HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

C BASIC

RECURSION

CONTENT

- Recursion
- The greatest common divisor problem (P.02.04.01)
- Converting integers to binary bit strings (P.02.04.02)
- Hanoi Tower (P.02.04.03)
- Memorized recursion
- Calculating Fibonacci sequence (P.02.04.04)
- Calculating combination constant (P.02.04.05)

RECURSION

- An object having recursive structures is defined/constructed based on itself with smaller sizes
- A recursive function is a function call to itself with smaller parameter sizes.
- A recursive algorithm (normally in the form of a recursive function) is suitable for processing, calculating recursive objects)

$$F(n) = \begin{cases} F(n-1) + n, & n \geq 2 \\ 1, & \text{khi } n = 1 \end{cases}$$

$$F(n) = \begin{cases} F(n-1) + F(n-2), & n \geq 2 \\ n, & \text{khi } n = 0, 1 \end{cases}$$

$$F(a,b) = \begin{cases} a, & \text{nếu } a = b \\ F(a-b, b), & \text{nếu } a > b \\ F(a, b-a), & \text{nếu } a < b \end{cases}$$

$$C(k, n) = \begin{cases} 1, & \text{khi } k = 0 \text{ hoặc } k = n \\ C(k,n-1) + C(k-1,n-1), & \text{ngược lại} \end{cases}$$

THE GREATEST COMMON DIVISOR PROBLEM (P.02.04.01)

- Given two positive integer a and b. Write a program to find the greatest common divisor of a and b.
- Data
  - Line 1: Two positive integer a and b (1 <= a, b <= 100000)
- Result
  - The greatest common integer of a and b

stdin	stdout
16 24	8

THE GREATEST COMMON DIVISOR PROBLEM– PSEUDOCODE

- If a = b then USCLN(a, b) = a
- If a > b then USCLN(a, b) = USCLN(a-b, b)
- If a < b then USCLN(a, b) = USCLN(a, b-a)

```
F(a, b){
   if a = b then return a;
   if a > b then return F(a-b, b);
   else return F(a, b-a);
}
```

CONVERTING INTEGERS TO BINARY BIT STRINGS (P.02.04.02)

- Given a positive integer N, write a program to convert N to a bit string (Ignore bits 0 on the lelftmost part)
- Data
  - One line with a positive integer N (1 <= N <= 2x10^{7})
- Result
  - One line for the bit string

stdin	stdout
20	10100

CONVERTING INTEGERS TO BINARY BIT STRINGS– PSEUDOCODE

- Calling recursively to convert N/2 to a bit string, after that, combine the result with the rightmost bit (N mod 2)

```
Convert(N){
   if N = 0 then return;
   Convert(N/2);
   b = N mod 2;
   print(b);
}
```

HANOI TOWER (P.02.04.03)

- Given n disks with different radii and 3 piles A, B, C. Initially n disks are located at pile A in order of small disk above and large disk below. Find a way to transfer n disks from pile A to pile B (using pile C as an intermediary) according to the principle
  - At each step, only 1 top disc can be transferred from 1 pile to 1 other pile (placed on top).
  - It is not allowed for a large disc to lie above a small disc at a certain pile
- Data
  - One line with 4 positive integers: n, A, B, C ($1 <= n <= 20$, $1 <= A, B, C <= 100$)
- Result
  - Line 1: an integer m (the number of steps)
  - Line i + 1 (i = 1, 2, …, m) contains 2 positive integers X and Y: at step i, move 1 disk from pile X to pile Y

stdin	stdout
2 11 22 33	3 11 33 11 22 33 22

HANOI TOWER - PSEUDOCODE

- Algorithm:
  - Move n-1 disks from pile A to pile C, taking B as the intermediate pile
  - Move 1 disc from peg A to peg B
  - Move n-1 disks from pile C to pile B, taking A as the intermediate pile

- The number of steps: 2^n-1

```
move(n, A, B, C){
  if n = 1 then print(A, B);
  else {
     move(n-1, A, C, B);
     move(1, A, B, C);
     move(n-1, C, B, A);
  }
}
```

CALCULATING FIBONACCI SEQUENCE (P.02.04.04)

- Given a positive integer n, calculate the nth Fibonacci number
- Data
  - Line 1: a positive integer n (2 <= n <= 100000)
- Result
  - Write the value of F(n) mod 10^9+7

$$F(n) = \begin{cases} F(n-1) + F(n-2), & n \geq 2 \\ n, & \text{khi } n = 0, 1 \end{cases}$$

stdin	stdout
10	55

CALCULATING FIBONACCI SEQUENCE - PSEUDOCODE

- Given a positive integer n, calculate the nth Fibonacci number
- Data
  - Line 1: a positive integer n (2 <= n <= 100000)
- Result
  - Write the value of F(n) mod 10^9+7

```
F(n){
  if n <= 1 then return n;
  return (F(n-1) + F(n-2)) mod 10^9+7;
}
```

CALCULATING COMBINATION CONSTANT (P.02.04.05)

- Given two non-negative integer k and n, calculating the combination constant C(k, n)
- Data
  - One line with two non-negative integer k and n (0 <= k, n <= 999)
- Result
  - Write the result of C(k,n) mod 10^9+7

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

stdin	stdout
3 5	10

CALCULATING COMBINATION CONSTANT - PSEUDOCODE

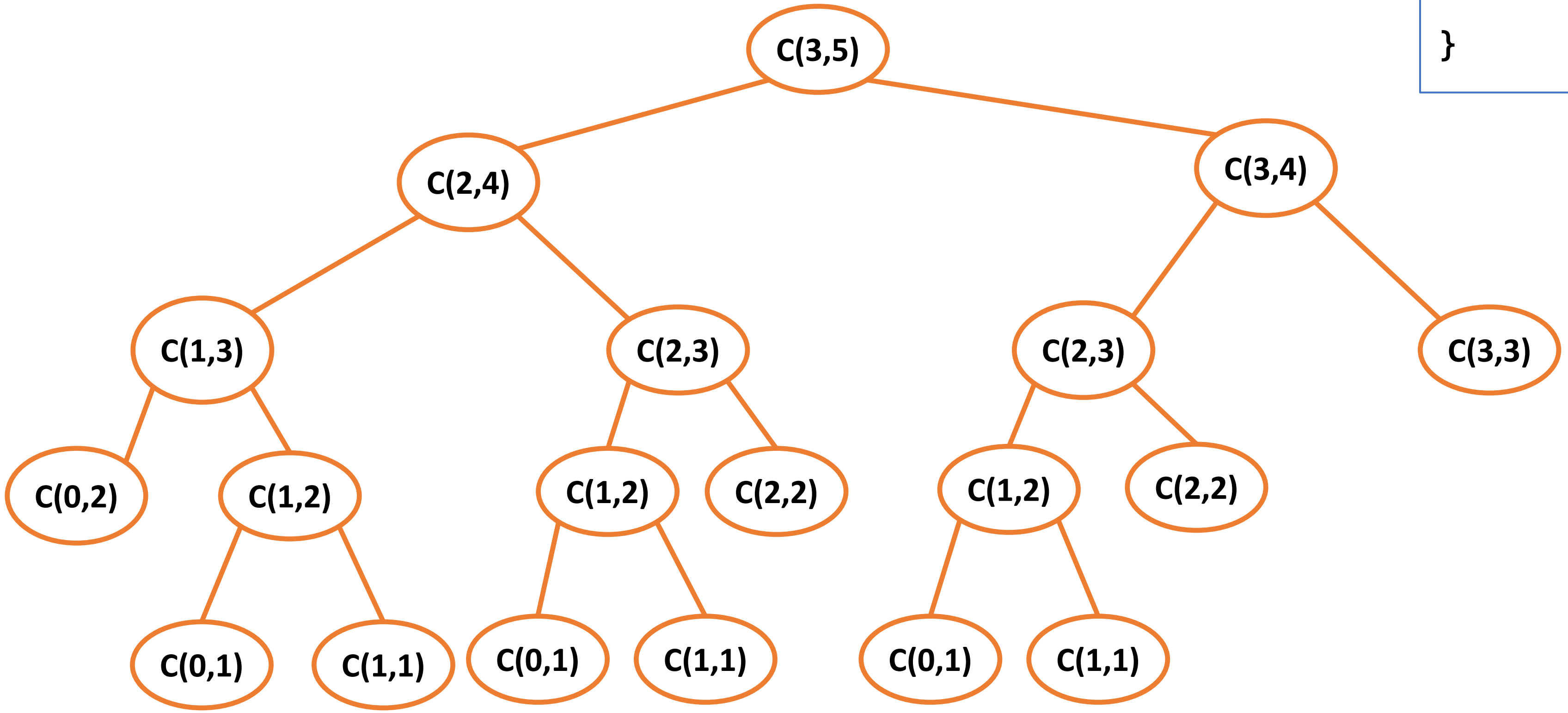
- Given two non-negative integer k and n, calculating the combination constant C(k, n)
- Data
  - One line with two non-negative integer k and n (0 <= k, n <= 999)
- Result
  - Write the result of C(k,n) mod 10^9+7

```
C(k, n){
   if k = 0 or k = n then return 1;
   return (C(k-1, n-1) + C(k, n-1)) mod 10^9+7;
}
```

CALCULATING COMBINATION CONSTANT– MEMORIZED RECURSION

- Recursive algorithm to calculate C(k , n)

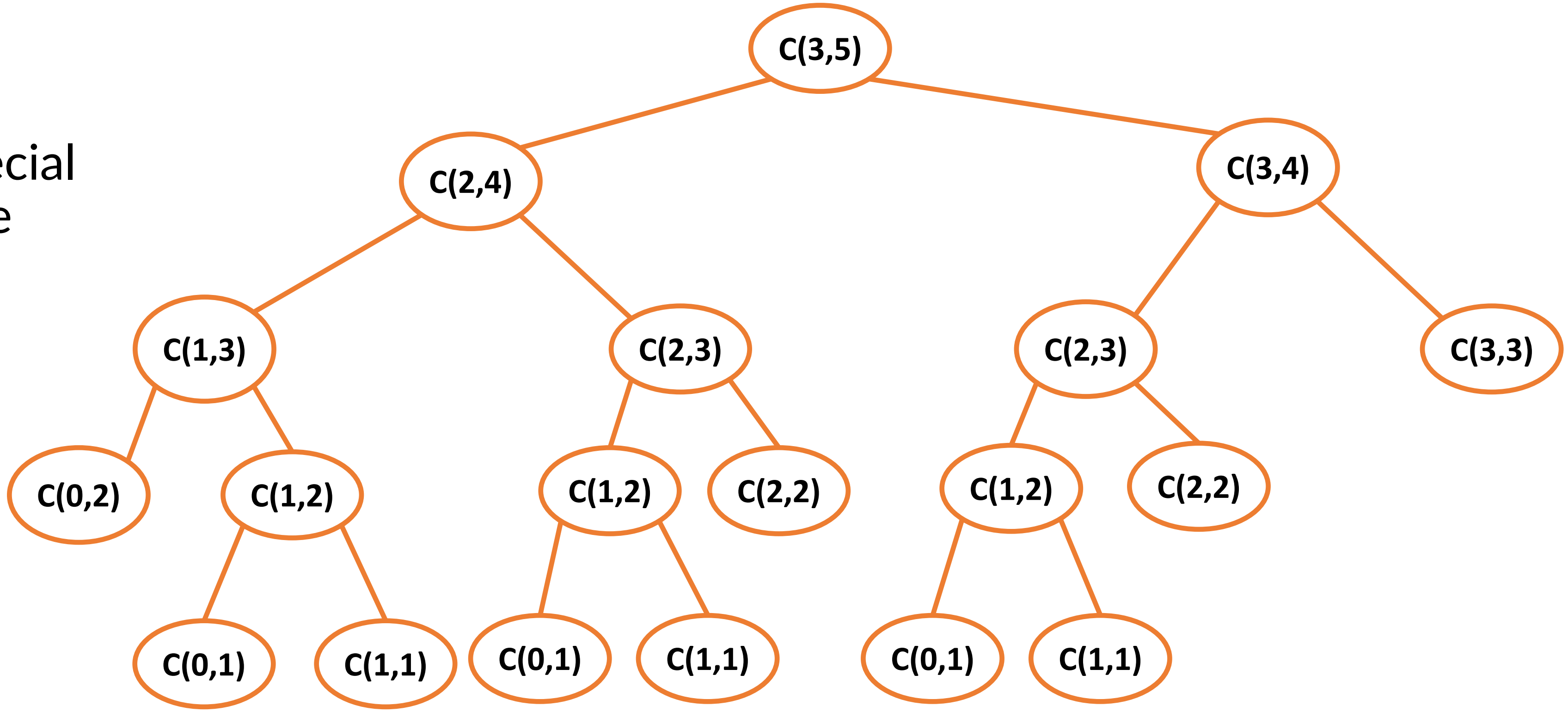
```
int C(int k, int n){
   if (k == 0 || k == n) return 1;
   return C(k-1, n-1) + C(k, n-1);
}
```



CALCULATING COMBINATION CONSTANT– MEMORIZED RECURSION

- Fix the situation where a subprogram with specified parameters is called recursively multiple times
- Use memory to store the results of a subroutine with specified parameters
- Memory is initialized with a special value to record each subroutine that has not been called yet
- The memory address will be mapped with the subroutine parameter values

```
int C(int k, int n){
   if (k == 0 || k == n) return 1;
   return C(k-1, n-1) + C(k, n-1);
}
```



CALCULATING COMBINATION CONSTANT– MEMORIZED RECURSION - PSEUDOCODE

- Fix the situation where a subprogram with specified parameters is called recursively multiple times
- Use memory to store the results of a subroutine with specified parameters
- The memory is initialized with a special value (for example, value 0) to record each subroutine that has not been called yet.
- The memory address will be mapped with the subroutine parameter values

```
M[N,N] = {0}; // Initialize 0-array as a memory
              // M[k,n] stores the value C(k,n)
C(k, n){
   if (k == 0 || k == n) M[k,n] = 1;
   else {
      if M[k,n] = 0 then {
         M[k,n] = C(k-1,n-1) + C(k,n-1);
      }
   }
   return M[k,n];
}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !