TRƯỜNG CAO ĐẲNG CNTT HỮU NGHỊ ViỆT - HÀN

KHOA KHOA HỌC MÁY TÍNH

----------***----------

THUẬT TOÁN

(Algorithms)

Nguyễn Thanh Cẩm

Nội Dung

C1 THUẬT TOÁN VÀ ĐỘ PHỨC TẠP

C2 CHIA ĐỂ TRỊ

C3 QUY HOẠCH ĐỘNG

C4 THUẬT TOÁN THAM LAM

C5 THUẬT TOÁN QUAY LUI

THUẬT TOÁN QUAY LUI

5.1 Thuật toán quay lui

5.2 Một số bài toán minh họa

THUẬT TOÁN QUAY LUI

Thuật toán quay lui

Đệ quy

Thuật toán quay lui tổng quát

5.1 Thuật toán quay lui

- **Quay lui** (*backtracking*) là một chiến lược tìm kiếm lời giải cho các bài toán thỏa mãn ràng buộc.
- Người đầu tiên đề ra thuật ngữ này (*backtrack*) là nhà toán học người Mỹ D. H. Lehmer vào những năm 1950.

5.1.1 Đệ quy

- Thí dụ 1: Tìm thuật toán đệ quy tính giá trị a^n với a là số thực không âm và n là số nguyên không âm.

- Thuật toán đệ quy tính a^n.

```
float power (a: float; n: int);
{
        if (n = 0) power(a, n) = 1
        else power(a, n) = a*power(a, n-1)
}
```

5.1.1 Đệ quy

- Thí dụ 2: Tìm thuật toán đệ quy tính UCLN của hai số nguyên a, b không âm và a < b.

- Thuật toán đệ quy tính UCLN(a, b)

```
int UCLN(int a, int b);
{
        if (a = 0) UCLN(a, b) = b
        else UCLN(a,b) = UCLN(b mod a,a)
 }
```

4.1.2 Thuật toán quay lui tổng quát

- Thuật toán quay lui dùng để giải bài toán liệt kê các cấu hình.
- Mỗi cấu hình được xây dựng bằng cách xây dựng từng phần tử,
- mỗi phần tử được chọn bằng cách thử tất cả các khả năng.

4.1.2 Thuật toán quay lui tổng quát

- Giả thiết cấu hình cần liệt kê có dạng (x_1, x_2, …, x_n). Khi đó thuật toán quay lui thực hiện các bước sau:
  - **Xét tất cả các giá trị x_1 có thể nhận, thử cho x_1 nhận lần lượt các giá trị đó. Với mỗi giá trị thử gán cho x_1 ta sẽ:**
  - **Xét tất cả các giá trị x_2 có thể nhận, lại thử cho x_2 nhận lần lượt các giá trị đó. Với mỗi giá trị thử gán cho x2 lại xét tiếp các khả năng chọn x3… cứ tiếp tục như thế đến bước:**
  - **Xét tất cả các giá trị x_n có thể nhận, thử cho x_n nhận lần lượt các giá trị đó, thông báo cấu hình tìm được (x_1, x_2, …, x_n).**
  - **Trên phương diện quy nạp, có thể nói rằng thuật toán quay lui liệt kê các cấu hình n phần tử dạng (x_1, x_2, …, x_n) bằng cách thử cho x_1 nhận lần lượt các giá trị có thể. Với mỗi giá trị gán cho x_1 lại liệt kê tiếp cấu hình n -1 phần tử (x_2, x_3, …, x_n).**

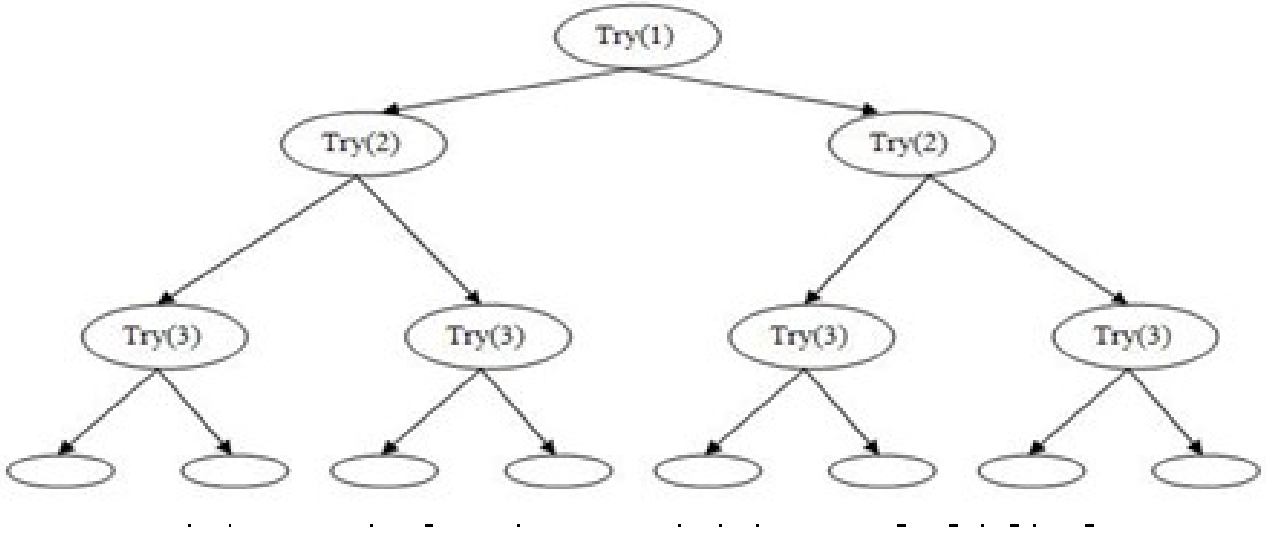
4.1.2 Thuật toán quay lui tổng quát

❖ Mô hình của thuật toán quay lui có thể mô tả như sau:

```
Void Try(int i)
{   For (mọi giá trị V có thể gán cho x_i)
      {     <thử cho x_i = V>;
             If (x_i là phần tử cuối cùng trong cấu hình)
             <Thông báo cấu hình tìm được>
             Else
            { <Ghi nhận cho x_i nhận giá trị V (nếu cần)>;
             Try(i + 1);
          <Nếu cần, bỏ ghi nhận việc thử x_i = V, để thử giá trị khác>;
           }
    }
}
```

5.1.2 Thuật toán quay lui tổng quát

- Ta có thể trình bày quá trình tìm kiếm lời giải của thuật toán quay lui bằng cây sau:



THUẬT TOÁN QUAY LUI

5.2 Một số bài toán minh họa

- 5.2.1 *Bài toán liệt kê dãy nhị phân độ dài n*
- 5.2.2 *Bài toán liệt kê các tập con k phần tử*
- 5.2.3 *Bài toán xếp hậu*
- 5.2.4 *Bài toán tô màu đồ thị*

5.2.1 Bài toán liệt kê dãy nhị phân độ dài n

- Biểu diễn nhị phân độ dài N dưới dạng $(x_1, x_2, \ldots, x_n)$.
- Ta sẽ liệt kê các dãy này bằng cách thử dùng các giá trị (0, 1) gán cho x_i.
- Với mỗi giá trị thử gán cho x_i lại thử các giá trị có thể gán cho x_{i+1}.
- Chương trình liệt kê bằng thuật toán quay lui có thể viết như dưới đây:

5.2.1 Bài toán liệt kê dãy nhị phân độ dài n

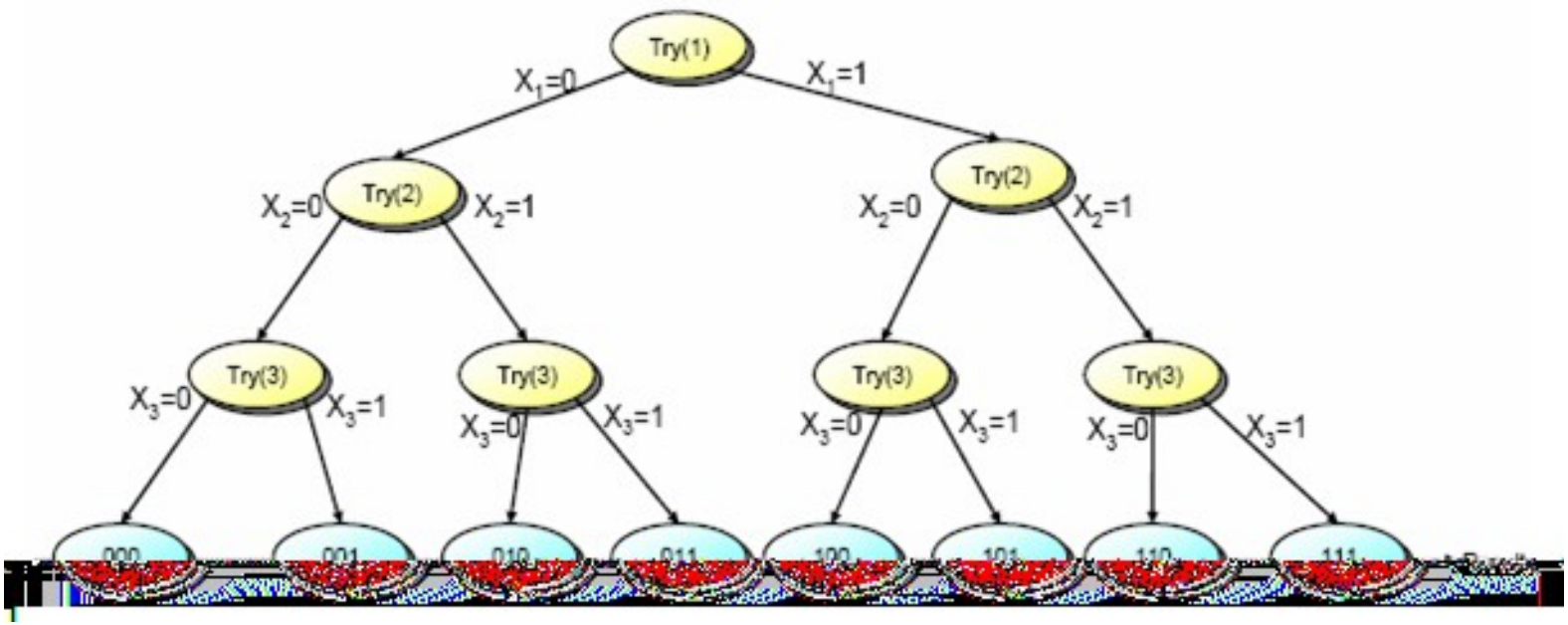
❖ *Thuật toán liệt kê các phần tử nhị phân*

```
void      Try(int i)
{ int j;
    For (j = 0;j<= 1;j++)
   <Xét các giá trị có thể gán chi x_i  với mỗi giá trị đó>
  {
            x[i] = j;
            if (i = n) Binary_result
            else  Try(i+1)
     }
}
```

5.2.1 Bài toán liệt kê dãy nhị phân độ dài n

❖ Mô tả bằng cây:

❖ Ví dụ: khi n = 3, cây tìm kiếm quay lui như sau:



THUẬT TOÁN QUAY LUI

5.2 Một số bài toán minh họa

- 5.2.1 *Bài toán liệt kê dãy nhị phân độ dài n*
- 5.2.2 *Bài toán liệt kê các tập con k phần tử*
- 5.2.3 *Bài toán xếp hậu*
- 5.2.4 *Bài toán tô màu đồ thị*

5.2.2 Bài toán liệt kê các tập con k phần tử

- Để liệt kê các tập con k phần tử của tập S = {1, 2,… , n}, ta có thể đưa về liệt kê các cấu hình (x_1, x_2, …, x_k).
- Ở đây các và $x_1 < x_2 < \ldots < x_k$. Ta có nhận xét:

$x_k \le n$

$x_{k-1} \le x_k - 1 \le n - 1$

…

$x_i \le n - k + i$

…

$x_1 \le n - k + 1$.

- Từ đó suy ra $x_{i-1} + 1 \le x_i \le n - k + i$ $(1 \le i \le k)$, ở đây ta giả thiết có thêm một số $x_0 = 0$ khi xét i = 1.

5.2.2 Bài toán liệt kê các tập con k phần tử

- Như vậy ta sẽ xét tất cả các cách:
  - **Chọn x_1 từ 1(= x_0 +1) đến n-k+1, với mỗi giá trị đó, xét tiếp tất cả các cách**
  - **Chọn x_2 từ x_1+1 đến n-k+2,…**
  - **Cứ như vậy khi chọn được đến x_k thì ta có một cấu hình cần liệt kê.**

5.2.2 Bài toán liệt kê các tập con k phần tử

❖ Thuật toán quay lui liệt kê các tập con k phần tử:

❖ void Try(int i)

```
{ int  j;
    For (j = x[i-1] + 1; j<= n-k+i; j++)
   {
       x[i] = j ;
       If (i == k)  Printresult
       Else Try(i+1);
    }
}
```

THUẬT TOÁN QUAY LUI

5.2 Một số bài toán minh họa

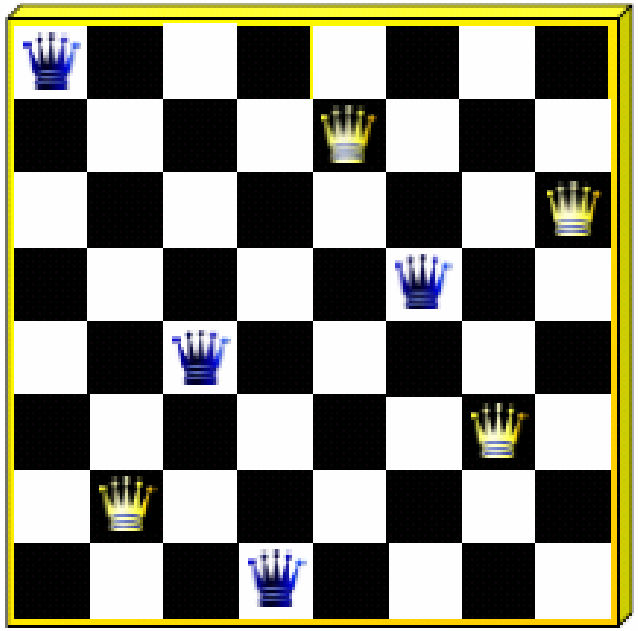
- 5.2.1 *Bài toán liệt kê dãy nhị phân độ dài n*
- 5.2.2 *Bài toán liệt kê các tập con k phần tử*
- 5.2.3 *Bài toán xếp hậu*
- 5.2.4 *Bài toán tô màu đồ thị*

5.2.3 Bài toán xếp hậu

- **Bài toán**
- Xét bàn cờ tổng quát kích thước n x n.
- Một quân hậu trên bàn cờ có thể ăn được các quân khác nằm tại các ô cùng hàng, cùng cột hoặc cùng đường chéo.
- Hãy tìm cách xếp n quân hậu trên bàn cờ sao cho không quân nào ăn quân nào.

5.2.3 Bài toán xếp hậu

- Ví dụ một cách xếp với n = 8:



5.2.3 Bài toán xếp hậu

❖ Phân tích

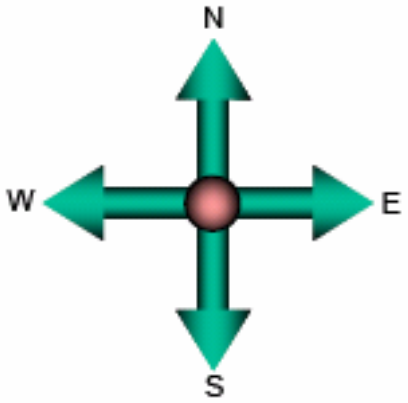
- ❖ Rõ ràng n quân hậu sẽ được đặt mỗi con một hàng, vì hậu ăn được ngang.
- ❖ Ta gọi quân hậu sẽ đặt ở hàng 1 là quân hậu 1,
- ❖ quân hậu ở hàng 2 là quân hậu 2…
- ❖ quân hậu ở hàng n là quân hậu n.
- ❖ Vậy một nghiệm của bài toán sẽ được biết khi ta tìm ra được vị trí cột của những quân hậu.
- ❖ Nếu ta định hướng Đông (Phải), Tây (Trái), Nam (Dưới), Bắc (Trên) thì ta nhận thấy rằng:

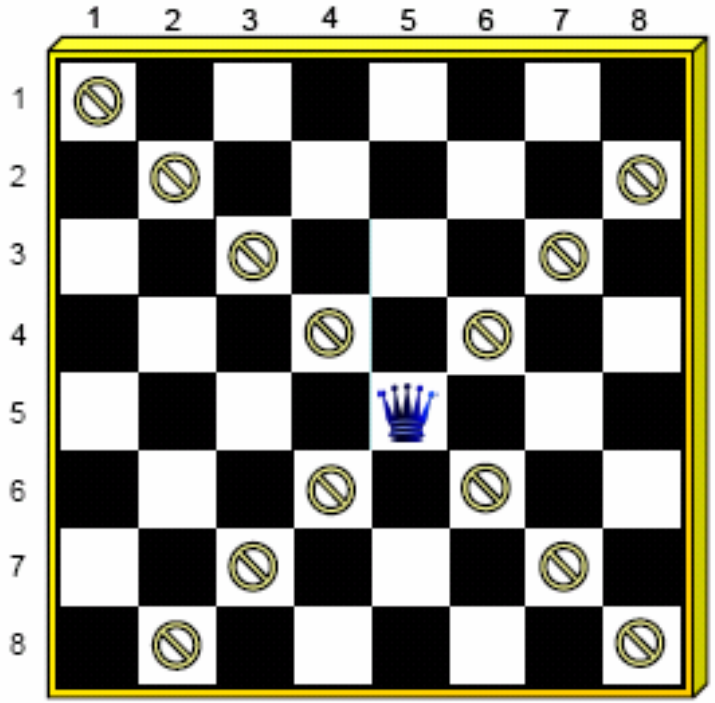
5.2.3 Bài toán xếp hậu

- Một đường chéo Đông Bắc - Tây Nam (ĐB-TN) bất kỳ sẽ đi qua một số ô, các ô đó có tính chất: Hàng + Cột = C (Const).
- Với mỗi đường chéo ĐB-TN ta có 1 hằng số C và với một hằng số C: 2 <= C <= 2n xác định duy nhất 1 đường chéo ĐB-TN vì vậy ta có thể đánh chỉ số cho các đường chéo ĐB- TN từ 2 đến 2n
- Một đường chéo Đông Nam - Tây Bắc (ĐN-TB) bất kỳ sẽ đi qua một số ô, các ô đó có tính chất: Hàng - Cột = C (Const).
- Với mỗi đường chéo ĐN-TB ta có 1 hằng số C và với một hằng số C: 1 - n <= C <= n - 1 xác định duy nhất 1 đường chéo ĐN-TB vì vậy ta có thể đánh chỉ số cho các đường chéo ĐN- TB từ 1 - n đến n - 1.

5.2.3 Bài toán xếp hậu

- Đường chéo ĐB-TN mang chỉ số 10 và đường chéo ĐN- TB mang chỉ số 0





5.2.3 Bài toán xếp hậu

❖ Cài đặt

- ❖ Ta có 3 mảng logic để đánh dấu:
- ❖ Mảng a[1..n]. a_i = TRUE nếu như cột i còn tự do, ai = FALSE nếu như cột i đã bị một quân hậu khống chế
- ❖ Mảng b[2..2n]. b_i = TRUE nếu như đường chéo ĐB-TN thứ i còn tự do, bi = FALSE nếu như đường chéo đó đã bị một quân hậu khống chế.
- ❖ Mảng c[1 - n..n - 1]. c_i = TRUE nếu như đường chéo ĐN-TB thứ i còn tự do, ci = FALSE nếu như đường chéo đó đã bị một quân hậu khống chế.
- ❖ Ban đầu cả 3 mảng đánh dấu đều mang giá trị TRUE. (Các cột và đường chéo đều tự do)

5.2.3 Bài toán xếp hậu

- Áp dụng thuật toán quay lui cho bài toán xếp hậu:
- Xét tất cả các cột, thử đặt quân hậu 1 vào một cột,
- với mỗi cách đặt như vậy, xét tất cả các cách đặt quân hậu 2 không bị quân hậu 1 ăn, lại thử 1 cách đặt và xét tiếp các cách đặt quân hậu 3…
- Mỗi cách đặt được đến quân hậu n cho ta 1 nghiệm
- Khi chọn vị trí cột j cho quân hậu thứ i, thì ta phải chọn ô(i, j) không bị các quân hậu đặt trước đó ăn, tức là phải chọn cột j còn tự do, đường chéo ĐB-TN (i+j) còn tự do, đường chéo ĐN-TB(i-j) còn tự do.
- Điều này có thể kiểm tra ($a_j = b_{i+j} = c_{i-j} = TRUE$).
- Khi thử đặt được quân hậu thứ i vào cột j, nếu đó là quân hậu cuối cùng (i = n) thì ta có một nghiệm. Nếu không:

5.2.3 Bài toán xếp hậu

- Trước khi gọi đệ quy tìm cách đặt quân hậu thứ i + 1, ta đánh dấu cột và 2 đường chéo bị quân hậu vừa đặt khống chế ($a_j = b_{i+j} = c_{i-j}$:= FALSE) để các lần gọi đệ quy tiếp sau chọn cách đặt các quân hậu kế tiếp sẽ không chọn vào những ô nằm trên cột j và những đường chéo này nữa.
- Sau khi gọi đệ quy tìm cách đặt quân hậu thứ i + 1, có nghĩa là sắp tới ta lại thử một cách đặt khác cho quân hậu thứ i, ta bỏ đánh dấu cột và 2 đường chéo bị quân hậu vừa thử đặt khống chế ($a_j = b_{i+j} = c_{i-j}$:= TRUE) tức là cột và 2 đường chéo đó lại thành tự do, bởi khi đã đặt quân hậu i sang vị trí khác rồi thì cột và 2 đường chéo đó hoàn toàn có thể gán cho một quân hậu khác

5.2.3 Bài toán xếp hậu

❖ **Thủ tục đặt hậu:**

```
void Try(int i)
    {int j;
      for (j = 1;j<= n;j++)
     if (a[j] and b[i + j] and c[i - j])
       {  x[i] = j;
          if (i == n) PrintResult
           else
       {    a[j] = False; b[i + j] = False; c[i - j] = False;
            Try(i + 1);
            a[j] = True; b[i + j] = True; c[i - j] = True;
        }
   }
}
```

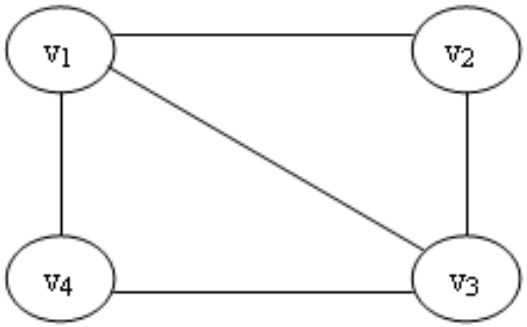
THUẬT TOÁN QUAY LUI

5.2 Một số bài toán minh họa

- 5.2.1 *Bài toán liệt kê dãy nhị phân độ dài n*
- 5.2.2 *Bài toán liệt kê các tập con k phần tử*
- 5.2.3 *Bài toán xếp hậu*
- 5.2.4 *Bài toán tô màu đồ thị*

5.2.3 Bài toán tô màu đồ thị

- Bài toán m màu đồ thị là bài toán tìm tất cả những cách để tô màu đồ thị vô hướng, sử dụng nhiều nhất m màu khác nhau, sao cho không có hai đỉnh kề cùng màu.
- Thí dụ: đồ thị sau cần 3 màu (nhưng 2 màu thì không đủ).

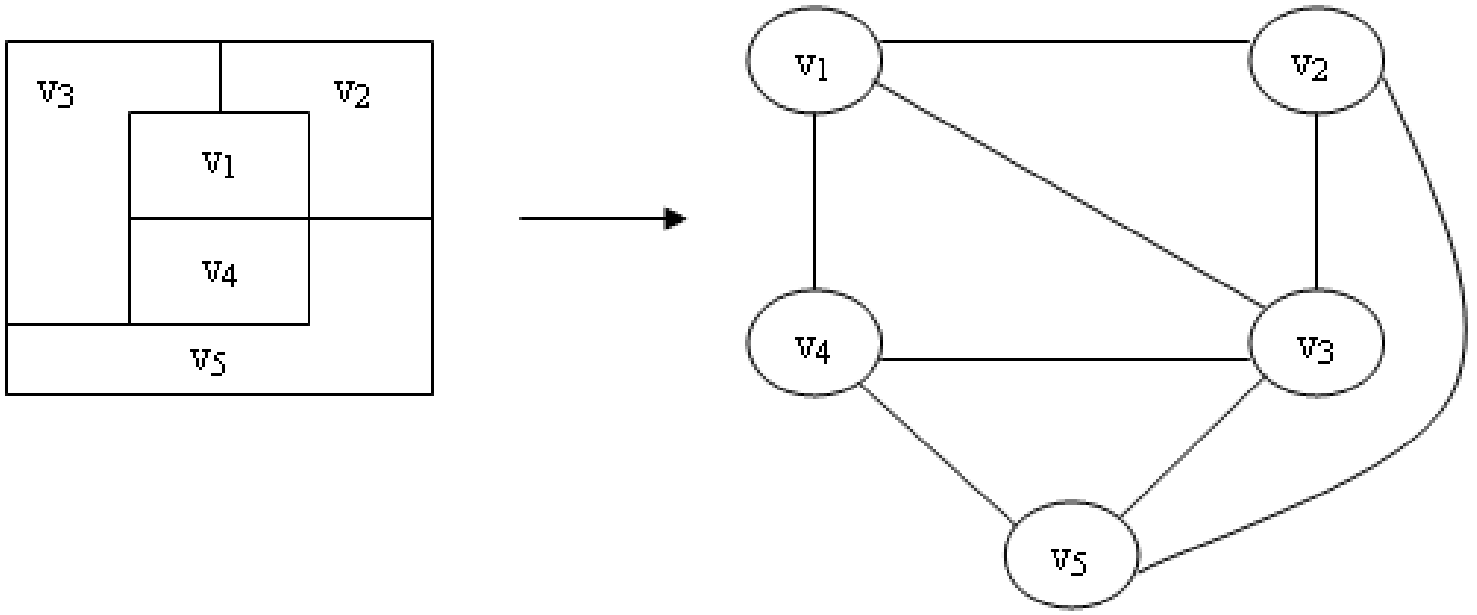


5.2.3 Bài toán tô màu đồ thị

- Một ứng dụng quan trọng của tô màu đồ thị là tô màu bản đồ.
- Một đồ thị được gọi là đồ thị phẳng nếu đồ thị đó nằm trong mặt phẳng và không có các cạnh giao nhau.
- Để mỗi bản đồ tương ứng với một đồ thị phẳng, mỗi vùng trong bản đồ được mô tả bởi một đỉnh của đồ thị.
- Hai vùng kề nhau trong bản đồ được mô tả bởi một cạnh nối hai đỉnh.

5.2.3 Bài toán tô màu đồ thị

- Thí dụ mô tả bản đồ bởi đồ thị

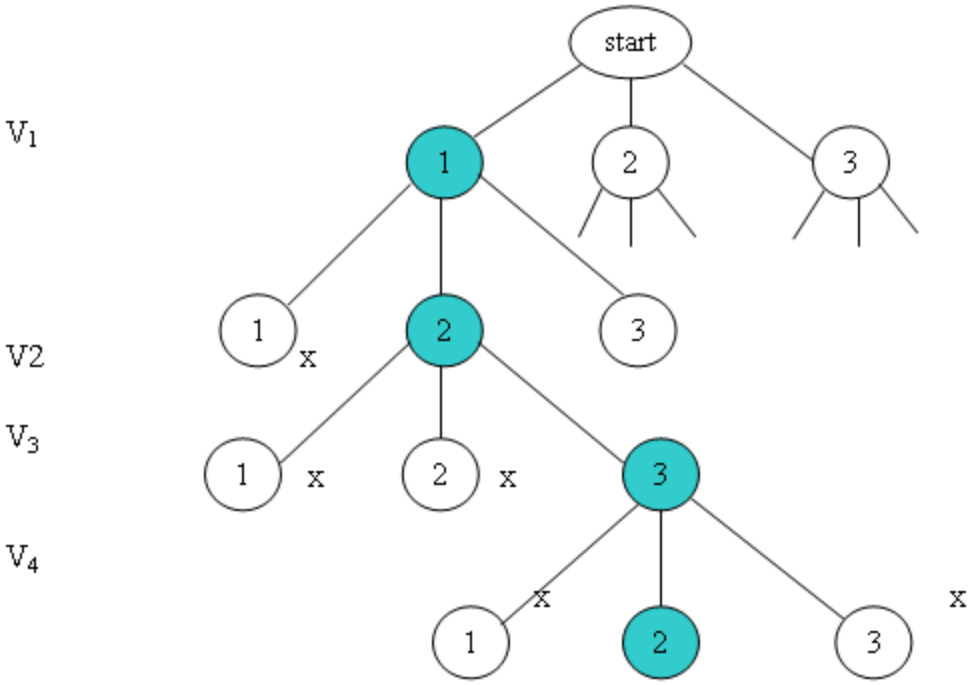


5.2.3 Bài toán tô màu đồ thị

- Bài toán tô màu cho đồ thị phẳng là để xác định có thể tô màu bản đồ sử dụng nhiều nhất m màu sao cho không có hai vùng kề nhau cùng một màu.
- Một cây không gian trạng thái dùng cho bài toán m màu, là mỗi một màu thử cho đỉnh v_1 ở mức một, màu cho đỉnh v_2 ở mức 2… màu cho đỉnh v_n là mức n.
- mỗi đường dẫn từ gốc đến lá một đáp án.
- Chúng ta có hay không một đáp án là xác định lời giải sao cho không có hai đỉnh kề cùng màu. Để tránh nhầm lẫn nhớ mô tả các nút trong cây trong cây trạng thái tương ứng với một đỉnh v của đồ thị tô màu.

5.2.3 Bài toán tô màu đồ thị

- Ví dụ bài toán 3 màu được mô tả



5.2.3 Bài toán tô màu đồ thị

- **Bài toán**
- Xác định tất cả những cách tô màu các đỉnh của đồ thị vô hướng, đúng chỉ m màu sao cho không có hai đỉnh kề cùng màu.
- **Input**:
- Cho n, m là hai số nguyên. Một đồ thị vô hướng chứa n đỉnh, đồ thị được mô tả bởi ma trận A hai chiều, các dòng và cột có chỉ số từ 1 đến n. Ở đây A[i][j] là đúng nếu có cạnh nối giữa đỉnh i và đỉnh j và false nếu ngược lại.
- **Output**:
- Tất cả các khả năng tô màu đồ thị dùng không quá m màu sao cho hai đỉnh kề không cùng màu. Xuất mỗi màu là một mảng vcolor chỉ số từ 1 đến n, ở đây vcolor[i] (1<= i<= n) là màu gán cho đỉnh i.

5.2.3 Bài toán tô màu đồ thị

❖ **Thuật toán**:

```
Void coloring(index i)
{ int color;
     If (promising(i))
     If (i==n)       Cout<<vcolor[1] cho đến vcolor[n];
     Else  For (color=1;color<=m;color++)
   {
            vcolor[i+1] = color;
            Coloring(i+1);
    }
}
```

5.2.3 Bài toán tô màu đồ thị

```
Bool promising(index i)
{index j;
     Bool switch;
     switch = true;
     J=1;
     While (j<I &&switch)
     {
            iF (A[i][j]&&vcolor[i]==vcolor[j])
     Switch=false;
     J++;
    }
    Return switch;
}
```

5.2.3 Bài toán tô màu đồ thị

- Thông thường n, m, A và vcolor là không nhập từ bàn phím. Trong cài đặt thuật toán chúng được khai báo toàn cục và các giá trị đó lấy vào bởi một thủ tục nhập. Chương trình chính gọi cho m_coloring(0).
- Số nút của cây không gian trạng thái cho thuật toán này là:

$$1 \quad m \quad m^2 \quad \ldots \quad m^n \qquad \frac{m^{n+1} - 1}{m - 1}$$

- Độ phức tạp của thuật toán là O(m.n)

TRƯỜNG CAO ĐẲNG CNTT HỮU NGHỊ ViỆT - HÀN

KHOA KHOA HỌC MÁY TÍNH

----------***----------

Thank You !

camcntt@yahoo.com