Khoa CNTT – ĐH Khoa học Tự nhiên
TP HCM, tháng 02/2022

Kỹ thuật lập trình

Đệ quy

Nguyễn Trọng Việt

Mục tiêu

Sau bài học, sinh viên có thể:

- Nêu khái niệm đệ quy và cấu trúc 1 hàm đệ quy
- Liệt kê ít nhất 3 loại đệ quy
- Liệt kê ít nhất 2 kỹ thuật liên quan đệ quy
- Ứng dụng đệ quy trong ít nhất 2 bài toán kinh điển

Nội dung

- Khái niệm đệ quy
- Phân loại đệ quy
- Các kỹ thuật liên quan
- Ứng dụng đệ quy

Nội dung

- Khái niệm đệ quy
- Phân loại đệ quy
- Các kỹ thuật liên quan
- Ứng dụng đệ quy

Khái niệm đệ quy ?

- Trong khoa học máy tính, đệ quy là 1 cách giải quyết bài toàn trong đó lời giải cho bài toán đang xét dựa trên lời giải của cùng bài toán đó nhưng với quy mô nhỏ hơn [1]
- Trong lập trình, đệ quy là quá trình 1 hàm chương trình gọi lại chính nó một cách trực tiếp hoặc gián tiếp [2]

[1] https://en.wikipedia.org/wiki/Recursion_(computer_science) (06/2020)
[2] https://www.geeksforgeeks.org/recursion/ (06/2020)

Cấu trúc 1 phát biểu đệ quy

- Trường hợp đơn giản
  - Lời giải của bài toán có quy mô nhỏ nhất
- Công thức truy hồi
  - Cách biểu diễn lời giải của bài toán hiện tại qua lời giải của bài toán con

Cấu trúc 1 phát biểu đệ quy

- Ví dụ:
  - Tính giai thừa
    - 0! = 1
    - n! = (n – 1)! * n, với mọi n nguyên và n > 0
  - Dãy Fibonacci
    - F(0) = 1
    - F(1) = 1
    - F(n) = F(n – 1) + F(n – 2), với mọi n nguyên và n > 1

Cấu trúc 1 hàm đệ quy

- Điều kiện dừng
  - Dừng gọi chính hàm đó trong trường hợp cơ bản
- Gọi hàm truy hồi
  - Thực hiện gọi lại chính hàm đó

Cấu trúc 1 hàm đệ quy

```
int factorial(int n)
{
   if ( n == 0 ) {
      return 1;
   }

   return factorial( n - 1 ) * n;
}
```

```
int fibonacci(int n)
{
   if ( n == 0 || n == 1 ) {
      return 1;
   }

   return fibonacci( n - 1 ) + fibonacci( n - 2 );
}
```

Nội dung

- Khái niệm đệ quy
- Phân loại đệ quy
- Các kỹ thuật liên quan
- Ứng dụng đệ quy

Phân loại đệ quy

- Đệ quy tuyến tính
- Đệ quy nhị phân
- Đệ quy tương hỗ
- Đệ quy phi tuyến

Phân loại đệ quy

- Đệ quy tuyến tính
- Đệ quy nhị phân
- Đệ quy tương hỗ
- Đệ quy phi tuyến

Đệ quy tuyến tính

- Thân hàm chứa duy nhất 1 lời gọi đệ quy
- Độ phức tạp O(n)

```
int factorial(int n)
{
   if ( n == 0 ) {
      return 1;
   }

   return factorial( n - 1 ) * n;
}
```

Phân loại đệ quy

- Đệ quy tuyến tính
- Đệ quy nhị phân
- Đệ quy tương hỗ
- Đệ quy phi tuyến

Đệ quy nhị phân

- Thân hàm chứa 2 lời gọi hàm
- Độ phức tạp: $O(2^n)$

```
int fibonacci(int n)
{
   if ( n == 0 || n == 1 ) {
      return 1;
   }

   return fibonacci( n - 1 ) + fibonacci( n - 2 );
}
```

Phân loại đệ quy

- Đệ quy tuyến tính
- Đệ quy nhị phân
- Đệ quy tương hỗ
- Đệ quy phi tuyến

Đệ quy tương hỗ

- Hàm `f1` gọi `f2`, trong `f2` gọi lại `f1`
- Độ phức tạp: phụ thuộc hàm tương hỗ
- Ví dụ: Tính số hạng thứ n

$$
\begin{aligned}
x_0 &= 1 \\
y_0 &= 0 \\
x_n &= 2x_{n-1} + 4y_{n-1} \\
y_n &= 5x_{n-1} + 3y_{n-1}
\end{aligned}
$$

Phân loại đệ quy

- Đệ quy tuyến tính
- Đệ quy nhị phân
- Đệ quy tương hỗ
- Đệ quy phi tuyến

Đệ quy phi tuyến

- Thân hàm chứa lời gọi đệ quy được đặt trong vòng lặp
- Độ phức tạp: O(n!)
- Ví dụ:

$$s_1 = 1$$
$$s_n = s_1 + s_2 + \ldots + s_{n-1}$$

$$x_0 = 1$$
$$x_n = n^2x_0 + (n-1)^2x_1 + \ldots + 1^2x_{n-1}$$

Nội dung

- Khái niệm đệ quy
- Phân loại đệ quy
- Các kỹ thuật liên quan
- Ứng dụng đệ quy

Công thức truy hồi

- Công thức truy hồi của 1 dãy A_n là công thức biểu diễn phần tử thứ n thông qua các phần tử trước của nó như $n - 1$, $n - 2$, …
- Ví dụ
  - Vi khuẩn A cứ mỗi giờ lại tự nhân đôi. Hỏi sau 6 giờ có bao nhiêu vi khuẩn A xuất hiện trên lam kính nếu ban đầu chỉ có 1 cặp vi khuẩn ?

$$A_0 = 2$$
$$A_n = 2 * A_{n-1}$$

Công thức truy hồi

- Công thức truy hồi của 1 dãy A_n là công thức biểu diễn phần tử thứ n thông qua các phần tử trước của nó như $n - 1$, $n - 2$, …
- Ví dụ
  - Anh B gửi ngân hàng 1000$, lãi xuất 15% năm. Hỏi sao 35 năm, anh B có được bao nhiêu tiền trong ngân hàng ?

$$B_0 = 1000$$
$$B_n = B_{n-1} + 0.15 * B_{n-1}$$

Đệ quy đuôi

- Tính:

  Sum(n) = 1 + 2 + … + n, với n > 0

```
int sum(int n)
{
   if ( n == 1 ) {
      return 1;
   }

   return sum( n - 1 ) + n;
}
```

Đệ quy đầu hay đuôi ?

Đệ quy đuôi

- Lời gọi đệ quy là chỉ thị cuối cùng được gọi thực hiện trong hàm
- Đệ quy đuôi được xem là tốt hơn đệ quy đầu
- Nhờ trình biên dịch tối ưu: vì lời gọi đệ quy đuôi là chỉ thị cuối cùng phải thực thi trong hàm, nên hàm hiện tại không còn cần thiết phải chiếm 1 slot trong call stack nữa

https://www.geeksforgeeks.org/tail-recursion/ (06/2020)

Đệ quy đuôi

- Tính:

  Sum(n) = 1 + 2 + … + n, với n > 0

```
int sum(int n, int total)
{
   if ( n == 1 ) {
      return total;
   }

   return sum( n - 1, total + n );
}
```

Đệ quy đuôi

- Tính:

  Giai thừa của n: n!

```
int factorial(int n, int fact)
{
   if ( n == 0 ) {
      return fact;
   }

   return factorial( n - 1, fact * n );
}
```

Kĩ thuật chia để trị

- Divide & conquer
- Ý tưởng
  - Nếu bài toán đủ nhỏ: giải quyết (**trị**) trực tiếp
  - Ngược lại:
    - **Chia** thành các bài toán nhỏ hơn
    - Lần lượt giải quyết từng bài toán nhỏ
    - Tổng hợp kết quả sau khi giải xong các bài toán nhỏ

Kĩ thuật chia để trị

```
function conquer(P)
start
   if P is small enough then
        solve P
        done
   end if

   divide P into smaller Pi
   conquer for each Pi
   build final solution
end
```

Kĩ thuật chia để trị

- Đếm số lượng phần tử mang giá trị âm trong mảng số nguyên

```
int countNegative(int* a, int left, int right)
{
   if ( left == right ) {
        return a[left] < 0 ? 1: 0;
   }

   int mid = (left + right)/2;
   int n_left = countNegative(a, left, mid);
   int n_right = countNegative(a, mid + 1, right);

   return n_left + n_right;
}
```

Kĩ thuật chia để trị

- Đếm số lượng phần tử mang giá trị âm trong mảng số nguyên

```
int countNegative(int* a, int n)
{
   int count_1 = a[0] < 0 ? 1: 0;
   if ( n == 1 ) {
        return count_1;
   }

   int count_2 = countNegative(a, n - 1);
   return count_1 + count_2;
}
```

Kĩ thuật chia để trị

- Sắp xếp mảng số nguyên tăng dần
- Giáo viên kiểm đếm bài thi sinh viên

Kĩ thuật chia để trị

- Ghi chú:

  Nếu độ lớn của bài toán con sau khi chia không nhỏ hơn nhiều so với bài toàn gốc thì không nên sử dụng kỹ thuật này để giải

Kĩ thuật lần ngược

- Back tracking
- Được nhà toán học Mỹ D. H. Lehmer đề xuất lần đầu tiên vào năm 1950
- Ý tưởng:
  - Xét bước hiện tại, nếu đã đến đích, ghi nhận lời giải
  - Ngược lại, nếu còn có thể bước tiếp
    - Chọn 1 bước để đi tiếp
    - Thử xem bước đó có dẫn đến lời giải
    - Quay lại bước hiện tại

Kĩ thuật lần ngược

```
tryToStep(step)
start
   if step is goal then
        dump the solution
        done
   end if

   while we can pick next step then
        pick the step'
        tryToStep(step')
        back to current step
   end while
end
```

Kĩ thuật lần ngược

- Cho mảng số nguyên dương
- Cho số nguyên K

Tìm tất cả các mảng con có tổng các phần tử bằng K

Kĩ thuật lần ngược

```
void findSubArray(int* a, int n, int start, int sum,
        int acc, bool* track) {
   if ( sum == acc ) {
        printSolution(a, track, start);  return;
   }

   for( int i = start ; i < n ; i++ ) {
        acc += a[i];
        track[i] = true;
        findSubArray(a, n, i + 1, sum, acc, track);
        acc -= a[i];
        track[i] = false;
   }
}
```

```
void printSolution(int* a, bool* track, int m) {
   for( int i = 0 ; i < m ; i++ ) {
        if ( track[i] )
            printf("%d ", a[i]);
   }
}
```

Kĩ thuật lần ngược

```
int main()
{
    int a[MAX_SIZE];
    bool track[MAX_SIZE] = {false};
    int n;
    int K;
    int S = 0;
    ...
    // input n, a, K
    ...

    findSubArray(a, n, 0, K, S, track);
}
```

Kĩ thuật khử đệ quy

- Ưu điểm
  - Thuật toán rõ ràng, sáng sủa
  - Chương trình ngắn gọn, dễ hiểu
  - Tìm lời giải dễ dàng, lập trình đơn giản
- Khuyết điểm
  - Khó debug
  - Tốn bộ nhớ, xử lý chậm
  - Dễ gây tràn stack nếu xử lý không khéo léo
  - Nhiều bài không thể giải/giải không hiểu quả bằng đệ quy

Kĩ thuật khử đệ quy

- Cách 1: Sử dụng cấu trúc dữ liệu mô phỏng cách làm việc của 1 chương trình đệ quy: dùng ngăn xếp (stack) để lưu trạng thái:
  - Push vào trạng thái hiện tại
  - Pop ra trạng thái trước đó
- Cách 2: Sử dụng vòng lặp

Bài tập

- Viết chương trình đệ quy tính

$x_0 = 1,$
$x_n = x * x^{n-1}$

$c_1 = 1,$
$c_n = 1 - 1/2 + 1/3 - ... (+/-)1/n$

$C(n, 0) = C(n, n) = 1,$
$C(n, k) = C(n-1, k-1) + C(n-1, k)$

Bài tập

- Viết chương trình in ra tam giác Pascal có chiều cao N nhập từ bàn phím

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
...
```

Bài tập

Cho danh sách liên kết đơn

Viết chương trình đệ quy:

1. Tính tổng các số chẵn trong danh sách
2. Tìm phần tử lớn nhất trong danh sách
3. Tìm phần tử cho trước trong danh sách
4. Đảo danh sách
5. Sắp xếp danh sách tăng dần theo thuật toán chèn trực tiếp

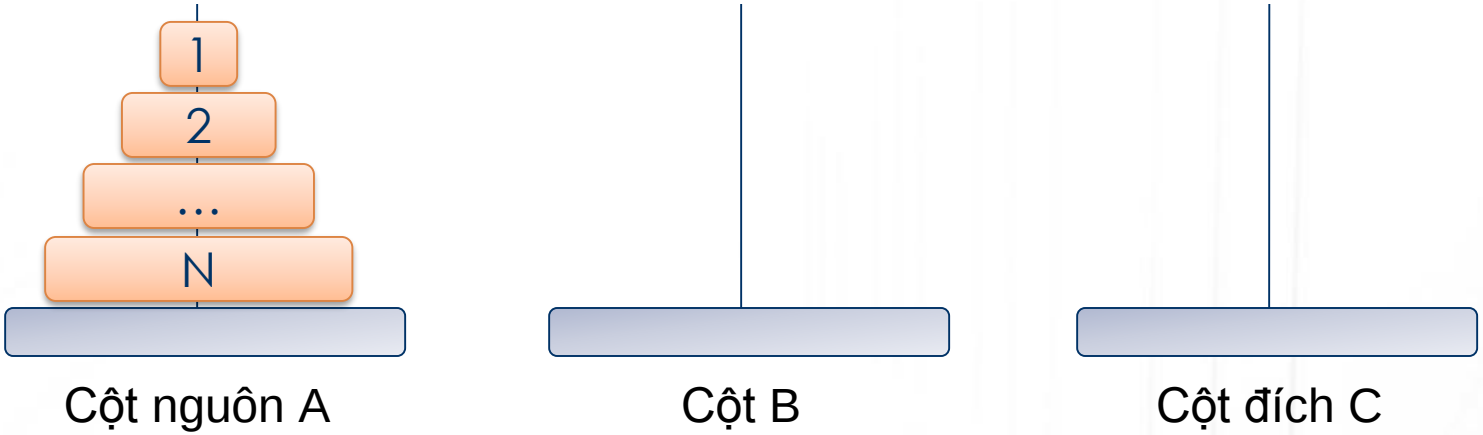
Nội dung

- Khái niệm đệ quy
- Phân loại đệ quy
- Các kỹ thuật liên quan
- Ứng dụng đệ quy

Bài toán tháp Hà Nội

- Cho 3 cột A, B và C
- Cột A chứa N đĩa, đĩa lớn nằm dưới, đĩa nhỏ nằm trên
- Tìm cách chuyển N đĩa từ cột A sang cột C sao cho
  - Mỗi lần chỉ được chuyển 1 đĩa
  - Đĩa lớn phải luôn nằm dưới
  - Có thể dùng các cột A, B, C làm trung gian

Bài toán tháp Hà Nội



Bài toán tháp Hà Nội

- Ý tưởng:
  - Sử dụng kỹ thuật chia để trị
  - Nếu chỉ có 1 đĩa ở cột A, di chuyển nó sang cột C. Kết thúc
  - Ngược lại:
    - Di chuyển N – 1 đĩa từ cột từ A sang cột B
    - Di chuyển đĩa thứ N từ A sang C
    - Di chuyển N – 1 đĩa từ B sang C

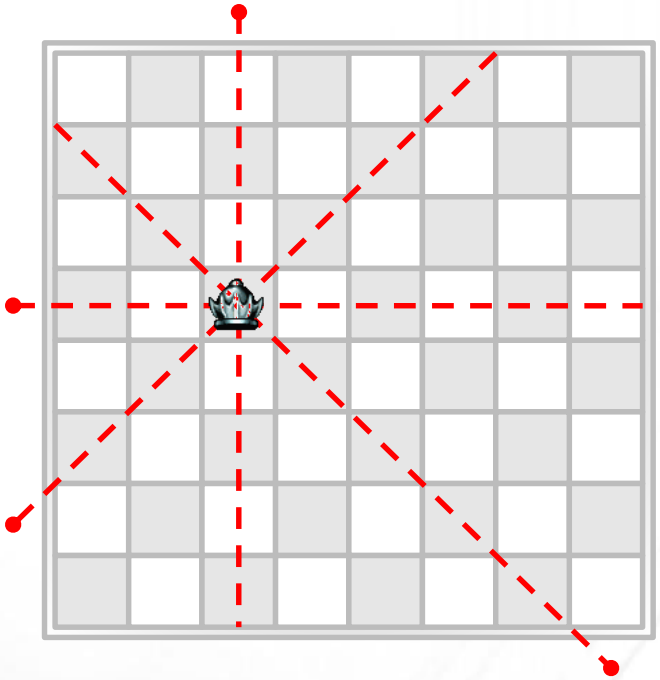
Bài toán tháp Hà Nội

```
move(N disks, A -> C, B: help)
{
    if ( N == 1 ) {
         move the disk from A -> C directly
         done
    }

    move(N-1, A -> B, C: help)
    move Nth disk from A -> C directly
    move(N-1, B -> C, A: help)
}
```

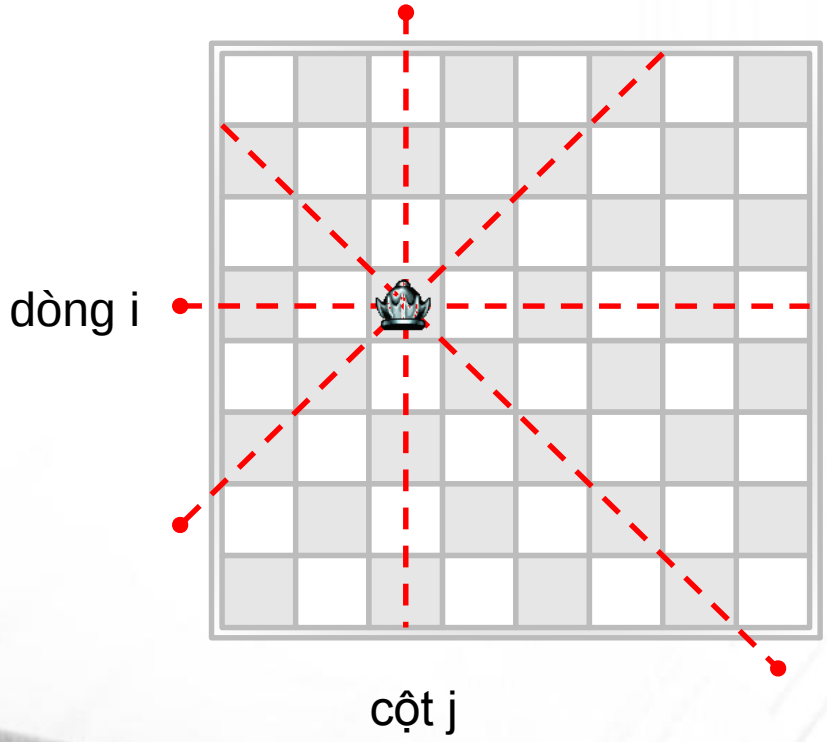
Bài toán 8 hậu

- Cho bàn cờ vua 8 x 8 ô
- Hãy đặt 8 quân hậu lên bàn cờ sao cho không quân nào được ăn quân nào
- Quy tắc ăn của quân hậu tuân theo luật cờ vua



Bài toán 8 hậu

- Chỉ có thể đặt quân hậu thứ N vào ô không bị khống chế bởi N – 1 quân hậu trước đó
- 1 quân hậu đặt tại ô (i, j) thì nó sẽ khống chế những ô nào ?

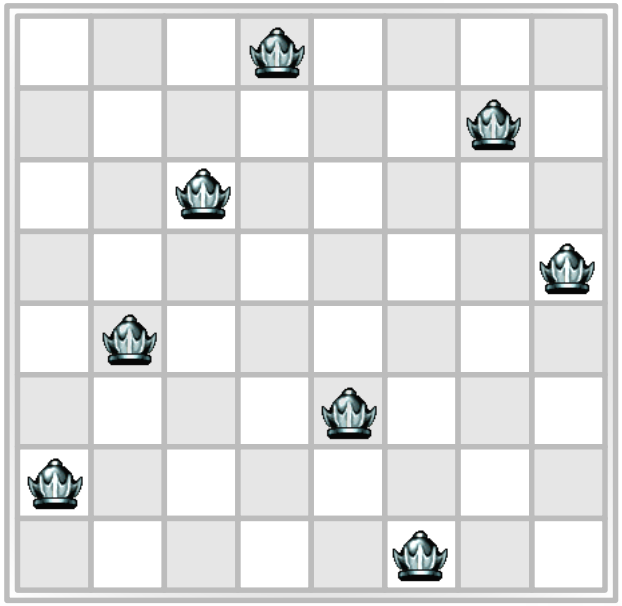


Bài toán 8 hậu

Ý tưởng:

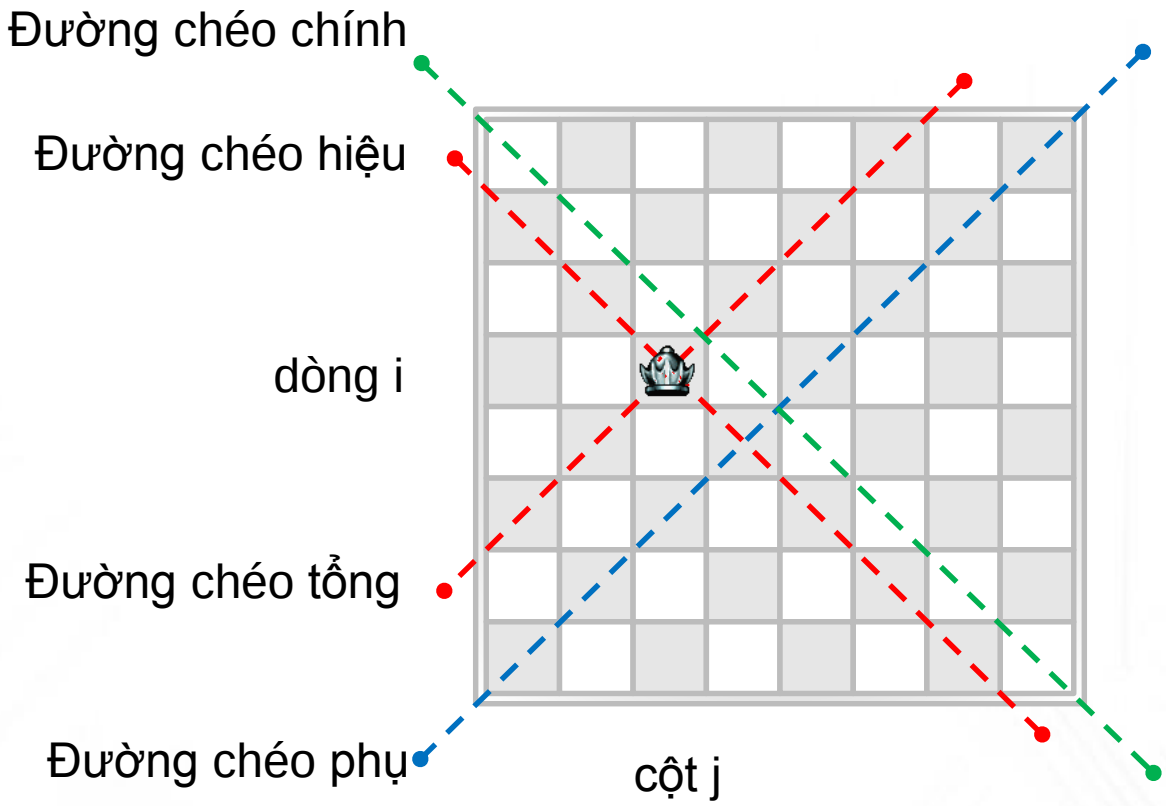
- Sử dụng kỹ thuật lần ngược
- Thử đặt hậu tại ô (i, j), nếu ô bị khống chế, lần ngược về
- Ngược lại, đánh dấu kiểm soát ô đó
- Nếu i là dòng cuối cùng, xuất đáp án
- Ngược lại
  - Lần lượt thử đặt hậu ở các tất cả cột ở dòng kế tiếp
  - Bỏ đánh dấu và quay lui

Bài toán 8 hậu



Một trong 12 đáp án tìm được

Bài toán 8 hậu



Bài toán 8 hậu

Nhận xét:

- Các phần tử nằm trên cùng hàng có chỉ số hàng bằng nhau
- Các phần tử nằm trên cùng cột có chỉ số cột bằng nhau
- Các phần tử nằm trên cùng một đường chéo song song với đường chéo chính có hiệu chỉ số hàng với chỉ số cột bằng nhau. Gọi là đường chéo hiệu
- Các phần tử nằm trên cùng một đường chéo song song với đường chéo phụ có tổng chỉ số hàng với chỉ số cột bằng nhau. Gọi là đường chéo tổng

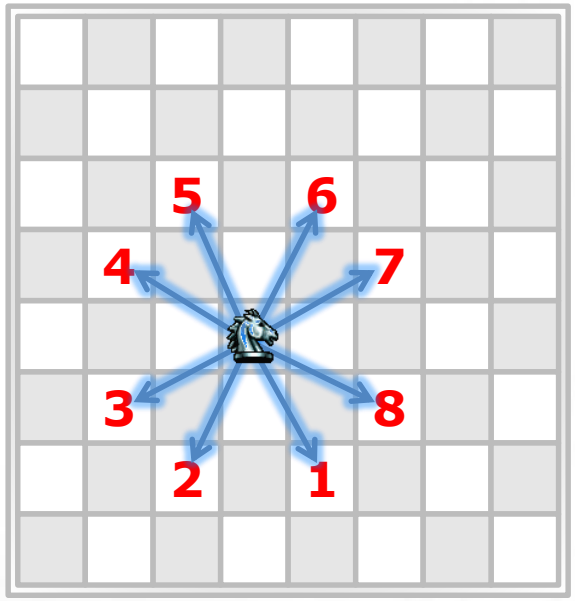
Bài toán 8 hậu

Quân hậu đặt ở vị trí (i, j), làm cách nào để biết nó có nằm trong đường khống chế của một quân đã được đặt trước đó hay không ?

- row[i] = false
- col[i] = false
- diagPlus[i + j – 1] = false
- diagMinus[i – j + n] = false

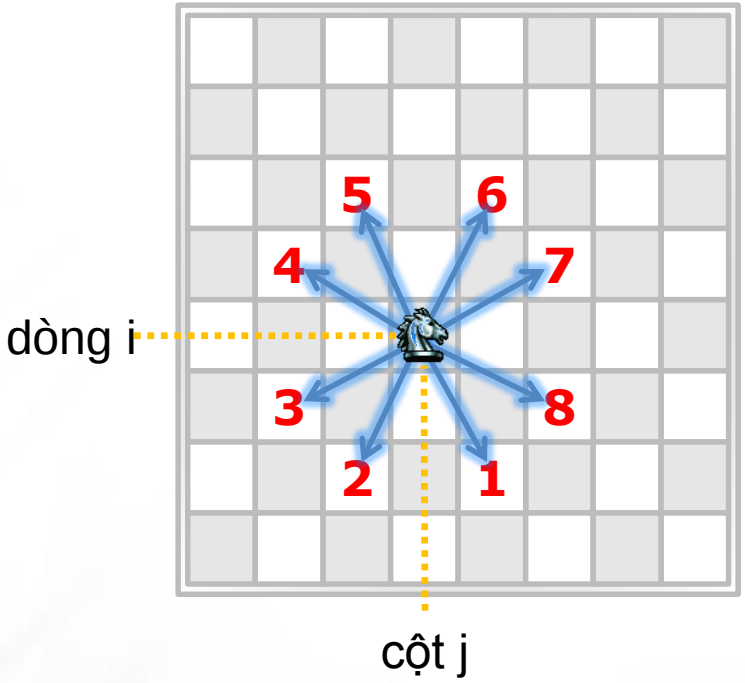
Bài toán mã đi tuần

- Cho bàn cờ vua 8 x 8
- Đặt quân mã ở 1 ô bất kì
- Hãy tìm lộ trình đi của quân mã sao cho mỗi ô đi đúng 1 lần và đi qua hết tất cả các ô của bàn cờ
- Quy tắc đi của quân mã tuân theo luật cờ vua



Bài toán mã đi tuần

- Chỉ có thể đặt quân mã tại những ô chưa được đi qua
- Nếu quân mã được đặt tại ô (i, j), những ô nào có thể đã đi qua



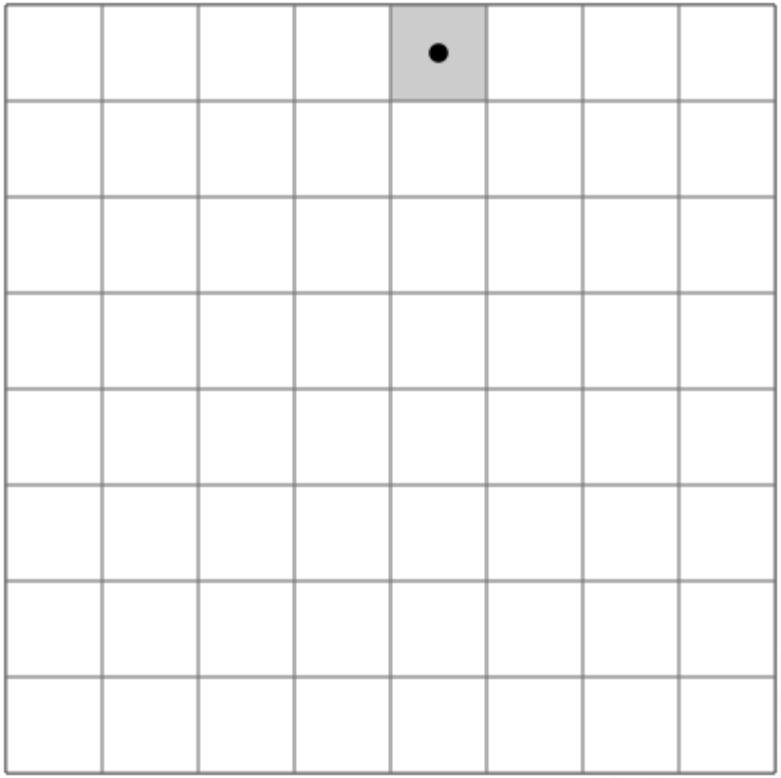
1: (i + 2, j + 1)
2: (i + 2, j – 1)
3: (i + 1, j – 2)
4: (i – 1, j – 2)
5: (i – 2, j – 1)
6: (i – 2, j + 1)
7: (i – 1, j + 2)
8: (i + 1, j + 2)

Bài toán mã đi tuần

Ý tưởng

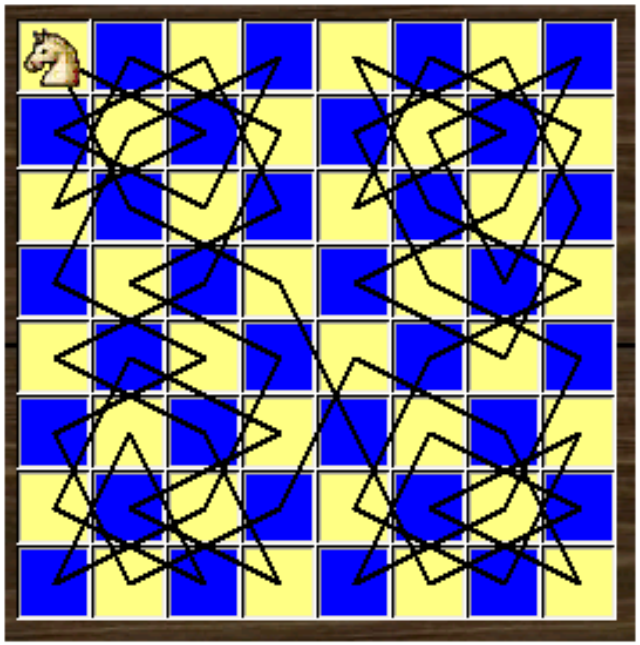
- Sử dụng kỹ thuật lần ngược
- Thử đi ô (i, j), nếu ô này đã đi, dừng, lần ngược về
- Ngược lại, cập nhật trạng thái cho ô
- Nếu là ô cuối cùng có thể đi, dừng xuất đáp án
- Ngược lại:
  - Lần lượt thử từng bước kế tiếp theo luật cờ vua
  - Bỏ đánh dấu ô (i, j) và lần ngược

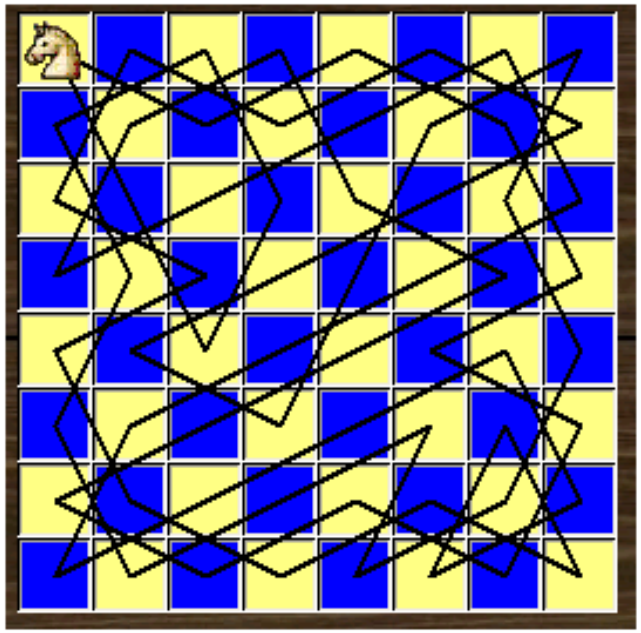
Bài toán mã đi tuần



https://vi.wikipedia.org/wiki/B%C3%A0i_to%C3%A1n_m%C3%A3_%C4%91i_tu%E1%BA%A7n (06/2020)

Bài toán mã đi tuần





https://vi.wikipedia.org/wiki/B%C3%A0i_to%C3%A1n_m%C3%A3_%C4%91i_tu%E1%BA%A7n (06/2020)

Đánh giá đạt mục tiêu

Sau buổi học, liệu sinh viên có thể:

- Nêu khái niệm đệ quy và cấu trúc 1 hàm đệ quy ?
- Liệt kê ít nhất 3 loại đệ quy ?
- Liệt kê ít nhất 2 kỹ thuật liên quan đệ quy ?
- Ứng dụng đệ quy trong ít nhất 2 bài toán kinh điển ?