REVIEW

- Xác định mối quan hệ giữa các cặp hàm $f(n)$ và $g(n)$ sau đây

a) $f(n) = n^{2.5} + 3(n-1), g(n) = 6n$

b) $f(n) = n^2 + 3\sqrt{n} - 1), g(n) = n^3$

c) $f(n) = 2n^{2.5} + n, g(n) = \sqrt{n^5}$

THUẬT TOÁN ĐỆ QUY

Nội dung

- Định nghĩa đệ quy
- Thuật toán đệ quy
- Phân tích thuật toán đệ quy
- Đệ quy có nhớ
- Thuật toán quay lui (backtracking algorithm)

Định nghĩa đệ quy

- ***Đối tượng bao gồm chính nó hoặc được định nghĩa dưới dạng chính nó***.

VD. Định nghĩa một ***công thức hợp lệ*** của các biến, số và các phép toán $\{+, -, *, /, \hat{}\}$

- x là công thức hợp lệ nếu x là biến hoặc số
- Nếu f, g là công thức hợp lệ thì $(f+g), (f-g), (f*g), (f/g), (f\hat{}g)$ cũng là công thức hợp lệ

Elena Filippova. Recursion. 2003
Acrylic on Canvas, 160 cm x 160 cm (63" x 63")

Hàm được định nghĩa đệ quy

- $n! = \begin{cases} 1 & nếu\ n = 0 \\ n \times (n-1)! & nếu\ n > 0 \end{cases}$
- $Fib(n) = \begin{cases} 1 & nếu\ n = 1, 2 \\ Fib(n-1) + Fib(n-2) & nếu\ n > 2 \end{cases}$
- $P(n) = \begin{cases} 0 & nếu\ n = 0 \\ 1 & nếu\ n = 1 \\ 2P(n-1) + P(n-2) & nếu\ n > 1 \end{cases}$

Định nghĩa đệ quy

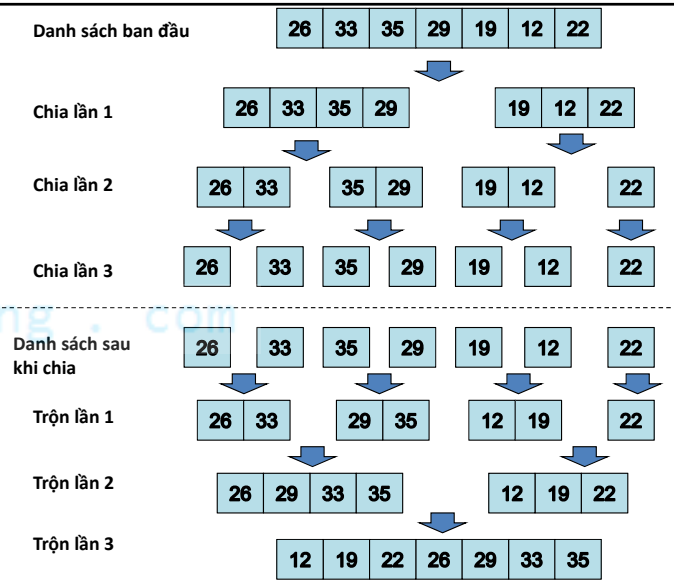
- Mọi định nghĩa đệ quy đều gồm 2 phần
  - Một trường hợp cơ sở (nhỏ nhất) có thể xử lý trực tiếp mà không cần đệ quy, và
  - Một phương thức tổng quát mà biến đổi một trường hợp cụ thể về các trường hợp nhỏ hơn. Do đó biến đổi các trường hợp cho đến khi về trường hợp cơ sở.

Thuật toán đệ quy

Thuật toán có chứa lời gọi đệ quy đến chính nó với đầu vào kích thước nhỏ hơn.

- VD. Sắp xếp trộn – MergeSort

```
MergeSort(int A[], int start, int end)
{
   if(start<end)
   {
         int mid = (start+end)/2;
         MergeSort(A, start, mid);
         MergeSort(A, mid+1, end);
         Merge(A,start,mid,end);
   }
}
```

Thuật toán đệ quy

- Mô tả thời gian thực hiện của thuật toán đệ quy bằng công thức đệ quy
- VD. MergeSort có

$$T(n) = \begin{cases} O(1) & nếu\ n = 1 \\ 2T\left(\frac{n}{2}\right) + O(n) & nếu\ n > 1 \end{cases}$$

Bỏ qua $T(n)$ với các giá trị n nhỏ (coi là hằng). Ta có thể viết lại $T(n)$ là

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

Phân tích thuật toán đệ quy

- Giải công thức đệ quy để tìm Θ hoặc O bằng:
  - Phương pháp thay thế
  - Phương pháp cây đệ quy
  - Dùng định lý thợ

Phương pháp thay thế

Phương pháp thay thế

- Gồm 2 bước:
  - Đoán dạng của lời giải
  - Sử dụng quy nạp toán học để tìm ra các hằng và chứng minh lời giải
- Xác định cận trên của công thức đệ quy

$$T(n) = 2T(\lfloor n/2 \rfloor) + n$$

Đoán $T(n) = O(n\log n)$

Cần chứng minh $T(n) \leq cn\log n$ với hằng số $n > 0$ được chọn phù hợp

Phương pháp thay thế

- Giả sử $T(n) \le cn\log n$ đúng với $\lfloor n/2 \rfloor$ tức là $T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \log \lfloor n/2 \rfloor$. Thay vào $T(n)$

$$T(n) \le 2(c\lfloor n/2 \rfloor \log \lfloor n/2 \rfloor) + n$$
$$\le cn\log n/2 + n = cn\log n - cn\log 2 + n$$
$$\le cn\log n$$

Đúng với $c \ge 1$

Ta cần chỉ ra kết quả quy nạp này đúng trong mọi trường hợp (đúng cả trong trường hợp cơ sở).

Phương pháp thay thế

- Giả sử trường hợp cơ sở $T(1) = 1$ nhưng $c1\log 1 = 0$. Kết quả quy nạp sai trong trường hợp cơ sở.
- Ta có thể giải quyết vấn đề này khi sử dụng các ký hiệu tiệm cận (O, Ω, Θ)
  $T(n) = cn\log n$ với $n \ge n_0$
- Chọn n_0 sao cho với mọi $n \ge n_0$ thì kết quả luôn đúng VD với $n = 2$ thì $T(2) = 2T(1) + 2 = 4 < c2\log 2$ với hằng số c ta chọn đủ lớn (VD $c = 5$).
- Vậy $T(n) = O(n\log n)$ với $c = 5$ và $n \ge 2$

Phương pháp thay thế

Đoán dạng lời giải tốt:

- Thêm bớt 1 hằng số không làm thay đổi dạng kết quả $T(n) = 2T\left(\frac{n}{2} + 12\right) + 3n$ vẫn có dạng $O(n\log n)$
- Ban đầu nới lỏng cận trên, dưới để chứng minh rồi sau đó giảm dần.

VD. Với $T(n)$ trong ví dụ ban đầu ta có thể chọn $\Omega(n)$ và $O(n^2)$ rồi sau đó giảm giới hạn trên, tăng giới hạn dưới cho tới khi hội tụ về giá trị chính xác

Phương pháp thay thế

- Tránh lỗi hay mắc

$$T(n) \le 2\left(c\lfloor n/2 \rfloor\right) + n \le cn + n = O(n)$$

Sai do ta không chứng minh $T(n) \le cn$

- Thay đổi biến

$$T(n) = 2T(\sqrt{n}) + \log n$$

đặt $m = \log n$ ta có $T(2^m) = 2T\left(2^{m/2}\right) + m$

đặt $S(m) = T(2^m)$ ta có $S(m) = 2S(m/2) + m = O(m\log m) = O(\log n \log\log n)$

- Ví dụ. Chứng minh rằng

$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + 1$ là $O(\log n)$

Một số tính chất của hàm mũ, loga, giai thừa

- Ta có các công thức:

$$a = b^{\log_b a} \; ; \quad \log_b a = 1/(\log_a b)$$

- Do đó, trong ký hiệu tiệm cận cơ số của log là không quan trọng:

$$O(\lg n) = O(\ln n) = O(\log n)$$

- Công thức Stirling:

$$n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right)$$

- Giai thừa và hàm mũ:

$$2^n < n! < n^n \quad \text{với } n > 5 \; ;$$

$$\log n! = \Theta(n \log n).$$

Vấn đề với phương pháp thay thế

T(n) = 1	n=1
T(n) = 4×T(n/2) + n	n>1

Dự đoán (chặt hơn!):
$T(n) \le c \times n^2 \quad \forall n > n_0$

Chuyển qui nạp:

Giả sử $T(k) \le c \times k^2$, $\forall k < n$. Chứng minh $T(n) \le c \times n^2$.

$T(n) = 4 \times T(n/2) + n$
$\le 4 \times c \times (n/2)^2 + n$
$= c \times n^2 + n$

Không $\le c \times n^2$!

Ví dụ 2 (tiếp)

T(n) = 1	n=1
T(n) = 4×T(n/2) + n	n>1

Sử dụng dự đoán chính xác hơn.

Trừ bớt đi một số hạng tăng chậm (là một kỹ thuật hay dùng).

Dự đoán:
$T(n) \le c \times n^2 - d \times n \quad \forall n > n_0$

Giả sử $T(k) \le c \times k^2 - d \times n$, $\forall k < n$. Cần CM $T(n) \le c \times n^2 - d \times n$.

Ví dụ 2 (tiếp)

$T(n) = 1 \quad n=1$
$T(n) = 4\times T(n/2) + n \quad n>1$

Dự đoán: $T(n) \le c\times n^2 - d\times n \quad \forall n>n_0$

Giả sử $T(k) \le c\times k^2 - d\times n$, $\forall k<n$. CM $T(n) \le c\times n^2 - d\times n$.

Khi n=1:

$T(n) = 1$ Theo định nghĩa.

$1 \le c-d$ Có thể chọn c, d thích hợp để có bất đẳng thức này

Ví dụ 2 (tiếp)

$T(n) = 1 \quad n=1$
$T(n) = 4\times T(n/2) + n \quad n>1$

Dự đoán: $T(n) \le c\times n^2 - d\times n \quad \forall n>n_0$

Giả sử $T(k) \le c\times k^2 - d\times n$, $\forall k<n$. CM $T(n) \le c\times n^2 - d\times n$.

Chuyển qui nạp, n>1:

$T(n) = 4\times T(n/2) + n$	(định nghĩa)
$\le 4\times(c\times(n/2)^2 - d\times(n/2)) + n$	(qui nạp)
$= c\times n^2 - 2\times d\times n + n$	(biến đổi)
$= c\times n^2 - d\times n - (d\times n - n)$	(biến đổi)
$\le c\times n^2 - d*n$	(Chọn $d\ge 1$)

Ví dụ 2 (tiếp)

$T(n) = 1 \quad n=1$
$T(n) = 4\times T(n/2) + n \quad n>1$

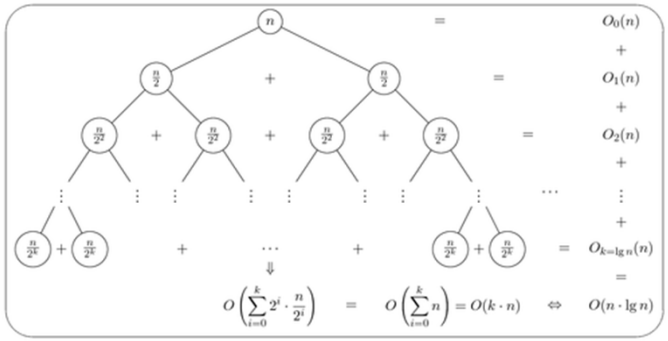
Đã chứng minh:

$T(n) \le 2\times n^2 - 1\times n \quad \forall n>0$

Vậy, $T(n) = O(n^2)$.

Phương pháp cây đệ quy

Phương pháp cây đệ quy

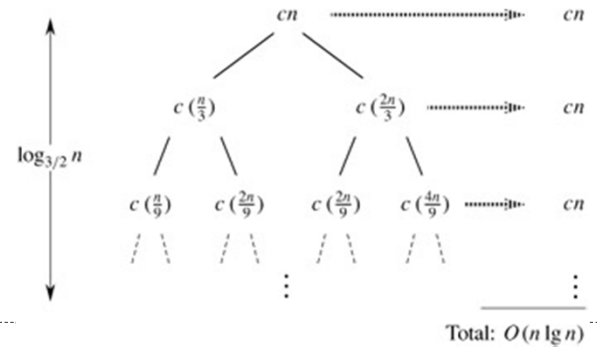


- Cây đệ quy cho mergeSort

$$T(n) = \begin{cases} O(1) & \text{nếu } n = 1 \\ 2T\left(\frac{n}{2}\right) + O(n) & \text{nếu } n > 1 \end{cases}$$

Phương pháp cây đệ quy

- Xét công thức đệ quy $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n)$



Phương pháp cây đệ quy

- Dùng phương pháp thay thế để chứng minh lời giải công thức đệ quy tìm được.

  VD. $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n) = O(n\log n)$

- $T(n) \le T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$

$$\le d\frac{n}{3}\log\frac{n}{3} + d\frac{2n}{3}\log\frac{2n}{3} + cn$$

$$= \frac{dn}{3}\log n - \frac{dn}{3}\log 3 + \frac{2dn}{3}\log 2n - \frac{2dn}{3}\log 3 + cn$$

$$= dn\log n - dn\log 3 + \frac{2dn}{3}\log 2 + cn$$

$$\le dn\log n$$

Với $d \ge \frac{c}{\log 3 - \frac{2}{3}}$ (chú ý $\log 2 = 1$)

Phương pháp cây đệ quy

- **Bài tập**: Xác định một cận trên tốt cho công thức đệ quy

$$T(n) = 3T\left(\frac{n}{2}\right) + n$$

dùng phương pháp thế để xác nhận lại kết quả.

Định lý thợ
Master theorem

Dùng định lý thợ

- Dùng để giải các công thức đệ quy dạng

$$T(n) = aT\left(\frac{n}{b}\right) + f(n),$$
$$trong\ đó\ a \geq 1, b > 1, f(n)\ là\ hàm\ tiệm\ cận\ dương$$

một cách hiệu quả.

- Bài toán ban đầu được chia thành $\boldsymbol{a}$ bài toán con có kích thước mỗi bài là $\boldsymbol{^n/_b}$, chi phí để tổng hợp các bài toán con là $\boldsymbol{f(n)}$

VD. Thuật toán sắp xếp trộn
chia thành 2 bài toán con, kích thước $n/2$. Chi phí tổng hợp 2 bài toán con là $O(n)$

Dùng định lý thợ

Định lý thợ (Master Theorem)

$a \geq 1, b > 1$ là các hằng số, $f(n)$ là một hàm. $T(n)$ định nghĩa đệ quy trên các tham số không âm
$T(n) = aT(^n/_b) + f(n)$, trong đó $^n/_b$ có thể hiểu là $\lfloor ^n/_b \rfloor$ hoặc $\lceil ^n/_b \rceil$. Thì $T(n)$ có thể bị giới hạn một cách tiệm cận như sau:

- Nếu $f(n) = O(n^{\log_b a - \epsilon})$, với hằng $\epsilon > 0$ thì $T(n) = \Theta(n^{\log_b a})$
- Nếu $f(n) = O(n^{\log_b a})$ thì $T(n) = \Theta(n^{\log_b a} \log n)$
- Nếu $f(n) = \Omega(n^{\log_b a + \epsilon})$, với hằng $\epsilon > 0$, và nếu $af(^n/_b) < cf(n)$ với hằng $c < 1$ và với mọi n đủ lớn thì
$$T(n) = \Theta(f(n))$$

Dùng định lý thợ

Áp dụng định lý thợ:

- $T(n) = 9T\left(\frac{n}{3}\right) + n.$
$a = 9, b = 3$ và $f(n) = n$ ta có $n^{\log_b a} \equiv n^{\log_3 9} = n^2$. Đây là trường hợp 1 (với $\epsilon = 1$) do đó $T(n) = \Theta(n^2)$
- $T(n) = T\left(\frac{2n}{3}\right) + 1.$
$a = 1, b = 3/2$ và $f(n) = 1$ ta có $n^{\log_{3/2} 1} = n^0 = 1$. Đây là trường hợp 2, do đó $T(n) = \Theta(\log n)$
- $T(n) = 3T\left(\frac{n}{4}\right) + n \log n$
$a = 3, b = 4$ và $f(n) = n \log n$ ta có $n^{\log_b a} \equiv n^{\log_4 3} = O(n^{0.793})$, $f(n) = \Omega(n^{\log_4 3 + \epsilon})$ với $\epsilon \approx 0.2$, $af(^n/_b) < cf(n) \equiv 3f\left(\frac{n}{4}\right) < cf(n \log n)$ với $c = \frac{3}{4}$ do vậy $T(n) = \Theta(n \log n)$ (TH3)

Dùng định lý thợ

- **Chú ý**: Không phải trường hợp nào cũng áp dụng được định lý thợ !
- VD. $T(n) = 2T\left(\frac{n}{2}\right) + n\log n$
  $a = 2, b = 2$ và $f(n) = n\log n$
  $n^{\log_b a} \equiv n^{\log_2 2} = n$ do đó có vẻ áp dụng trường hợp 3.
  Tuy nhiên $f(n) = n\log n$ tiệm cận lớn hơn $2f\left(\frac{n}{2}\right)$ với mọi hằng số ϵ do đó không thể áp dụng được.

Đệ quy có nhớ

Đệ quy có nhớ

- Trong thuật toán đệ quy, những bài toán con có thể được giải đi giải lại nhiều lần!
- VD. Tính số Fibonacci

$$f(n) = \begin{cases} 1 & nếu\ n = 0,1 \\ f(n-1) + f(n-2) & nếu\ n \geq 2 \end{cases}$$

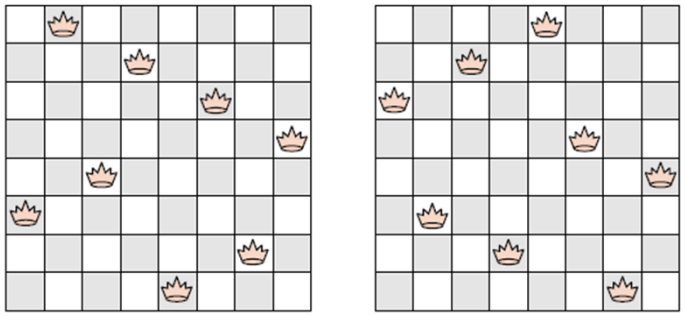
  Tính $f(5)$

- **Ghi nhận lời giải: dùng mảng**
- Khi gặp bài toán con cần giải: Kiểm tra xem bài toán con đã được giải chưa:
  - Nếu đã giải: lấy kết quả
  - Ngược lại, giải bài toán con và cập nhật lời giải vào bảng

Thuật toán quay lui
Back-tracking algorithm

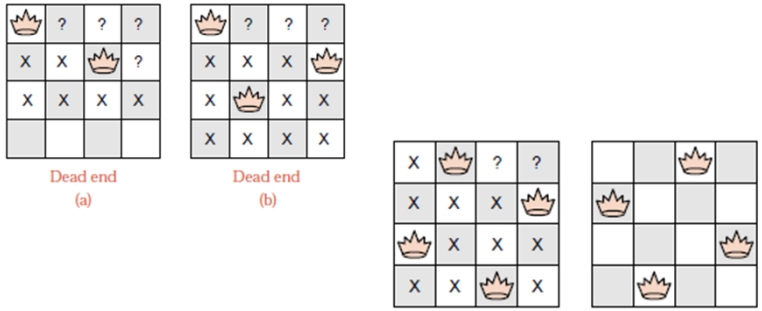
Thuật toán quay lui

- Bài toán 8 con hậu: "Hãy xếp 8 con hậu trên bàn cờ 8x8 sao cho chúng không thể ăn lẫn nhau"



Thuật toán quay lui

- **Thuật toán xếp hậu**: đặt lần lượt các quân hậu lên bàn cờ (theo 1 cách nào đó) sao cho quân hậu đặt sau không ăn được quân đã đặt trước đó.

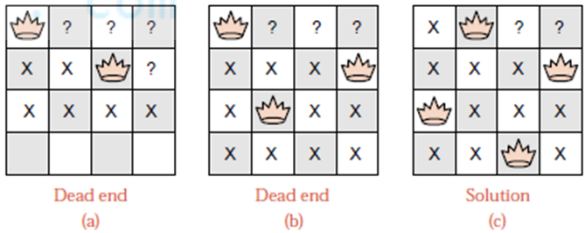


Thuật toán quay lui

solve_from (**Current_config**)
if Current_config đã chứa đủ 8 hậu
    print **Current_config**
else
  Với tập p các ô trên bàn cờ mà chưa bị ảnh hưởng bởi **Current_config**
{
  Thêm 1 quân hậu vào p;
  Cập nhật lại **Current_config**
  solve_from(**Current_config**);
  Loại bỏ quân hậu khỏi p của **Current_config**;
}

Thuật toán quay lui

- Dead end: trạng thái chưa kết thúc, nhưng ta không thể đặt thêm được 1 quân hậu nào nữa.
- Khi rơi vào trạng thái dead end ta phải tiến hành quay lui (backtrack) lại lựa chọn gần nhất để thử một khả năng có thể khác.



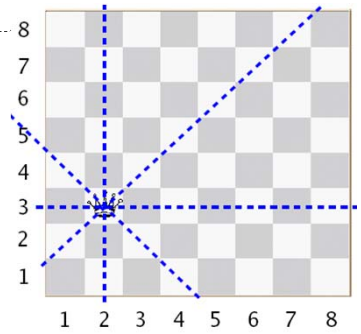
Thuật toán quay lui

Thuật toán quay lui – backtracking algorithm:

- Thử tìm kiếm lời giải đầy đủ cho bài toán từ việc xây dựng lời giải bộ phận, trong đó lời giải bộ phận phải luôn phù hợp với yêu cầu bài toán.
- Trong quá trình thực hiện, thuật toán mở rộng dần lời giải bộ phận. Nếu việc mở rộng khiến lời giải bộ phận vi phạm yêu cầu bài toán thì tiến hành quay lui, loại bỏ sửa đổi gần nhất và thử một khả năng xây dựng lời giải bộ phận có thể (hợp lệ) khác.

Bài toán 8 con hậu

- Nhận xét:
  - Mỗi cột phải có 1 con hậu
    - Con hậu 1 nằm trên cột 1
    - …
    - Con hậu j nằm trên cột j
    - …
    - Con hậu 8 nằm trên cột 8
  - Các con hậu phải không cùng hàng
  - Các con hậu phải không nằm trên đường chéo của nhau



Giải thuật

```
function Try (column) {
  for (row = 1; row <= 8; row++) {
      if ( [row, column] là an toàn) {
          Đặt con hậu vào vị trí [row, column];
          if (column == 8)
              In kết quả;
          else
              Try (column + 1);
          Xóa con hậu khỏi vị trí [row, column];
      }
  }
}
```

Thử lần lượt từng vị trí hàng

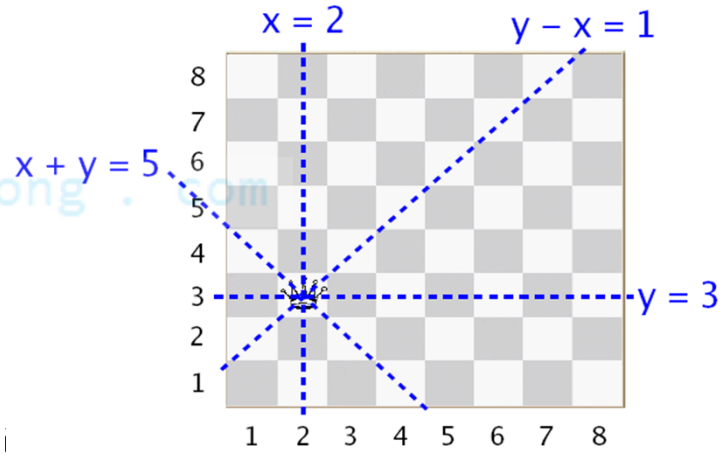
Nếu vị trí thử không bị con hậu nào tấn công

Con hậu thứ 8 là an toàn

Đệ quy để với con hậu tiếp

Xóa để tiếp tục thử vị trí [row+1, column]

Kiểm tra An toàn



cuu duong than cong . com