ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

LẬP TRÌNH C CƠ BẢN

Đệ quy

NỘI DUNG

- Đệ quy
- Đệ quy có nhớ
- Đệ quy quay lui

ĐỆ QUY

- Một chương trình con (thủ tục/hàm) đưa ra lời gọi đến chính nó nhưng với dữ liệu đầu vào nhỏ hơn
- Tình huống cơ sở
  - Dữ liệu đầu vào nhỏ đủ để đưa ra kết quả một cách trực tiếp mà không cần đưa ra lời gọi đệ quy
- Tổng hợp kết quả
  - Kết quả của chương trình còn được xây dựng từ kết quả của lời gọi đệ quy và một số thông tin khác

- $f(n) = 1 + 2 + \ldots + n$

Other form

- $f(n) = \begin{cases} 1, \text{ if } n = 1 \\ f(n-1) + n, \text{ if } n > 1 \end{cases}$

```
#include <stdio.h>
int f(int n){
    if(n == 1) return 1;
    return n + f(n-1);
}
int main(){
    printf("%d\n",f(4));
}
```

ĐỆ QUY

- Dãy Fibonacci: 1, 1, 2, 3, 5, 8, 13, 21, 34, …

$$f(n) = \begin{cases} 1, \text{ if } n = 0 \text{ or } n = 1 \\ f(n-1) + f(n-2), \text{ if } n > 1 \end{cases}$$

```
#include <stdio.h>
int f(int n){
    if(n <= 1) return 1;
    return f(n-1) + f(n-2);
}
int main(){
    for(int i = 0; i <= 10; i++)
        printf("%d ",f(i));
}
```

ĐỆ QUY

- Tổ hợp chập k của n phần tử

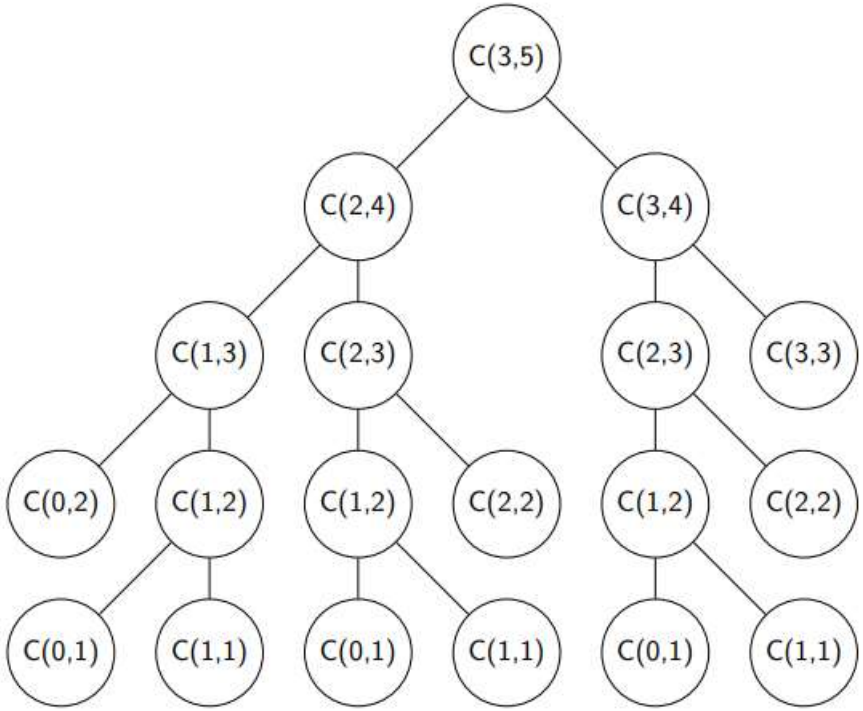
$$C(k,n) = \begin{cases} 1, \text{ if } k = 0 \text{ or } k = n \\ C(k,n-1) + f(k-1,n-1), \text{ ngược lại} \end{cases}$$

```
#include <stdio.h>
int C(int k, int n){
    if(k == 0 || k == n) return 1;
    return C(k,n-1) + C(k-1,n-1);
}
int main(){
    printf("%d ",C(3,5));
}
```

ĐỆ QUY CÓ NHỚ

- Tổ hợp chập k của n phần tử

$$C(k,n) = \begin{cases} 1, \text{ if } k = 0 \text{ or } k = n \\ C(k,n-1) + f(k-1,n-1), \text{ otherwise} \end{cases}$$

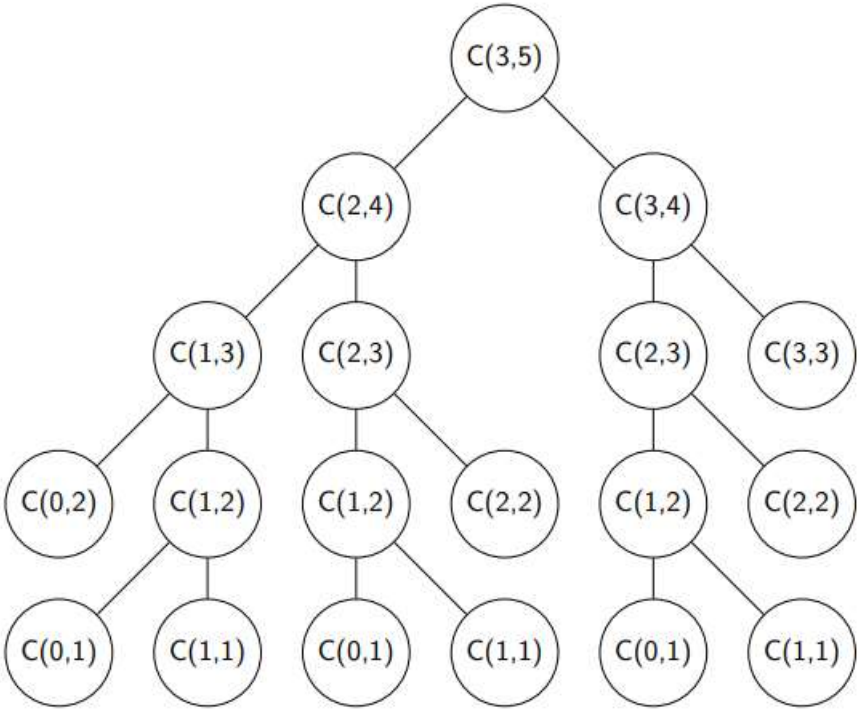


ĐỆ QUY CÓ NHỚ

- Tổ hợp chập k của n phần tử

$$C(k,n) = \begin{cases} 1, \text{ if } k = 0 \text{ or } k = n \\ C(k,n-1) + f(k-1,n-1), \text{ otherwise} \end{cases}$$

- Dư thừa
  - Một hàm với cùng giá trị tham số được gọi lặp đi lặp lại nhiều lần



ĐỆ QUY CÓ NHỚ

- Khắc phục tình trạng một chương trình con với tham số xác định được gọi đệ quy nhiều lần
- Sử dụng bộ nhớ để lưu trữ kết quả của một chương trình con với tham số cố định
- Bộ nhớ được khởi tạo với giá trị đặc biệt để ghi nhận mỗi chương trình con chưa được gọi lần nào
- Địa chỉ bộ nhớ sẽ được ánh xạ với các giá trị tham số của chương trình con

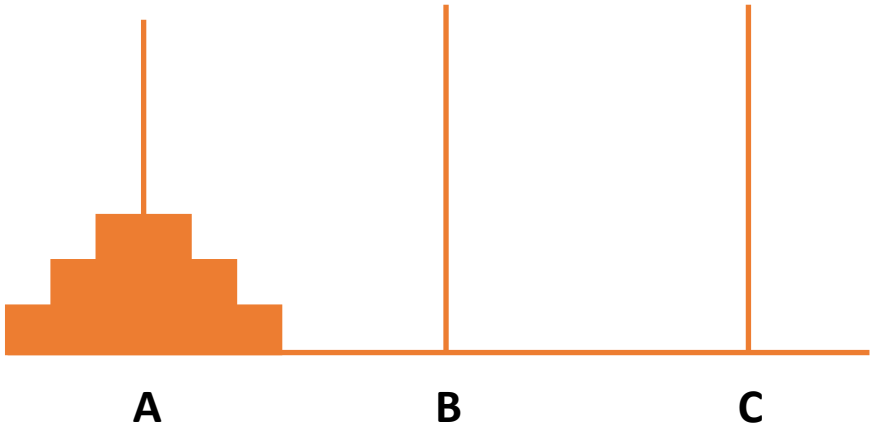
ĐỆ QUY CÓ NHỚ

- Khắc phục tình trạng một chương trình con với tham số xác định được gọi đệ quy nhiều lần
- Sử dụng bộ nhớ để lưu trữ kết quả của một chương trình con với tham số cố định
- Bộ nhớ được khởi tạo với giá trị đặc biệt để ghi nhận mỗi chương trình con chưa được gọi lần nào
- Địa chỉ bộ nhớ sẽ được ánh xạ với các giá trị tham số của chương trình con

```
#include <stdio.h>
#define MAX 100
int M[MAX][MAX];// M[k][n] store the value of
                // C(k,n)
int C(int k,int n){
    if(k == 0 || k == n) M[k][n] = 1;
    else if(M[k][n] == 0)
      M[k][n] = C(k,n-1) + C(k-1,n-1);
    return M[k][n];
}
int main(){
    memset(M,0,sizeof(M));
    printf("%d ",C(3,5));
}
```

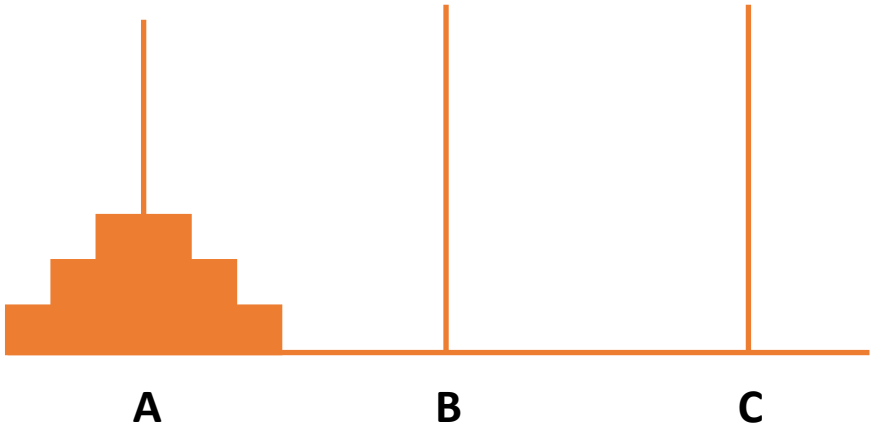
THÁP HÀ NỘI

- Bài toán tháp Hà Nội
  - Có n đĩa với kích thước khác nhau và 3 cọc A, B, C
  - Ban đầu n đĩa nằm ở cọc A theo thứ tự đĩa nhỏ nằm trên và đĩa lớn nằm dưới



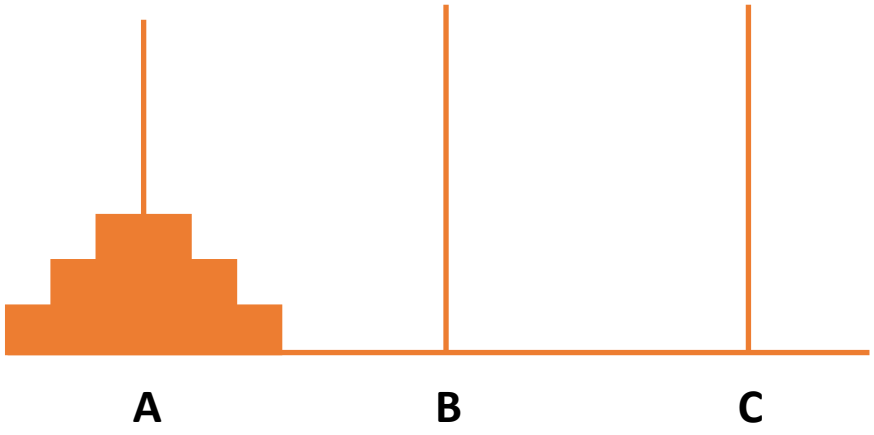
THÁP HÀ NỘI

- Bài toán tháp Hà Nội
  - Có n đĩa với kích thước khác nhau và 3 cọc A, B, C
  - Ban đầu n đĩa nằm ở cọc A theo thứ tự đĩa nhỏ nằm trên và đĩa lớn nằm dưới
  - Tìm cách chuyển n đĩa này từ cọc A sang cọc B, sử dụng cọc C làm trung gian theo nguyên tắc
    - Mỗi lần chỉ được chuyển 1 đĩa trên cùng từ 1 cọc sang cọc khác
    - Không được phép để xảy ra tình trạng đĩa to nằm bên trên đĩa nhỏ



THÁP HÀ NỘI

- Bài toán tháp Hà Nội
  - Có n đĩa với kích thước khác nhau và 3 cọc A, B, C
  - Ban đầu n đĩa nằm ở cọc A theo thứ tự đĩa nhỏ nằm trên và đĩa lớn nằm dưới
  - Tìm cách chuyển n đĩa này từ cọc A sang cọc B, sử dụng cọc C làm trung gian theo nguyên tắc
    - Mỗi lần chỉ được chuyển 1 đĩa trên cùng từ 1 cọc sang cọc khác
    - Không được phép để xảy ra tình trạng đĩa to nằm bên trên đĩa nhỏ



Lời giải

- B1: A → B
- B2: A → C
- B3: B → C
- B4: A → B
- B5: C → A
- B6: C → B
- B7: A → B

THÁP HÀ NỘI

```
#include <stdio.h>
int cnt = 0;
void move(int n, char A, char B, char C){
    if(n == 1){
        cnt++;
        printf("Step %d: Move a disk from %c to %c\n",cnt,A,B);
    }else{
        move(n-1,A,C,B);
        move(1,A,B,C);
        move(n-1,C,B,A);
    }
}
int main(){
    move(3,'A','B','C');
}
```

ĐỆ QUY QUAY LUI

- Áp dụng để giải các bài toán liệt kê, bài toán tối ưu tổ hợp
- $A = \{(x_1, x_2, \ldots, x_n) \mid x_i \in A_i, \forall\, i = 1,\ldots, n\}$
- Liệt kê tất cả các bộ $x \in A$ thoả mãn một thuộc tính P nào đó
- Thủ tục TRY(k):
  - Thử các giá trị v có thể gán cho x_k mà không vi phạm thuộc tính P
  - Với mỗi giá trị hợp lệ v:
    - Gán v cho x_k
    - Nếu $k < n$: gọi đệ quy TRY(k+1) để thử tiếp giá trị cho x_{k+1}
    - Nếu $k = n$: ghi nhận cấu hình

ĐỆ QUY QUAY LUI

```
TRY(k)
  Begin
    Foreach v thuộc A_k
     if check(v,k) /* kiểm tra xem v có hợp lệ không */
       Begin
         x_k = v;
         if(k = n) ghi_nhan_cau_hinh;
         else TRY(k+1);
       End
  End
Main()
Begin
  TRY(1);
End
```

ĐỆ QUY QUAY LUI: liệt kê xâu nhị phân

- Mô hình hoá cấu hình:
  - Mảng **x[n]** trong đó **x[i]** $\in$ {0,1}
    là bít thứ **i** của xâu nhị phân

    (**i= 0, . . ., n-1**)

ĐỆ QUY QUAY LUI: liệt kê xâu nhị phân

- Mô hình hoá cấu hình:
  - Mảng **x[n]** trong đó **x[i]** ∈{0,1} là bít thứ **i** của xâu nhị phân

    (**i= 0, . . ., n-1**)

```
void printSolution(){
  for(int k = 0; k < n; k++)
    printf("%d",x[k]);
  printf("\n");
}

int TRY(int k) {
  for(int v = 0; v <= 1; v++){
    x[k] = v;
    if(k == n-1) printSolution();
    else TRY(k+1);
  }
}

int main() {
  TRY(0);
}
```

ĐỆ QUY QUAY LUI: liệt kê xâu nhị phân

- Liệt kê các xâu nhị phân sao cho không có 2 bit 1 nào đứng cạnh nhau
- Mô hình hoá cấu hình:
  - Mảng **x[n]** trong đó **x[i]** $\in$ {0,1} là bít thứ **i** của xâu nhị phân

    (**i= 1, . . ., n**)
  - Thuộc tính *P*: không có 2 bít 1 nào đứng cạnh nhau

ĐỆ QUY QUAY LUI: liệt kê xâu nhị phân

- Liệt kê các xâu nhị phân sao cho không có 2 bit 1 nào đứng cạnh nhau
- Mô hình hoá cấu hình:
  - Mảng **x[n]** trong đó **x[i]** $\in$ {0,1} là bít thứ **i** của xâu nhị phân

    (**i= 1, . . ., n**)
  - Thuộc tính *P*: không có 2 bít 1 nào đứng cạnh nhau

```
int TRY(int k) {
  for(int v = 0; v <= 1; v++){
    if(x[k-1] + v < 2){
      x[k] = v;
      if(k == n)
        printSolution();
      else TRY(k+1);
    }
  }
}

int main() {
  x[0] = 0;
  TRY(1);
}
```

ĐỆ QUY QUAY LUI: liệt kê tổ hợp

- Liệt kê các tổ hợp chập k của 1, 2, …, n
- Mô hình hoá cấu hình:
  - Mảng $x[k]$ trong đó $x[i] \in \{1, . . ., n\}$ là phần tử thứ i của cấu hình tổ hợp ($i = 1, . . ., k$)
  - Thuộc tính P: $x[i] < x[i+1]$, với mọi $i = 1, 2, …, k-1$

ĐỆ QUY QUAY LUI: liệt kê tổ hợp

- Liệt kê các tổ hợp chập *k* của 1, 2, …, *n*
- Mô hình hoá cấu hình:
  - Mảng *x*[*k*] trong đó *x*[*i*] ∈{1, . . ., *n*} là phần tử thứ *i* của cấu hình tổ hợp (*i* = 1, . . ., *k*)
  - Thuộc tính *P*: *x*[*i*] < *x*[*i*+1], với mọi *i* = 1, 2, …, *k*-1

```
int TRY(int i) {
  for(int v = x[i-1]+1; v <= n-k+i;
          v++){
     x[i] = v;
     if(i == k)
        printSolution();
      else TRY(i+1);
     }
}

int main() {
  x[0] = 0;
  TRY(1);
}
```

ĐỆ QUY QUAY LUI: liệt kê hoán vị

- Liệt kê các hoán vị của 1, 2, …, n
- Mô hình hoá cấu hình:
  - Mảng $x[1,. . . , n]$ trong đó $x[i] \in \{1, . . ., n\}$ là phần tử thứ i của cấu hình hoán vị ($i = 1, . . ., n$)
  - Thuộc tính P:
    - $x[i] \neq x[j]$, với mọi $1 \leq i < j \leq n$
  - Mảng đánh dấu $m[v]$ = true (false) nếu giá trị v đã xuất hiện (chưa xuất hiện) trong cấu hình bộ phận, với mọi $v = 1, …, n$

ĐỆ QUY QUAY LUI: liệt kê hoán vị

- Liệt kê các hoán vị của 1, 2, …, n
- Mô hình hoá cấu hình:
  - Mảng $x[1,\ldots,n]$ trong đó $x[i] \in \{1, \ldots, n\}$ là phần tử thứ i của cấu hình hoán vị ($i = 1, \ldots, n$)
  - Thuộc tính P:
    - $x[i] \neq x[j]$, với mọi $1 \leq i < j \leq n$
  - Mảng đánh dấu $m[v]$ = true (false) nếu giá trị v đã xuất hiện (chưa xuất hiện) trong cấu hình bộ phận, với mọi $v = 1, \ldots, n$

```
void TRY(int i) {
  for(int v = 1; v <= n; v++){
    if(!m[v]) {
      x[i] = v;
      m[v] = true; // đánh dấu
      if(i == n)
        printSolution();
      else TRY(i+1);
      m[v] = false;// khôi phục
    }
  }
}
void main() {
  for(int v = 1; v <= n; v++)
     m[v] = false;
  TRY(1);
}
```

ĐỆ QUY QUAY LUI: bài toán xếp hậu

- Xếp n quân hậu trên một bàn cờ quốc tế sao cho không có 2 quân hậu nào ăn được nhau
- Mô hình hoá
  - $x[1, \ldots, n]$ trong đó $x[i]$ là hàng của quân hậu xếp trên cột i, với mọi $i = 1, \ldots, n$
  - Thuộc tính P
    - $x[i] \neq x[j]$, với mọi $1 \leq i < j \leq n$
    - $x[i] + i \neq x[j] + j$, với mọi $1 \leq i < j \leq n$
    - $x[i] - i \neq x[j] - j$, với mọi $1 \leq i < j \leq n$

	1	2	3	4
1		X		
2				X
3	X			
4			X	

Lời giải $x = (3, 1, 4, 2)$

ĐỆ QUY QUAY LUI: bài toán xếp hậu

```
int check(int v, int k) {
  // kiểm tra xem v có thể gán được
  // cho x[k] không
  for(int i = 1; i <= k-1; i++) {
    if(x[i] == v) return 0;
    if(x[i] + i == v + k) return 0;
    if(x[i] – i == v – k) return 0;
  }
  return 1;
}
```

```
void TRY(int k) {
  for(int v = 1; v <= n; v++) {
    if(check(v,k)) {
      x[k] = v;
      if(k == n) printSolution();
      else TRY(k+1);
    }
  }
}
void main() {
    TRY(1);
}
```

ĐỆ QUY QUAY LUI

- Liệt kê tất cả các nghiệm nguyên dương của phương trình:

$$x_1 + x_2 + \ldots + x_n = M$$

ĐỆ QUY QUAY LUI

- Liệt kê tất cả các nghiệm nguyên dương của phương trình:

$$x_1 + x_2 + \ldots + x_n = M$$

- Duy trì biến T là tổng giá trị các biến đã được thử giá trị
- Hàm TRY(k)
  - Các biến $x_1, x_2, \ldots, x_{k-1}$ đã được thử giá trị
  - $T = x_1 + x_2 + \ldots + x_{k-1}$
  - $x_{k+1} + x_{k+2} + \ldots + x_n \geq n-k$

  → $1 \leq x_k \leq M - T - (n - k)$

ĐỆ QUY QUAY LUI

- Liệt kê tất cả các nghiệm nguyên dương của phương trình:

$$x_1 + x_2 + \ldots + x_n = M$$

```
#include <stdio.h>
#define N 100
int n,M,T;
int x[N];
void solution(){
    for(int i = 1; i <= n; i++)
        printf("%d ",x[i]);
    printf("\n");
}
int check(int v, int k){
    if(k == n) return T + v == M;
    return 1;
}
```

```
void Try(int k){
    for(int v = 1; v <= M - T - (n-k); v++){
        if(check(v,k)){
            x[k] = v;
            T += v;
            if(k == n) solution();
            else Try(k+1);
            T -= v;
        }
    }
}
int main(){
    n = 3; M = 5; T = 0;
    Try(1);
}
```

ĐỆ QUY QUAY LUI: bài toán Sudoku

- Điền các chữ số từ 1 đến 9 vào các ô trong bảng vuông 9x9 sao cho trên mỗi hàng, mỗi cột và mỗi bảng vuông con 3x3 đều có mặt đầy đủ 1 chữ số từ 1 đến 9

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	1	4	3	6	5	8	9	7
3	6	5	8	9	7	2	1	4
8	9	7	2	1	4	3	6	5
5	3	1	6	4	2	9	7	8
6	4	2	9	7	8	5	3	1
9	7	8	5	3	1	6	4	2

ĐỆ QUY QUAY LUI: bài toán Sudoku

- Mô hình hoá
  - Mảng 2 chiều $x[0..8, 0..8]$
  - Thuộc tính P
    - $x[i, j_2] \neq x[i, j_1]$, với mọi $i = 0,\ldots,8$, và $0 \leq j_1 < j_2 \leq 8$
    - $x[i_1, j] \neq x[i_2, j]$, với mọi $j = 0,\ldots,8$, và $0 \leq i_1 < i_2 \leq 8$
    - $x[3I+i_1, 3J+j_1] \neq x[3I+i_2, 3J+j_2]$, với mọi $I, J = 0,\ldots, 2$, và $i_1, j_1, i_2, j_2 \in \{0,1, 2\}$ sao cho $i_1 \neq i_2$ hoặc $j_1 \neq j_2$

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	1	4	3	6	5	8	9	7
3	6	5	8	9	7	2	1	4
8	9	7	2	1	4	3	6	5
5	3	1	6	4	2	9	7	8
6	4	2	9	7	8	5	3	1
9	7	8	5	3	1	6	4	2

ĐỆ QUY QUAY LUI: bài toán Sudoku

- Thứ tự duyệt: từ ô (0,0), theo thứ tự từ trái qua phải và từ trên xuống dưới

1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
7	8	9	1	2	3	4	5	6
2	1	4	3	6	5	8	9	7
3	6	5	8	9	★			

ĐỆ QUY QUAY LUI: bài toán Sudoku

```
bool check(int v, int r, int c){
  for(int i = 0; i <= r-1; i++)
    if(x[i][c] == v) return false;
  for(int j = 0; j <= c-1; j++)
    if(x[r][j] == v) return false;
  int I = r/3;  int J = c/3;
  int i = r - 3*I;  int j = c - 3*J;
  for(int i1 = 0; i1 <= i-1; i1++)
    for(int j1 = 0; j1 <= 2; j1++)
      if(x[3*I+i1][3*J+j1] == v)
        return false;
  for(int j1 = 0; j1 <= j-1; j1++)
    if(x[3*I+i][3*J+j1] == v)
       return false;
  return true;
}
```

```
void TRY(int r, int c){
  for(int v = 1; v <= 9; v++){
    if(check(v,r,c)){
      x[r][c] = v;
      if(r == 8 && c == 8){
        printSolution();
      }else{
        if(c == 8) TRY(r+1,0);
        else TRY(r,c+1);
      }
    }
  }
}

void main(){
  TRY(0,0);
}
```

ĐỆ QUY QUAY LUI: bài tập

- Cho số nguyên dương M, N và N số nguyên dương A_1, A_2, …, A_N. Liệt kê các nghiệm nguyên dương của phương trình

$$A_1X_1 + A_2X_2 + \ldots + A_NX_N = M$$

- Giải bài toán xếp hậu và sudoku sử dụng kỹ thuật đánh dấu

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!