Recursion

Yun Tian (Tony)
Department of Computer Science
Eastern Washington University
CSCD300-01

Outline

- Why Recursion?
- Definition
- Examples

Why Recursion?

- Want to work with Google, Amazon, Facebook or Microsoft ?
- 80% of technical interview questions designed to use recursion.

Why Recursion?

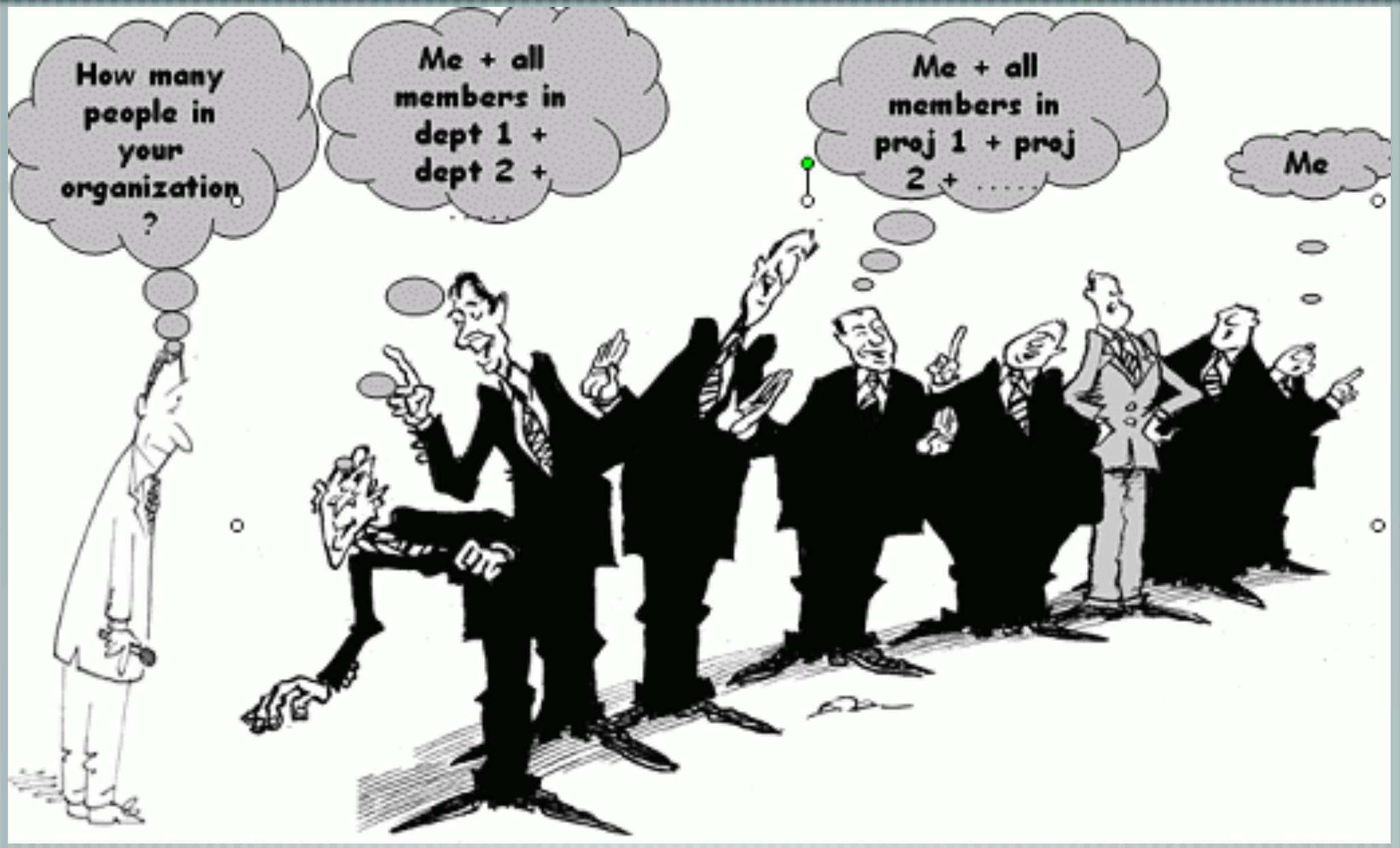
- Recursion is more than programming skills, but problem solving skills.
- Well related to many algorithm design paradigms and analysis, such as Divide and Conquer, Dynamic programming, even exhaustive search.
- Well related to many data structures,

  such as binary tree and linked list.

Definition of Recursion

- We divide a problem into smaller subproblems of the same type.
- Repeat this process till subproblems are solvable.
- Solving higher level subproblems uses solutions of lower level of subproblems.

Definition of Recursion



Figure, Recursive Pattern

www.codeproject.com/Articles/29036/Patterns-in-Real-Life

Example of Recursion

- Factorial Example
- $5! = 5 \times 4 \times 3 \times 2 \times 1$

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times .. \times 2 \times 1 \quad (1)$$

$$n! = n \times (n-1)! \quad (2)$$

Example of Recursion

- Factorial Example Continuted
- We reduce the size of problem to (n-1). We could further reduce the problem towards 0! = 1.
- We represent original problem using subproblem(s) of the same type, but with smaller size.

Call Stack

Call Stacks

- A call stack is a data structure used by the program to store information about the active subroutines (like functions in C++ or methods in Java) in a program.
- The main reason for having a call stack is so that the program can keep track of where a subroutine should return control to once it finishes executing.
- A stack frame is an element of the call stack, and a new stack frame is created every time a subroutine is called.

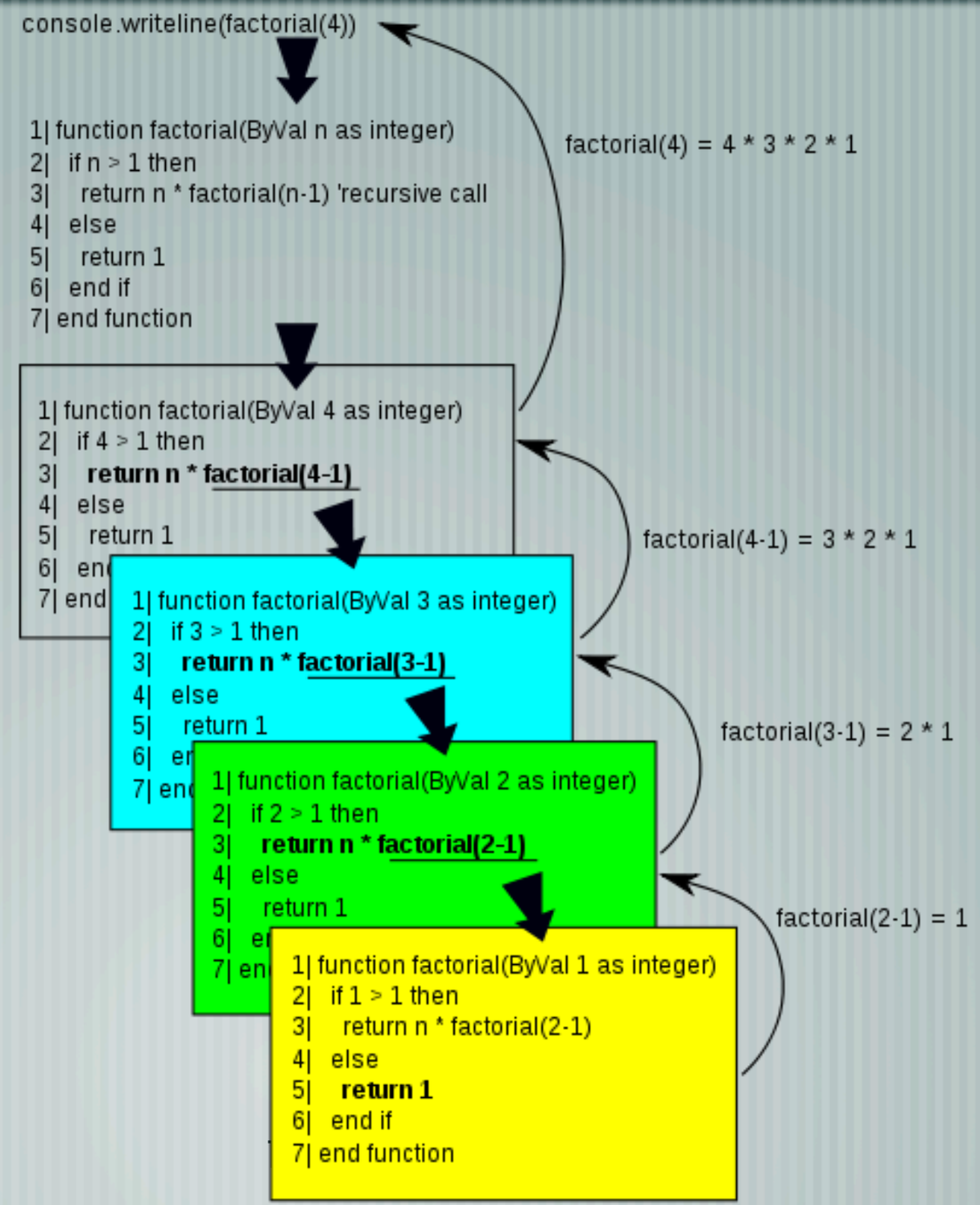
Call Stack

Call Stacks

- Example, suppose we have a method “CreateBox” which calls another method “CreateLine” in 4 different places.
- If the program has finished executing the method CreateLine, then it needs to know where in the CreateBox method it needs to return to.

Example of Recursion

Diagram showing function calls and return values on recursive solution to factorial(4)



Each Function(or method) call results in a record in the call stack.(Data in stack frame include local variables, arguments,

http://en.wikibooks.org/wiki/A-level_Computing/AQA/Problem_Solving,_Programming,_Operating_Systems,_Databases_and_Networking/Programming_Concepts/Recursive_Techniques

Formal Definition

- Recursion is defined by two properties:
  - One or more simple base cases (lowest level subproblems that are solvable)
  - A set of rules which reduce all other cases toward the base case. (recursive definition or repeated division of a larger problem)

Formal Definition

- Example of Fibonacci Sequence
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...
- Two properties of Recursion in defining Fibonacci Sequence
  - Fib(0) is 0 and Fib(1) is 1. [base case]
  - For all integers n > 1: Fib(n) is Fib(n-1) + Fib(n-2). [Recursive Definition]

Fibonacci Sequence

Implementing Fibonacci Sequence

- Recursion is characterized by calling itself.

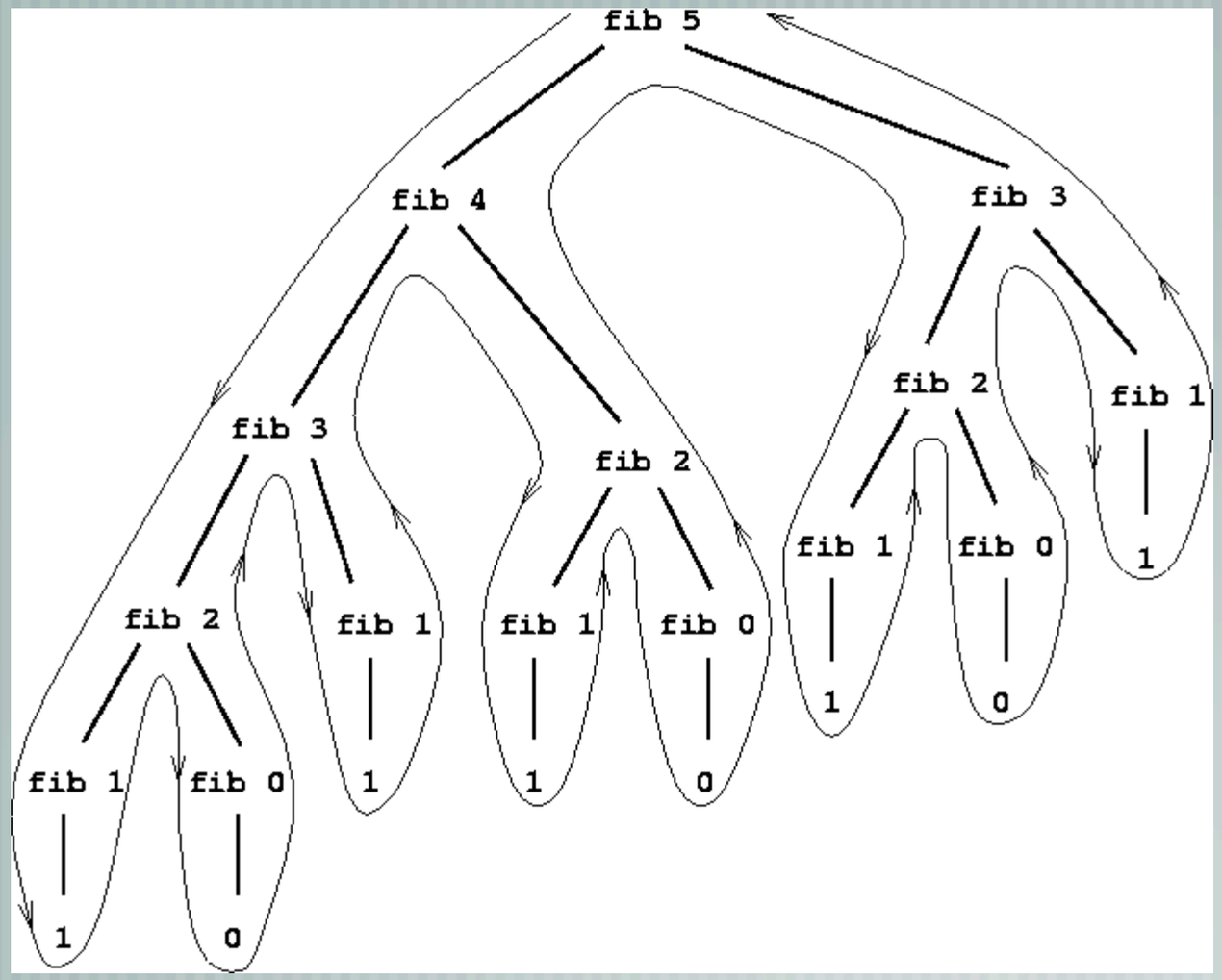
```
int fib(int n) {
    if (n <= 1)            // base case
        return n;          //what if we forget the base case?

    return fib(n-1) + fib(n-2);  //recursive definition
}
```

This implementation is NOT optimal.

Fibonacci Sequence

Implementing Fibonacci Sequence

Recursion is characterized by calling itself.



Picture form http://mitpress.mit.edu/sicp/full-text/sicp/book/node16.html.

Time Complexity: T(n) = T(n-1) + T(n-2) which is exponential, around 1.6^n
We can observe that this implementation does a lot of repeated work (see the left recursion tree). So this is a bad implementation for Nth Fibonacci number.

Take Home Summary

- Definition of Recursion
- Why we have to learn Recursion?
- Call stack and stack frame
- Fibonacci Sequence

Next Class

- Improved Fibonacci Sequence
- Memoization
- More examples