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?

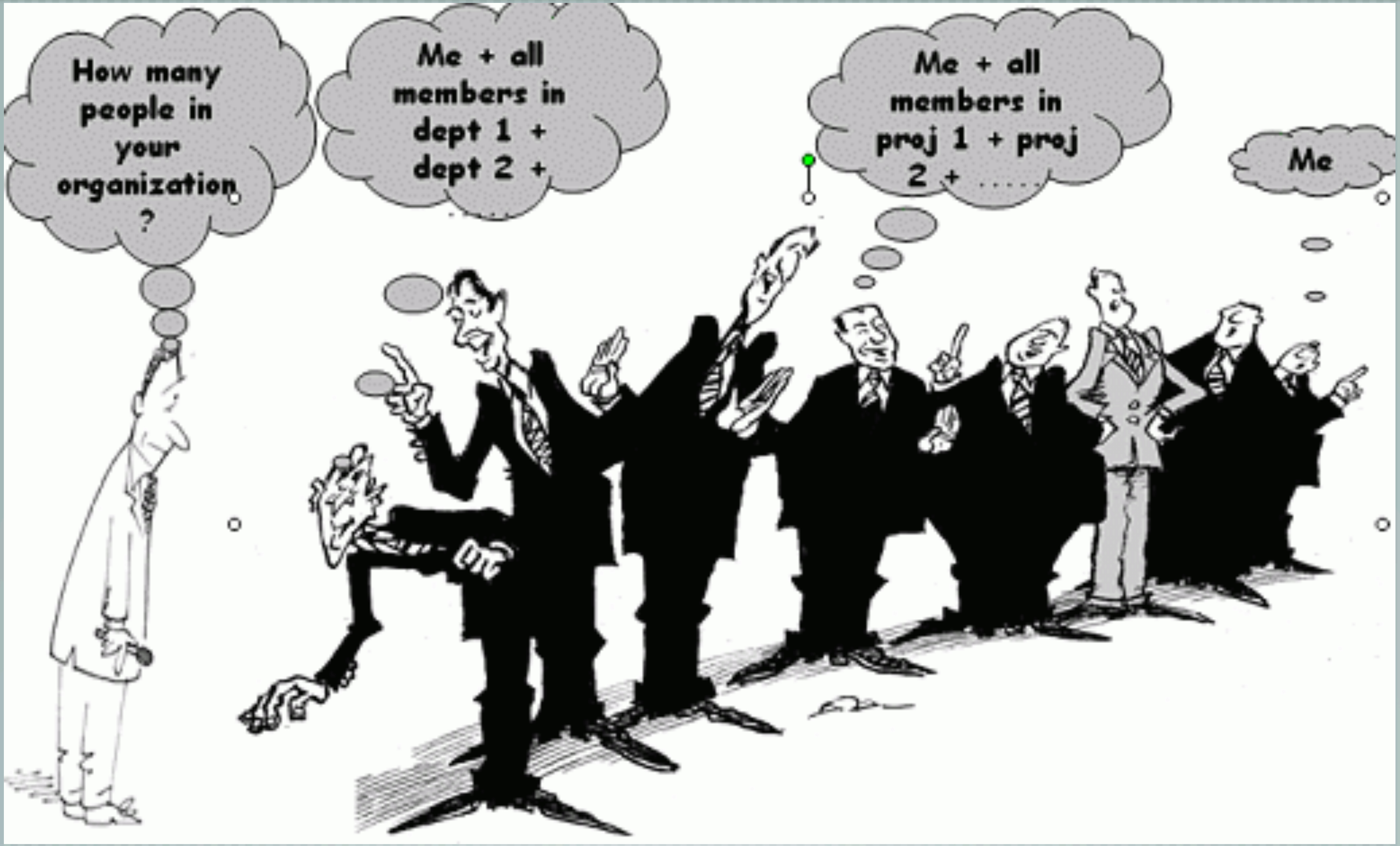
- Recursion is more than programming skills, but problem solving skills.
  - Well related to many algorithm design paradigms and analysis, such as Divide and Conquer, Dynamic programming, even exhaustive search.
  - Well related to many data structures,

    such as binary tree and linked list.

Pattern 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


How many people in your organization?
Me + all members in dept 1 + dept 2 +
Me + all members in proj 1 + proj 2 + ...
Me

Figure, Recursive Pattern

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

Example of Recursion

- Factorial Example
- $5! = 5 \times 4 \times 3 \times 2 \times 1$
- $n! = n \times \underbrace{(n-1) \times (n-2) \times (n-3) \times .. \times 2 \times 1}_{\downarrow}$ (1)

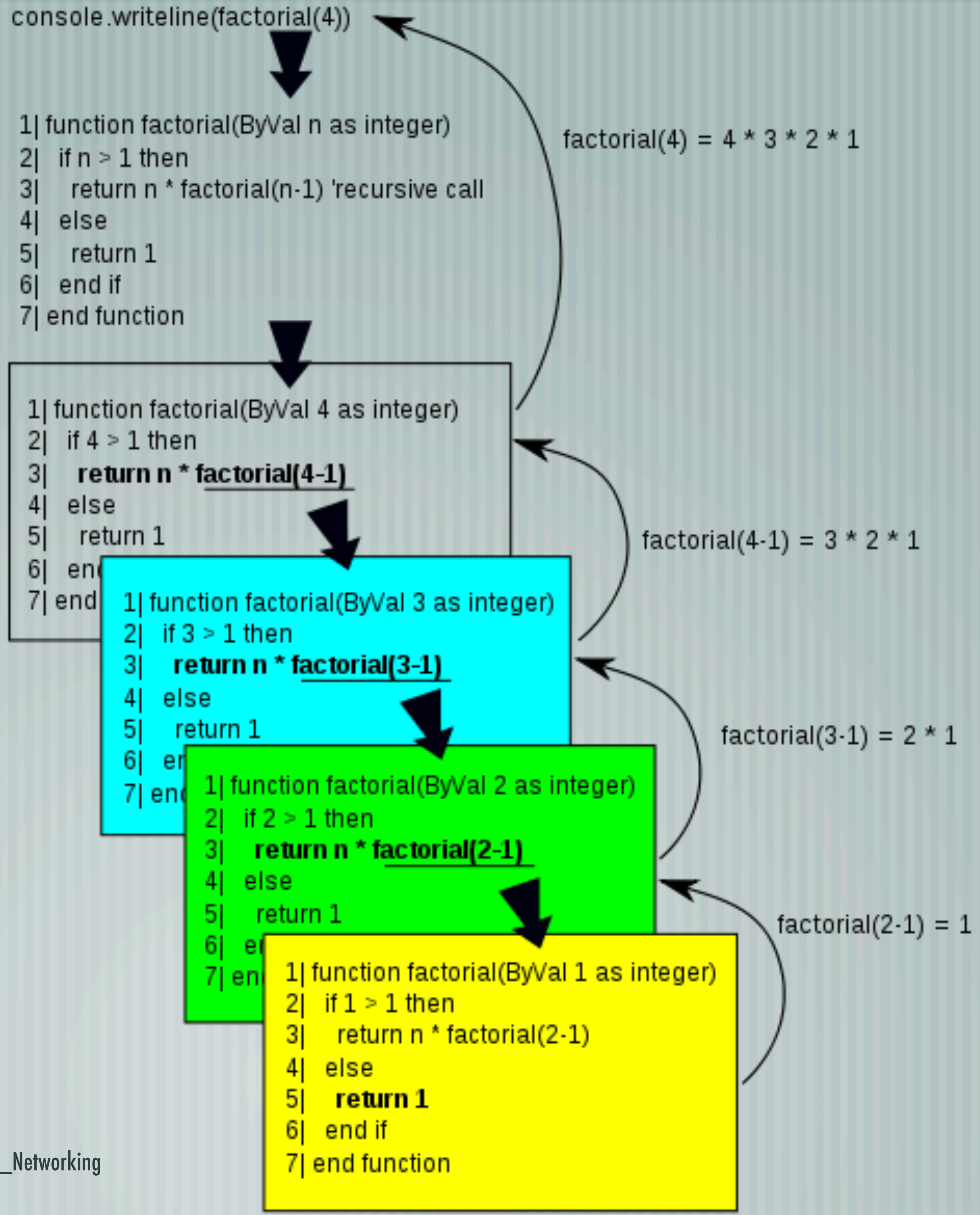
$$n! = n \times (n-1)! \qquad (2)$$

Example of Recursion

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

Example of Recursion

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



```
console.writeline(factorial(4))
```

```
function factorial(ByVal n as integer)
  if n > 1 then
    return n * factorial(n-1) 'recursive call
  else
    return 1
  end if
end function
```

```
function factorial(ByVal 4 as integer)
  if 4 > 1 then
    return n * factorial(4-1)
  else
    return 1
  en
end
```

```
function factorial(ByVal 3 as integer)
  if 3 > 1 then
    return n * factorial(3-1)
  else
    return 1
  er
en
```

```
function factorial(ByVal 2 as integer)
  if 2 > 1 then
    return n * factorial(2-1)
  else
    return 1
  er
en
```

```
function factorial(ByVal 1 as integer)
  if 1 > 1 then
    return n * factorial(2-1)
  else
    return 1
  end if
end function
```

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)
  - How to divide(represent) the original problem into(with) subproblem(s). (recursive definition)

Formal Definition

- Two properties of Recursion in factorial
  - Fac(0) is 1.   [base case]
  - For all integers n > 1: Fac(n) is n * Fac(n-1).   [Recursive Definition]

Implementation – call itself

```
void fac(int n) {
        if( n == 0 ) return 1;
        else
                return n * fac(n - 1);
}
```

Two Secret Weapons for Recursion:

- You make the recursive call as if you are calling a regular method(or an existing API).
- You HAVE TO trust that the recursive call will surely do the work for you!

Take Home Summary

- Definition of Recursion
- Why we have to learn Recursion?
- Secret Weapons for Recursion