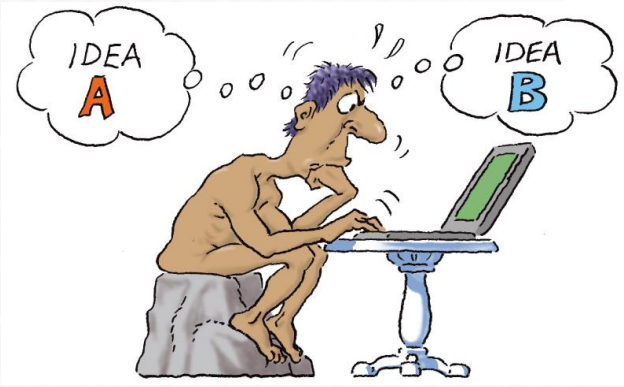
Yêu cầu tìm hiểu cách giải nhánh cận.



- PHƯƠNG PHÁP NHÁNH CẬN (Branch and Bound)

4. Bài toán ô số Puzzle: Cho một bảng 3 × 3 với 8 ô (mỗi ô có một số từ 1 đến 8) và một khoảng trống. Mục tiêu là di chuyển các số trên các ô để khớp với cấu hình cuối cùng (sử dụng khoảng trống để dịch chuyển vị trí) . Chúng ta có thể trượt 1 số đến bốn ô liền kề (trái, phải, trên và dưới) vào khoảng trống.





https://www.geeksforgeeks.org/8-puzzle-problem-using-branch-and-bound/

Yêu cầu: Sử dụng phương pháp nhánh cận.

Cám ơn đã theo dõi