Algoritma & Pemrograman Saintifik

Gatot F. Hertono, Ph.D

Departemen Matematika

SCMA601401

What do we know about these algorithm?

```
Function Fib1(n)
Input: n (n ≥ 0)
Output: Bilangan Fibonacci ke-n

if n ≤ 1 then
  return(n)
else
  return(Fib1(n-1) + Fib1(n-2))
endif

end Fib1
```

Recursive

```
Function Fib2(n)
Input: n (n ≥ 0)
Output: Bilangan Fibonacci ke-n

if n ≤ 1 then
    return(n)
else
     Prev:=0;
     Curr:=1
     for i:=2 to n do
        Next := Prev + Curr;
        Prev:=Curr;
        Curr := Next;
     endfor
endif
return(Curr)
end Fib2
```

Iterative

Iterative vs Recursive

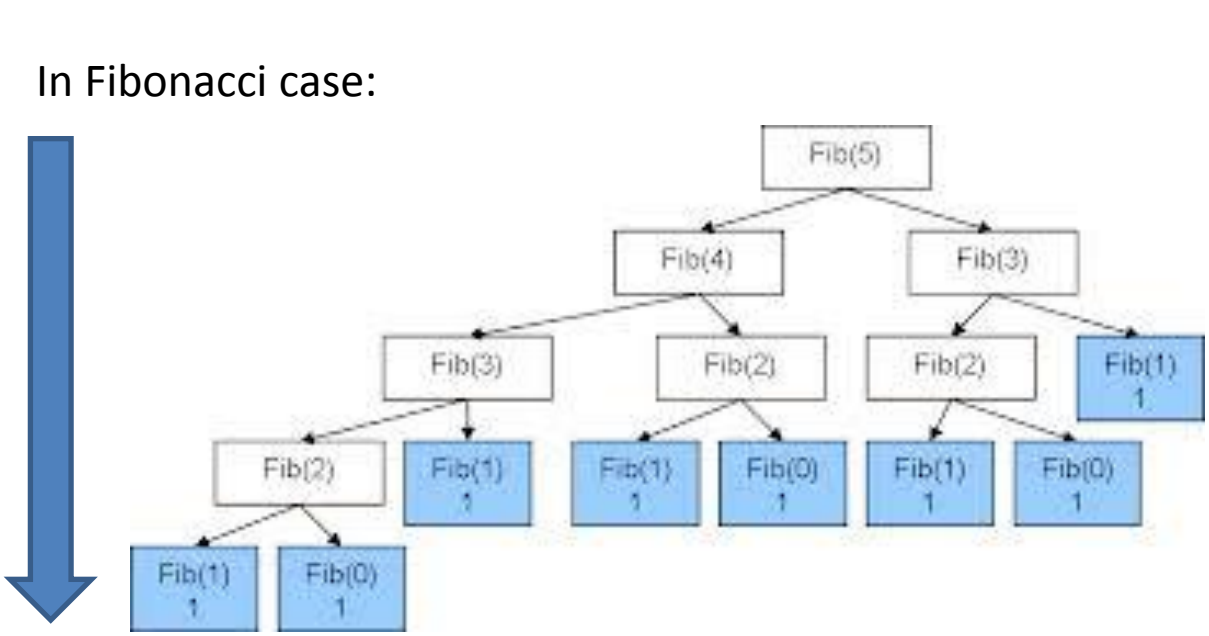
In computer programs, repetition is accomplished in one of two ways: either through *recursion* or through *iteration*.

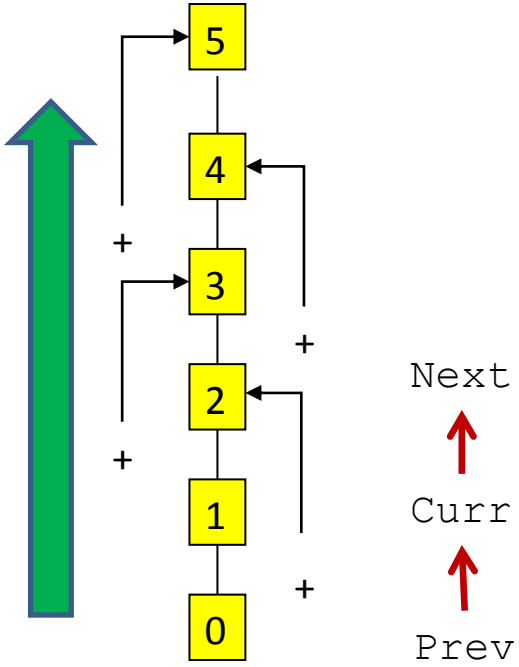
In general, recursion and iteration perform the same kinds of tasks:

solve a complicated task one piece at a time, and combine the results.

Both approaches result in a process being repeated several times.

In Fibonacci case:



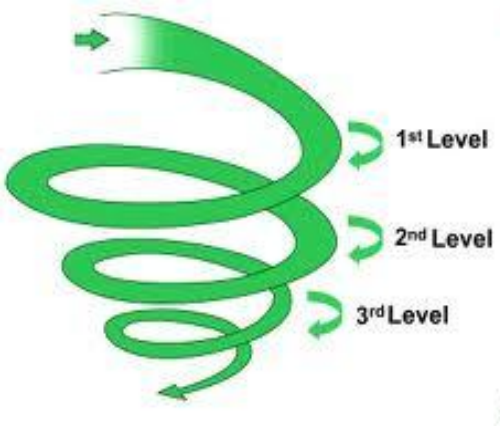


Top-Down **Bottom-Up**

Iterative

Iterative structures usually refers to structures that contain explicit repetitions of a process, that is,

loops.



A loop must have some sort of stopping criterion. Usually it is of one of two type:

- ✓ predetermined number of iterations through the loop;
- ✓ a specific condition that is achieved.

```
FOR i = 0 TO 9 DO
    Procedure
```

```
i:=0
WHILE (i < 10) DO
   Procedure
   i := i + 1
```

```
i:=0
REPEAT
   Procedure
   i := i + 1
UNTIL (i >= 10)
```

Iterative

Emphasis of iteration: keep repeating until a task is "done"

e.g., loop counter reaches limit

Example 1: $n!$

```
Function fact(n)
Input: n (integer) (n ≥ 0)
Output: n!

running_total := 1
while (n > 1)
  running_total := running_total × n
  n := n -1
end
return (running_total)

end fact
```

Example 2: x^n

```
Function Power(x: real, n: integer)
Input: x (real),  n (n (integer) ≥ 0)
Output: x^n

Hasil := x
for i:=1 to n-1 do
        Hasil := Hasil * x
endfor
return(Hasil)
end Power
```

Recursive

Recursion in computer science is a method where the solution to a problem depends on solutions to smaller instances of the same problem. The approach can be applied to many types of problems, and recursion is one of the central ideas of computer science. (Wikipedia)

Suatu fungsi rekursif $f(x)$: adalah suatu fungsi dimana evaluasinya untuk suatu input x_i (x_i bukan initial input x_0) memerlukan evaluasi fungsi dirinya sendiri untuk input x_j yang lain.

Contoh:

Fact(n) = n * Fact(n-1), dengan kondisi awal: Fact(1) = 1
Fibo(n) = Fibo(n-1)+Fibo(n-2), dengan kondisi awal: Fibo(0)=0, Fibo(1)=1
Gcd(a,b)=Gcd(b, a mod b), dengan kondisi awal: Gcd(a,0) = a

A mathematical definition of a function is recursive if the function is defined in terms of itself (with a slightly smaller argument), and a computer function (subroutine) is recursive if it invokes itself (with slightly different arguments).

Recursive

Emphasis of recursion: solve a large problem by breaking it up into smaller and smaller pieces until you can solve it; combine the results

A classic example:

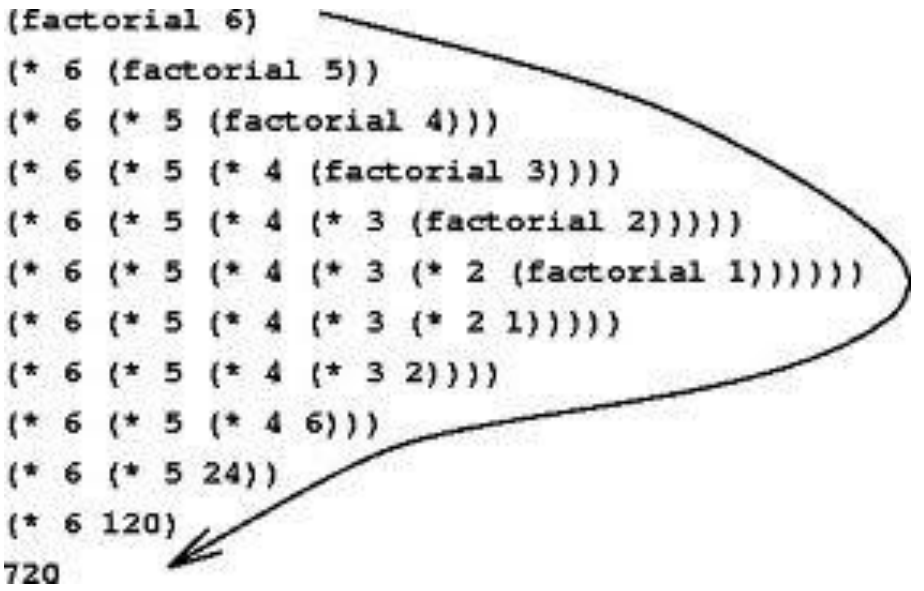
$$fact(n) = \begin{cases} 1 & , \text{ if } n = 0 \\ n.fact(n-1) & , \text{ if } n > 0 \end{cases}$$

Recursive version

```
Function fact(n)
Input: n (integer) (n ≥ 0)
Output: n!

if n = 0 then
  return(1)
else
  return(n * fact(n-1))
endif

end fact
```



Notes: Recursive codes have no loops. Repetition is achieved when the subprogram calls itself repeatedly until it reaches the base case.

Iterative vs Recursive

In performance:

Example 1: *N!*

N	Recursive	Iterative
10	334 ticks	11 ticks
100	846 ticks	23 ticks
1000	3368 ticks	110 ticks
10000	9990 ticks	975 ticks
100000	stack overflow	9767 ticks

Example 2: *Fibonacci(n)*

(see the slide #3)

Which One is Better?

- No clear answer, but there are known trade-offs.
- "Mathematicians" often prefer recursive approach.
  - Solutions often shorter, closer in spirit to abstract mathematical entity.
  - Good recursive solutions may be more difficult to design and test.
- "Programmers", esp. w/o college CS training, often prefer iterative solutions.
  - Somehow, it seems more appealing to many.
  - Control stays local to loop, less "magical".

Exercises

1. Write an algorithm to compute the mean and variance of 1000 data
2. Write a recursive algorithm to compute: $a_n = 2^{n-1}, \text{ with } a_1 = 1$
3. Design an algorithm to compute:

$$f(t) = \frac{\sqrt{t}(t-5)^2}{4}, \quad t \in [0, 10]$$

with

$$\Delta t = 0.25$$