Intro: Fibonacci Numbers

Daniel Kane

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

Data Structures and Algorithms
Algorithmic Toolbox

Learning Objectives

- Understand the definition of the Fibonacci numbers.
- Show that the naive algorithm for computing them is slow.
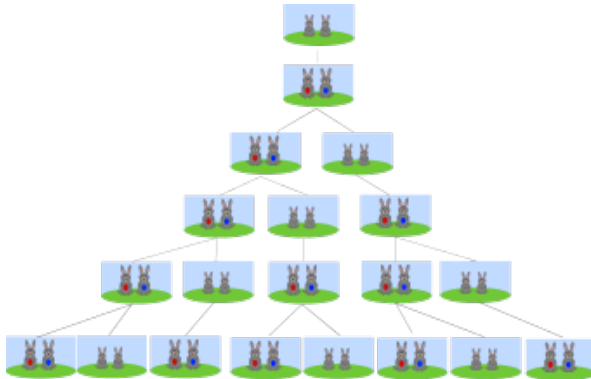- Efficiently compute large Fibonacci numbers.

Outline

Definition

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

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \ldots$$

Developed to Study Rabbit Populations

Rapid Growth

Lemma

$F_n \geq 2^{n/2}$ for $n \geq 6$.

Proof

By induction

Base case: $n = 6, 7$ (by direct computation).

Inductive step:

$$F_n = F_{n-1} + F_{n-2} \geq 2^{(n-1)/2} + 2^{(n-2)/2} \geq$$
$$2 \cdot 2^{(n-2)/2} = 2^{n/2}. \quad \square$$

Formula

Theorem

$$F_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1 + \sqrt{5}}{2} \right)^n - \left(\frac{1 - \sqrt{5}}{2} \right)^n \right).$$

Example

$$F_{20} = 6765$$
$$F_{50} = 12586269025$$
$$F_{100} = 354224848179261915075$$
$$F_{500} = 139423224561697880139724382870407283950070256587697307264108962948325571622863290691557658876222521294125$$

Computing Fibonacci numbers

Compute F_n

Input: An integer $n \geq 0$.

Output: F_n.

Outline

Algorithm

FibRecurs(n)

if $n \leq 1$:
  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 $n \leq 1$

$T(n) = 2.$

If $n \geq 2$

FibRecurs(n)

if $n \leq 1$:

  return n

else:

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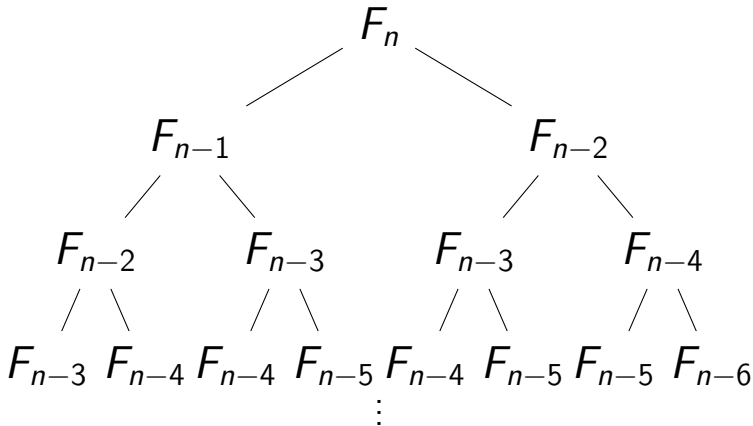
$T(n) = 3 + T(n-1) + T(n-2).$

Running Time

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

Therefore $T(n) \geq F_n$
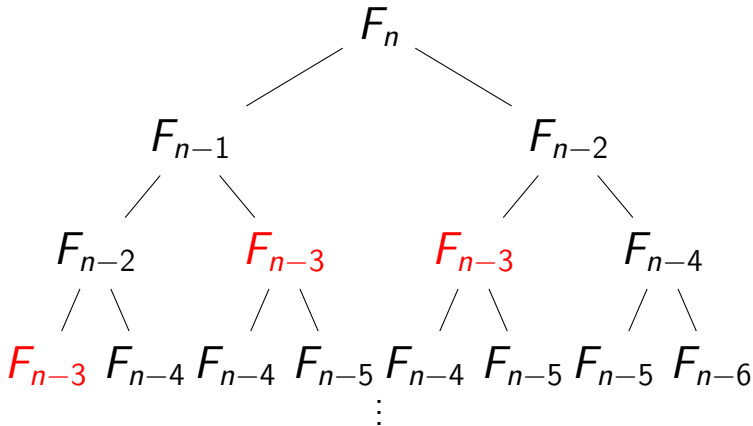
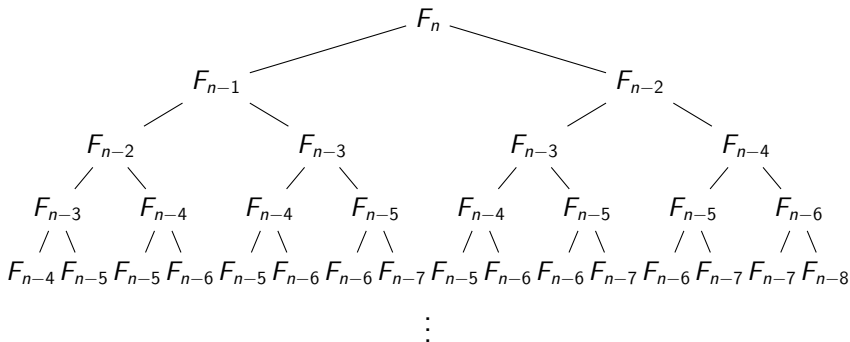$T(100) \approx 1.77 \cdot 10^{21}$ (1.77 sextillion)

Takes 56,000 years at 1GHz.

Why so slow?

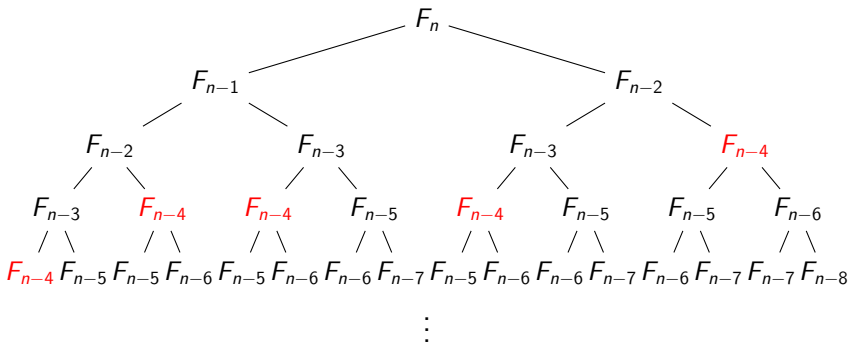# Why so slow?

Why so slow?

Why so slow?

Outline

Another Algorithm

Imitate hand computation:
0, 1, 1, 2, 3, 5

$0 + 1 = 1$
$1 + 1 = 2$
$1 + 2 = 3$
$2 + 3 = 5$

Another Algorithm

Imitate hand computation:
0, 1, 1, 2, 3, 5, 8

$0 + 1 = 1$
$1 + 1 = 2$
$1 + 2 = 3$
$2 + 3 = 5$
$3 + 5 = 8$

New Algorithm

FibList(n)

```
create an array F[0...n]
F[0] ← 0
F[1] ← 1
for i from 2 to n:
    F[i] ← F[i-1] + F[i-2]
return F[n]
```

- $T(n) = 2n + 2$. So $T(100) = 202$.
- Easy to compute.

Summary

- Introduced Fibonacci numbers.
- Naive algorithm takes ridiculously long time on small examples.
- Improved algorithm incredibly fast.

Moral: The right algorithm makes all the difference.