Algorithms & Recursions

Kuan-Yu Chen (陳冠宇)

2018/09/12 @ TR-212, NTUST

Algorithm

- The concept of an algorithm is fundamental to computer science
  - A program **does not** have to satisfy the fourth condition

Definition: An *algorithm* is a finite set of instructions that, if followed, accomplishes a particular task. In addition, all algorithms must satisfy the following criteria:

(1) **Input.** Zero or more quantities are externally supplied.

(2) **Output.** At least one quantity is produced.

(3) **Definiteness.** Each instruction is clear and unambiguous. (明確性)

(4) **Finiteness.** If we trace out the instructions of an algorithm, then for all cases, the algorithm terminates after a finite number of steps. (有限性)

(5) **Effectiveness.** Every instruction must be basic enough to be carried out, in principle, by a person using only pencil and paper. It is not enough that each operation be definite as in (3); it also must be feasible. □ (有效性)

Recursive Algorithms.

- The recursive mechanisms are extremely powerful, because they often can express a complex process very clearly

- Recursive functions can be categorized into three classes
  - Direct Recursion
    - The function may call itself before it is done
  - Indirect Recursion
    - The function may call other functions that again invoke the calling function
  - Tail Recursion
    - The function may call itself at the end of the function
    - A special case of direct recursion

Recursive Algorithms..

Direct Recursion

```
function A()
{
    ...

    A() ;

    ...
}
```

Indirect Recursion

```
function A()
{
    ...

    B() ;

    ...
}

function B()
{
    ...

    A() ;

    ...
}
```

calling cycle

Tail Recursion

```
function A()
{
    ...

    A() ;
}
```

Recursive Algorithms…

- Let's make a comparison

Recursion	Non-Recursion
Codes are more compact	Codes are complicated
Easy to understand	Hard to read
Time-consuming	Time-saving

Examples – 1

- Please write down a recursive program to do factorial.

$$0! = 1$$

$$1! = 1$$

$$2! = 1 \times 2$$

$$3! = 1 \times 2 \times 3$$

$$n! = 1 \times 2 \times 3 \times \cdots \times n$$

```
int factorial( int a )
{
    if( a == 0 )
        return 1 ;
    else
        return factorial(a-1)*a ;
}
```

Examples – 2.

- Please (1) write a recursive program, $Fib(\ int\ a\ )$, to calculate the Fibonacci number; (2) how many function calls do you need to do when we want to calculate $Fib(5)$?

Fibonacci number

From Wikipedia, the free encyclopedia

In mathematics, the **Fibonacci numbers** are the numbers in the following integer sequence, called the **Fibonacci sequence**, and characterized by the fact that every number after the first two is the sum of the two preceding ones:[1][2]

$$1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ 144,\ \ldots$$

Often, especially in modern usage, the sequence is extended by one more initial term:

$$0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ 55,\ 89,\ 144,\ \ldots$$ [3]

By definition, the first two numbers in the Fibonacci sequence are either 1 and 1, or 0 and 1, depending on the chosen starting point of the sequence, and each subsequent number is the sum of the previous two.

The sequence F_n of Fibonacci numbers is defined by the recurrence relation:

$$F_n = F_{n-1} + F_{n-2},$$

with seed values[1][2]

$$F_1 = 1,\ F_2 = 1$$

or[5]

$$F_0 = 0,\ F_1 = 1.$$

Examples – 2..

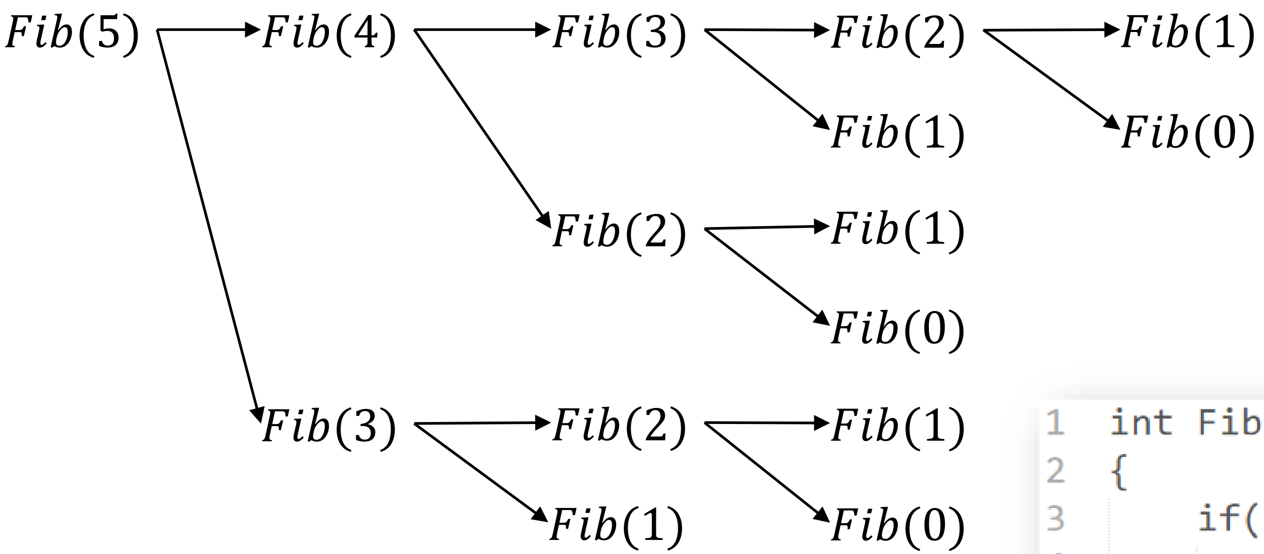
- Please (1) write a recursive program, $Fib(\ int\ a\ )$, to calculate the Fibonacci number; (2) how many function calls do you need to do when we want to calculate $Fib(5)$?

$$Fib(a) = \begin{cases} 0, & if\ a = 0 \\ 1, & if\ a = 1 \\ Fib(a-1) + Fib(a-2), & otherwise \end{cases}$$

```
int Fib( int a )
{
    if( a == 0 )
        return 0 ;
    else if( a == 1 )
        return 1 ;
    else
        return Fib(a-1)+Fib(a-2) ;
}
```

Examples – 2...

- Please (1) write a recursive program, $Fib(\ int\ a\ )$, to calculate the Fibonacci number; (2) how many function calls do you need to do when we want to calculate $Fib(5)$?




```
int Fib( int a )
{
    if( a == 0 )
        return 0 ;
    else if( a == 1 )
        return 1 ;
    else
        return Fib(a-1)+Fib(a-2) ;
}
```

Examples – 3.

- Given an Ackerman's function $A(m,n)$, please calculate $A(1,2)$.

$$A(m,n) = \begin{cases} n+1, & if\ m = 0 \\ A(m-1,1), & if\ n = 0 \\ A\big(m-1, A(m,n-1)\big), & otherwise \end{cases}$$

$$A(1,2) = A\big(0, A(1,1)\big)$$

$$A(1,1) = A\big(0, A(1,0)\big)$$

$$A(1,0) = A(0,1)$$

$$A(0,1) = 2$$

Examples – 3..

- Given an Ackerman's function $A(m,n)$, please calculate $A(1,2)$.

$$A(m,n) = \begin{cases} n+1, & if\ m=0 \\ A(m-1,1), & if\ n=0 \\ A\big(m-1, A(m,n-1)\big), & otherwise \end{cases}$$

$$A(1,2) = A\big(0, A(1,1)\big) = A(0,3) = 4$$

$$A(1,1) = A\big(0, A(1,0)\big) = A(0,2) = 3$$

$$A(1,0) = A(0,1) = 2$$

$$A(0,1) = 2$$

Questions?

kychen@mail.ntust.edu.tw