Data Structures & Algorithms

Recursion

A Short Review on Recursion

- **Recursion**: when a method calls itself

- Also occurs frequently in mathematical definitions:

$$f(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot f(n-1) & else \end{cases}$$

Iterative calculation

Computing a factorial iteratively:

```
factorial(n)
    f ← 1
    for i  = 2 ... n do
        f = f * i
```

Recursion

- Computing a factorial recursively:

```
factorial(n)
        return n * factorial(n-1)
```

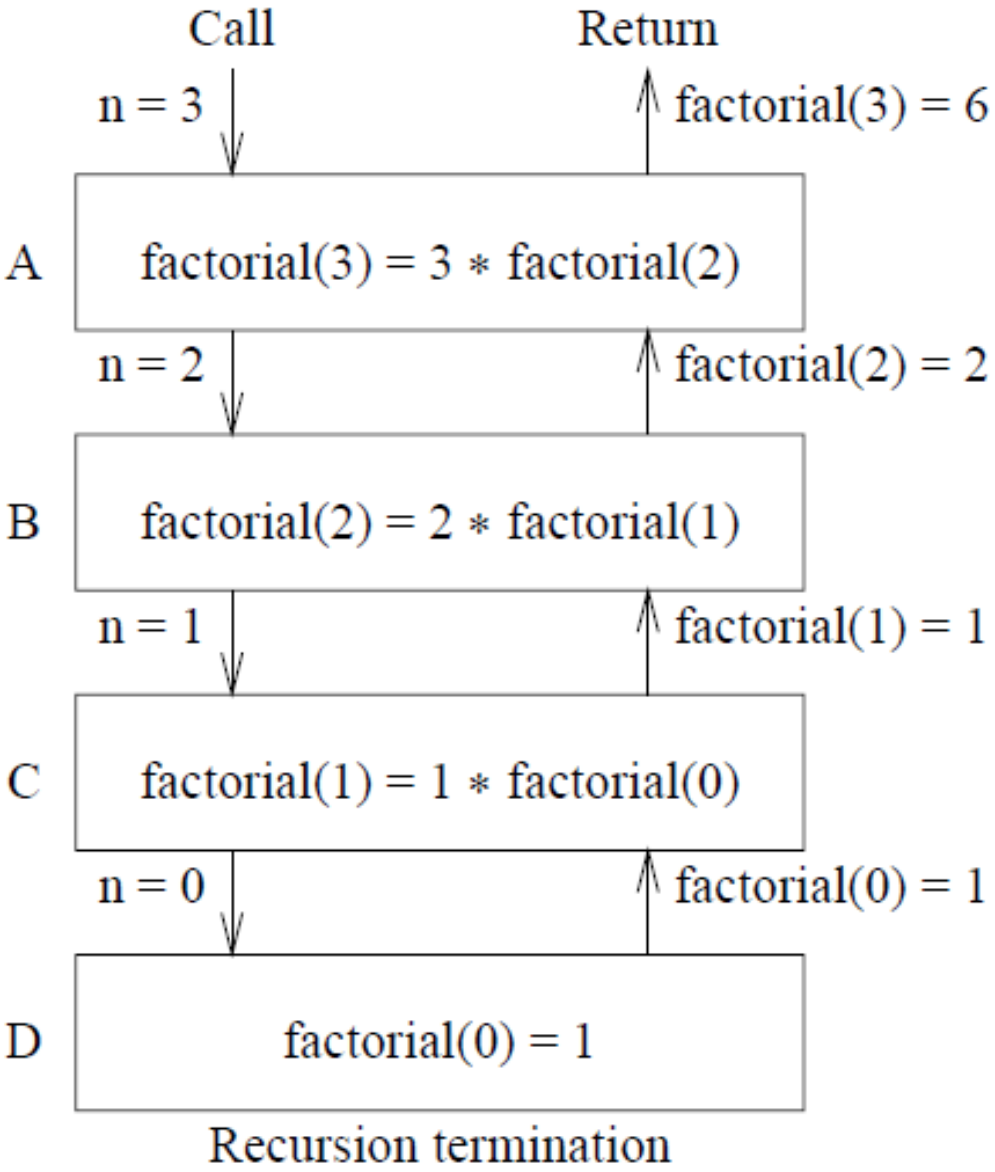
- Problem?

Recursion

- Always define a **base case**

```
factorial(n)
    if n = 0
        return 1
    else
        return n * factorial(n-1)
```

Recursion


Call
Return
n = 3
factorial(3) = 6
A
factorial(3) = 3 * factorial(2)
n = 2
factorial(2) = 2
B
factorial(2) = 2 * factorial(1)
n = 1
factorial(1) = 1
C
factorial(1) = 1 * factorial(0)
n = 0
factorial(0) = 1
D
factorial(0) = 1
Recursion termination

(a)

Recursion

- Figure out base case:
  - ”what is the simplest argument I could possibly get?”

- Make a recursive call with a simpler argument:
  - Simplify your problem
  - assume this new problem will simply work.
  - This is called the “leap of faith”

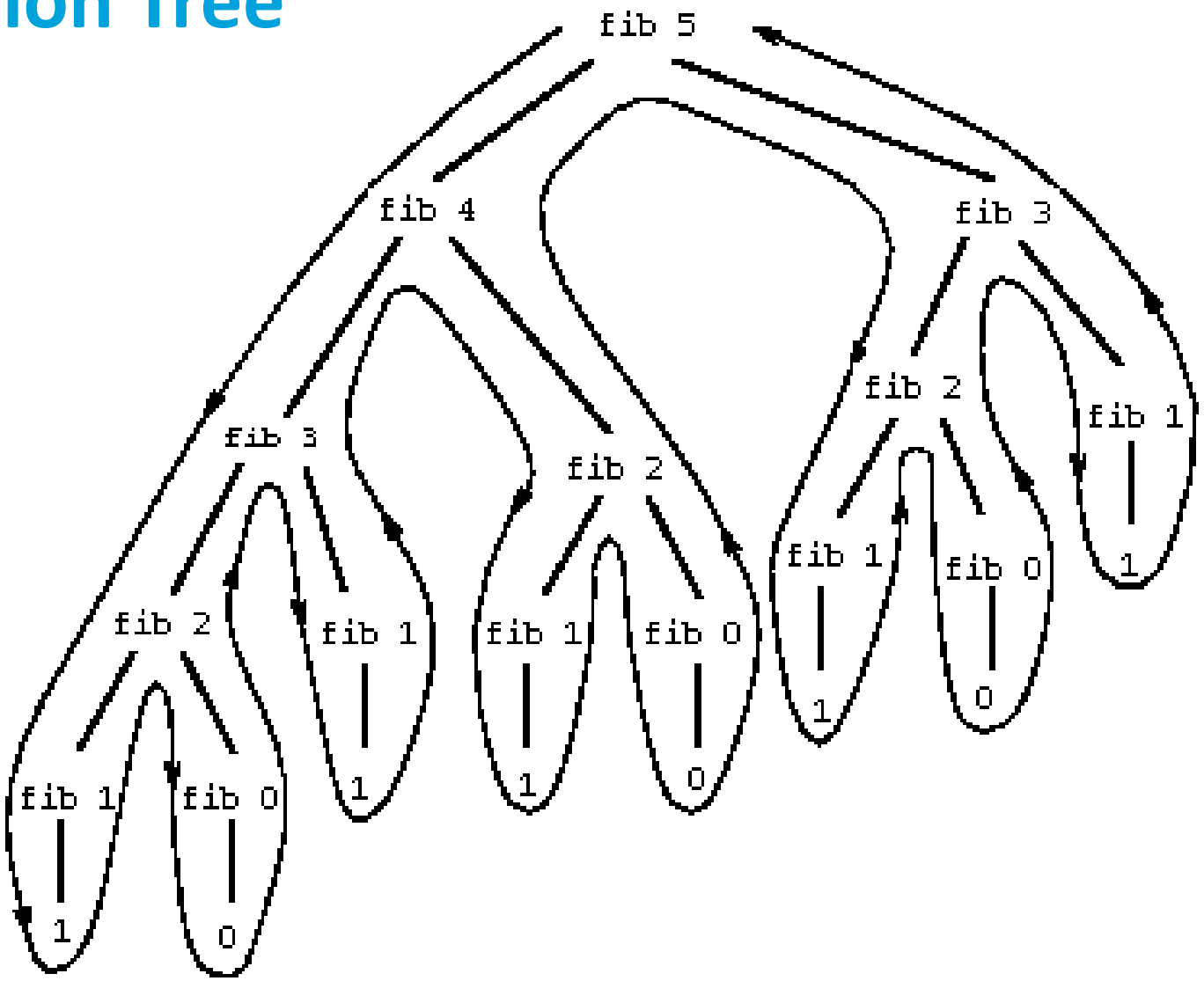
Recursion

Another example:

$$F_n = \begin{cases} n & for\ n \in \{0,1\} \\ F_{n-1} + F_{n-2} & for\ n \geq 2 \end{cases}$$

```
fib(n)
    if n = 0 return 0
    if n = 1 return 1
    return fib(n - 1) + fib(n – 2)
```

Recursion Tree



Implementing recursion in Java

- A recursive function should only use variables passed as parameters

- Avoid using class attributes in the recursive function
  - It may cause unexpected behavior

- When testing your recursive function, try executing it multiple times