

# Introduction: Fibonacci Numbers III

Daniel Kane

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

Algorithmic Toolbox  
Data Structures and Algorithms

# Learning Objectives

- Compute Fibonacci numbers efficiently.

## Definition

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

# Algorithm

FibRecurs( $n$ )

```
if  $n \leq 1$ :  
    return  $n$   
else:  
    return FibRecurs( $n - 1$ ) + FibRecurs( $n - 2$ )
```

Too slow!

# Another Algorithm

Imitate hand computation:

0, 1

# Another Algorithm

Imitate hand computation:

0, 1, 1

$$0 + 1 = 1$$

# Another Algorithm

Imitate hand computation:

0, 1, 1, 2

$$0 + 1 = 1$$

$$1 + 1 = 2$$

# Another Algorithm

Imitate hand computation:

0, 1, 1, 2, 3

$$0 + 1 = 1$$

$$1 + 1 = 2$$

$$1 + 2 = 3$$



# 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  $F[0 \dots n]$

$F[0] \leftarrow 0$

$F[1] \leftarrow 1$

for  $i$  from 2 to  $n$ :

$F[i] \leftarrow F[i - 1] + F[i - 2]$

return  $F[n]$

# New Algorithm

**FibList( $n$ )**

create an array  $F[0 \dots n]$

$F[0] \leftarrow 0$

$F[1] \leftarrow 1$

for  $i$  from 2 to  $n$ :

$F[i] \leftarrow F[i - 1] + F[i - 2]$

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.

# 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.**