Nội dung: Thuật toán đệ quy

* Tóm tắt lý thuyết về thuật toán đệ quy
* Một số bài toán đệ quy đơn giản
* Đệ quy quay lui – back tracking

Thuật toán đệ quy

* Trong phần thân có lời gọi đến chính nó
* Thời gian gọi đệ quy sẽ ảnh hưởng đến thời gian thực hiện của thuật toán --> nên giảm tối thiểu số lần phải gọi đệ quy khi tính toán
  + Dùng biến phụ để tránh phải gọi lại nhiều lần
  + Dùng bảng tra để lưu trữ lời giải các bài toán con

VD. Tính số Fibonacci thứ n

F(n) = F(n-1) + F(n-2)

Và F(0)=1, và F(1) = 1

==============================

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE\_MAX 45

int M[SIZE\_MAX+2];

int Fib1(int n)

{

if(n==0||n==1) return 1;

return Fib1(n-1)+Fib1(n-2);

}

// cach 1, dung them bang tra top-down

int Fib2(int n)

{

if(n==0||n==1) return 1;

// check xem bai toan con da giai chua, neu giai roi thi tra ve ket qua

if(M[n]!=0) return M[n];

// chua giai--> di giai va ghi ket qua

M[n] = Fib2(n-1)+Fib2(n-2);

return M[n];

}

// cach 2, dung them bang tra bottom-up

int Fib3(int n)

{

M[0]=1; M[1]=1;

for(int i=2;i<=n;i++)

M[i] = M[i-1]+M[i-2];

return M[n];

}

int main() {

memset(M, 0, SIZE\_MAX\*sizeof(int));

//printf("Fib(n) = %d\n",Fib2(SIZE\_MAX));

printf("Fib(n) = %d\n",Fib3(SIZE\_MAX));

//printf("Fib(n) = %d\n",Fib1(SIZE\_MAX));

return 0;

}

===============================

Thuật toán back tracking – quay lui

Sinh hoán vị

* Sinh chuỗi n phần tử thỏa mãn điều kiện

VD. Sinh chuỗi nhị phân độ dài 5

===============

void printSolution(int \*x, int n) {

for (int k = 0; k < n; k++)

printf("%d", x[k]);

printf("\n");

}

int TRY(int \*x, int n, int k) {

for (int v = 0; v <= 1; v++) {

x[k] = v;

if (k == n - 1) printSolution(x,n);

else TRY(x, n, k + 1);

}

}

int main() {

int x[5],n=5;

TRY(x,n,0);

}

=============-=

VD 2. Sinh chuỗi nhị phân mà KHÔNG có 2 bit 1 liên tiếp

===============

void printSolution(int \*x, int n) {

for (int k = 0; k < n; k++)

printf("%d", x[k]);

printf("\n");

}

int TRY(int \*x, int n, int k) {

for (int v = 0; v <= 1; v++) {

if (k>0 && v + x[k - 1] == 2) continue; // check DK

x[k] = v;

if (k == n - 1) printSolution(x, n);

else TRY(x, n, k + 1);

}

}

int main() {

int x[5], n = 5;

TRY(x, n, 0);

}

===============

Bài tập 1. Sinh ra chuỗi nhị phân độ dài n (n<10) mà KHÔNG có 2 số 0 hoặc 2 số 1 liên tiếp

Bài tập 2. Sinh ra số hệ 10 có n chữ số (các số đầu không được là 0) mà không có 2 số liên tiếp giống nhau.  
1212121212  
KHÔNG 1221

====================================

In ra hoán vị của tập hợp {1,2,3,4,..,MAX\_SIZE}

================

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_SIZE 3

#define true 1

#define false 0

int m[MAX\_SIZE], x[MAX\_SIZE];

void printSolution(){

for(int k = 0; k < MAX\_SIZE; k++)

printf("%d",x[k]);

printf("\n");

}

void TRY(int i) {

for(int v = 1; v <= MAX\_SIZE; v++){

if(!m[v]) {

x[i] = v;

m[v-1] = true; // đánh dấu

if(i == MAX\_SIZE-1)

printSolution();

else TRY(i+1);

m[v-1] = false;// khôi phục

}

}

}

void main() {

for(int v = 0; v < MAX\_SIZE; v++)

m[v] = false;

TRY(0);

}

================

**Bài tập 3**. Viết chương trình in ra các hoán vị của các ký tự trong xâu

“ABACD”, hoặc “ABCD” Có thể có ký tự trùng (vẫn được sử dụng)

AAB --> AAB, ABA, BAA

**Bài tập 4**. Viết chương trình in ra các số có n (n<10) chữ số sao cho

* tổng các chữ số bằng 9
* Các chữ số theo thứ tự không giảm: 12234, 13445,…

**Bài tập 5**. Viết chương trình in ra các giá trị x1, x2, x3 thỏa mãn các điều kiện sau

* X1 + x2 + x3 = 1000
* X1 = 2 \* x2
* X2 > x3