

Recursion Trees

Recursion Trees

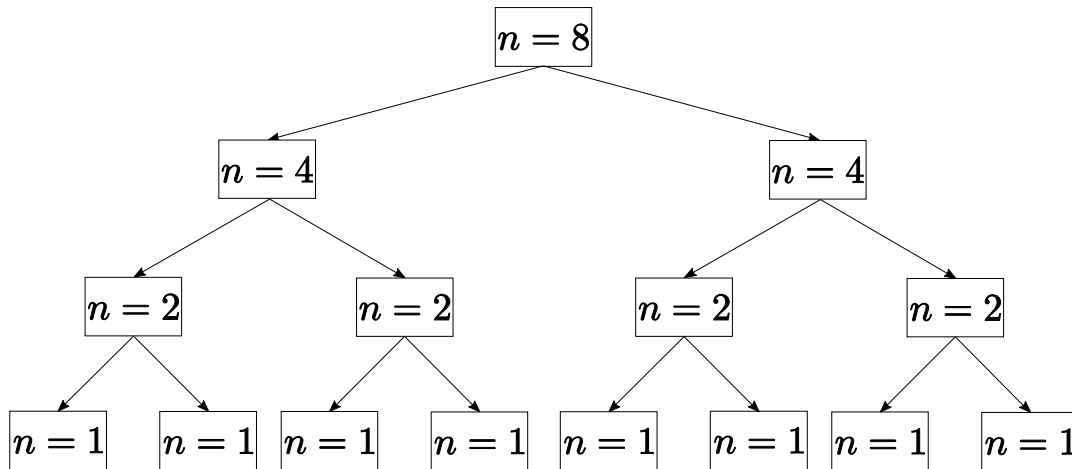
- Visualize call structure

Recursion Trees

- Visualize call structure

- Example: `fnc(8)`

```
unsigned int fnc (unsigned int n) {  
    ...  
    return fnc(n/2) + fnc(n/2);  
}
```



Fibonacci Tree Problem

Fibonacci - Recursion Tree

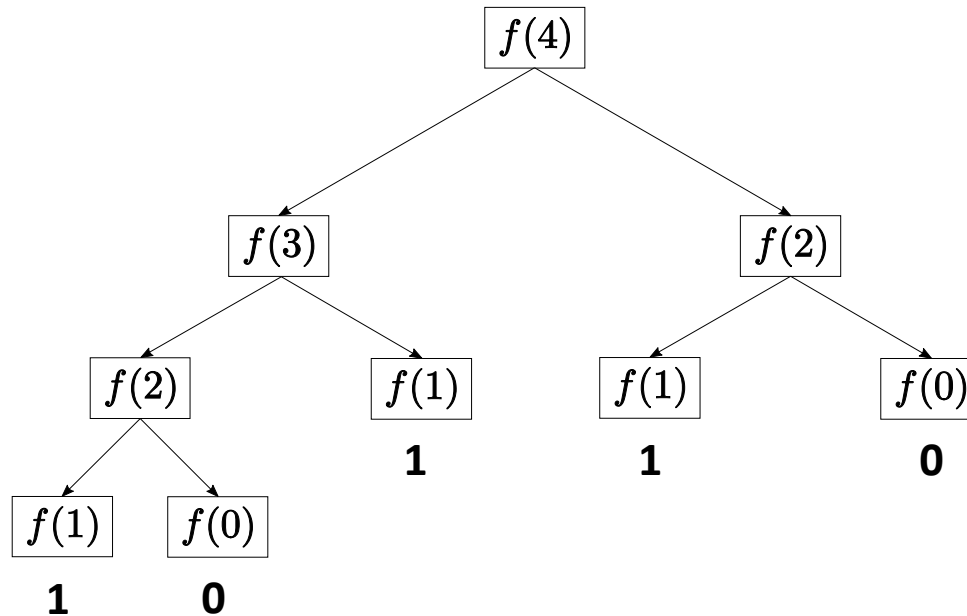
```
// POST: return value is the n-th
//      Fibonacci number F(n)
ifmp::integer fib (const unsigned int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n-1) + fib(n-2); // n > 1
}
```

fib(4) :

Fibonacci - Recursion Tree

```
// POST: return value is the n-th
//       Fibonacci number F(n)
ifmp::integer fib (const unsigned int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n-1) + fib(n-2); // n > 1
}
```

fib(4) :

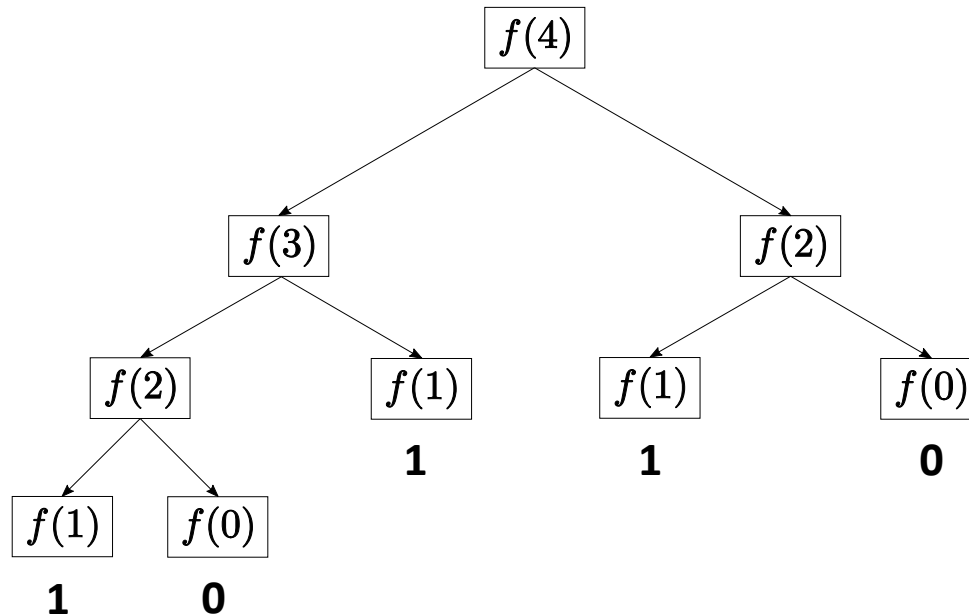


Fibonacci - Recursion Tree

Fibonacci-number VS function calls

n=4 :

fib(4) :



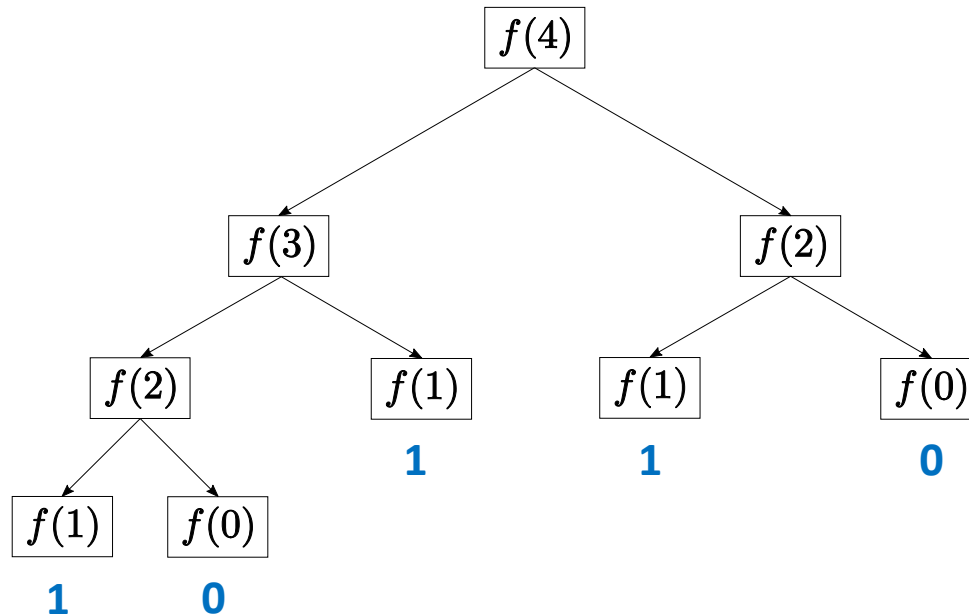
Fibonacci - Recursion Tree

Fibonacci-number VS **function calls**

n=4 :

3

fib(4) :



Fibonacci - Recursion Tree

Fibonacci-number

VS

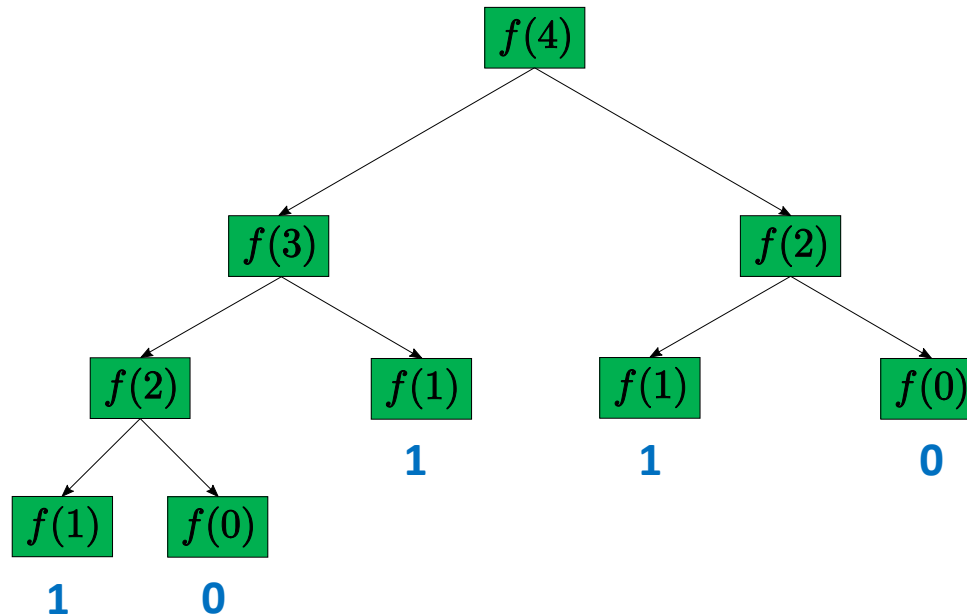
function calls

n=4 :

3

9

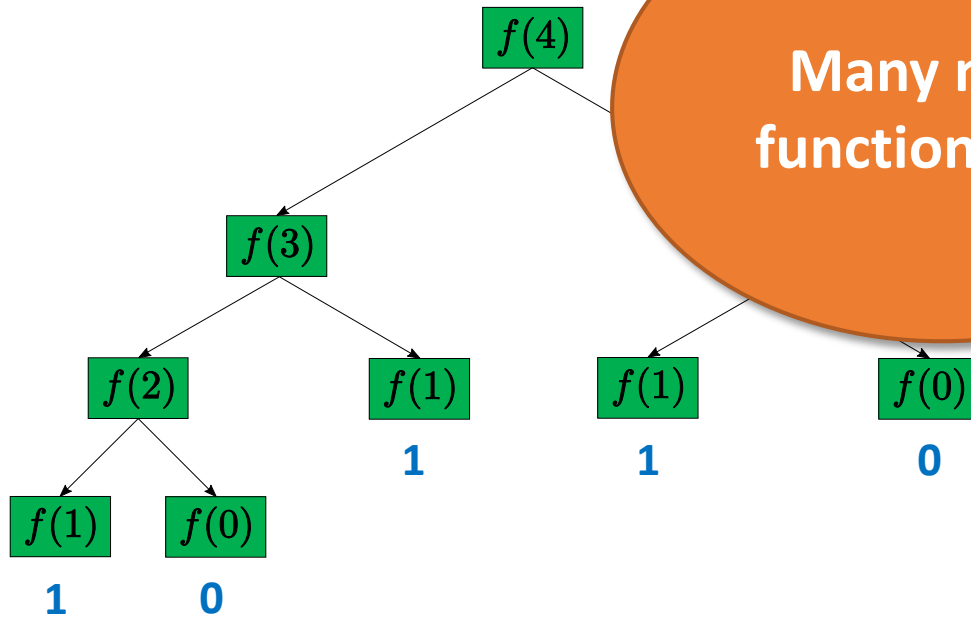
fib(4) :



Fibonacci - Recursion Tree

	Fibonacci-number	VS	function calls
n=4 :	3		9

`fib(4)` :



Many more function calls!

Fibonacci - Recursion Tree

Fibonacci number VS function calls

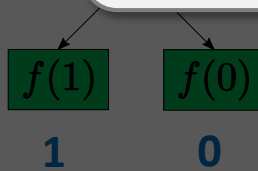
n=4 :

Fibonacci growth:

$\text{fib}(n) \sim c^n$
(for n large, c golden ratio)

n	fib(n)
5	5
10	55
20	6'765
40	102'334'155
80	23'416'728'348'467'685

fib(4) :



Fibonacci - Recursion Tree

Fibonacci number VS function calls

n=4 :

Fibonacci growth:

$$\text{fib}(n) \sim c^n$$

(for n large, c golden ratio)

n	fib(n)
5	5
10	55
20	6'765
40	102'334'155
80	23'416'728'348'467'685

fib(4) :

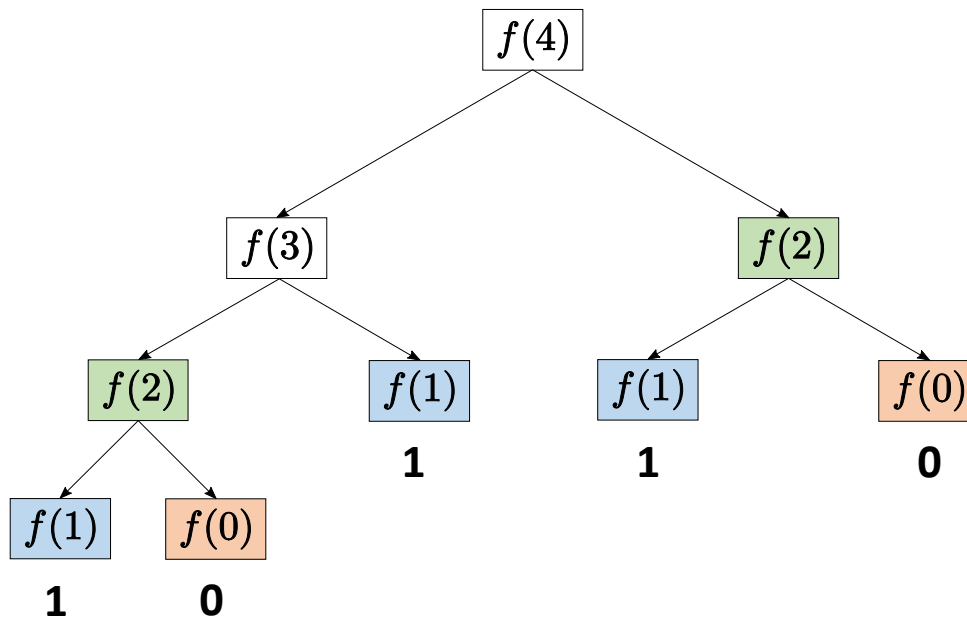
Requires
unbelievably
many recursive
calls!

The Problem

- Problem: Same computation multiple times

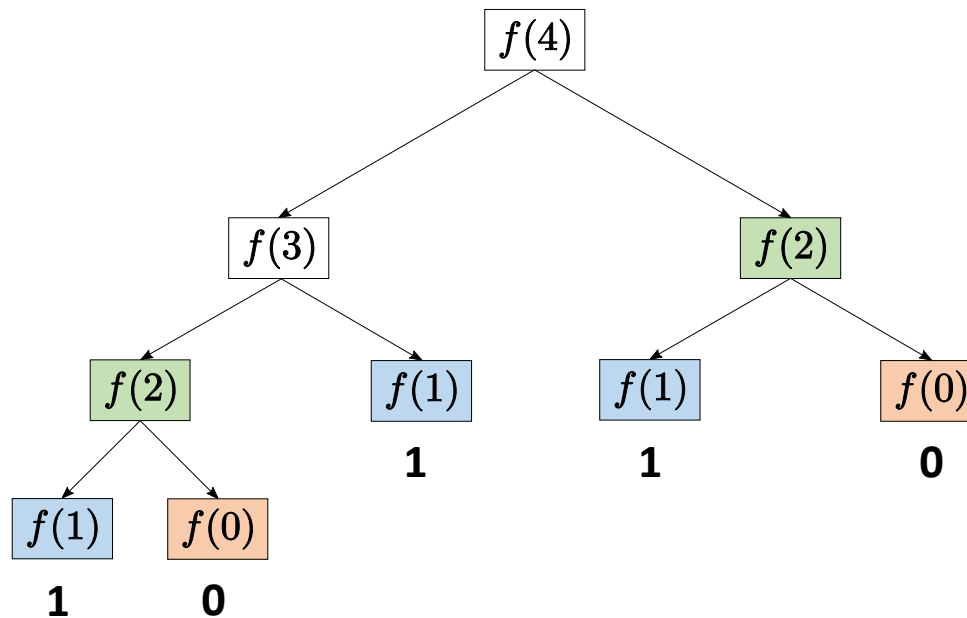
The Problem

- Problem: Same computation multiple times



The Problem

- Problem: Same computation multiple times
- Gets worse as n increases :-)



The Problem

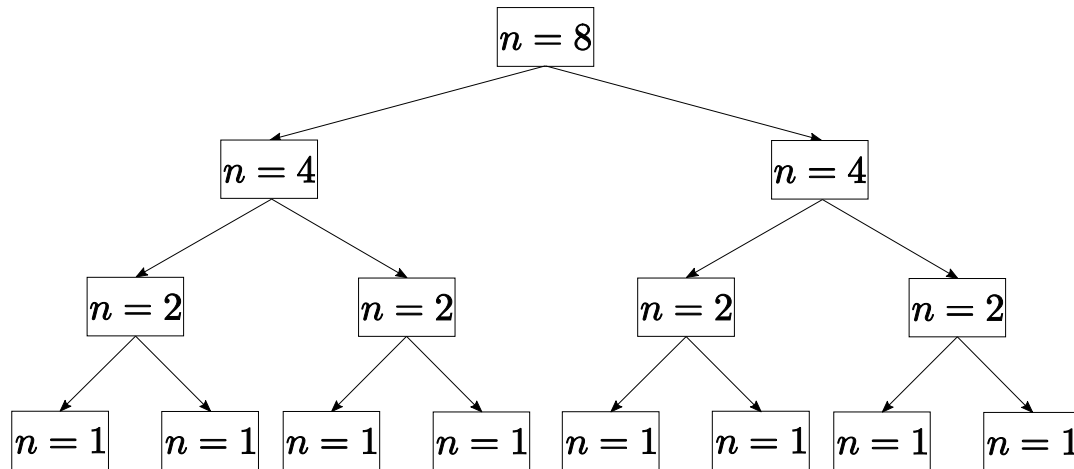
- Not all recursive functions are this inefficient.

The Problem

- Not all recursive functions are this inefficient.

- Example:

```
unsigned int fnc (unsigned int n) {  
    ...  
    return fnc(n/2) + fnc(n/2);  
}
```



The Problem

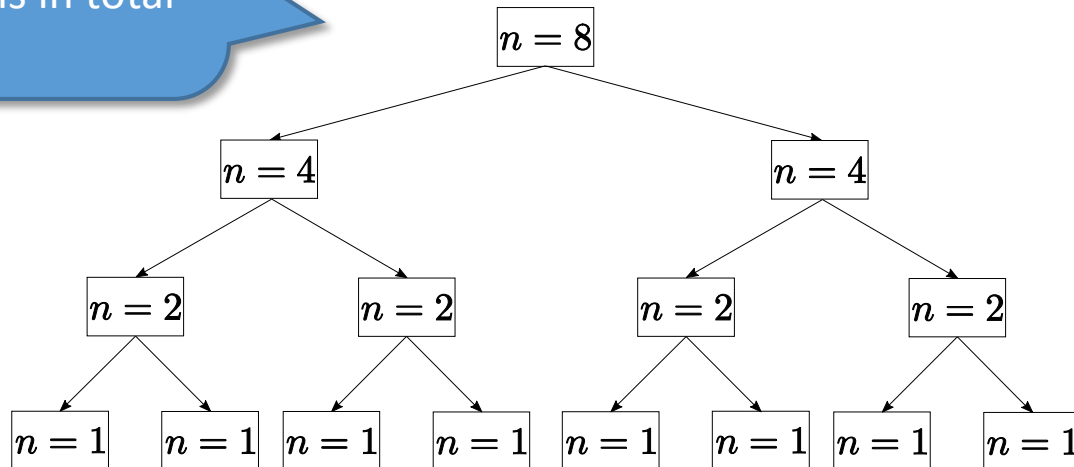
- Not all recursive functions are this inefficient.

• Example:

$$2n - 1$$

recursive calls in total

```
unsigned int fnc (unsigned int n) {  
    ...  
    return fnc(n/2) + fnc(n/2);  
}
```

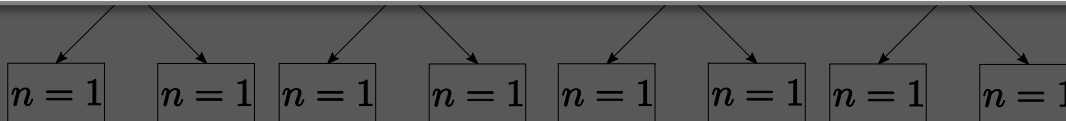


The Problem

- Not all recursive functions are this inefficient.

Number of Recursive Calls

n	fib	fnc
5	> 5	9
10	> 55	19
20	> 6'765	39
40	> 102'334'155	79
80	> 23'416'728'348'467'685	159



The Problem

- Not all recursive functions are this inefficient.

Number of Recursive Calls

n	fib	fnc
5	> 5	9
10	> 55	19
20	> 6'765	39
40	> 102'334'155	79
80	> 23'416'728'348'467'685	159

Mindblowing
difference!

$n = 1$ $n = 1$ $n = 1$ $n = 1$ $n = 1$ $n = 1$

The Problem

- Reason: $n/2$ falls much faster than $n - 1$ and $n - 2$
 - $n/2 \rightarrow$ sub-tree of height: $\log_2(n)$
 - $n - 1, n - 2 \rightarrow$ sub-tree of height: $n - 1, n - 2$

