Introduction: Fibonacci Numbers II

Daniel Kane

Department of Computer Science and Engineering
University of California, San Diego

Algorithmic Design and Techniques
Algorithms and Data Structures at edX

Learning Objectives

- Produce a simple algorithm to compute Fibonacci numbers.
- Show that this algorithm is very slow.

Definition

$$F_n = \begin{cases} 0, & n = 0\,, \\ 1, & n = 1\,, \\ F_{n-1} + F_{n-2}, & n > 1\,. \end{cases}$$

Definition

$$F_n = \begin{cases} 0, & n = 0\,, \\ 1, & n = 1\,, \\ F_{n-1} + F_{n-2}, & n > 1\,. \end{cases}$$

Grows rapidly.

Computing Fibonacci numbers

Compute F_n

Input: An integer $n \geq 0$.

Output: F_n.

Algorithm

FibRecurs(n)

```
if n ≤ 1:
  return n
```

Algorithm

FibRecurs(n)

```
if n ≤ 1:
  return n
else:
  return FibRecurs(n − 1) + FibRecurs(n − 2)
```

Running time

Let $T(n)$ denote the number of lines of code executed by `FibRecurs`(n).

If $n \leq 1$

FibRecurs(n)

```
if n ≤ 1:
  return n
else:
  return FibRecurs(n − 1) + FibRecurs(n − 2)
```

If $n \leq 1$

FibRecurs(n)

```
if n ≤ 1:
  return n
else:
  return FibRecurs(n − 1) + FibRecurs(n − 2)
```

$T(n) = 2.$

If $n \geq 2$

FibRecurs(n)

```
if n ≤ 1:
  return n
else:
  return FibRecurs(n − 1) + FibRecurs(n − 2)
```

If $n \geq 2$

```
FibRecurs(n)

if n ≤ 1:
  return n
else:
  return FibRecurs(n − 1) + FibRecurs(n − 2)
```

$T(n) = 3$

If $n \geq 2$

FibRecurs(n)

```
if n ≤ 1:
  return n
else:
  return FibRecurs(n − 1) + FibRecurs(n − 2)
```

$$T(n) = 3 + T(n-1) + T(n-2).$$

Running Time

$$T(n) = \begin{cases} 2 & \text{if } n \leq 1 \\ T(n-1) + T(n-2) + 3 & \text{else.} \end{cases}$$

Running Time

$$T(n) = \begin{cases} 2 & \text{if } n \leq 1 \\ T(n-1) + T(n-2) + 3 & \text{else.} \end{cases}$$

Therefore $T(n) \geq F_n$

Running Time

$$T(n) = \begin{cases} 2 & \text{if } n \leq 1 \\ T(n-1) + T(n-2) + 3 & \text{else.} \end{cases}$$

Therefore $T(n) \geq F_n$

$T(100) \approx 1.77 \cdot 10^{21}$ (1.77 sextillion)

Running Time

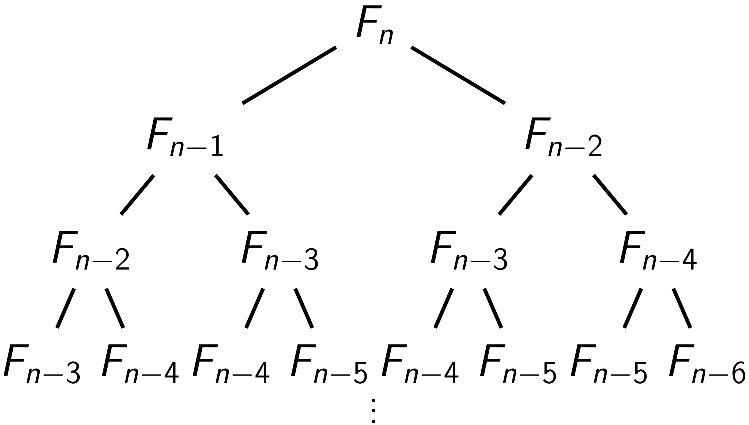
$$T(n) = \begin{cases} 2 & \text{if } n \leq 1 \\ T(n-1) + T(n-2) + 3 & \text{else.} \end{cases}$$

Therefore $T(n) \geq F_n$

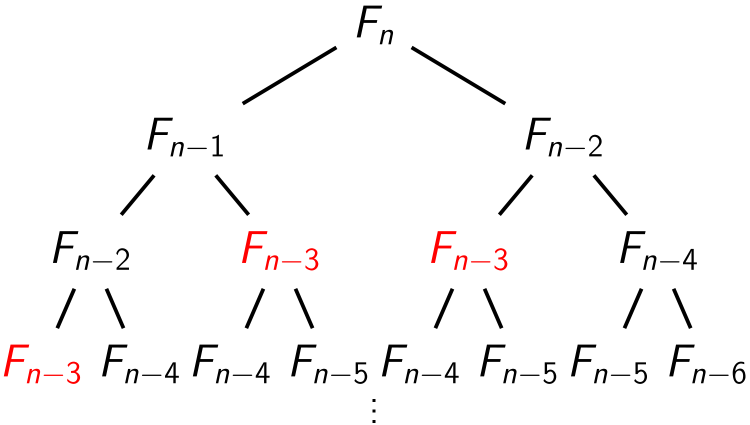
$T(100) \approx 1.77 \cdot 10^{21}$ (1.77 sextillion)

Takes 56,000 years at 1GHz.

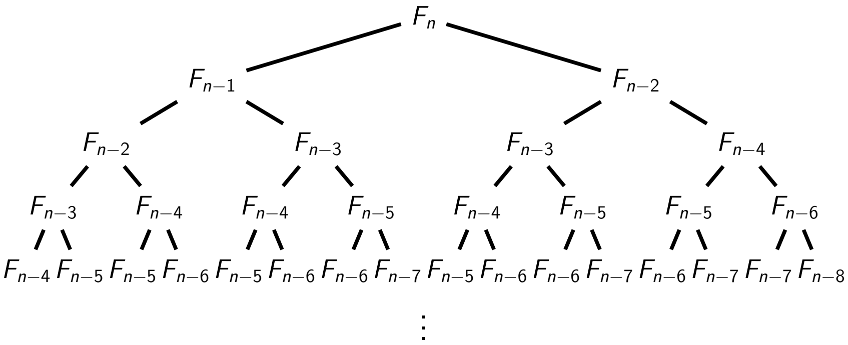
Why so slow?



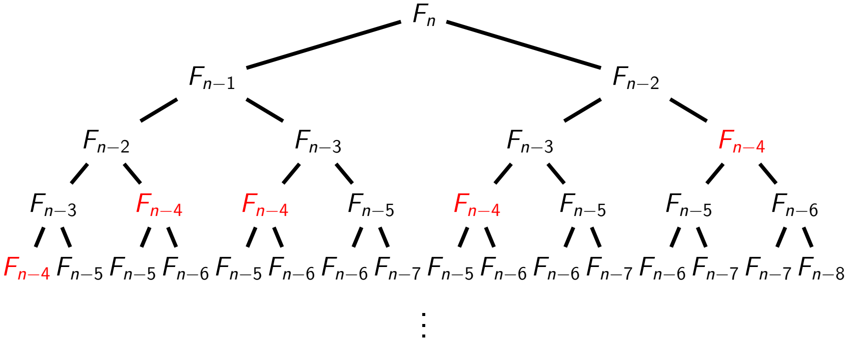
Why so slow?



Why so slow?


F_n
F_{n-1} F_{n-2}
F_{n-2} F_{n-3} F_{n-3} F_{n-4}
F_{n-3} F_{n-4} F_{n-4} F_{n-5} F_{n-4} F_{n-5} F_{n-5} F_{n-6}
F_{n-4} F_{n-5} F_{n-5} F_{n-6} F_{n-5} F_{n-6} F_{n-6} F_{n-7} F_{n-5} F_{n-6} F_{n-6} F_{n-7} F_{n-6} F_{n-7} F_{n-7} F_{n-8}
⋮

Why so slow?


F_n
F_{n-1} F_{n-2}
F_{n-2} F_{n-3} F_{n-3} F_{n-4}
F_{n-3} F_{n-4} F_{n-4} F_{n-5} F_{n-4} F_{n-5} F_{n-5} F_{n-6}
F_{n-4} F_{n-5} F_{n-5} F_{n-6} F_{n-5} F_{n-6} F_{n-6} F_{n-7} F_{n-5} F_{n-6} F_{n-6} F_{n-7} F_{n-6} F_{n-7} F_{n-7} F_{n-8}
⋮