HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

TUẦN 3 : ĐỆ QUY QUAY LUI

ONE LOVE. ONE FUTURE.

NỘI DUNG

- Sơ đồ chung đệ quy quay lui
- Bài toán liệt kê xâu nhị phân
- Bài toán liệt kê hoán vị
- Bài toán điền số Sudoku

SƠ ĐỒ CHUNG

- Bài toán liệt kê tổ hợp: Liệt kê các bộ $x = (x[1], x[2],..., x[k], x[k+1],. . ., x[n])$ với $x[i] \in A_i$, $i = 1, 2, ..., n$ và thỏa mãn tập các ràng buộc P cho trước.
- Ví dụ:
  - "Bài toán liệt kê xâu nhị phân độ dài *n*" *dẫn về liệt kê các bộ* $x = (x[1], x[2],..., x[k], x[k+1],. . ., x[n])$ với $x[i] \in \{0, 1\}$, $i = 1, 2, ..., n$
  - "Bài toán liệt kê xâu nhị phân độ dài *n có số lượng bit 0 là một số chẵn*" *dẫn về liệt kê các bộ* $x = (x[1], x[2],..., x[k], x[k+1],. . ., x[n])$ với $x[i] \in \{0, 1\}$, $i = 1, 2, ..., n$ và thỏa mãn ràng buộc: số lượng phần tử $x[i] = 0$ với $i = 1, 2, ..., n$ là một số chẵn.
- Thuật toán quay lui cho phép giải các bài toán liệt kê tổ hợp. Có hai cách cài đặt thuật toán quay lui: đệ quy hoặc không đệ quy.

SƠ ĐỒ CHUNG

- Bài toán liệt kê tổ hợp: Liệt kê các bộ $x = (x[1], x[2],..., x[k], x[k+1],. . ., x[n])$ với $x[i] \in A_i$, $i = 1, 2, ..., n$ và thỏa mãn tập các ràng buộc P cho trước.

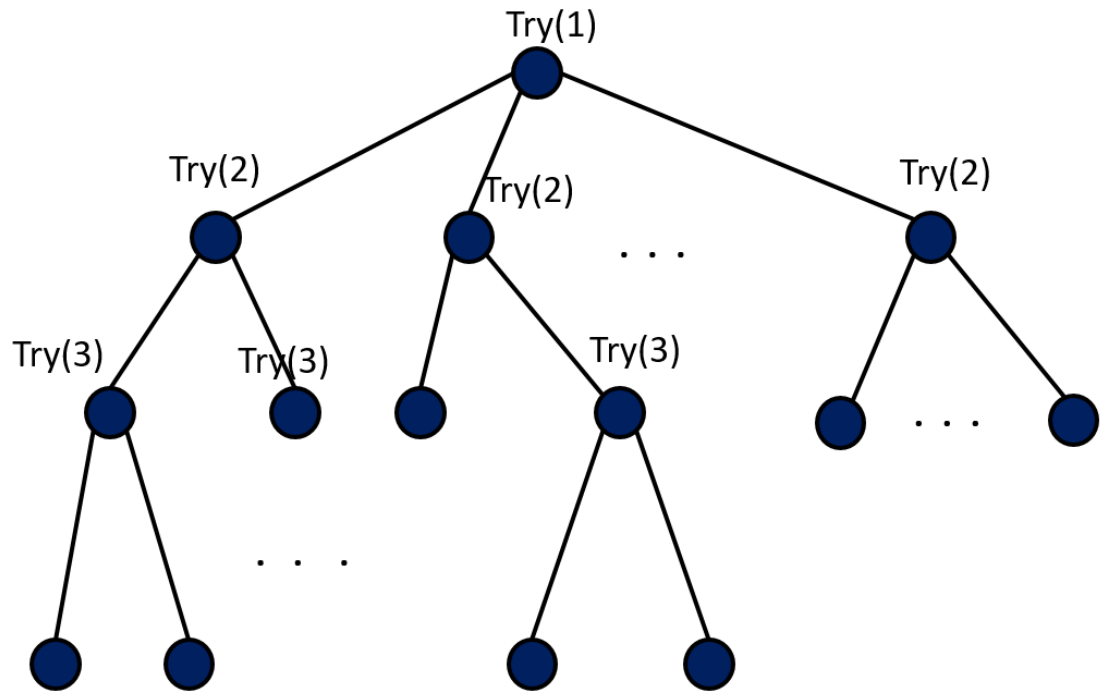
- Lệnh gọi để thực hiện thuật toán đệ quy quay lui là: Try(1);
- Nếu chỉ cần tìm một lời giải thì cần tìm cách chấm dứt các thủ tục gọi đệ qui lồng nhau sinh bởi lệnh gọi Try(1) sau khi ghi nhận được lời giải đầu tiên.
- Nếu kết thúc thuật toán mà ta không thu được một lời giải nào thì điều đó có nghĩa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

```
Try(k){//thử các giá trị có thể gán cho x[k]
  for v in candidates(k) do {
     if (check(v,k)) then {
        x[k] = v;
        [Update the data structure D]
        if (k == n) then solution();
        else Try(k+1);
        [Recover the data structure D]
     }
  }
}
```

SƠ ĐỒ CHUNG

- Bài toán liệt kê tổ hợp: Liệt kê các bộ $x = (x[1], x[2],..., x[k], x[k+1],. . ., x[n])$ với $x[i] \in A_i$, $i = 1, 2, ..., n$ và thỏa mãn tập các ràng buộc P cho trước.

- Lệnh gọi để thực hiện thuật toán đệ quy quay lui là: Try(1);




```
Try(k){//thử các giá trị có thể gán cho x[k]
  for v in candidates(k) do {
     if (check(v,k)) then {
        x[k] = v;
        [Update the data structure D]
        if (k == n) then solution();
        else Try(k+1);
        [Recover the data structure D]
     }
  }
}
```

BÀI TOÁN LIỆT KÊ XÂU NHỊ PHÂN

- Cho số nguyên dương $n \geq 1$. Hãy liệt kê tất cả các xâu nhị phân độ dài n theo thứ tự từ điển.
- Ví dụ: $n = 3$, ta có các xâu nhị phân độ dài 3 cần liệt kê theo thứ tự như sau:
  - 000
  - 001
  - 010
  - 011
  - 100
  - 101
  - 110
  - 111

BÀI TOÁN LIỆT KÊ XÂU NHỊ PHÂN

- Cho số nguyên dương $n \geq 1$. Hãy liệt kê tất cả các xâu nhị phân độ dài n theo thứ tự từ điển.
- Biểu diễn lời giải: mỗi xâu nhị phân được biểu diễn bởi mảng (x[1], x[2], . . ., x[n]) trong đó x[k] $\in$ {0,1} là bít thứ k trong xâu nhị phân.

```
Try(k){//thử các giá trị có thể gán cho x[k]
  for v in candidates(k) do {
     if (check(v,k)) then {
        x[k] = v;
        [Update the data structure D]
        if (k == n) then solution();
        else Try(k+1);
        [Recover the data structure D]
     }
  }
}
```

Cần xác định:
- candidates(k)
- check(v, k)

```
void Try(int k){
    for (int v = 0; v <= 1; v++){
        x[k] = v;
        if (k == n) solution();
        else Try(k+1);
    }
}
```

BÀI TOÁN LIỆT KÊ XÂU NHỊ PHÂN

- Cho số nguyên dương $n \geq 1$. Hãy liệt kê tất cả các xâu nhị phân độ dài n theo thứ tự từ điển.
- Biểu diễn lời giải: mỗi xâu nhị phân được biểu diễn bởi mảng (x[1], x[2], . . ., x[n]) trong đó x[k] $\in$ {0,1} là bít thứ k trong xâu nhị phân.

```
Try(k){//thử các giá trị có thể gán cho x[k]
  for v in candidates(k) do {
     if (check(v,k)) then {
        x[k] = v;
        [Update the data structure D]
        if (k == n) then solution();
        else Try(k+1);
        [Recover the data structure D]
     }
  }
}
```

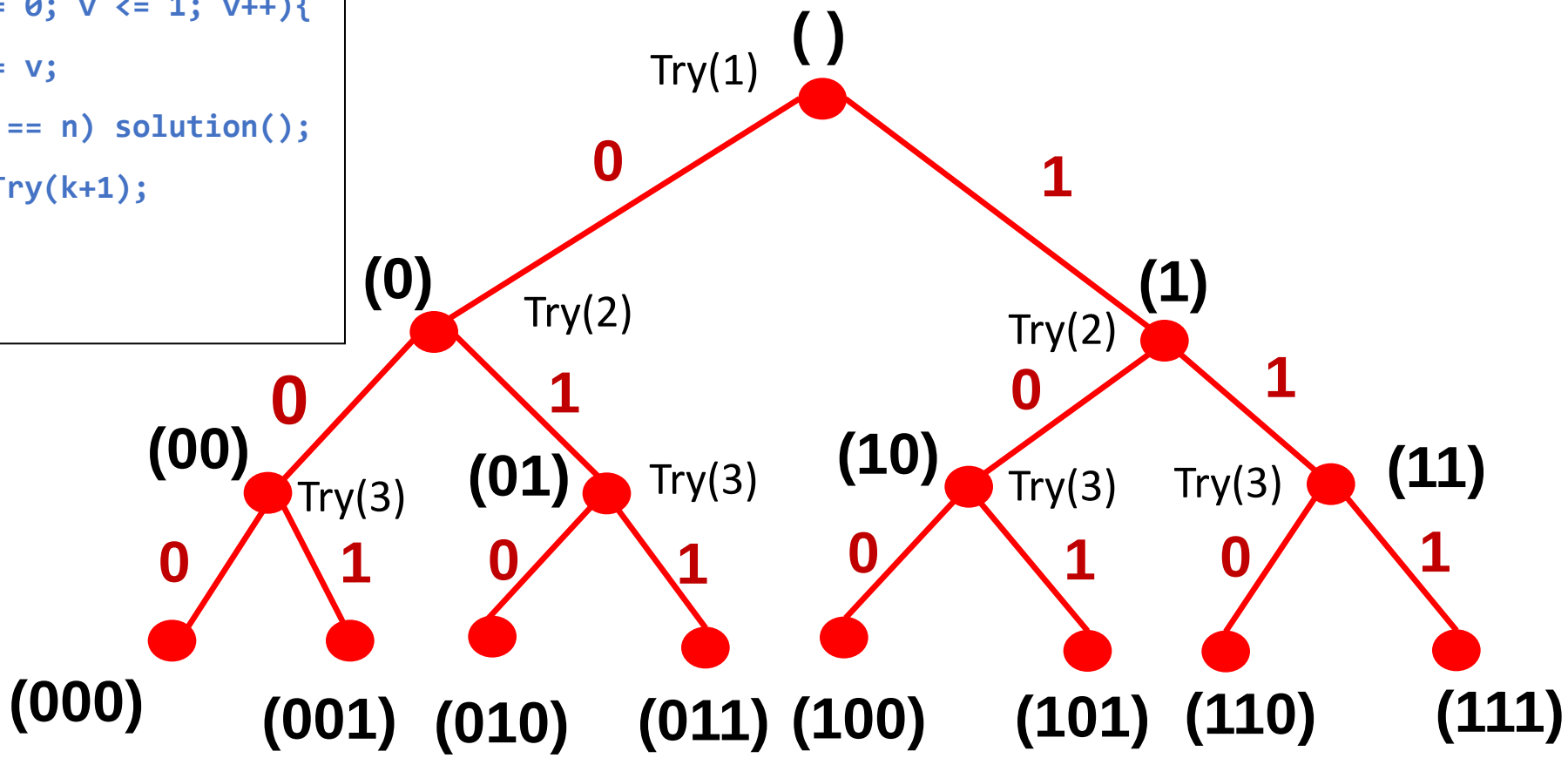
Cần xác định:
- candidates(k)
- check(v, k)

```
void Try(int k){
    for (int v = 0; v <= 1; v++){
        x[k] = v;
        if (k == n) solution();
        else Try(k+1);
    }
}
```

BÀI TOÁN LIỆT KÊ XÂU NHỊ PHÂN

```
void Try(int k){
  for (int v = 0; v <= 1; v++){
        x[k] = v;
        if (k == n) solution();
        else Try(k+1);
  }
}
```

Cây liệt kê các xâu nhị phân độ dài 3



BÀI TOÁN LIỆT KÊ HOÁN VỊ

- Cho số nguyên dương $n \geq 1$. Hãy liệt kê tất cả các hoán vị của n số 1, 2, ..., n theo thứ tự từ điển.
- Ví dụ: $n = 3$, ta có các hoán vị của 1, 2, 3 theo thứ tự từ điển như sau:
  - (1, 2, 3)
  - (1, 3, 2)
  - (2, 1, 3)
  - (2, 3, 1)
  - (3, 1, 2)
  - (3, 2, 1)

BÀI TOÁN LIỆT KÊ HOÁN VỊ

- Biểu diễn lời giải: mỗi hoán vị *n* phần tử được biểu diễn bởi mảng (x[1], x[2], . . ., x[*n*]) trong đó:
  - x[*k*] ∈{1, 2,..., *n*} là phần tử thứ *k* trong hoán vị
  - x[k] ≠ x[1], x[2],... x[*k*-1], x[*k*+1], ...,x[*n*]

```
try(k){//thử các giá trị có thể gán cho x[k]
  for v in candidates(k) do {
   if (check(v,k)) then {
     x[k] = v;
     [Update the data structure D]
     if (k == n) then solution();
     else try(k+1);
     [Recover the data structure D]
    }
  }
}
```

Cần xác định:
- candidates(k)
- check(v, k)

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

BÀI TOÁN LIỆT KÊ HOÁN VỊ

```
#include <stdio.h>
int n;
int x[100];
void solution(){
    for(int k = 1; k <= n; k++)
        printf("%d ",x[k]);
    printf("\n");
}
int check(int v, int k) {
  for (int i = 1; i <= k-1; i++)
    if (x[i] == v) 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) solution();
            else Try(k+1);
        }
    }
}
int main(){
   scanf("%d",&n);
   Try(1);
}
```

BÀI TOÁN LIỆT KÊ HOÁN VỊ

- Kỹ thuật đánh dấu
  - used[v] = 1: v đã xuất hiện
  - used[v] = 0: v chưa xuất hiện

```
try(k){//thử các giá trị có thể gán cho x[k]
  for v in candidates(k) do {
     if (check(v,k)) then {
        x[k] = v;
        [Update the data structure D]
        if (k == n) then solution();
        else try(k+1);
        [Recover the data structure D]
     }
  }
}
```

```
void Try(int k){
    for(int v = 1; v <= n; v++){
        if (used[v]==0){
            x[k] = v;
            used[v] = 1;
            if (k == n) solution();
            else Try(k+1);
            used[v] = 0;
        }
    }
}
int main(){
   scanf("%d",&n);
   for(int v = 1; v <= n; v++) used[v] = 0;
   Try(1);
}
```

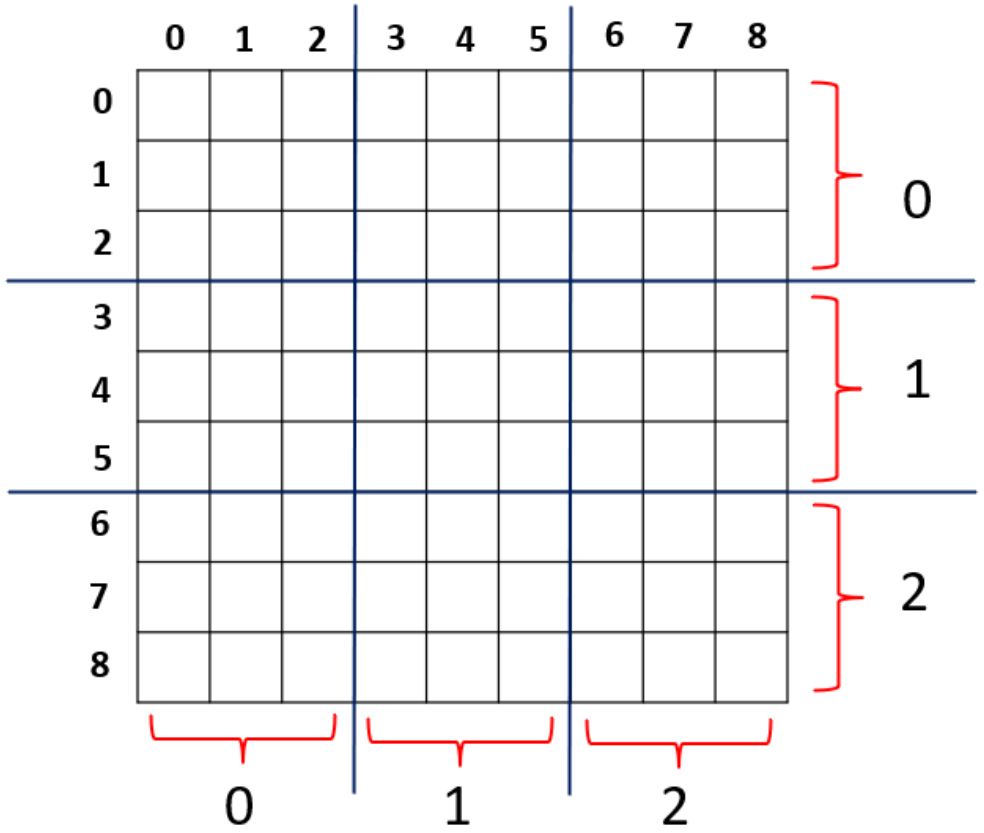
BÀI TOÁN LIỆT KÊ HOÁN VỊ

```
#include <stdio.h>
int n;
int x[100];
int used[100];
void solution(){
    for(int k = 1; k <= n; k++)
         printf("%d ",x[k]);
    printf("\n");
}
```

```
void Try(int k){
    for(int v = 1; v <= n; v++){
        if (used[v]==0){
            x[k] = v;
            used[v] = 1;
            if (k == n) solution();
            else Try(k+1);
            used[v] = 0;
        }
    }
}
int main(){
   scanf(“%d”,&n);
   for(int v = 1; v <= n; v++) used[v] = 0;
   Try(1);
}
```

BÀI TOÁN ĐIỀN SỐ SUDOKU

- Phát biểu bài toán:
  - Cho lưới ô vuông 9x9 được chia thành 9 lưới 3x3.
  - Các hàng, cột được đánh số 0, 1, 2, …, 8
  - Liệt kê tất cả các cách điền các số 1, 2. …, 9 vào các ô thuộc lưới ô vuông 9x9 sao cho:
    - Các số trên mỗi dòng là khác nhau,
    - Các số trên mỗi cột là khác nhau,
    - Các số trên mỗi lưới 3x3 là khác nhau.



BÀI TOÁN ĐIỀN SỐ SUDOKU

- Phát biểu bài toán:
  - Cho lưới ô vuông 9x9 được chia thành 9 lưới 3x3.
  - Các hàng, cột được đánh số 0, 1, 2, …, 8
  - Liệt kê tất cả các cách điền các số 1, 2. …, 9 vào

    các ô thuộc lưới ô vuông 9x9 sao cho:
    - Các số trên mỗi dòng là khác nhau,
    - Các số trên mỗi cột là khác nhau,
    - Các số trên mỗi lưới 3x3 là khác nhau.

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

BÀI TOÁN ĐIỀN SỐ SUDOKU

- Biểu diễn lời giải: $X[i, j]$ là giá trị số được điền vào ô hàng i cột j ($i, j = 0, 1, 2, \ldots, 8$)
- Mảng đánh dấu
  - markR[r, v] = 1: giá trị v đã xuất hiện trên hàng r và markR[r, v] = 0, ngược lại ($r = 0, 1, \ldots, 8$ và $v = 1, 2, \ldots, 9$)
  - markC[c, v] = 1: giá trị v đã xuất hiện trên cột c và markC[c, v] = 0, ngược lại ($c = 0, 1, 2, \ldots, 8$ và $v = 1, 2, \ldots, 9$)
  - markS[i, j, v] = 1: giá trị v đã xuất hiện trong lưới 3x3 hàng thứ i và cột thứ j và markS[i, j, v] = 0, ngược lại ($i, j = 0, 1, 2$ và $v = 1, 2, \ldots, 9$)

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

BÀI TOÁN ĐIỀN SỐ SUDOKU

- Thứ tự duyệt các ô để thử giá trị: từ trên xuống dưới và từ trái qua phải
- Hàm đệ quy Try(*r*, *c*): thử giá trị cho ô hàng *r* cột *c*

```
check(v, r, c){
  if markR[r,v] = 1 then return 0;
  if markC[c,v] = 1 then return 0;
  if markS[r/3,c/3,v] = 1 then return 0;
  return 1;
}
```

```
Try(r, c){
  for v = 1 to 9 do {
    if (check(v,r,c)) then {
      X[r,c] = v;
      markR[r,v] = 1; markC[c,v] = 1; markS[r/3,c/3,v] = 1;
      if r = 8 and c = 8 then solution();
      else {
        if c = 8 then Try(r+1, 0); else Try(r, c+1);
      }
      markR[r,v] = 0; markC[c,v] = 0; markS[r/3,c/3,v] = 0;
    }
  }
}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !