

## 61A Lecture 20

Friday, October 14

## Tree Recursion

Tree-shaped processes arise whenever executing the body of a function entails making **more than one** call to that function.

**n:** 1, 2, 3, 4, 5, 6, 7, 8, 9, ... , 35

**fib(n):** 0, 1, 1, 2, 3, 5, 8, 13, 21, ... , 5,702,887

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

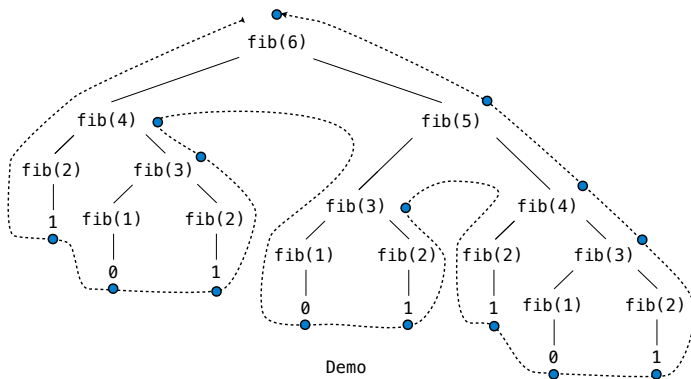


<http://en.wikipedia.org/wiki/File:Fibonacci.jpg>

2

## A Tree-Recursive Process

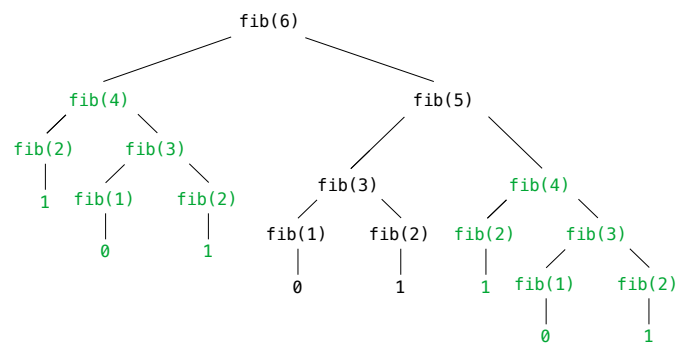
The computational process of fib evolves into a tree structure



3

## Repetition in Tree-Recursive Computation

This process is highly repetitive; fib is called on the same argument multiple times



4

## Memoization

**Idea:** Remember the results that have been computed before

```
def memo(f):  
    cache = {}  
    def memoized(n):  
        if n not in cache:  
            cache[n] = f(n)  
        return cache[n]  
    return memoized
```

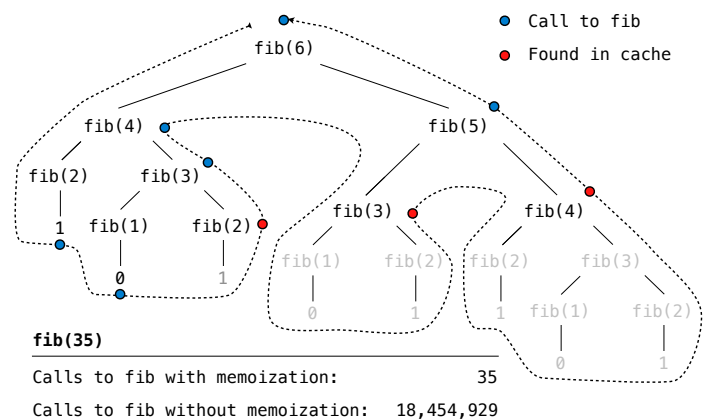
Keys are arguments that map to return values

Same behavior as f, if f is a pure function

Demo

5

## Memoized Tree Recursion



6

## Iteration vs Memoized Tree Recursion

Iterative and memoized implementations are not the same.

```
def fib_iter(n):
    prev, curr = 1, 0
    for _ in range(n-1):
        prev, curr = curr, prev + curr
    return curr
```

The first Fibonacci number

Time	Space
n steps	3 names

Independent of problem size

```
@memo
def fib(n):
    if n == 1:
        return 0
    if n == 2:
        return 1
    return fib(n-2) + fib(n-1)
```

n steps	n entries
---------	-----------

Scales with problem size

## Counting Change

\$1 = \$0.50 + \$0.25 + \$0.10 + \$0.10 + \$0.05

\$1 = 1 half dollar, 1 quarter, 2 dimes, 1 nickel

\$1 = 2 quarters, 2 dimes, 30 pennies

\$1 = 100 pennies

How many ways are there to change a dollar?

How many ways to change \$0.11 with nickels & pennies?

\$0.11 can be changed with nickels & pennies by

A. Not using any more nickels; \$0.11 with just pennies

B. Using at least one nickel; \$0.06 with nickels & pennies

## Counting Change Recursively

How many ways are there to change a dollar?

The number of ways to change an amount **a** using **n** kinds =

- The number of ways to change **a** using all but the first kind
- +
  - The number of ways to change (**a - d**) using all **n** kinds, where **d** is the denomination of the first kind of coin.

```
def count_change(a, kinds=(50, 25, 10, 5, 1)):
    <base cases>
    d = kinds[0]
    return count_change(a, kinds[1:]) + count_change(a-d, kinds)
```

Demo

## Space Consumption

Which environment frames do we need to keep during evaluation?

Each step of evaluation has a set of **active** environments.

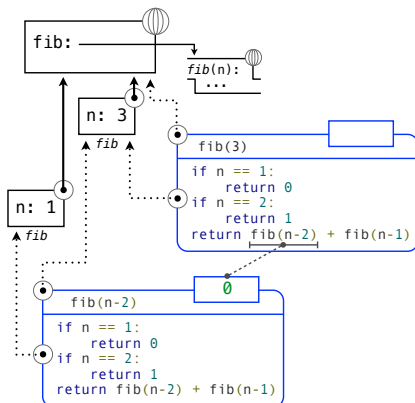
Values and frames referenced by active environments are kept.

Memory used for other values & frames can be reclaimed.

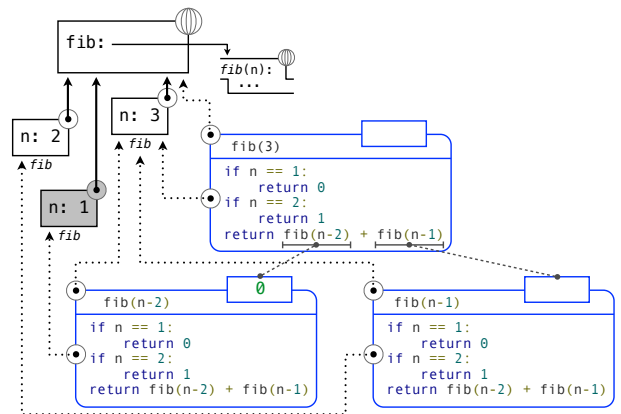
**Active environments:**

- The environment for the current expression being evaluated
- All environments for expressions that depend upon the value of the current expression
- All environments associated with values referenced by active environments

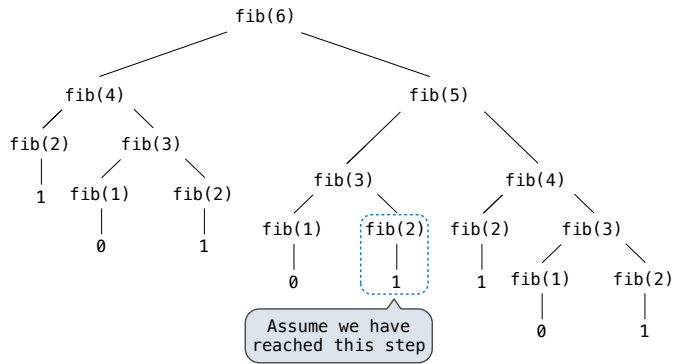
## Fibonacci Environment Diagram



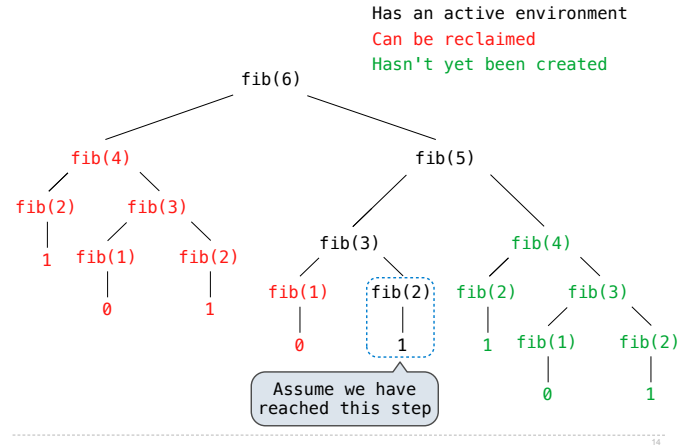
## Fibonacci Environment Diagram



### Fibonacci Memory Consumption



### Fibonacci Memory Consumption



### Active Environments for Returned Functions

