Phần thứ nhất

LÝ THUYẾT TỔ HỢP

Combinatorial Theory

GV: Nguyễn Huy Đức

Bộ môn: Khoa học Máy tính – ĐH Thủy lợi

0903 402 655 ducnghuy@gmail.com

Nội dung

Chương 4

BÀI TOÁN LIỆT KÊ

NỘI DUNG

4.1 Giới thiệu bài toán

4.2 Thuật toán quay lui

4.3 Một số ví dụ

4.1. Giới thiệu bài toán

- Bài toán đưa ra danh sách tất cả cấu hình tổ hợp thoả mãn một số tính chất cho trước được gọi là *bài toán liệt kê tổ hợp*.
- Do số lượng cấu hình tổ hợp cần liệt kê thường là rất lớn ngay cả khi kích thước cấu hình chưa lớn:
  - Số hoán vị của n phần tử là n!
  - Số tập con m phần tử của n phần tử là n!/(m!(n-m)!
- Do đó cần có quan niệm thế nào là giải bài toán liệt kê tổ hợp

4.1. Giới thiệu bài toán

- Bài toán liệt kê tổ hợp là giải được nếu như ta có thể xác định một ***thuật toán*** để theo đó có thể lần lượt xây dựng được tất cả các cấu hình cần quan tâm.
- Một thuật toán liệt kê phải đảm bảo 2 yêu cầu cơ bản:
  - Không được lặp lại một cấu hình,
  - không được bỏ sót một cấu hình.

4.2. Thuật toán quay lui

Backtracking Algorithm

NỘI DUNG

- Sơ đồ thuật toán
- Ví dụ cho các bài toán
  - ✓ Liệt kê xâu nhị phân độ dài n
  - ✓ Liệt kê hoán vị
  - ✓ Liệt kê tập con m phần tử của tập n phần tử

Sơ đồ thuật toán

- Thuật toán quay lui (Backtracking Algorithm) là một thuật toán cơ bản được áp dụng để giải quyết nhiều vấn đề khác nhau.
- ***Bài toán liệt kê (Q):*** *Cho $A_1, A_2, \ldots, A_n$ là các tập hữu hạn. Ký hiệu*

  $$X = A_1 \times A_2 \times \ldots \times A_n = \{ (x_1, x_2, \ldots, x_n) \colon \ x_i \in A_i \ , i=1, 2, \ldots, n\}.$$

  Giả sử P là tính chất cho trên X. Vấn đề đặt ra là liệt kê tất cả các phần tử của X thoả mãn tính chất P:

  $$D = \{ x = (x_1, x_2, \ldots, x_n) \in X \colon \ x \text{ thoả mãn tính chất } P \}.$$

- Các phần tử của tập D được gọi là các ***lời giải chấp nhận được.***

Ví dụ

- Tất cả các bài toán liệt kê tổ hợp cơ bản đều có thể phát biểu dưới dạng bài toán (Q)
- Bài toán liệt kê xâu nhị phân độ dài n dẫn về việc liệt kê các phần tử của tập

$$B^n = \{(x_1, ..., x_n): x_i \in \{0, 1\}, i=1, 2, ..., n\}.$$

- Tập các hoán vị của các số tự nhiên 1, 2, ..., n là tập

$$\Pi_n = \{(x_1,..., x_n) \in N^n: x_i \neq x_j ; i \neq j\}.$$

- Bài toán liệt kê các tập con m phần tử của tập $N = \{1, 2, ..., n\}$ đòi hỏi liệt kê các phần tử của tập:

$$S(m,n) = \{(x_1,..., x_m) \in N^m: 1 \leq x_1 < ... < x_m \leq n\}.$$

Lời giải bộ phận

- **Định nghĩa.** Ta gọi lời giải bộ phận cấp k ($0 \le k \le n$) là bộ có thứ tự gồm k thành phần

$$(a_1, a_2, ..., a_k),$$

trong đó $a_i \in A_i$, $i = 1, 2, ..., k$.

- Khi $k = 0$, lời giải bộ phận cấp 0 được ký hiệu là () và còn được gọi là lời giải rỗng.
- Nếu $k = n$, ta có lời giải đầy đủ hay đơn giản là một lời giải của bài toán.

Ý tưởng chung

- Thuật toán quay lui được xây dựng dựa trên việc xây dựng dần từng thành phần của lời giải.
- Thuật toán bắt đầu từ lời giải rỗng (). Trên cơ sở tính chất *P* ta xác định được những phần tử nào của tập A_1 có thể chọn vào vị trí thứ nhất của lời giải. Những phần tử như vậy ta sẽ gọi là những ứng cử viên (viết tắt là UCV) vào vị trí thứ nhất của lời giải. Ký hiệu tập các UCV vào vị trí thứ nhất của lời giải là S_1. Lấy $a_1 \in S_1$, bổ sung nó vào lời giải rỗng đang có ta thu được lời giải bộ phận cấp 1: (a_1).

Bước tổng quát

- Tại bước tổng quát, giả sử ta đang có lời giải bộ phận cấp k-1: $(a_1, a_2, ..., a_{k-1})$.
- Trên cơ sở tính chất P ta xác định được những phần tử nào của tập A_k có thể chọn vào vị trí thứ k của lời giải.
- Những phần tử như vậy ta sẽ gọi là những ứng cử viên (viết tắt là UCV) vào vị trí thứ k của lời giải khi k-1 thành phần đầu của nó đã được chọn là $(a_1, a_2, ..., a_{k-1})$. Ký hiệu tập các ứng cử viên này là S_k.

Xét hai tình huống

- **Tình huống 1:** $S_k \neq \varnothing$. Khi đó lấy $a_k \in S_k$, bổ sung nó vào lời giải bộ phận cấp k-1 đang có

  $$(a_1, a_2, \ldots, a_{k-1})$$

  ta thu được lời giải bộ phận cấp k:

  $$(a_1, a_2, \ldots, a_{k-1}, a_k).$$

- Khi đó
  - Nếu $k = n$ thì ta thu được một lời giải,
  - Nếu $k < n$, ta tiếp tục đi xây dựng thành phần thứ k+1 của lời giải.

Tình huống ngõ cụt

- **Tình huống 2:** $S_k = \varnothing$. Điều đó có nghĩa là lời giải bộ phận $(a_1, a_2, ..., a_{k-1})$ không thể tiếp tục phát triển thành lời giải đầy đủ. Trong tình huống này ta quay trở lại tìm ứng cử viên mới vào vị trí thứ k-1 của lời giải.
- Nếu tìm thấy UCV như vậy, thì bổ sung nó vào vị trí thứ k-1 rồi lại tiếp tục đi xây dựng thành phần thứ k.
- Nếu không tìm được thì ta lại quay trở lại thêm một bước nữa tìm UCV mới vào vị trí thứ k-2, ... Nếu quay lại tận lời giải rỗng mà vẫn không tìm được UCV mới vào vị trí thứ 1, thì thuật toán kết thúc.

Thuật toán quay lui

```
procedure Bactrack(k: integer);
begin
      Xây dựng S_k;
       for y ∈ S_k do (* Với mỗi UCV y từ S_k *)
       begin
             a_k := y;
             (ghi nhận trạng thái mới);
             if k = n then <Ghi nhận lời giải (a_1, a_2, ..., a_n) >
             else Backtrack(k+1);
             (trả về trạng thái cũ);
       end;
end;
```

Lệnh gọi để thực hiện thuật toán quay lui là: Bactrack(1)

Hai vấn đề mấu chốt

- Để cài đặt thuật toán quay lui giải các bài toán tổ hợp cụ thể ta cần giải quyết hai vấn đề cơ bản sau:
  - Tìm thuật toán xây dựng các tập UCV S_k.
  - Tìm cách mô tả các tập này để có thể cài đặt thao tác liệt kê các phần tử của chúng (cài đặt vòng lặp qui ước **for y $\in$ S_k do).**
- Hiệu quả của thuật toán liệt kê phụ thuộc vào việc ta có xác định được chính xác các tập UCV này hay không.

Chú ý

- Nếu chỉ cần tìm một lời giải thì cần tìm cách chấm dứt các thủ tục gọi đệ qui lồng nhau sinh bởi lệnh gọi Backtrack(1) sau khi ghi nhận được lời giải đầu tiên.
- Nếu kết thúc thuật toán mà ta không thu được một lời giải nào thì điều đó có nghĩa là bài toán không có lời giải.

Chú ý

- Thuật toán dễ dàng mở rộng cho bài toán liệt kê trong đó lời giải có thể mô tả như là bộ (a_1, a_2, ..., a_n,...) độ dài hữu hạn, tuy nhiên giá trị của độ dài là không biết trước và các lời giải cũng không nhất thiết phải có cùng độ dài.
- Khi đó chỉ cần sửa lại câu lệnh

  if k = n then <Ghi nhận lời giải (a_1, a_2, ..., a_k) >

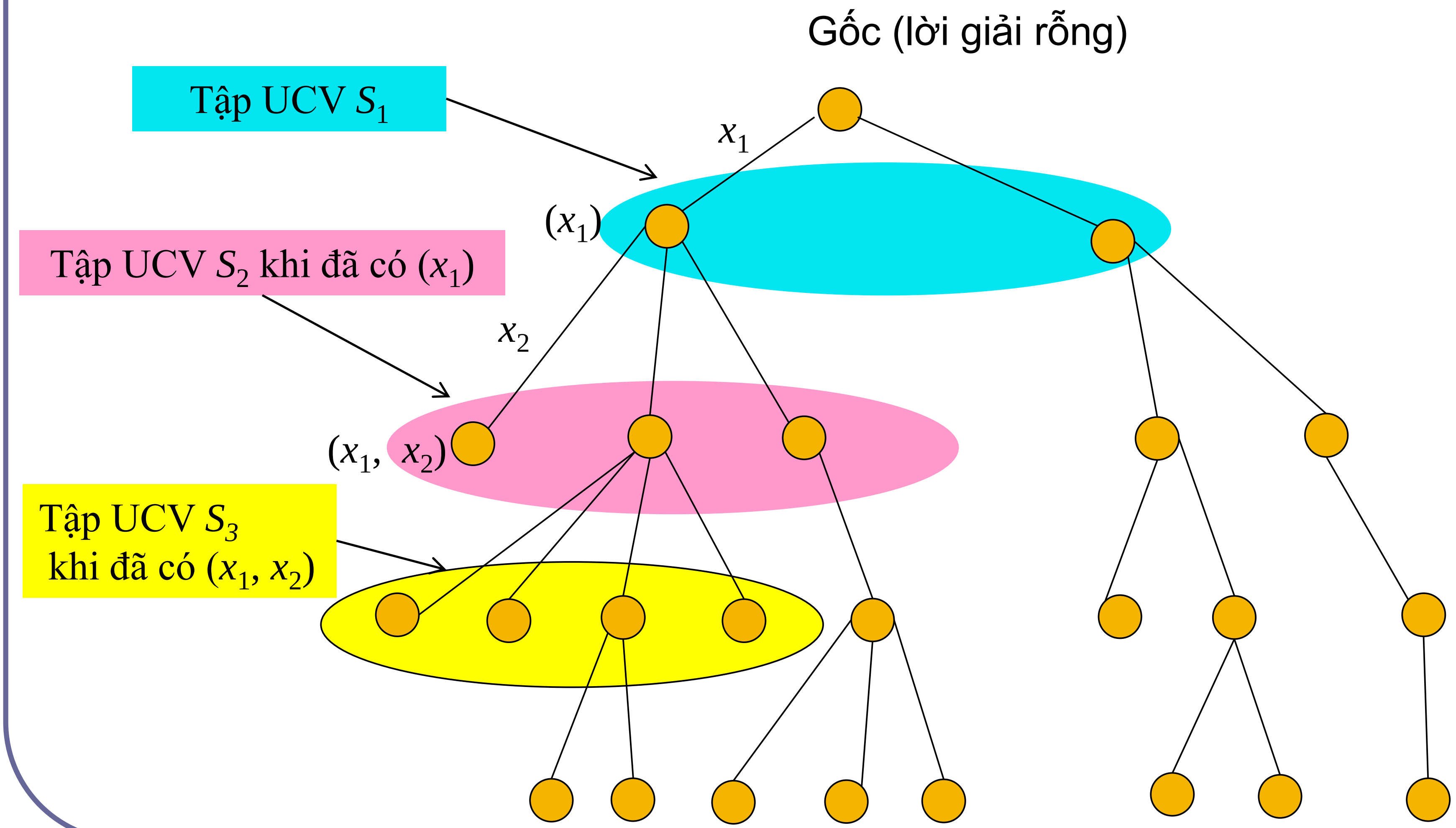
  else Backtrack(k+1);

  thành

  if <(a_1, a_2, ..., a_k) là lời giải> then <Ghi nhận (a_1, a_2, ..., a_k) >

  else Backtrack(k+1);
- Cần xây dựng hàm nhận biết (a_1, a_2, ..., a_k) đã là lời giải hay chưa.

Cây liệt kê lời giải theo thuật toán quay lui



4.3. Một số ví dụ áp dụng

Liệt kê xâu nhị phân độ dài *n*

Liệt kê xâu nhị phân độ dài *n*

- Bài toán liệt kê xâu nhị phân độ dài n dẫn về việc liệt kê các phần tử của tập

  $$B^n = \{(x_1, ..., x_n): x_i \in \{0, 1\}, i=1, 2, ..., n\}.$$

- Ta xét cách giải quyết hai vấn đề cơ bản để cài đặt thuật toán quay lui:
  - Rõ ràng ta có $S_1 = \{0, 1\}$. Giả sử đã có xâu nhị phân cấp k-1 $(b_1, ..., b_{k-1})$, khi đó rõ ràng $S_k = \{0,1\}$.

    Như vậy, tập các UCV vào các vị trí của lời giải được đã xác định.
  - Để cài đặt vòng lặp liệt kê các phần tử của S_k, dễ thấy là ta có thể sử dụng vòng lặp for

for (j=0; j<=1; j++)

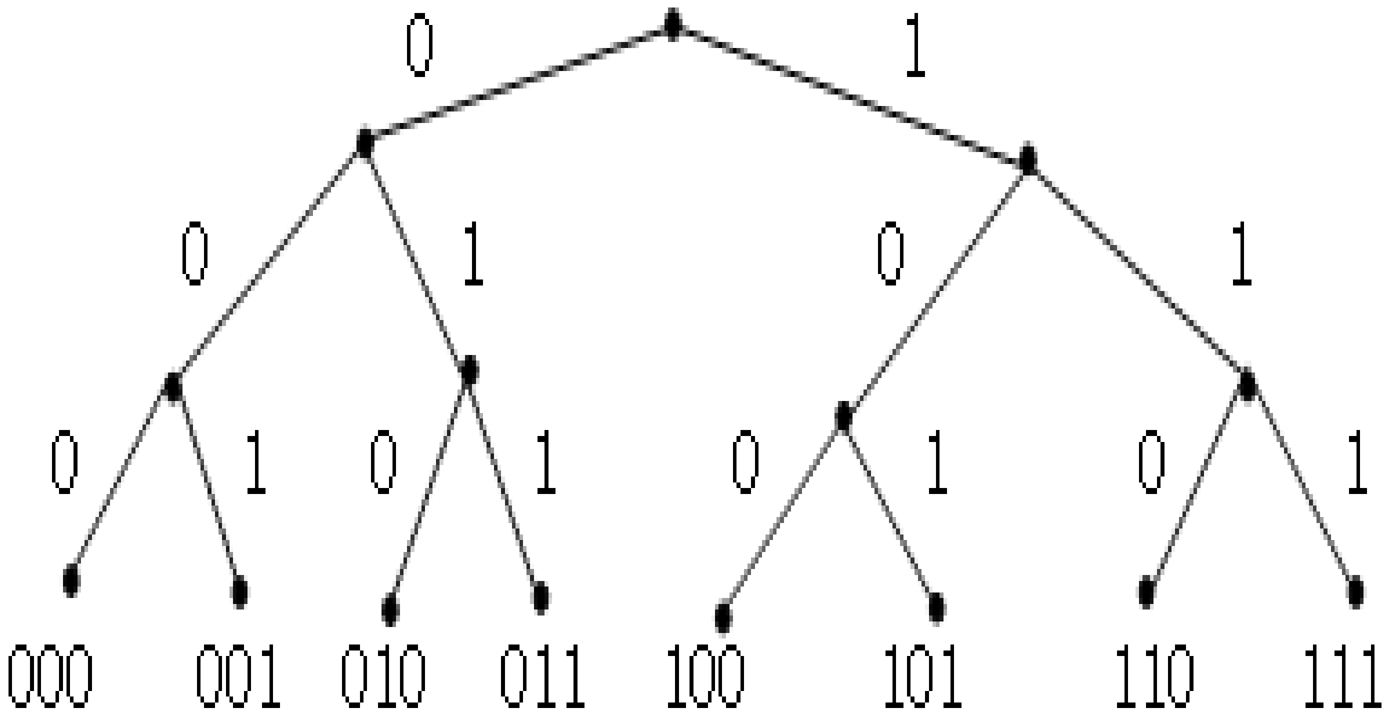
Chương trình trên C++

```
#include <iostream>
using  namespace std;
int n, dem, b[15];
void khoitao()
{  cout <<"Nhap n= "; cin >> n;
   dem=0;
}
void xuat()
{  dem++; cout <<dem <<". ";
   for (int j=1; j<=n; j++)
           cout<< b[j];
   cout << endl;
}
```

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

int main()
{
   khoitao();
   Try(1);
}
```

Cây liệt kê dãy nhị phân độ dài 3



Liệt kê hoán vị của tập n phần tử

Liệt kê hoán vị

- Tập các hoán vị của các số tự nhiên N ={1, 2, ..., n} là tập:

$$\Pi_n = \{(x_1,\ldots, x_n) \in N^n\text{:}\ \ x_i \neq x_{\mathrm{j}}\ ,\ i \neq j\ \}.$$

- **Bài toán:**

Liệt kê tất cả các phần tử của Π_n

Giải quyết 2 vấn đề mấu chốt

- Rõ ràng $S_1 = N$. Giả sử ta đang có hoán vị bộ phận $(a_1, a_2, ..., a_{k-1})$, từ điều kiện $a_i \neq a_j$, với mọi $i \neq j$ ta suy ra

$$S_k = N \setminus \{ a_1, a_2, ..., a_{k-1} \}.$$

- Như vậy ta đã có cách xác định được tập các UCV vào các vị trí của lời giải.
- Mô tả S_k thế nào ?

Mô tả S_k

- Đánh dấu những phần tử đã có mặt trong hoán vị bộ phận (a_1, a_2, ..., a_{k-1})
- Mảng đánh dấu dd[1..n]:

  dd[j] = false nếu j chưa có mặt trong (a_1, a_2, ..., a_{k-1}).
- Tập UCV ở bước k:

$$S_k = \{ j \in N \,/\, dd[j] = false \}$$

- Khi đặt a_k =j thì đánh dấu bằng cách gán:

  dd[j] = True

Chương trình trên C++

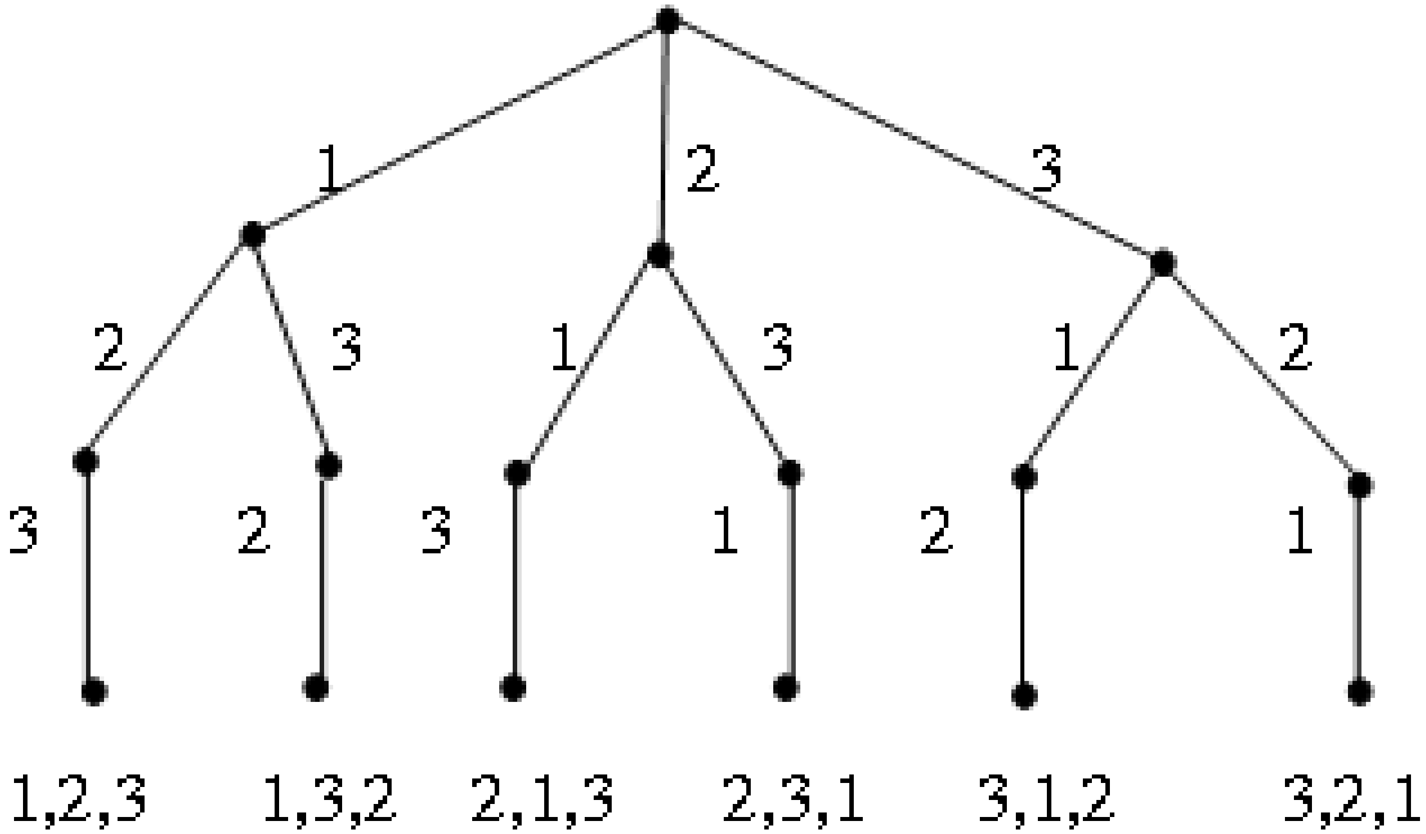
```
#include <iostream>
using  namespace std;
 int n, dem, hv[10];
bool dd[10];
void khoitao()
{   cout <<"Nhap n= "; cin >> n;
    for (int i=1; i<=n; i++)
         dd[i]=false;
    dem=0; }

void xuat()
{   dem++; cout <<dem <<". ";
    for (int j=1; j<=n; j++) cout<<
     hv[j]<<" ";
    cout << endl; }
```

```
void Try(int i)
{
    for (int j=1; j<=n; j++)
        if (dd[j]==false)
        {
            hv[i]=j;
            dd[j]=true;
            if (i==n) xuat();
            else Try(i+1);
            dd[j]=false;
        }
}
int main()
{  khoitao(); Try(1); }
```

Cây liệt kê hoán vị của 1, 2, 3

-



Liệt kê chỉnh hợp chập m của n phần tử

Liệt kê chỉnh hợp chập m của n phần tử

- Tập các chỉnh hợp chập m của các số tự nhiên $N = \{1, 2, ..., n\}$ là tập:

$$A(n,m) = \{(x_1,\ldots, x_m) \in N^m: \ x_i \neq x_j, \ i \neq j\}.$$

- **Bài toán: Liệt kê tất cả các phần tử của *A(n,m)***
- Phân tích tương tự như hoán vị, có chương trình như sau:

Chương trình trên C++

```
#include <iostream>
using  namespace std;
int n, m, dem, kq[10], dd[10];
void khoitao()
{   cout <<"Nhap n= "; cin >> n;
    cout <<"Nhap m= "; cin >> m;
    for (int i=1; i<=n; i++)  dd[i]=0;
    dem=0;
}
void xuat()
{  dem++; cout <<dem <<". ";
   for (int j=1; j<=m; j++)
        cout<< kq[j]<<" ";
   cout << endl;
}
```

```
Void Try(int i)
{
   for (int j=1; j<=n; j++)
      if (dd[j]==0)
      {
         kq[i]=j;
         dd[j]=1;
         if (i==m) xuat();
         else Try(i+1);
         dd[j]=0;
      }
}

int main()
{  khoitao(); Try(1); }
```

Liệt kê các tổ hợp chập m của n phần tử

Liệt kê các m-tập con của n-tập

- **Bài toán:** Liệt kê các tập con *m* phần tử của tập $N = \{1, 2, ..., n\}$.

- Bài toán dẫn về: Liệt kê các phần tử của tập:

$$S(m,n)=\{(a_1,..., a_m)\in N^m: 1 \leq a_1<...<a_m\leq n\}$$

Giải quyết 2 vấn đề mấu chốt

- Từ điều kiện: $1 \le a_1 < a_2 < \dots < a_m \le n$

  suy ra $S_1 = \{1, 2, \dots, n-(m-1)\}$.

- Giả sử đã có tập con $(a_1, \dots, a_{k-1})$.

  Từ điều kiện $a_{k-1} < a_k < \dots < a_m \le n$, ta suy ra

  $S_k = \{a_{k-1}+1, a_{k-1}+2, \dots, n-(m-k)\}$.

- Để cài đặt vòng lặp liệt kê các phần tử của S_k, dễ thấy là ta có thể sử dụng vòng lặp

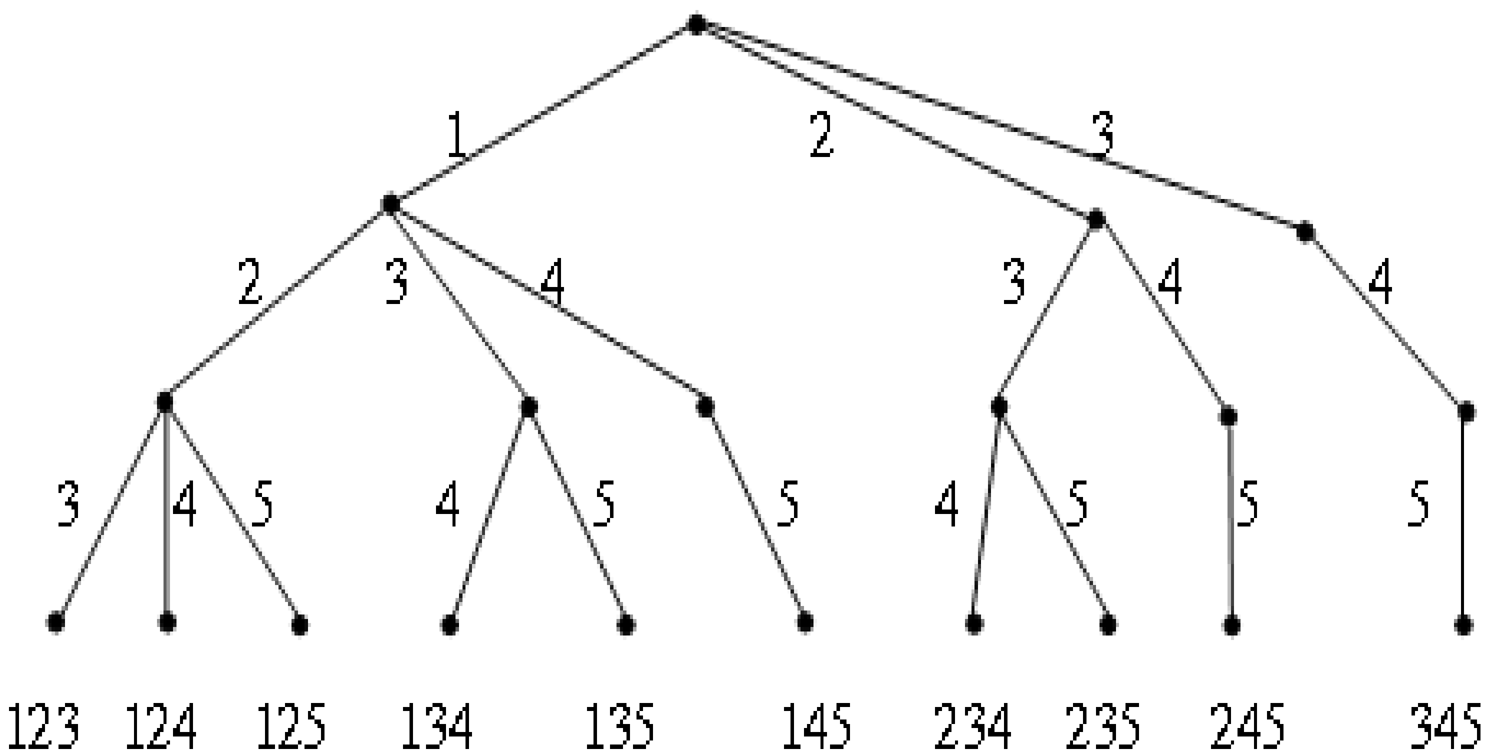
  for (j=a[k-1]+1; j<=n-m+k; j++) **...**

Chương trình trên C++

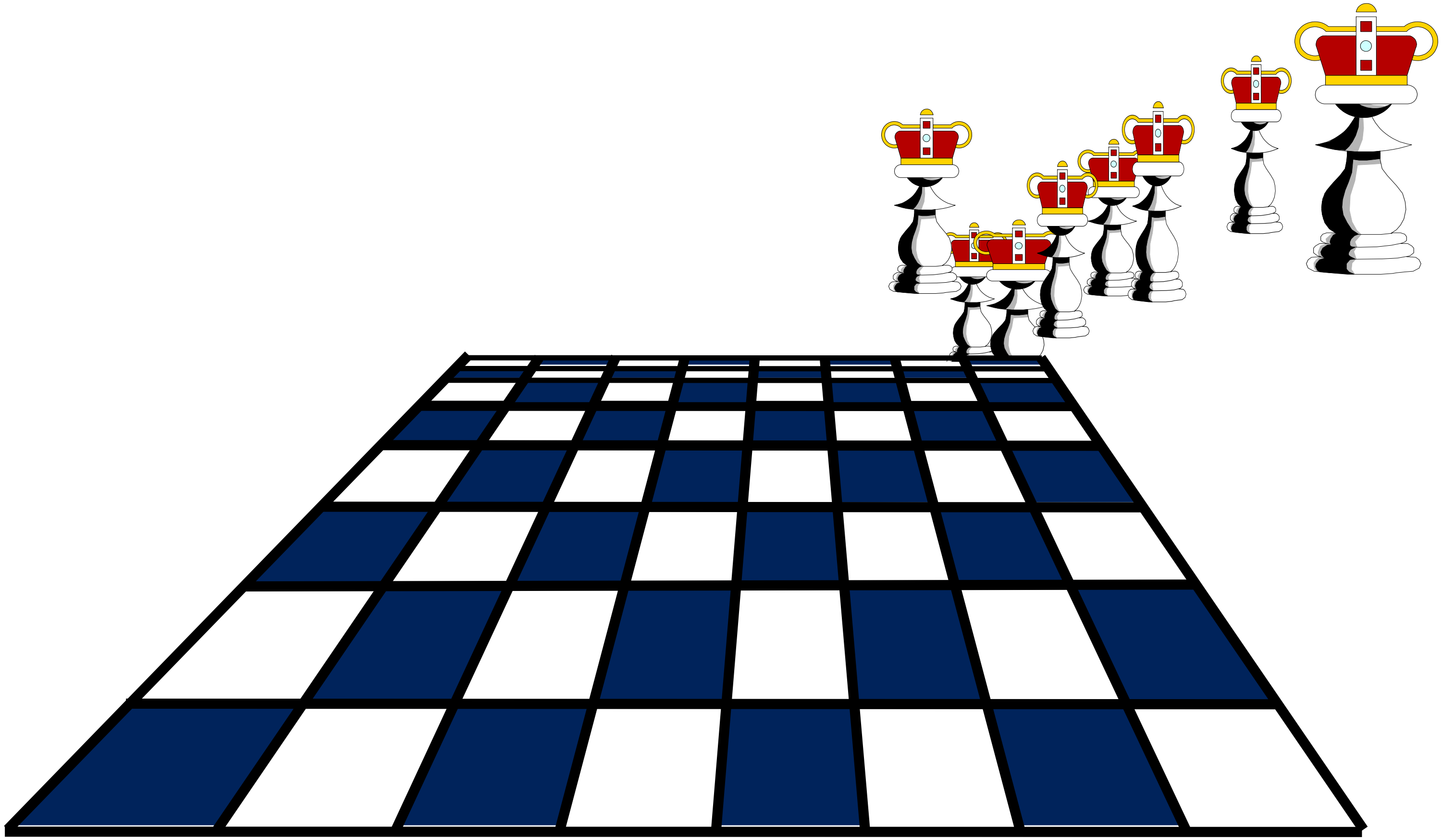
```
#include <iostream>
using  namespace std;
int n, m, dem, a[15];
void khoitao()
{   cout <<"Nhap n= "; cin >> n;
    cout <<"Nhap m= "; cin >> m;
    dem=0; a[0]=0;
}
 void xuat()
{
   dem++; cout <<dem <<". ";
   for (int j=1; j<=m; j++)
           cout<< a[j]<<" ";
   cout << endl;
}
```

```
void Try(int i)
{
   for (int j=a[i-1]+1; j<=n-m+i; j++)
      {
         a[i]=j;
         if (i==m) xuat();
         else
           Try(i+1);
      }
}
 int main()
{
   khoitao();
   Try(1);
}
```

Cây liệt kê S(5,3)

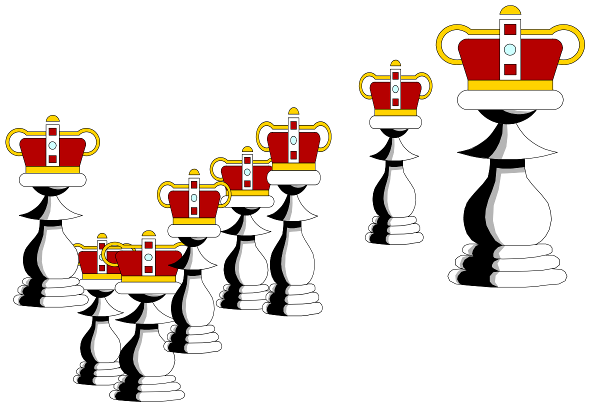


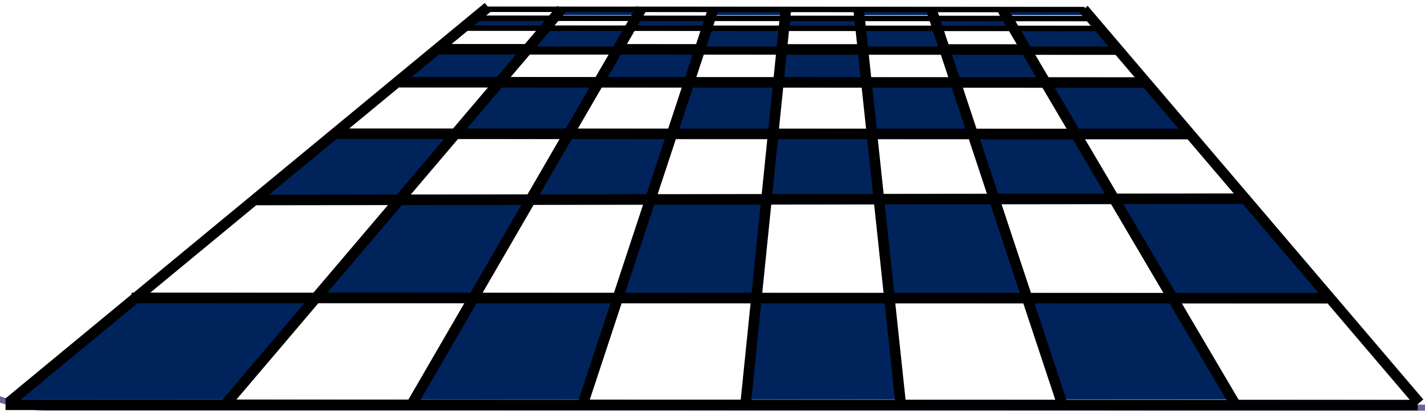
The *n*-Queens Problem



The *n*-Queens Problem

- Giả sử ta có 8 con hậu...
- ...và bàn cờ quốc tế



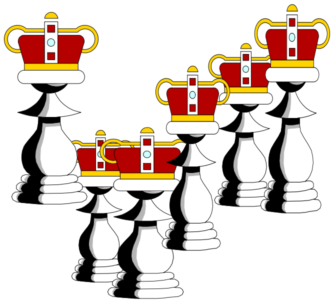


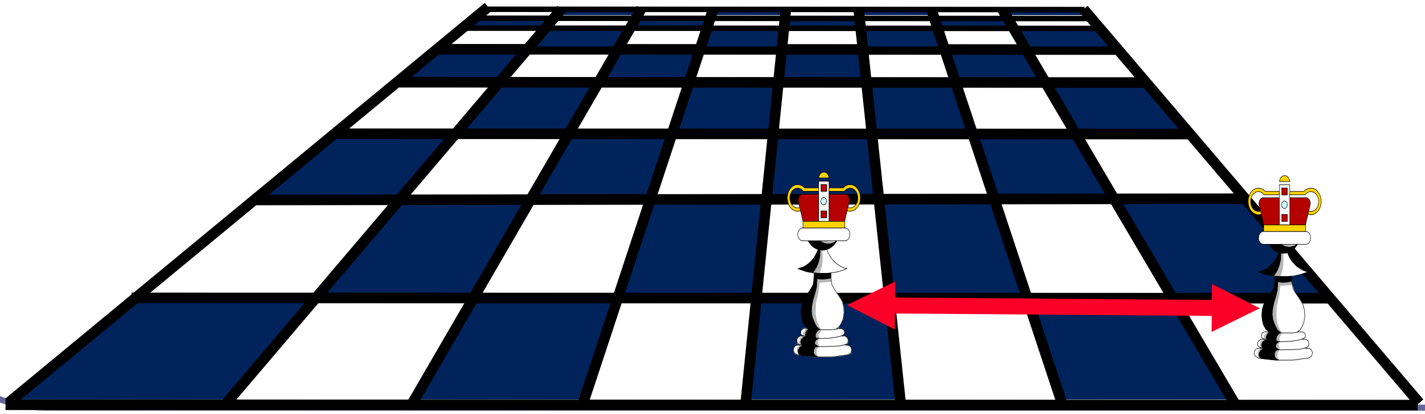
The *n*-Queens Problem

Có thể xếp các con hậu sao cho không có hai con nào ăn nhau hay không ?

The n-Queens Problem

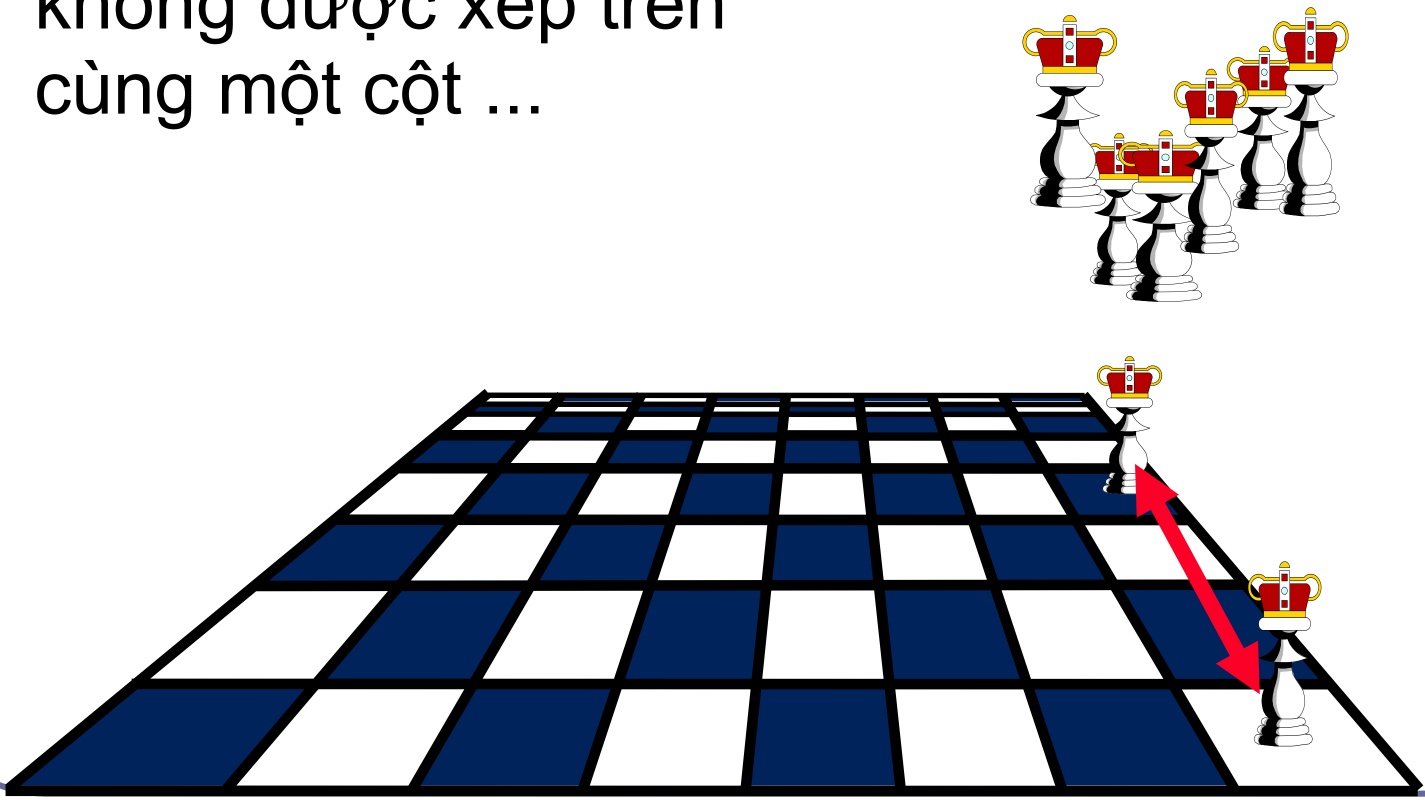
Hai con hậu bất kỳ không được xếp trên cùng một dòng ...





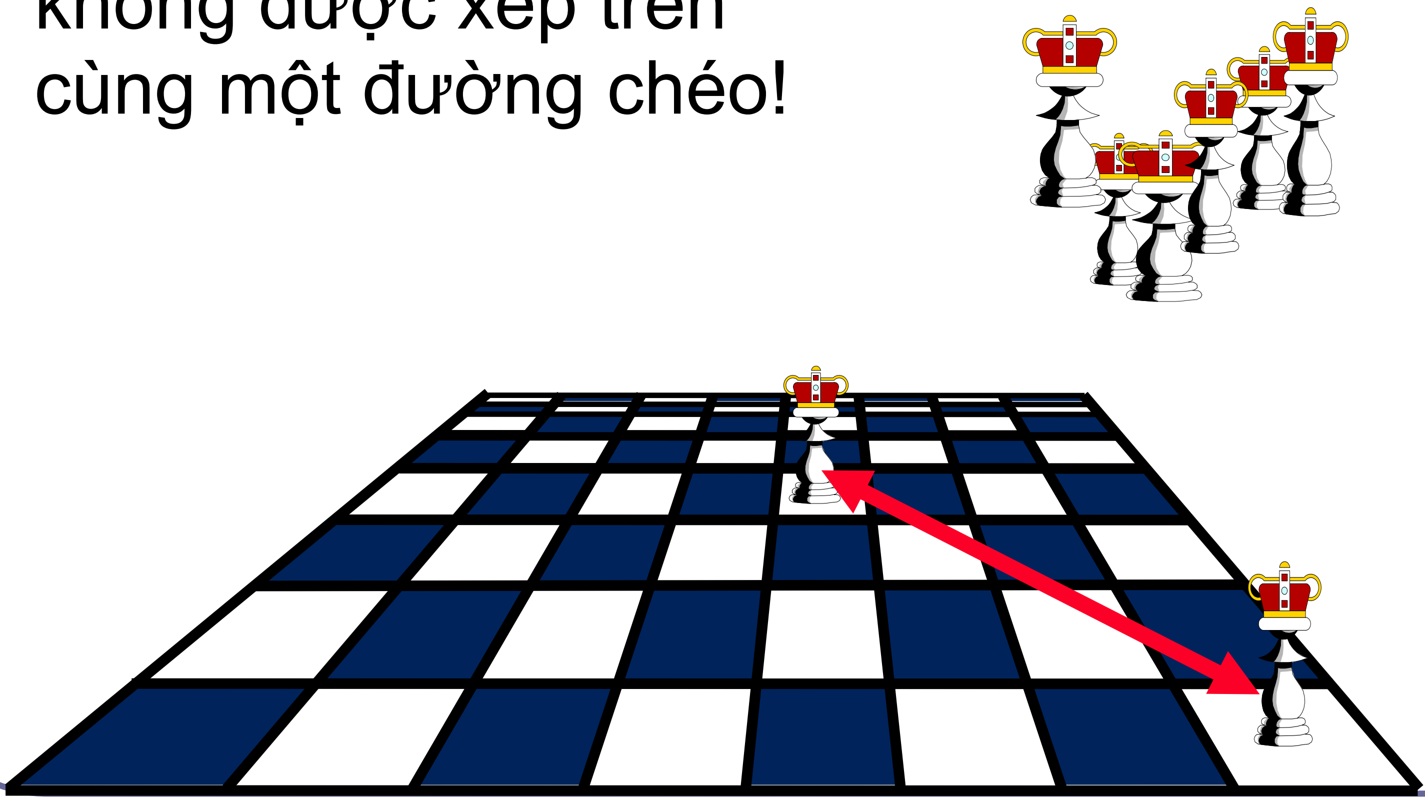
The n-Queens Problem

Hai con hậu bất kỳ không được xếp trên cùng một cột ...



The n-Queens Problem

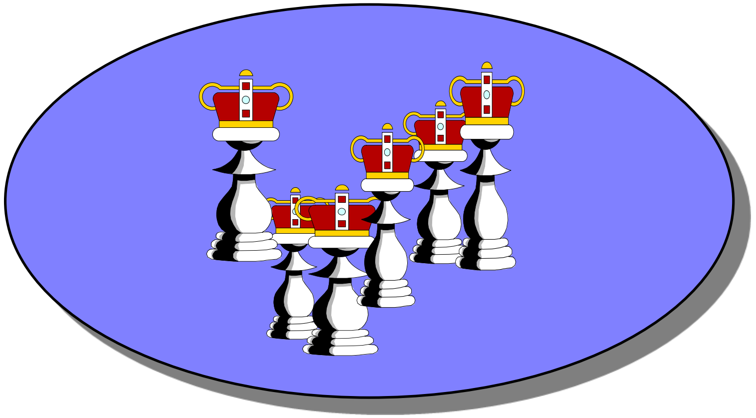
Hai con hậu bất kỳ không được xếp trên cùng một đường chéo!



The n-Queens Problem

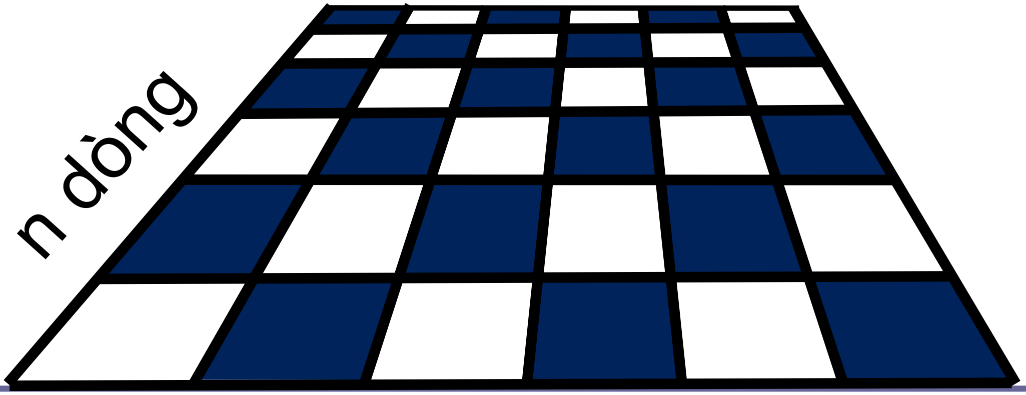
Kích thước n*n

n con hậu



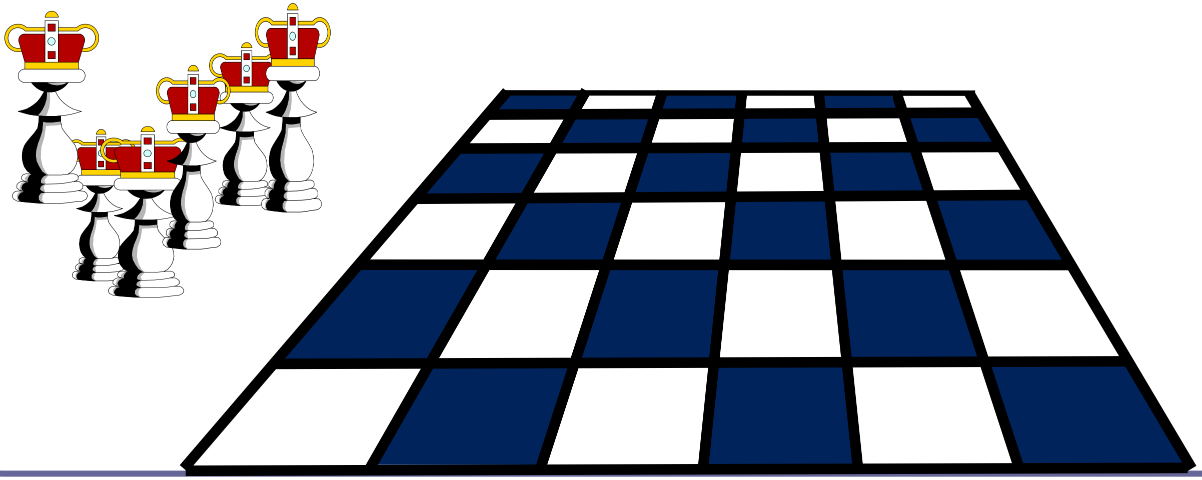
n cột

n dòng



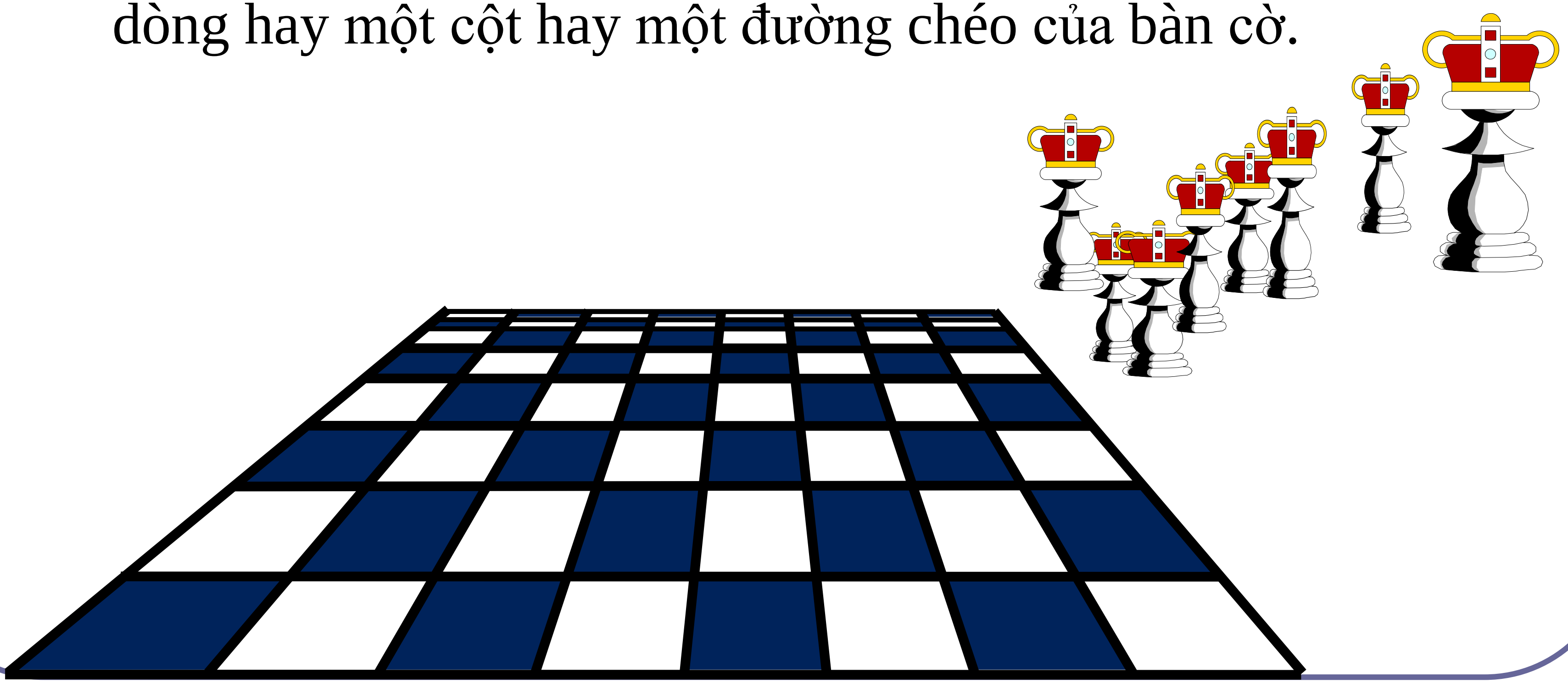
The n-Queens Problem

Xét bài toán xếp n con hậu lên bàn cờ kích thước n x n.



Bài toán xếp hậu

- Liệt kê tất cả các cách xếp n quân Hậu trên bàn cờ $n \times n$ sao cho chúng không ăn được lẫn nhau, nghĩa là sao cho không có hai con nào trong số chúng nằm trên cùng một dòng hay một cột hay một đường chéo của bàn cờ.



Biểu diễn lời giải

- Đánh số các cột *(trái sang phải)* và dòng *(dưới lên trên)* của bàn cờ từ 1 đến *n*.
- Mỗi dòng xếp đúng 1 quân hậu, vấn đề là xếp hậu vào cột nào ?
- Biểu diễn một cách xếp hậu bằng bộ n thành phần:

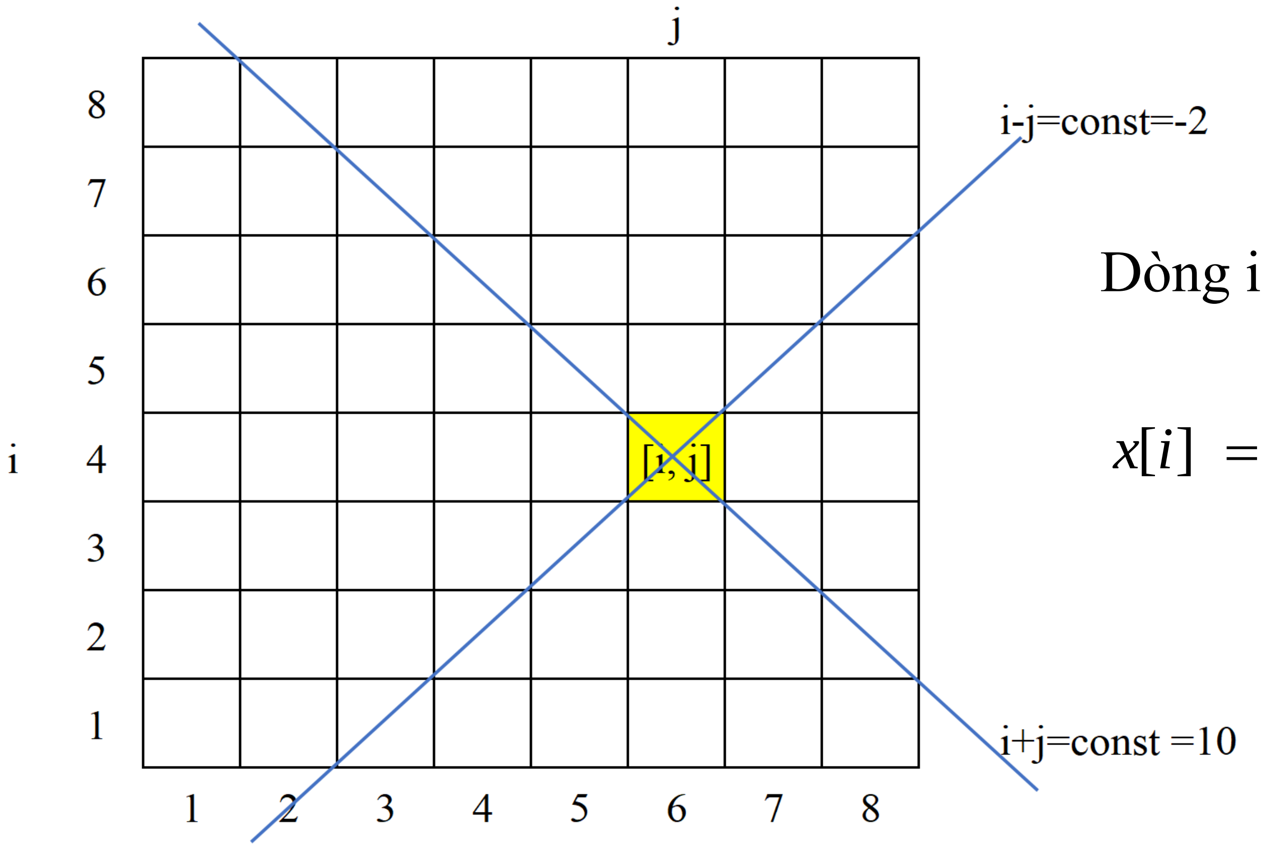
$$(x_1, x_2, ..., x_n),$$

  trong đó x_i =j nghĩa là quân hậu dòng i xếp vào cột j.
- Các giá trị đề cử cho x_i là 1 đến n. Giá trị j là được chấp nhận nếu ô [i, j] chưa bị các quân hậu chiếu đến.
- Để kiểm soát: cần ghi nhận trạng thái bàn cờ trước cũng như sau khi xếp được một quận hậu.

Biểu diễn lời giải

- Kiểm soát chiều ngang: mỗi dòng xếp đúng một hậu.
- Kiểm soát chiều dọc: dùng dãy biến logic a_j (j= 1,2,..n) với quy ước a_j= true nếu cột j chưa có hậu.
- Với 2 đường chéo nhận xét thấy:
  - ✓ Một đường có phương trình i+j= const ($2 \le i+j \le 2n$);
  - ✓ Đường kia có phương trình i-j = const ($1-n \le i-j \le n-1$)
- Ghi nhận trạng thái đường chéo thứ nhất bằng dãy biến logic b_j ($2 \le j \le 2n$), đường chéo thứ hai bằng dãy biến logic c_j ($1-n \le j \le n-1$) với quy ước các đường này còn trống nếu biến tương ứng có giá trị true.

Biểu diễn lời giải



Dòng i đặt hậu ở cột j:

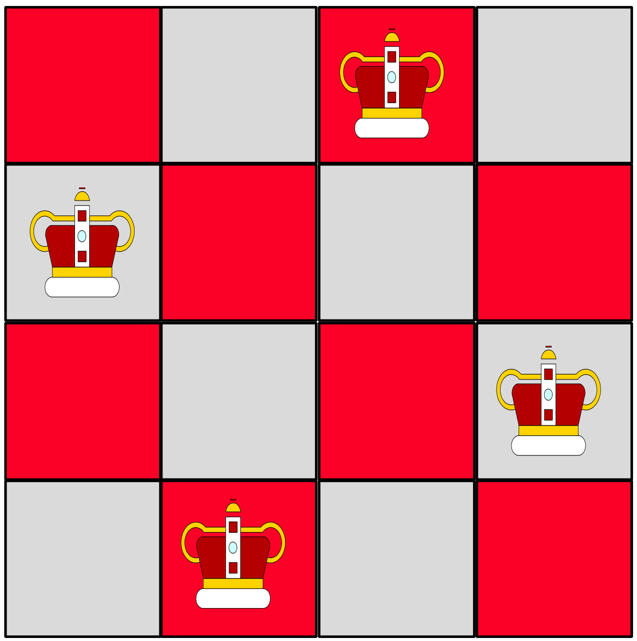
$$x[i] = j \Leftrightarrow \begin{cases} a[j] = true \\ b[i+j] = true \\ c[i-j] = true \end{cases}$$

Chương trình trên C++

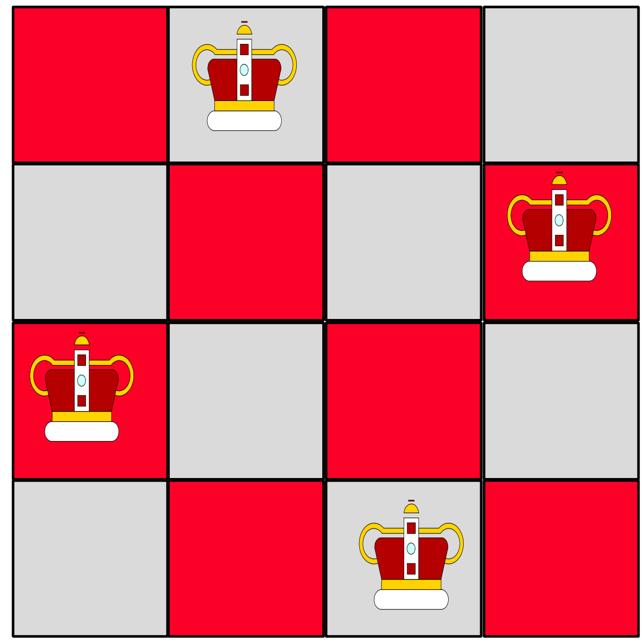
```
#include <iostream>
using  namespace std;
int n, dem, x[20];
bool a[20], b[40], c[40];
void khoitao()
{  cout <<"Nhap n= "; cin >> n;
    for (int i=1; i<=n; i++) a[i]=true;
    for (int i=2; i<=2*n; i++) b[i]=true;
    for (int i=1; i<=2*n-1; i++) c[i]=true;
     //Dich chi so mang c tu [1-n; n-1]
     //thanh [1; 2n-1]
     dem=0; }
void xuat()
{   dem++; cout <<dem <<". ";
     for (int i=1; i<=n; i++)
              cout<< x[i]<<“ ";
     cout << endl; }
```

```
void hau(int i)
{
 for (int j=1; j<=n; j++)
    if (a[j] and b[i+j] and c[i-j+n])
    {   x[i]=j;
        a[j]=false; b[i+j]=false;
        c[i-j+n]=false;
        if (i==n) xuat();
        else hau(i+1);
        a[j]=true; b[i+j]=true;
        c[i-j+n]=true;  }
}
int main()
{
   khoitao();
   hau(1);
}
```

Hai lời giải của bài toán xếp hậu khi n=4



$x = (2, 4, 1, 3)$

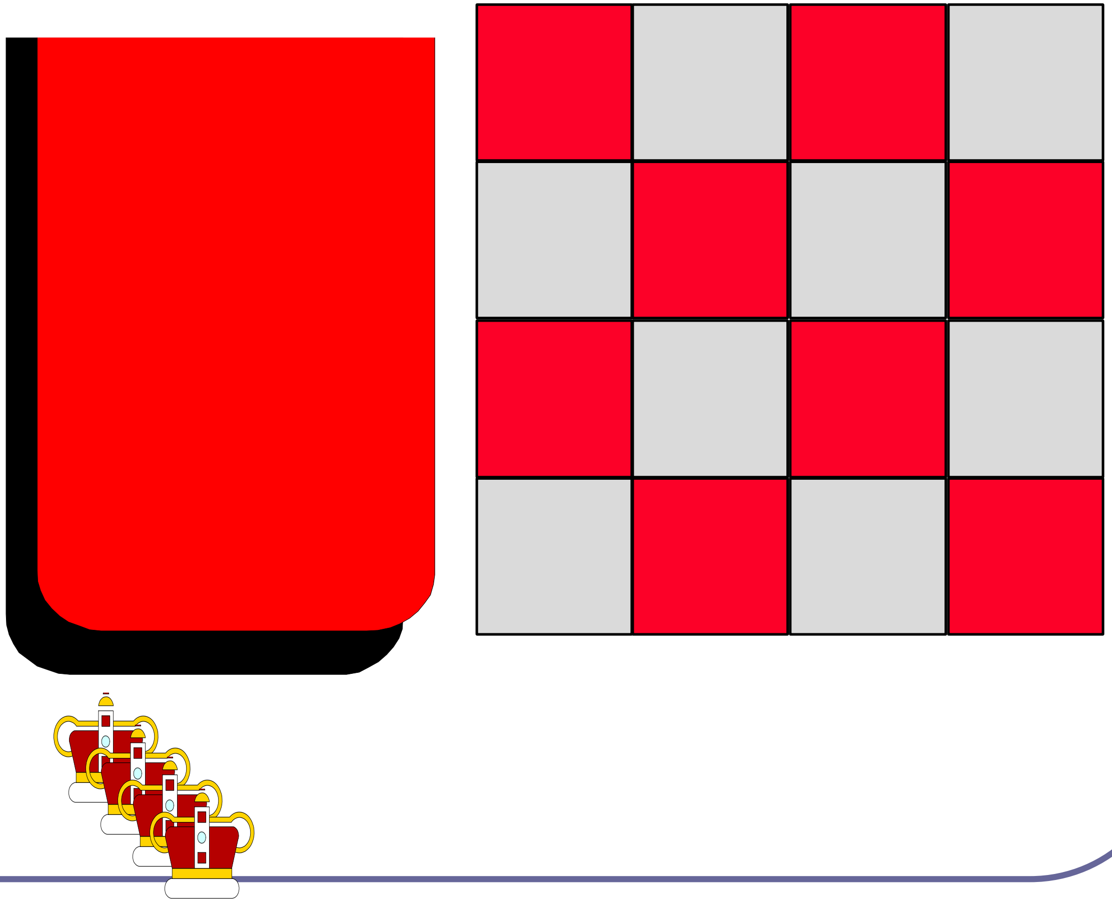


$x = (3, 1, 4, 2)$

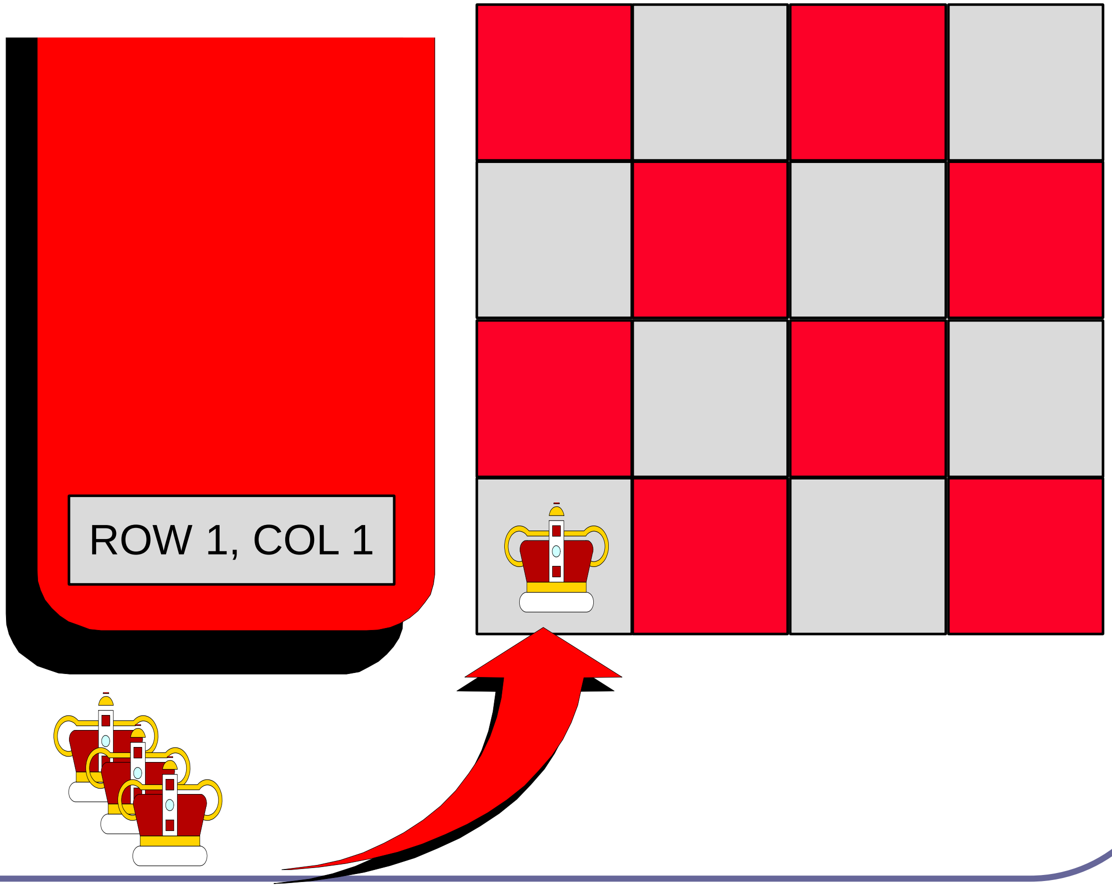
Chú ý

- Rõ ràng là bài toán xếp hậu không phải là luôn có lời giải, chẳng hạn bài toán không có lời giải khi n=2, 3. Do đó điều này cần được thông báo khi kết thúc thuật toán.

Thuật toán làm việc như thế nào

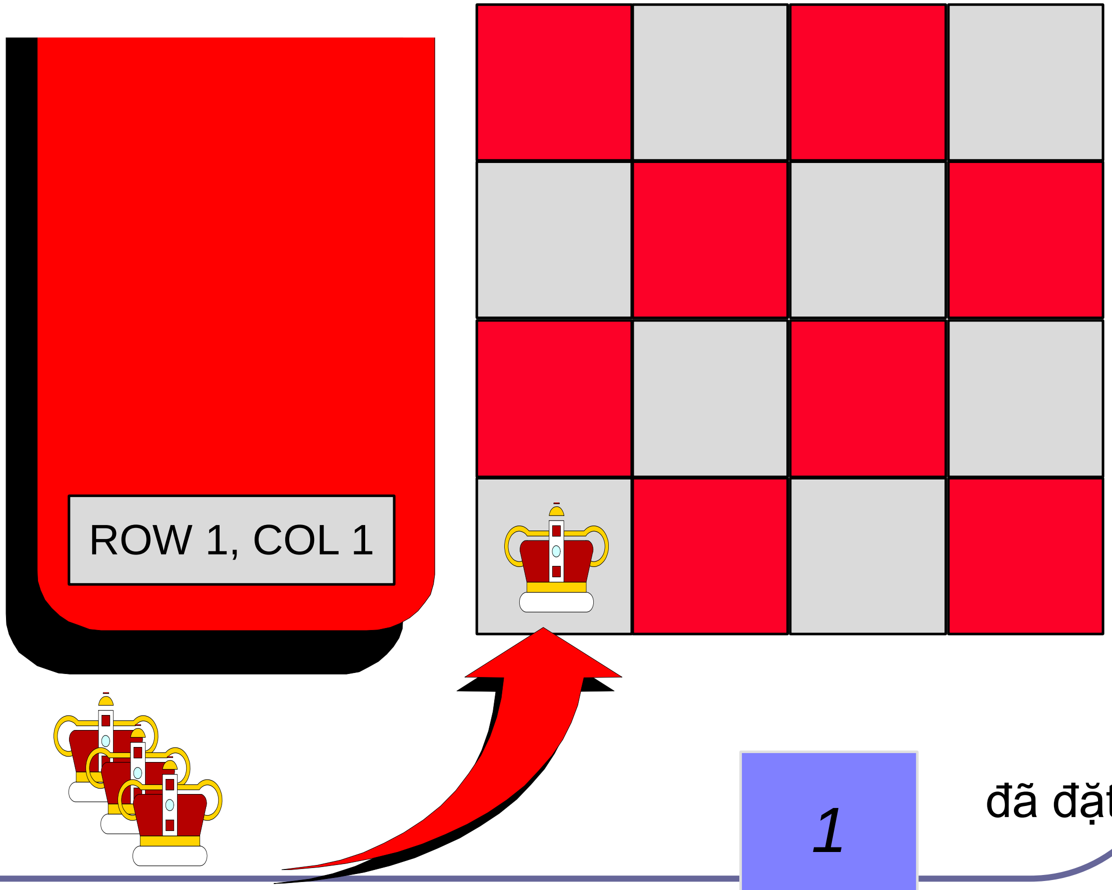


Thuật toán làm việc như thế nào



Thuật toán làm việc như thế nào

- Xếp con hậu ở dòng 1
  vào vị trí cột 1

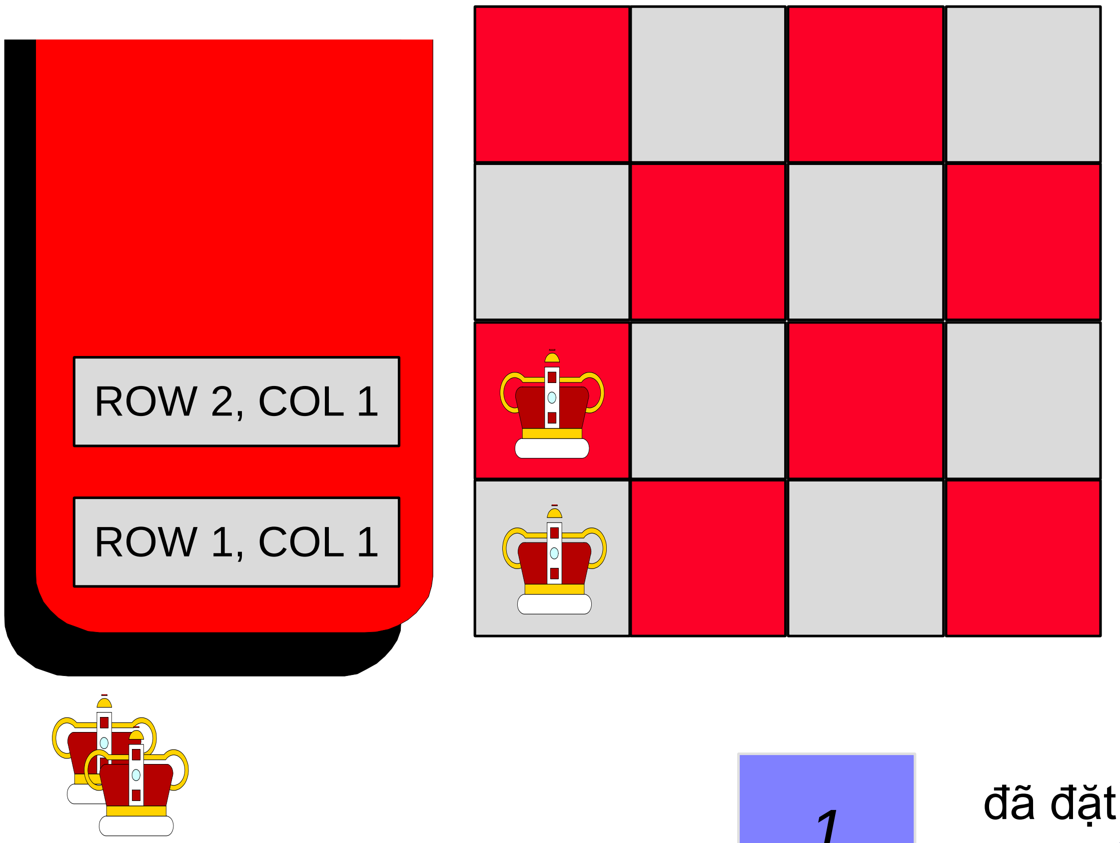


1

đã đặt

Thuật toán làm việc như thế nào

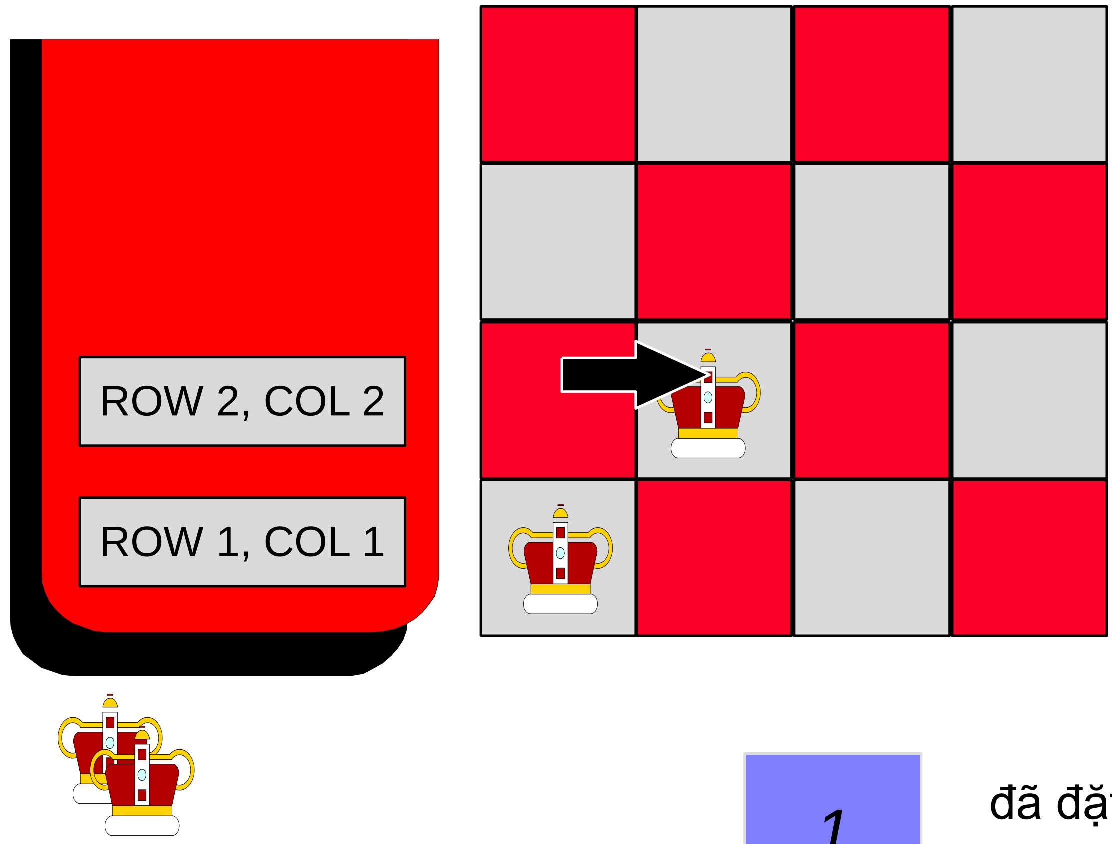
- Thử xếp con hậu ở dòng 2 vào vị trí cột 1



1 đã đặt

Thuật toán làm việc như thế nào

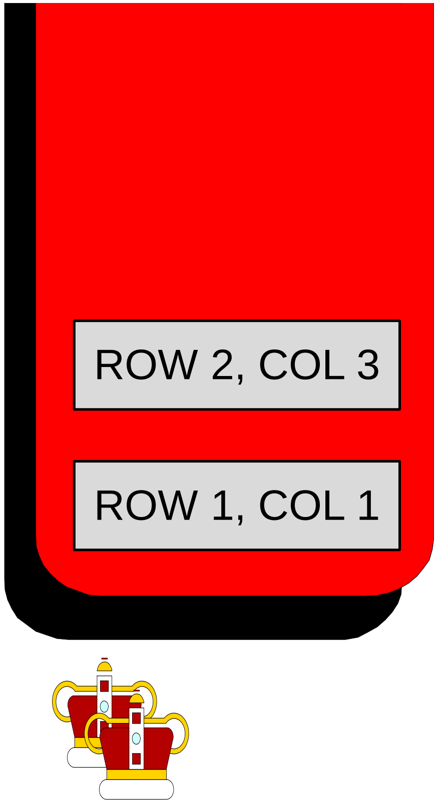
- Thử xếp con hậu ở dòng 2 vào vị trí cột 2

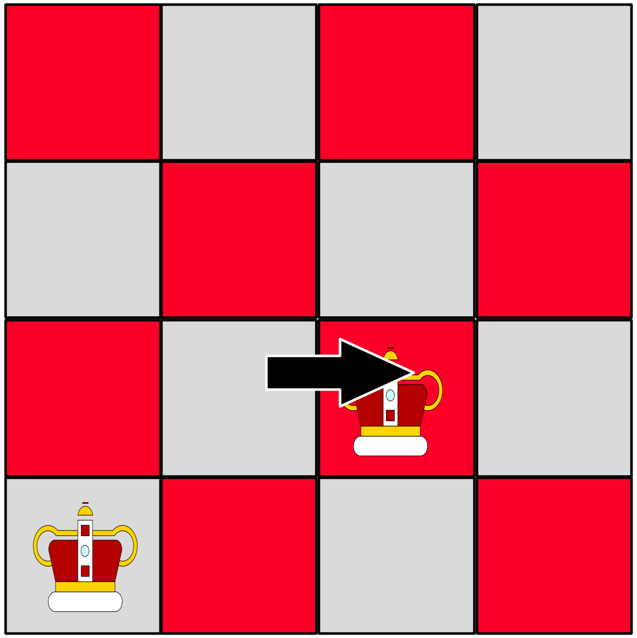


1 đã đặt

Thuật toán làm việc như thế nào

- Thử xếp con hậu ở dòng 2 vào vị trí cột 3



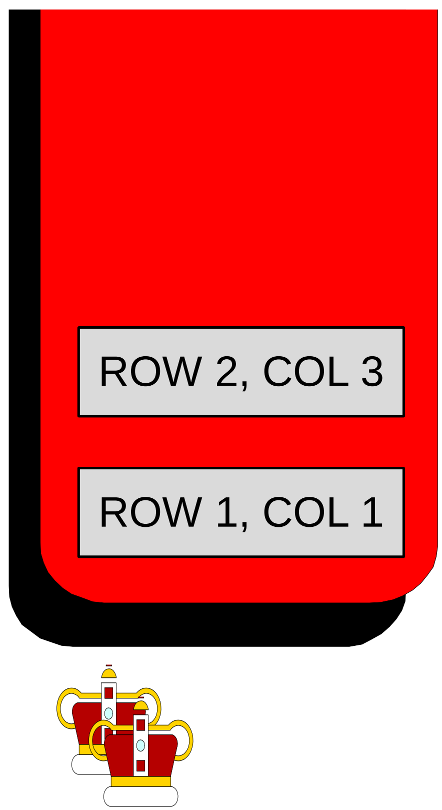


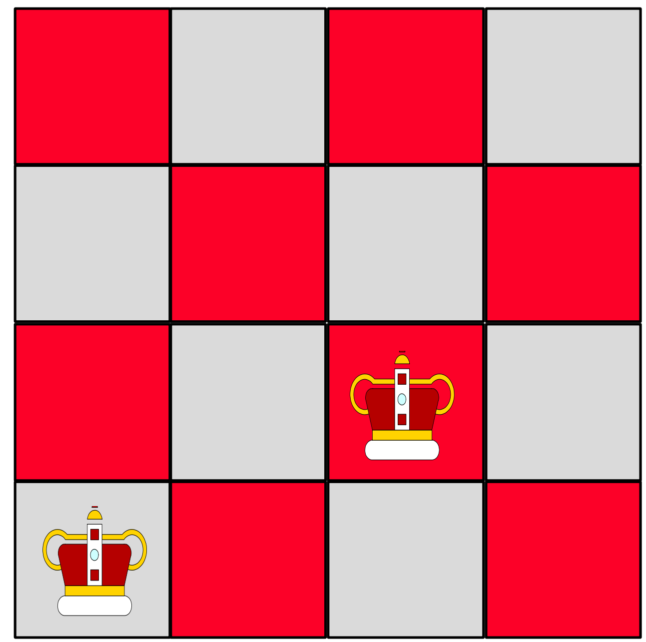
1

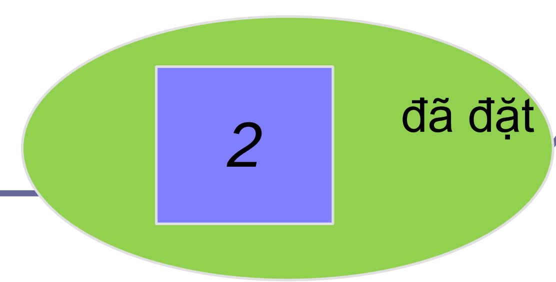
đã đặt

Thuật toán làm việc như thế nào

- Chấp nhận xếp con hậu ở dòng 2 vào vị trí cột 3

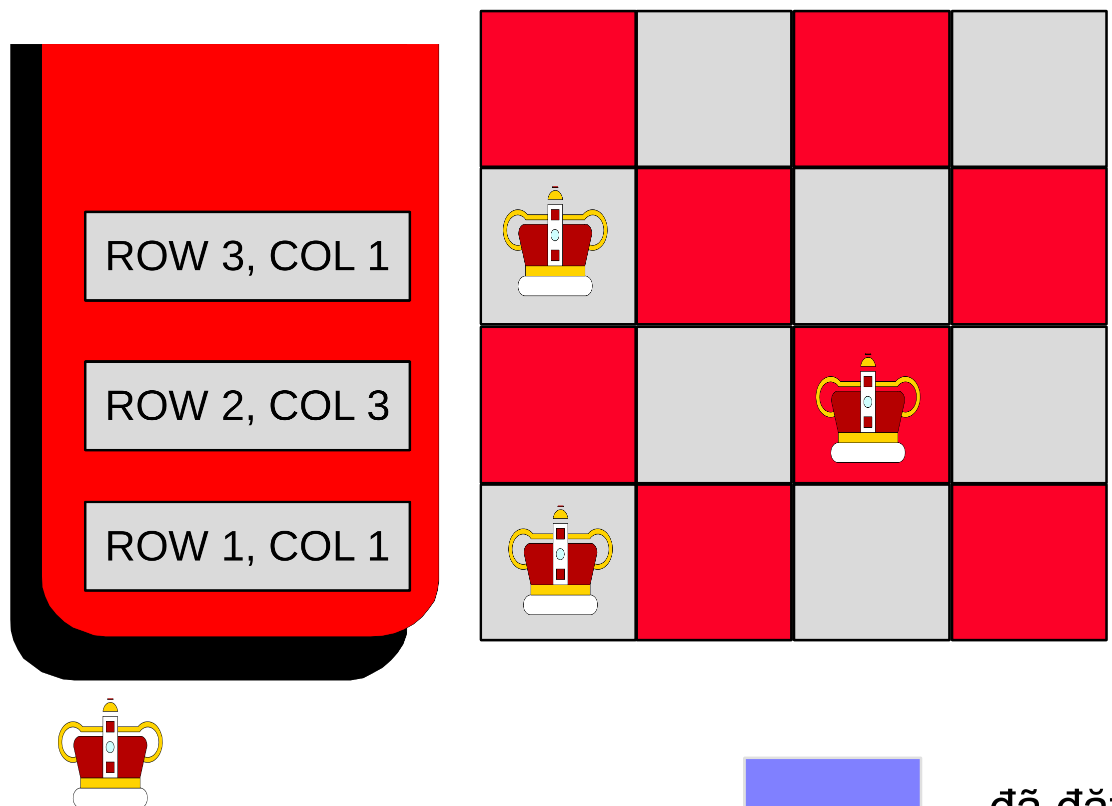






Thuật toán làm việc như thế nào

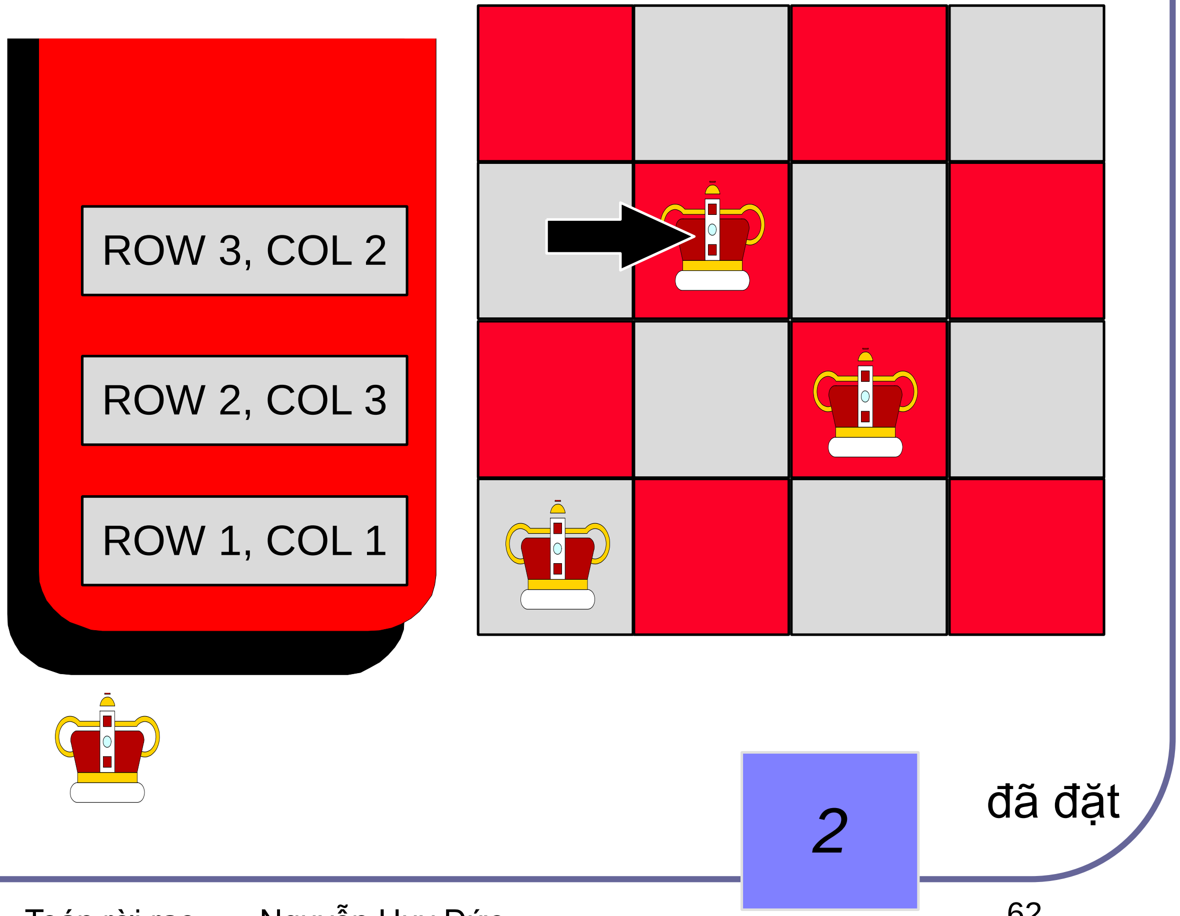
Thử xếp con hậu ở
dòng 3
vào cột đầu tiên



2 đã đặt

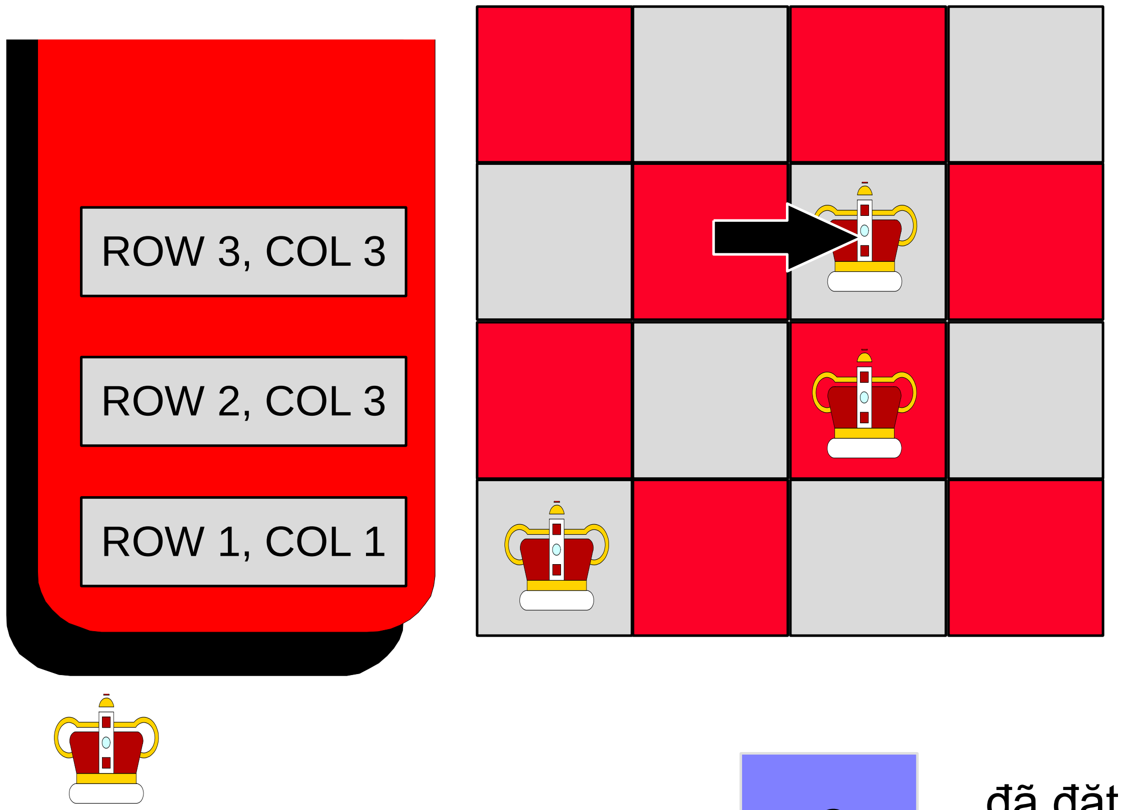
Thuật toán làm việc như thế nào

Thử cột tiếp theo



Thuật toán làm việc như thế nào

- Thử cột tiếp theo

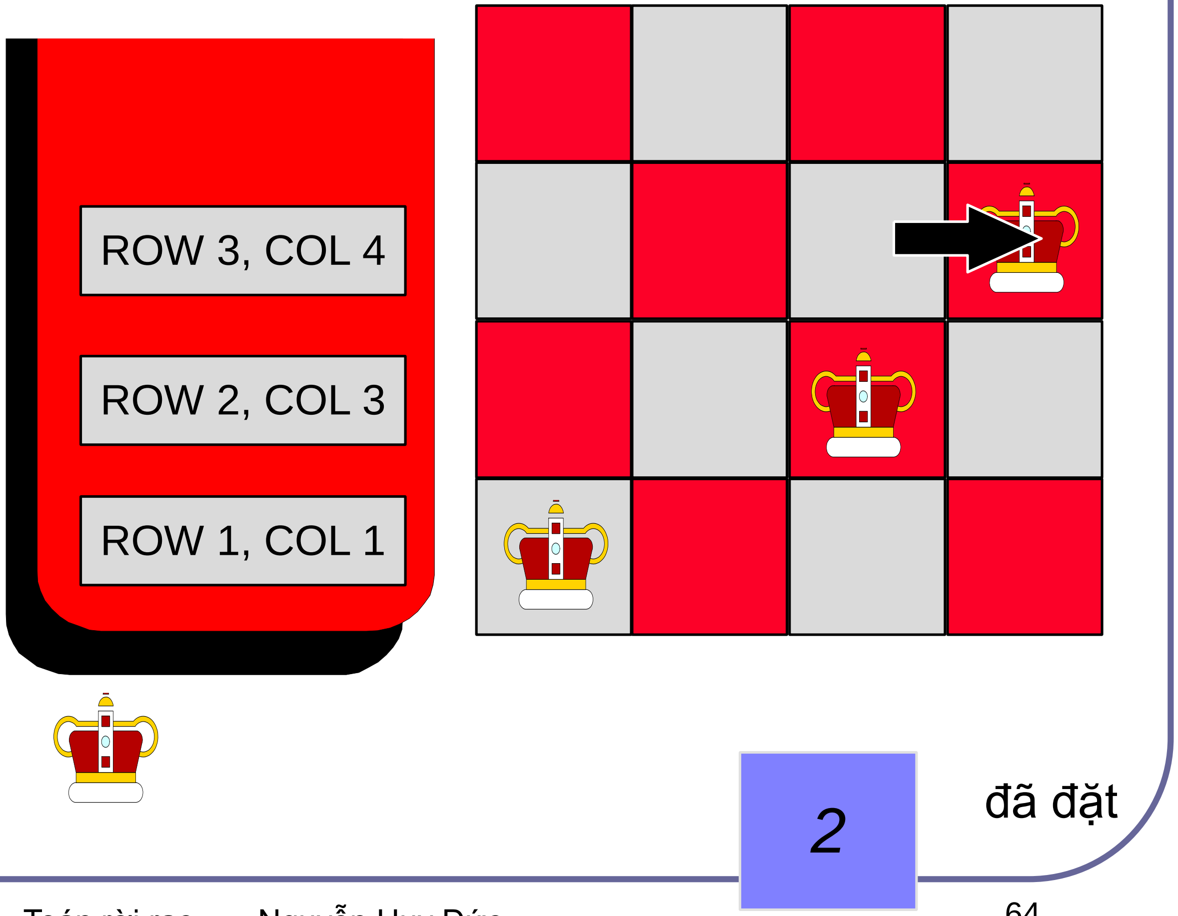


2

đã đặt

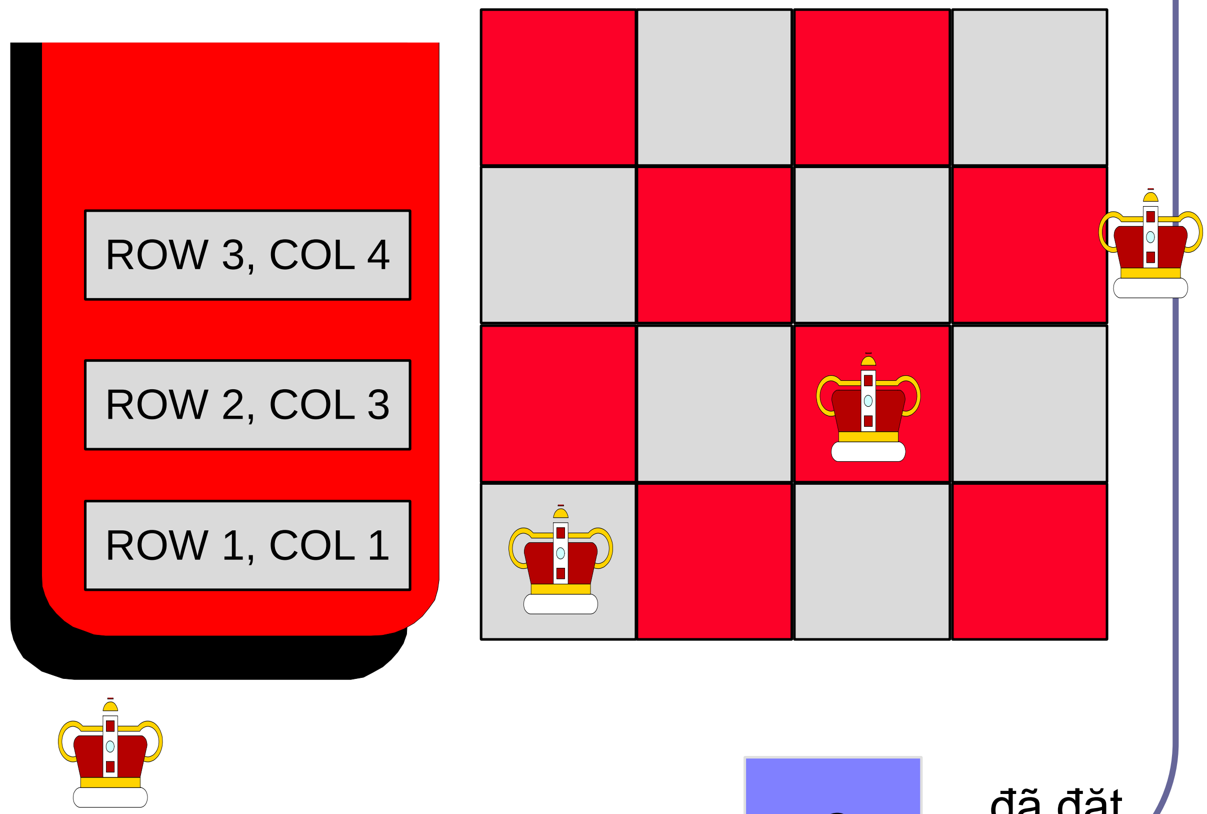
Thuật toán làm việc như thế nào

- Thử cột tiếp theo



Thuật toán làm việc như thế nào

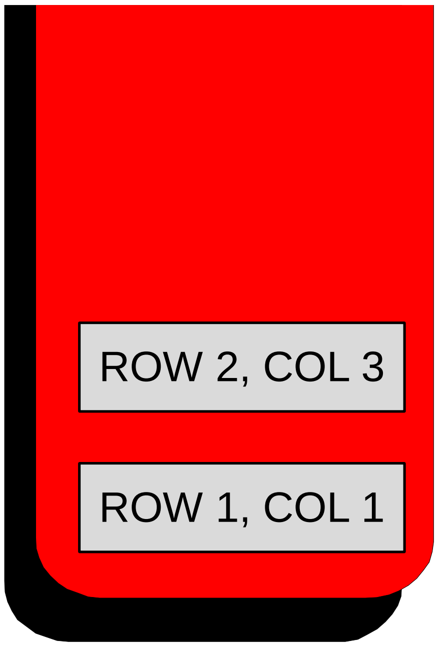
...không có vị trí đặt con hậu ở dòng 3.

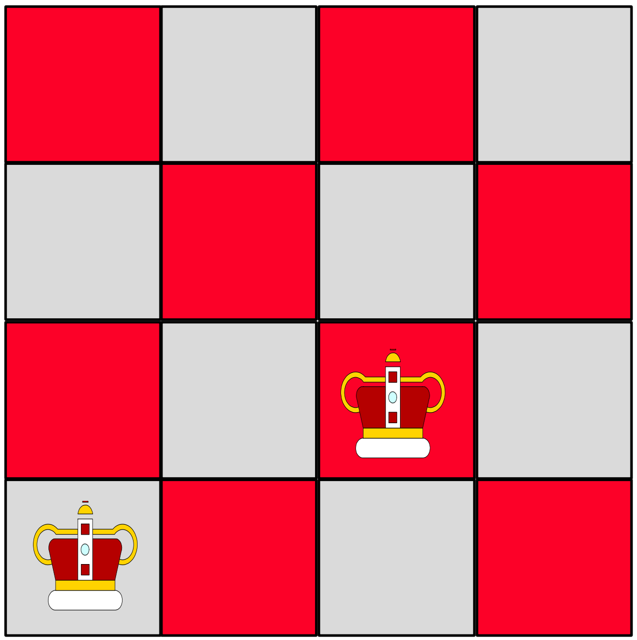


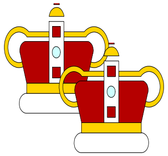
2 đã đặt

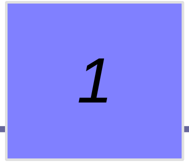
Thuật toán làm việc như thế nào

Quay lại dịch chuyển con hậu ở dòng 2





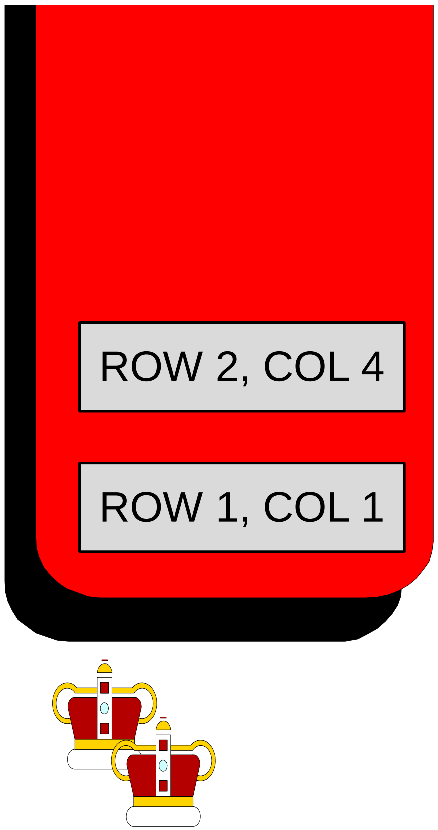


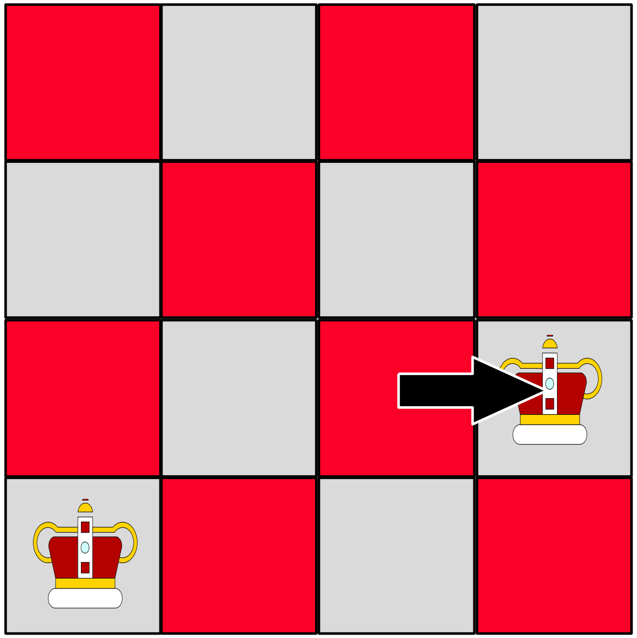


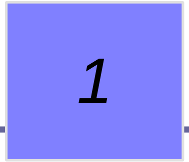
đã đặt

Thuật toán làm việc như thế nào

Đẩy con hậu ở dòng 2 sang cột thứ 4.



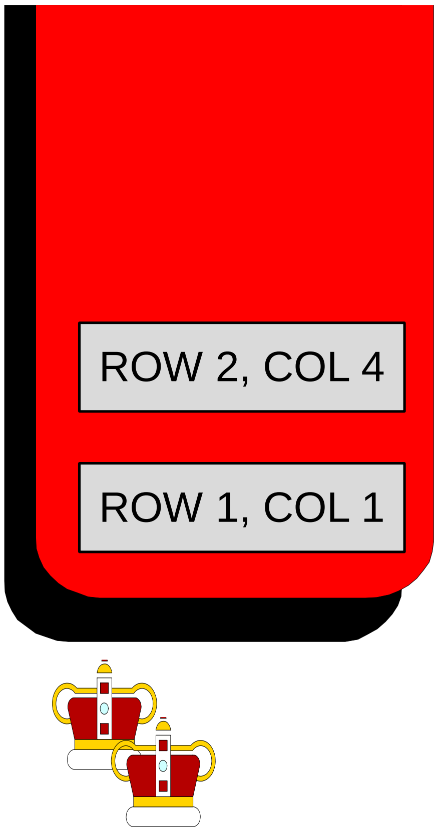


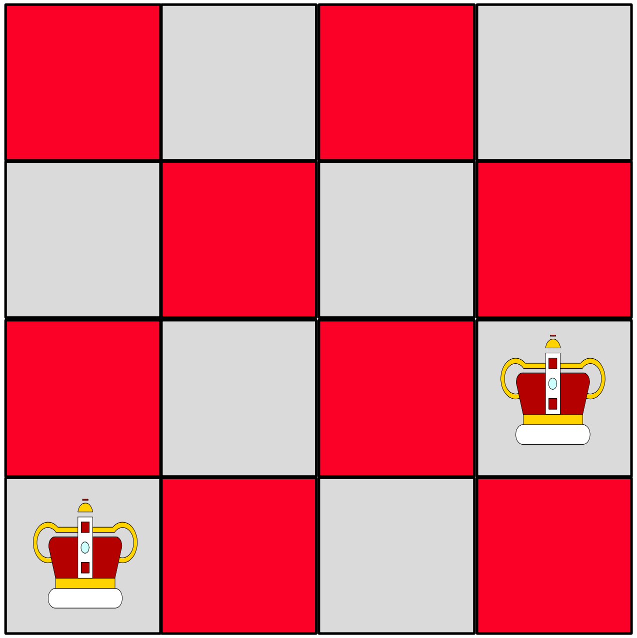


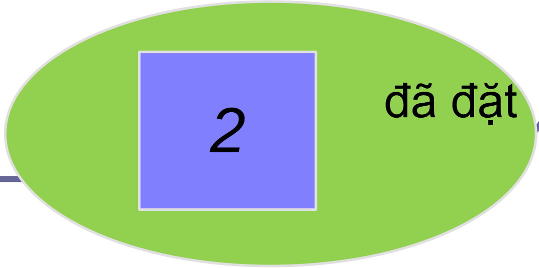
đã đặt

Thuật toán làm việc như thế nào

Xếp được con hậu ở dòng 2 ta tiếp tục xếp con hậu ở dòng 3

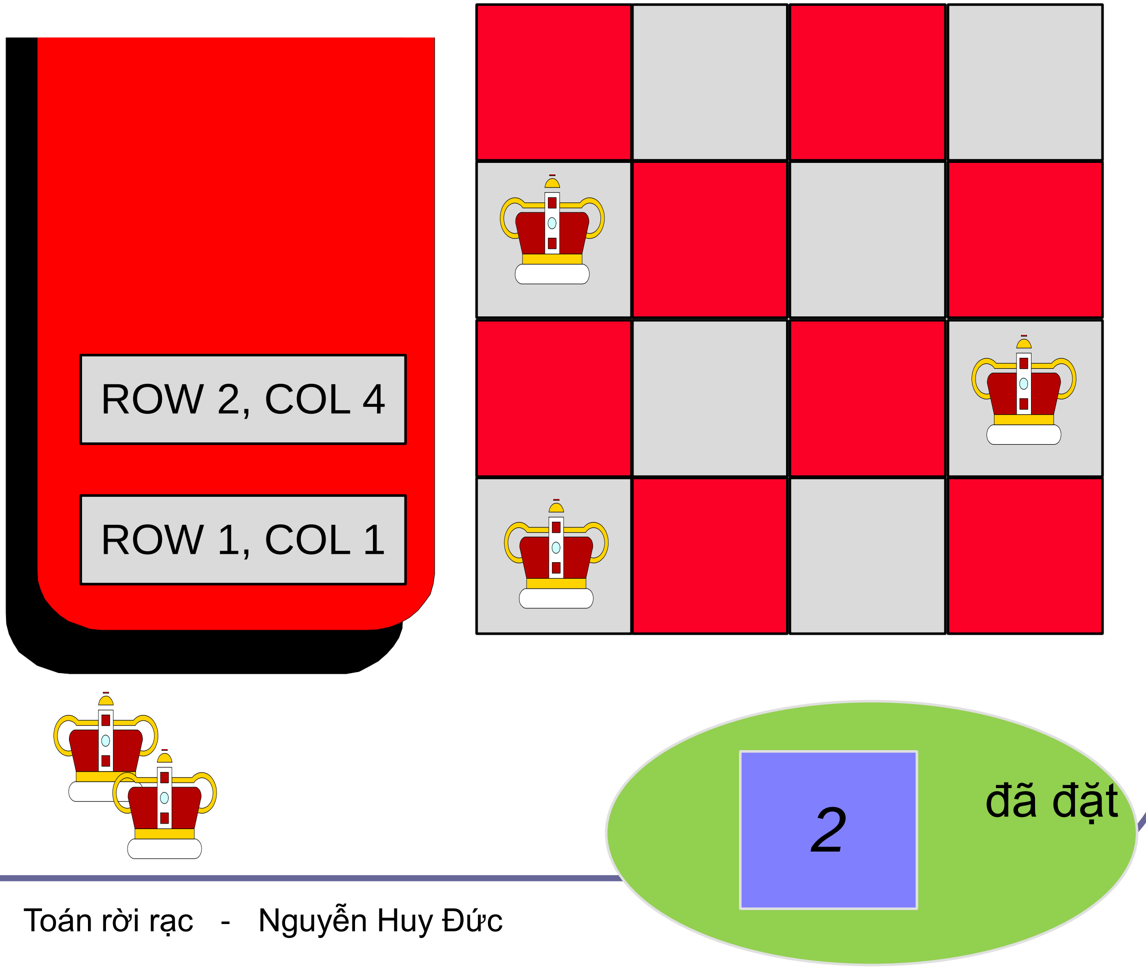






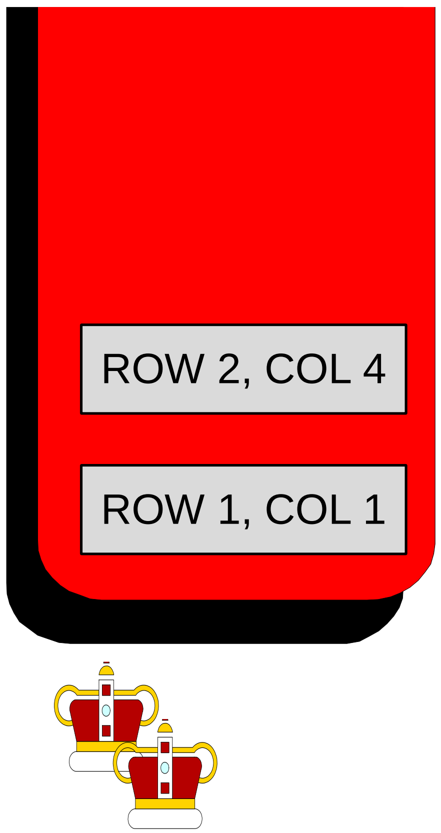
Thuật toán làm việc như thế nào

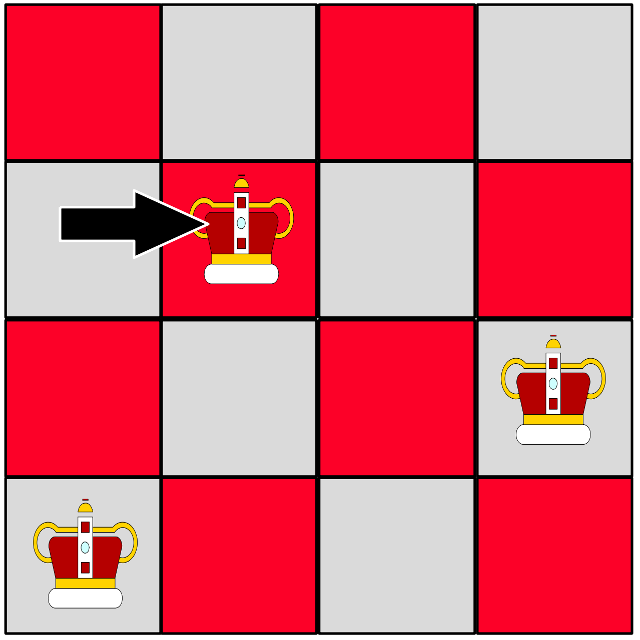
Thử xếp con hậu ở dòng 3 vào cột 1

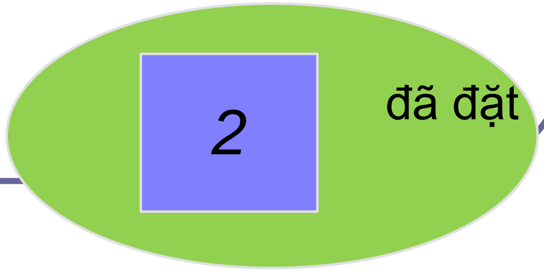


Thuật toán làm việc như thế nào

Thử xếp con hậu ở dòng 3 vào cột 2

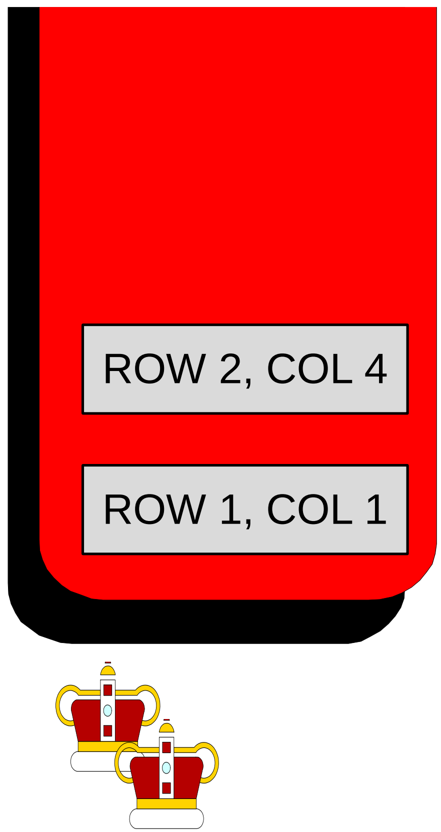


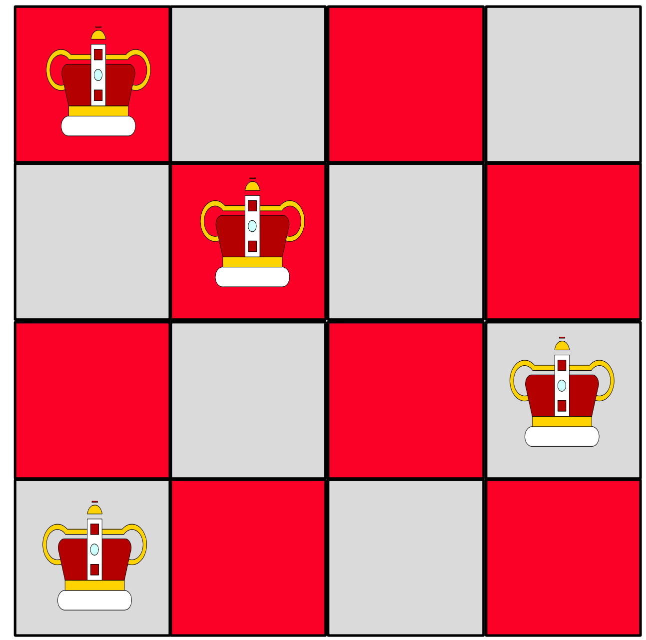


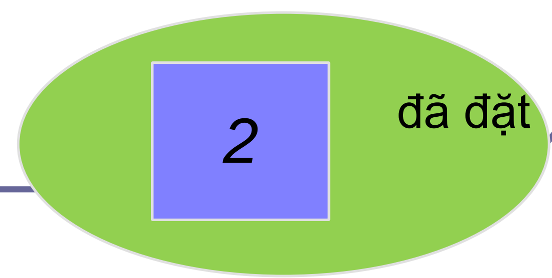


Thuật toán làm việc như thế nào

Xếp được con hậu ở dòng 3 ta tiếp tục xếp con hậu ở dòng 4: Thử cột 1

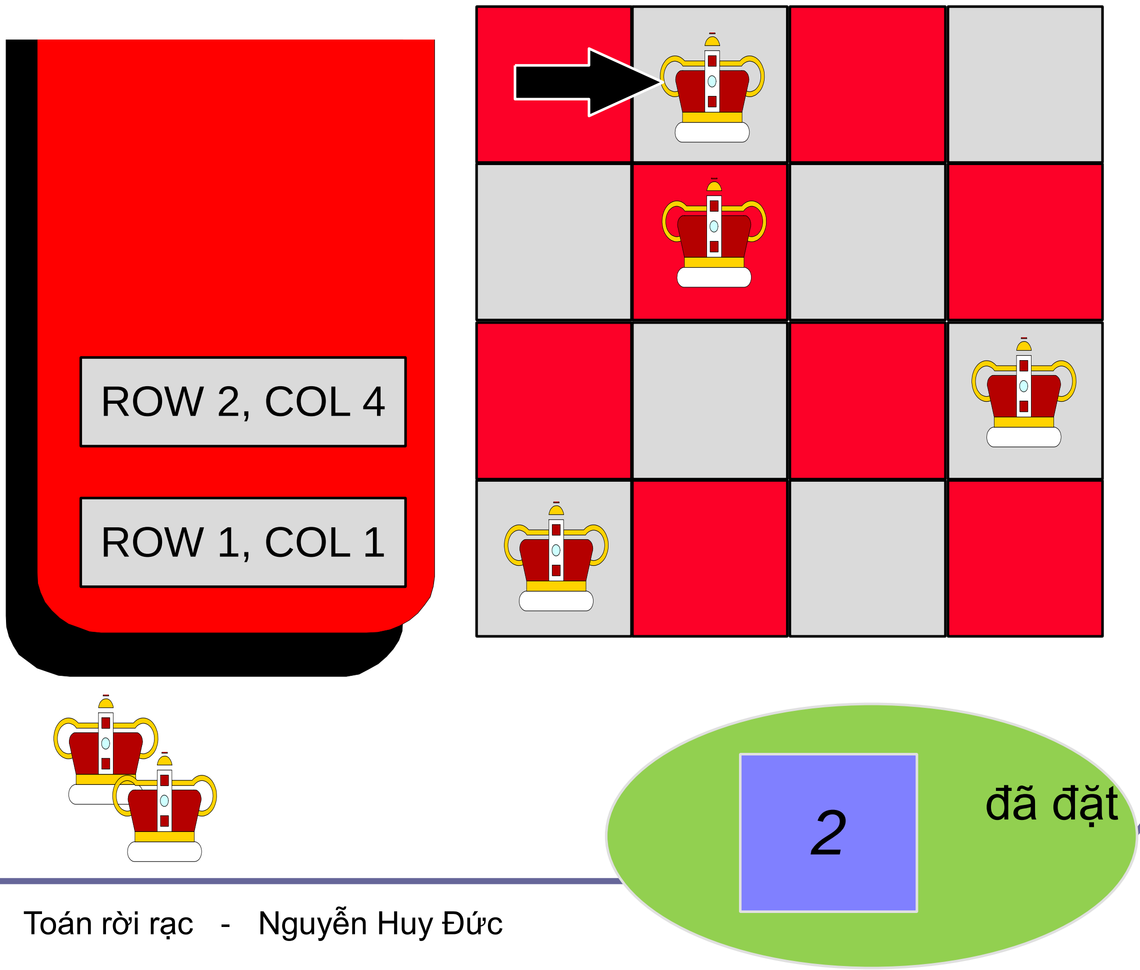






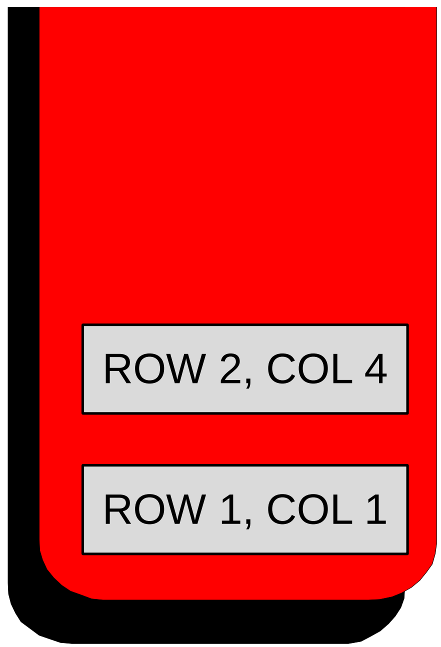
Thuật toán làm việc như thế nào

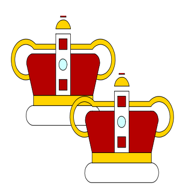
Thử xếp được con hậu ở dòng 4 vào cột 2

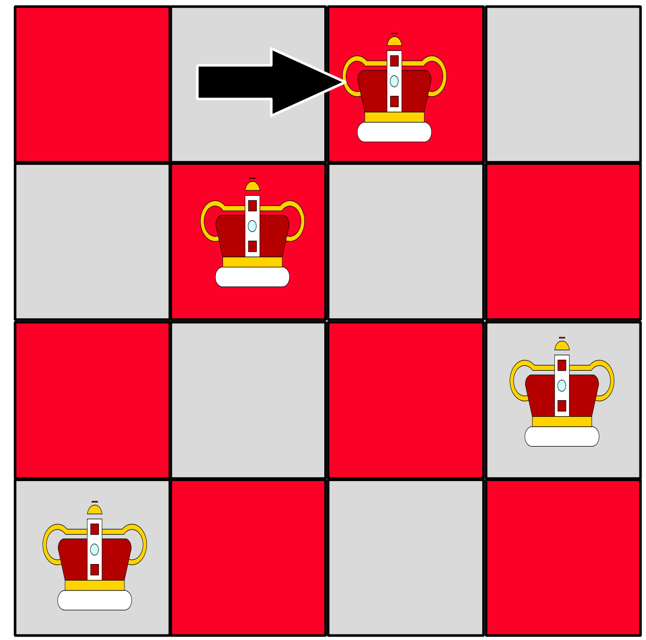


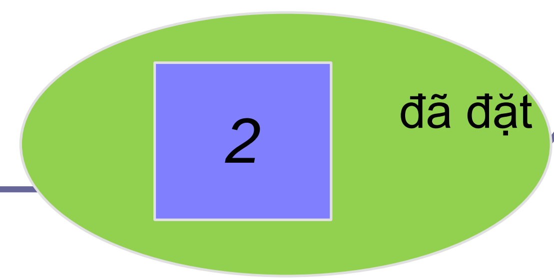
Thuật toán làm việc như thế nào

Thử xếp được con hậu ở dòng 4 vào cột 3



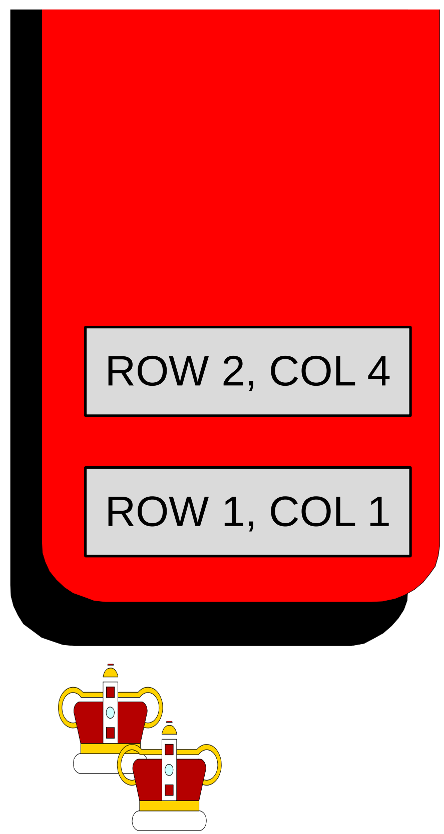


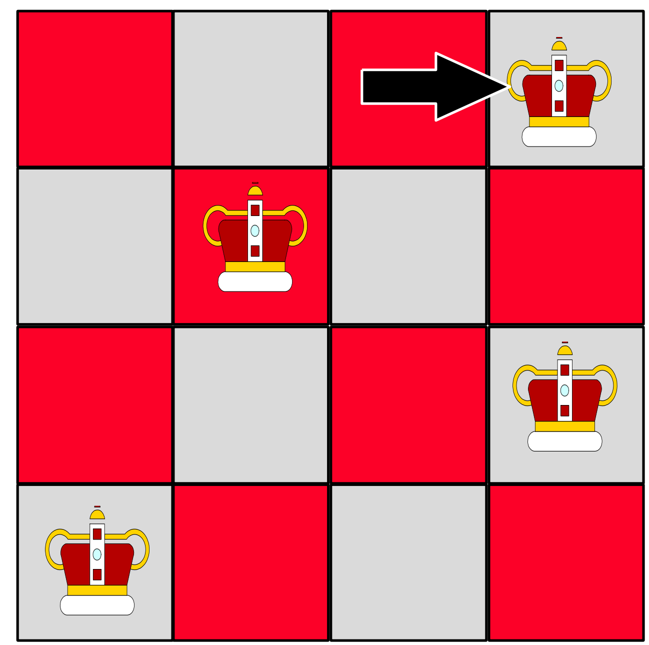


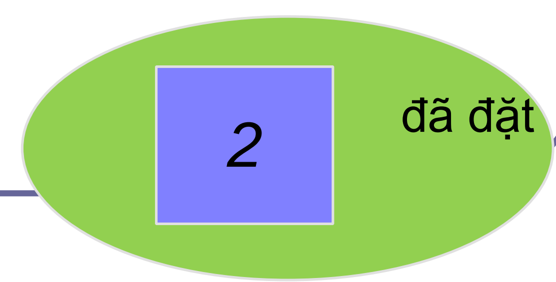


Thuật toán làm việc như thế nào

Thử xếp được con hậu ở dòng 4 vào cột 4

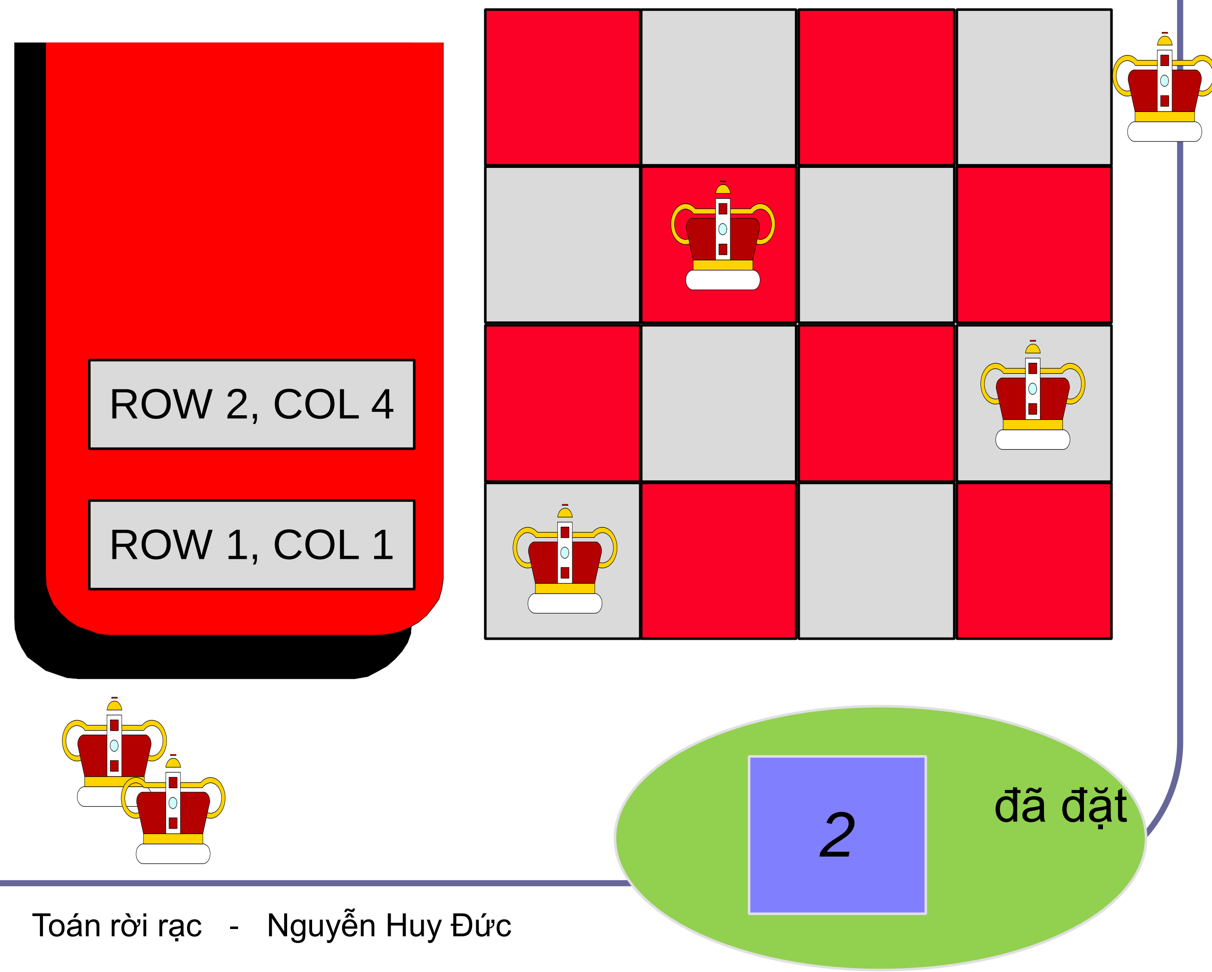






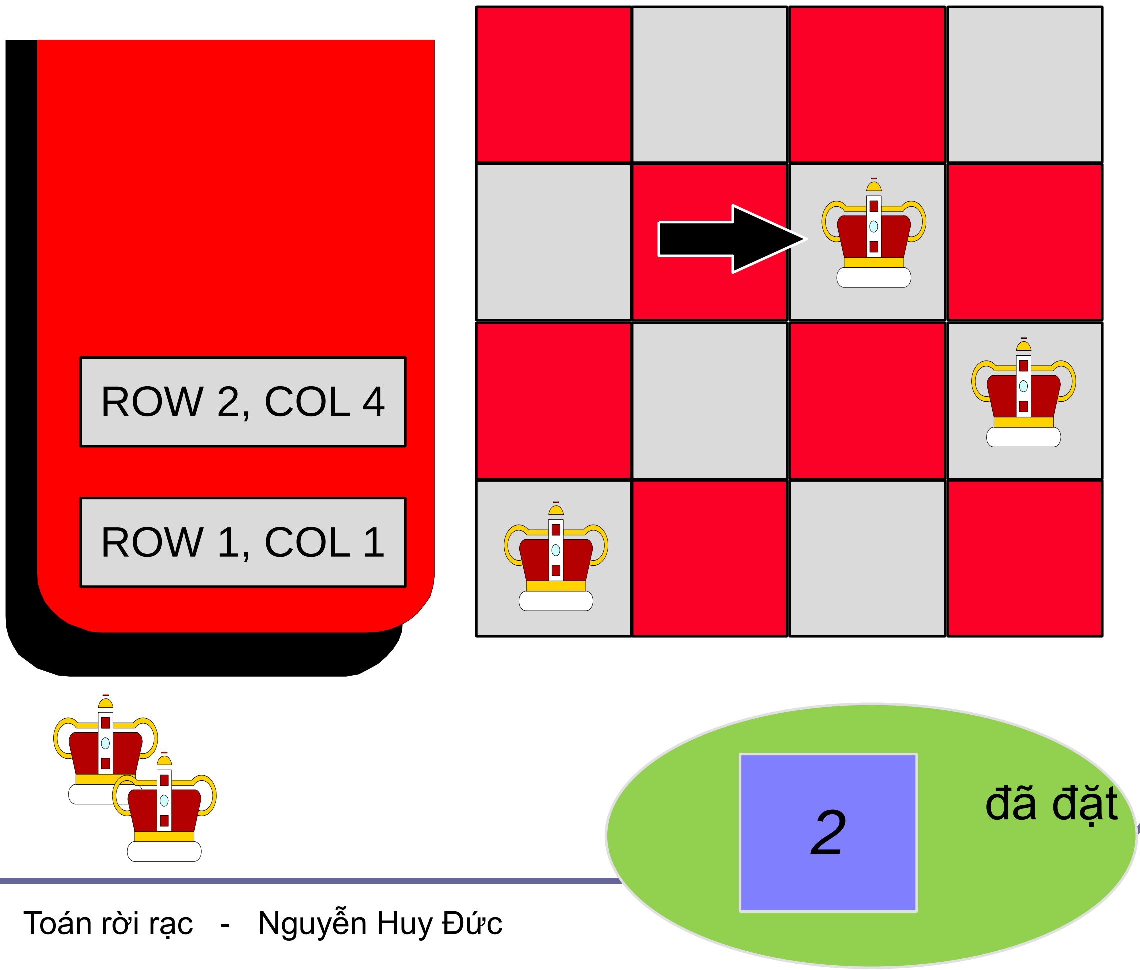
Thuật toán làm việc như thế nào

Không xếp được
con hậu ở dòng 4



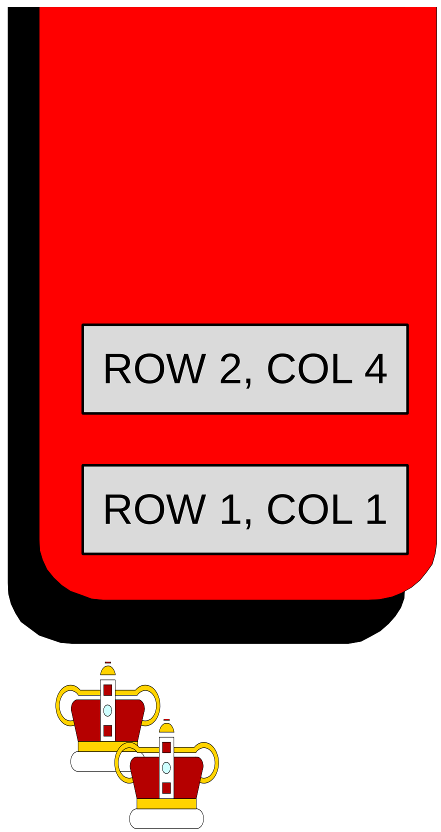
Thuật toán làm việc như thế nào

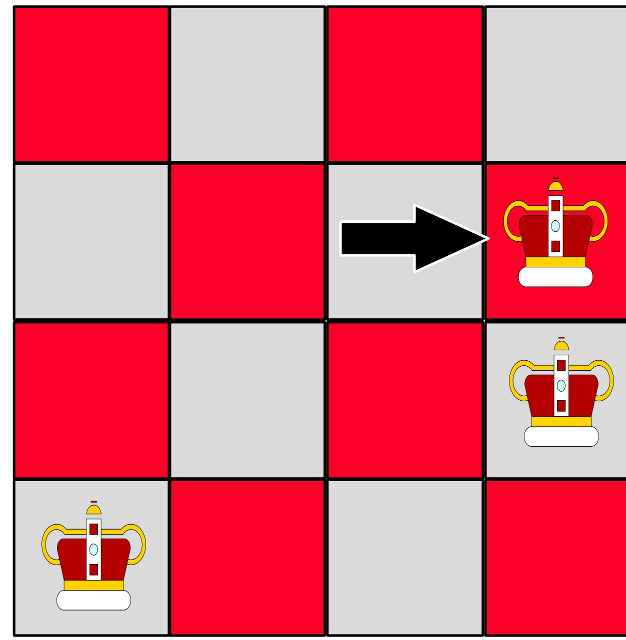
Quay lại tìm vị trí mới cho con hậu ở dòng 3: Thử cột 3

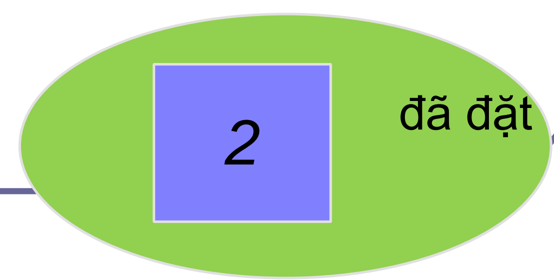


Thuật toán làm việc như thế nào

Quay lại tìm vị trí mới cho con hậu ở dòng 3: Thử cột 4

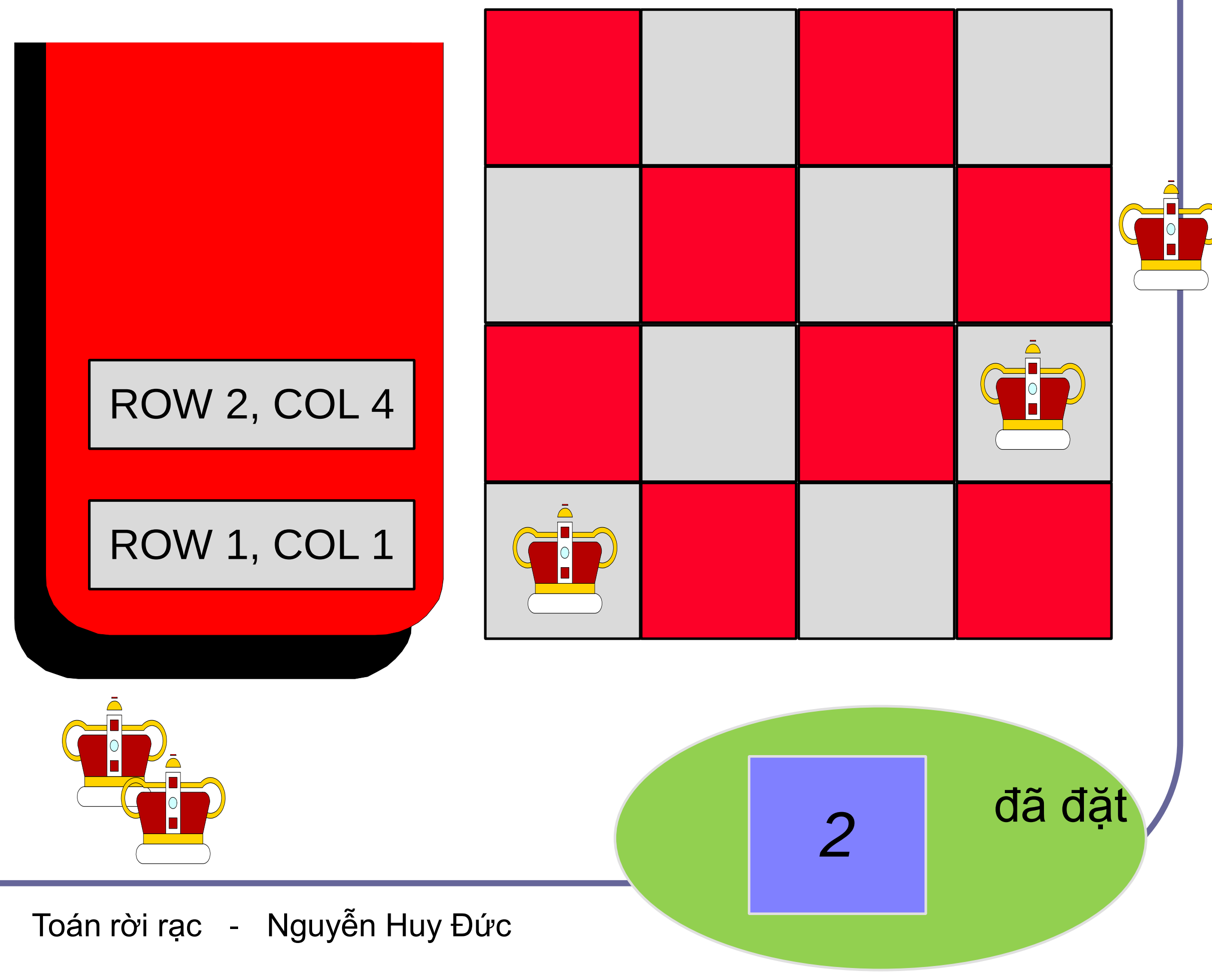






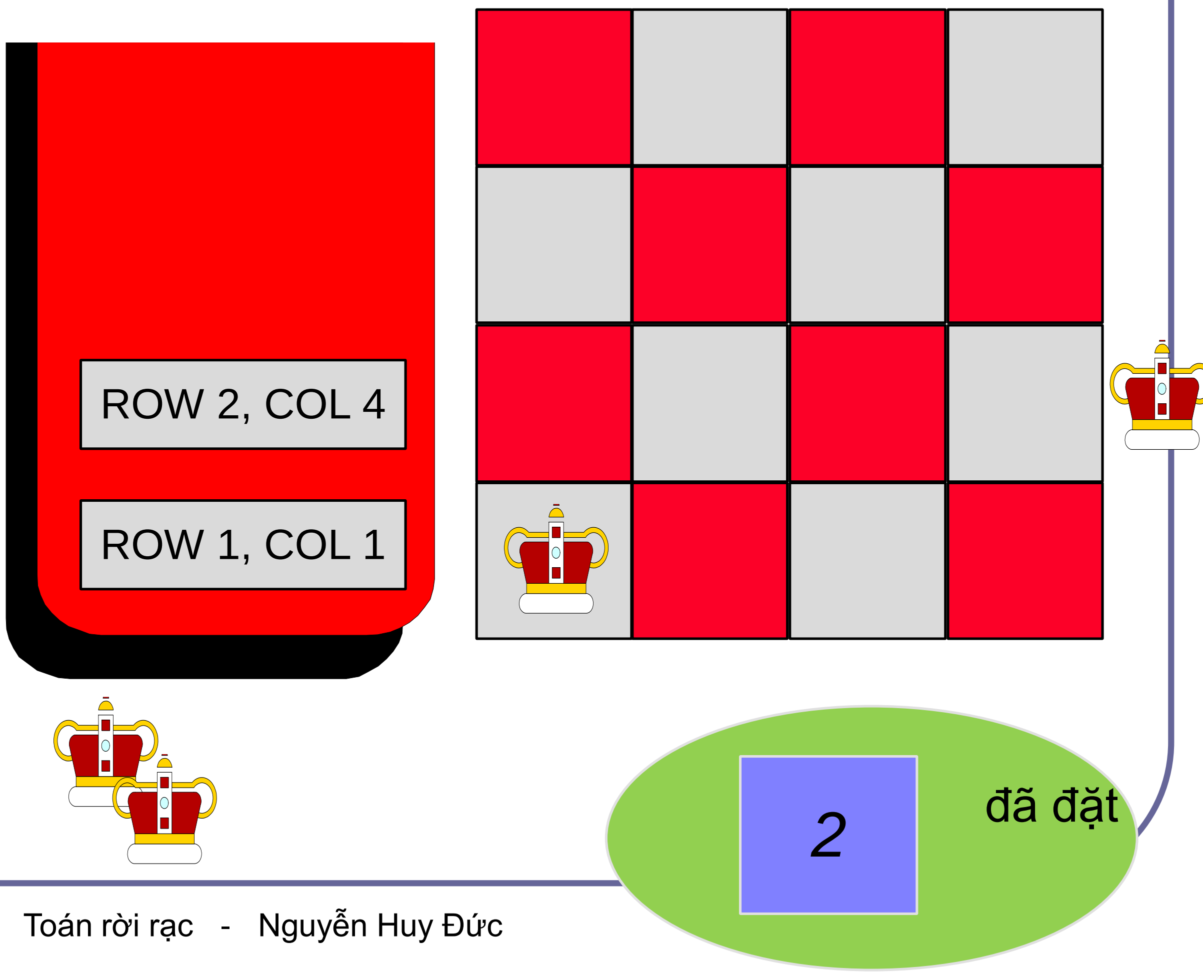
Thuật toán làm việc như thế nào

Không có cách xếp mới cho con hậu ở dòng 3



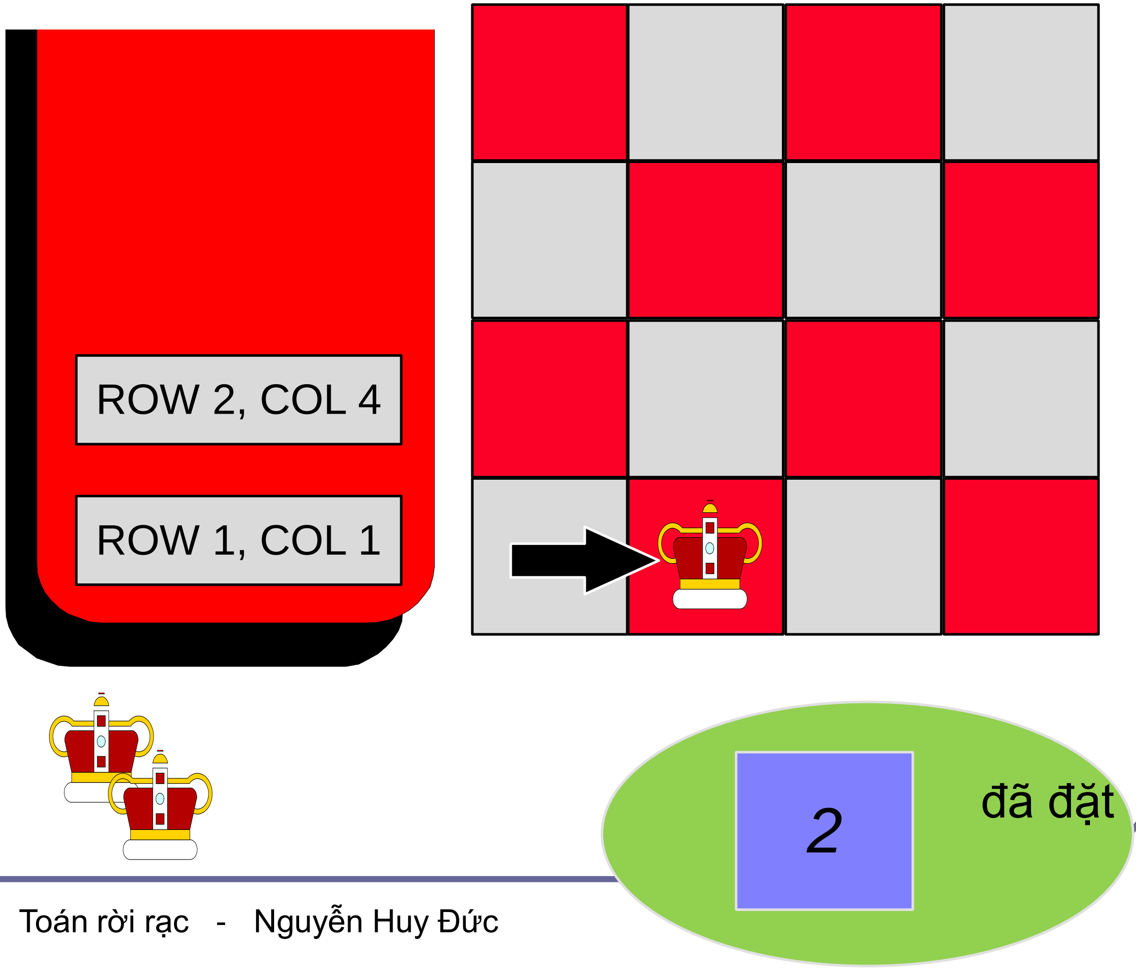
Thuật toán làm việc như thế nào

Quay lại tìm cách xếp mới cho con hậu ở dòng 2: Không có



Thuật toán làm việc như thế nào

Quay lại tìm cách xếp mới cho con hậu ở dòng 1: Chuyển sang cột 2



Một lời giải của bài toán xếp hậu khi *n* = 8