HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

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

C BASIC

ĐỆ QUY QUAY LUI

ONE LOVE. ONE FUTURE.

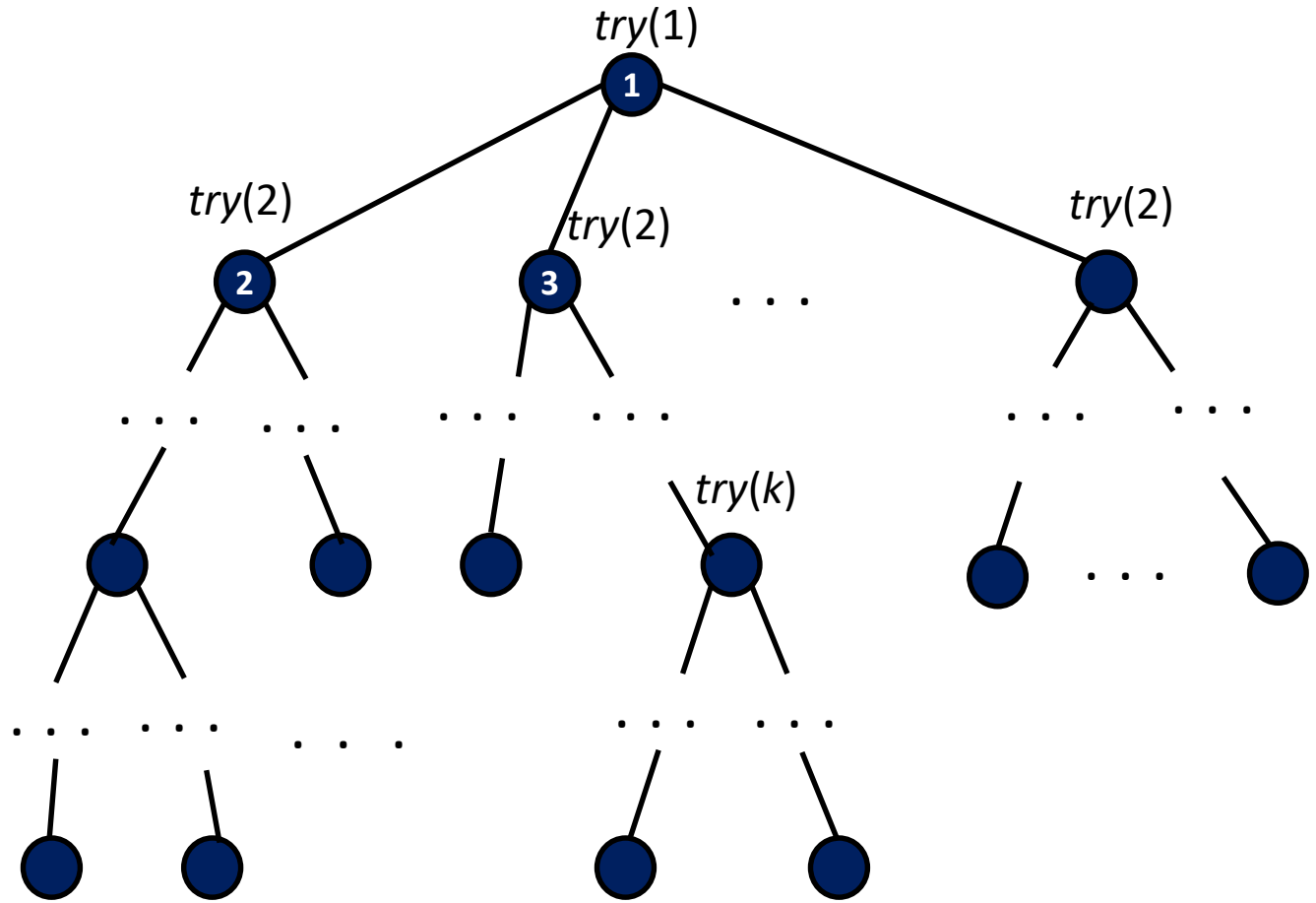
NỘI DUNG

- Đệ quy quay lui
- Bài toán liệt kê xâu nhị phân độ dài *n* (P.02.05.01)
- Bài toán liệt kê xâu nhị phân độ dài n không có 2 bít 1 đứng cạnh nhau (P.02.05.02)
- Bài toán liệt kê hoán vị (P.02.05.03)
- Bài toán liệt kê nghiệm nguyên dương của phương trình tuyến tính (P.02.05.04)

ĐỆ QUY QUAY LUI

- Thuật toán quay lui cho phép ta giải các bài toán liệt kê tổ hợp và bài toán tối ưu tổ hợp
- Phương án được mô hình hóa bằng một dãy các biến quyết định $X_1, X_2, \ldots, X_n$
- Cần tìm cho mỗi biến X_i một giá trị từ 1 tập rời rạc A_i cho trước sao cho
  - Các ràng buộc của bài toán được thỏa mãn
  - Tối ưu một hàm mục tiêu cho trước
- Tìm kiếm quay lui
  - Duyệt qua tất cả các biến (ví dụ thứ tự từ $X_1, X_2, \ldots, X_n$), với mỗi biến X_k
    - Duyệt lần lượt qua tất cả các giá trị có thể gán cho X_k, với mỗi giá trị v
      - Kiểm tra ràng buộc
      - Gán cho X_k
      - Nếu $k = n$ thì ghi nhận một phương án
      - Ngược lại, xét tiếp biến X_{k+1}

ĐỆ QUY QUAY LUI



Bài toán liệt kê

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

BÀI TOÁN LIỆT KÊ XÂU NHỊ PHÂN ĐỘ DÀI *n* (P.02.05.01)

- Cho số nguyên dương *n*. Hãy viết chương trình liệt kê tất cả các xâu nhị phân độ dài n theo thứ tự từ điển.
- Dữ liệu
  - Dòng 1: ghi số nguyên dương *n* (1 <= *n* <= 20)
- Kết quả
  - Ghi ra trên mỗi dòng một xâu nhị phân độ dài *n*

stdin	stdout
3	000 001 010 011 100 101 110 111

BÀI TOÁN LIỆT KÊ XÂU NHỊ PHÂN ĐỘ DÀI *n* – MÃ GIẢ

- Biểu diễn lời giải: $X_1, X_2, \ldots, X_n$
- Hàm try(*k*):
  - Duyệt các giá trị v từ 0 đến 1
- Hàm check(*v, k*):
  - luôn trả về true

```
check(v, k){
  return true;
}

try(k){
  for v = 0 to 1 do {
     if check(v, k) then {
       Xk = v;
       if k = n then solution();
       else try(k+1);
     }
  }
}
```

BÀI TOÁN LIỆT KÊ XÂU NHỊ PHÂN ĐỘ DÀI *n* – CODE HOÀN CHỈNH

```
#include <stdio.h>
#define N 21
int X[N];
int n;
int check(int v, int k){
    return 1;
}
void solution(){
    for(int i = 1; i <= n; i++) printf("%d",X[i]);
    printf("\n");
}
```

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

LIỆT KÊ XÂU NHỊ PHÂN ĐỘ DÀI *n* KHÔNG CHỨA 2 BÍT 1 CẠNH NHAU (B02-05-02)

- Cho số nguyên dương *n*. Hãy viết chương trình liệt kê tất cả các xâu nhị phân độ dài *n* (theo thứ tự từ điển) không chứa 2 bít 1 đứng cạnh nhau.
- Dữ liệu
  - Dòng 1: ghi số nguyên dương *n* (1 <= *n* <= 20)
- Kết quả
  - Ghi ra trên mỗi dòng một xâu nhị phân độ dài *n*

stdin	stdout
3	000 001 010 100 101

LIỆT KÊ XÂU NHỊ PHÂN ĐỘ DÀI *n* KHÔNG CHỨA 2 BÍT 1 CẠNH NHAU

- Biểu diễn lời giải: $X_1, X_2, \ldots, X_n$
- Hàm try(*k*):
  - Duyệt các giá trị v từ 0 đến 1
- Hàm check(*v, k*):
  - Nếu *k* = 1 thì trả về true
  - Nếu *k* > 1
    - Nếu *x*[*k*-1] = 1 và *v* = 1 thì trả về false
    - Ngược lại, trả về true

```
check(v, k){
  if k = 1 then return true;
  return x[k-1] + v <= 1;
}

try(k){
  for v = 0 to 1 do {
     if check(v, k) then {
       X_k = v;
       if k = n then solution();
       else try(k+1);
     }
  }
}
```

LIỆT KÊ XÂU NHỊ PHÂN ĐỘ DÀI *n* KHÔNG CHỨA 2 BÍT 1 CẠNH NHAU - CODE

```
#include <stdio.h>
#define N 21
int X[N];
int n;
int check(int v, int k){
    if(k==1) return 1;
    return X[k-1] + v <= 1;
}
void solution(){
    for(int i = 1; i <= n; i++) printf("%d",X[i]);
    printf("\n");
}
```

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

LIỆT KÊ HOÁN VỊ (P.02.05.03)

- Cho số nguyên dương *n*. Hãy viết chương trình liệt kê tất cả các hoán vị của 1, 2, . . ., *n* theo thứ tự từ điển
- Dữ liệu
  - Dòng duy nhất chứa số nguyên dương n (1 <= *n* <= 10)
- Kết quả
  - Ghi ra trên mỗi dòng 1 hoán vị (sau mỗi phần tử của hoán vị là 1 ký tự SPACE)

stdin	stdout
3	1 2 3 1 3 2 2 1 3 2 3 1 3 1 2 3 2 1

LIỆT KÊ HOÁN VỊ

- Biểu diễn lời giải: $X_1, X_2, \ldots, X_n$
- Mảng đánh dấu:
  - mark[v] = 1: có nghĩa v đã xuất hiện
  - mark[v] = 0: có nghĩa v chưa xuất hiện
- Hàm try(*k*):
  - Duyệt các giá trị v từ 1 đến n
- Hàm check(*v, k*):
  - Nếu mark[v] = 1 thì trả về false
  - Ngược lại, trả về true
- Khi gán $X_k = v$:
  - Thực hiện đánh dấu trạng thái mark[*v*] = 1

```
check(v, k){
  if mark[v] = 1 then return false;
  else return true;
}

try(k){
  for v = 1 to n do {
    if check(v, k) then {
      X_k = v;
      mark[v] = 1; // update status
      if k = n then solution();
      else try(k+1);
      mark[v] = 0; // recover when backtracking
    }
  }
}
```

LIỆT KÊ HOÁN VỊ – CODE HOÀN CHỈNH

```
#include <stdio.h>
#define N 20
int  n;
int x[N];
int mark[N];

int check(int v, int k){
    return mark[v] == 0;
}
void solution(){
    for(int i =1 ; i <= n; i++)
        printf("%d ",x[i]);
    printf("\n");
}
```

```
void Try(int k){
    for(int v = 1; v <= n; v++){
        if(check(v,k)){
            x[k] = v;
            mark[v] = 1; // update status of v
            if(k == n) solution();
            else Try(k+1);
            mark[v] = 0; // recover status when backtracking
        }
    }
}
int main(){
    scanf("%d",&n);
    for(int v = 1; v <= n; v++) mark[v] = 0;
    Try(1);
    return 0;
}
```

LIỆT KÊ NGHIỆM NGUYÊN DƯƠNG PHƯƠNG TRÌNH TUYẾN TÍNH (P.02.05.04)

- Cho số nguyên dương n và M, hãy viết chương trình liệt kê tất cả các bộ X_1, X_2, ..., X_n (theo thứ tự từ điển) sao cho

$$X_1 + X_2 + \ldots + X_n = M$$

- Dữ liệu
  - Dòng 1: chứa số nguyên dương n và M ($2 <= n <= 10$, $1 <= M <= 20$)
- Kết quả
  - Ghi ra trên mỗi dòng một bộ giá trị của X_1, X_2, ..., X_n (sau mỗi phần tử là 1 ký tự SPACE)

stdin	stdout
3 5	1 1 3 1 2 2 1 3 1 2 1 2 2 2 1 3 1 1

LIỆT KÊ NGHIỆM NGUYÊN DƯƠNG PHƯƠNG TRÌNH TUYẾN TÍNH – MÃ GIẢ

- Biểu diễn lời giải: $X_1, X_2, \ldots, X_n$
- Biến trung gian T: tổng giá trị của các biến đã được gán giá trị
- Hàm try(k):
  - $X_1 + X_2 + \ldots + X_{k-1} + X_k + X_{k+1} + \ldots + X_n = M$
  - $T = X_1 + X_2 + \ldots + X_{k-1} \rightarrow X_k \leq M - T - n + k$ (do $X_{k+1}, \ldots, X_n \geq 1$)
  - Duyệt các giá trị v từ 1 đến $M - T - n + k$
- Hàm check(v, k):
  - Nếu $k < n$ thì trả về true
  - Ngược lại
    - Nếu $T + v = M$ thì trả về true, ngược lại thì trả về false
- Khi gán $X_k = v$:
  - Thực hiện cập nhật $T = T + v$

```
check(v, k){
  if k < n then return true;
  else return (T + v = M);
}

try(k){
  for v = 1 to M – T – n + k do {
    if check(v, k) then {
      Xk = v;
      T = T + v; // update status
      if k = n then solution();
      else try(k+1);
      T = T - v; // recover when
backtracking
    }
  }
}
```

LIỆT KÊ NGHIỆM NGUYÊN DƯƠNG PHƯƠNG TRÌNH TUYẾN TÍNH – CODE

```
#include <stdio.h>
#define N 20
int  n, M;
int x[N];
int T;

int check(int v, int k){
  if(k < n) return 1;
  return T + v == M;
}
void solution(){
    for(int i = 1 ; i <= n; i++)
        printf("%d ",x[i]);
    printf("\n");
}
```

```
void Try(int k){
    for(int v = 1; v <= M - T - n + k; v++){
        if(check(v,k)){
            x[k] = v;
            T = T + v; // update status of T
            if(k == n) solution();
            else Try(k+1);
            T = T - v; // recover status when backtracking
        }
    }
}
int main(){
    scanf("%d%d",&n, &M);
    T = 0;
    Try(1);
    return 0;
}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !