LECTURE 10

RECURSION VS ITERATION

MCS 275 Spring 2023
Emily Dumas

LECTURE 10: RECURSION VS ITERATION

Reminders and announcements:

- Homework 4 due Tuesday at Noon
- Project 1 autograder open! 🥳
- Project 1 due Friday at 6pm
- Project 2 will be posted next Monday, due Feb 28

ITERATIVE SOLUTIONS

Let's write iterative versions of factorial, Fibonacci, and paper folding. (Or as many as time allows.)

TIMING COMPARISON

Let's compare the running time of the iterative and recursive solutions.

QUESTION

Why is recursive `fact()` somewhat competitive, but `fib()` is dreadfully slow?

Let's track the number of function calls.

FACT CALL GRAPH

FACT CALL GRAPH

FACT CALL GRAPH

$\longrightarrow$ Call

$$\begin{tikzcd}
5 \arrow[r] & 4
\end{tikzcd}$$

FACT CALL GRAPH

$\rightarrow$ Call

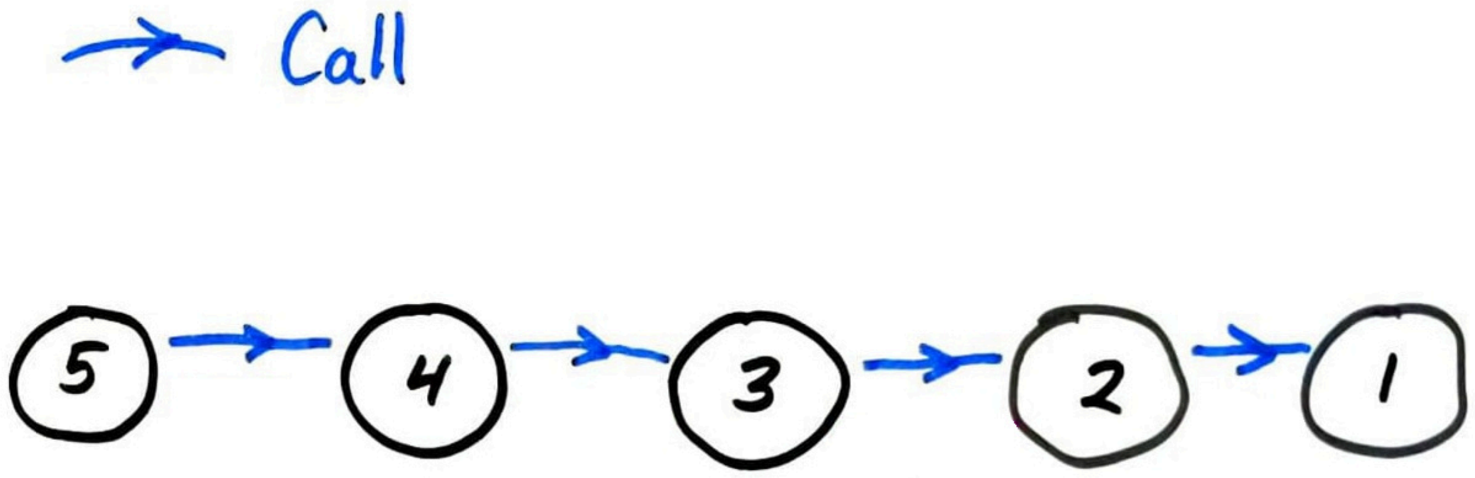
$$\begin{tikzcd}
5 \arrow[r] & 4 \arrow[r] & 3
\end{tikzcd}$$

FACT CALL GRAPH

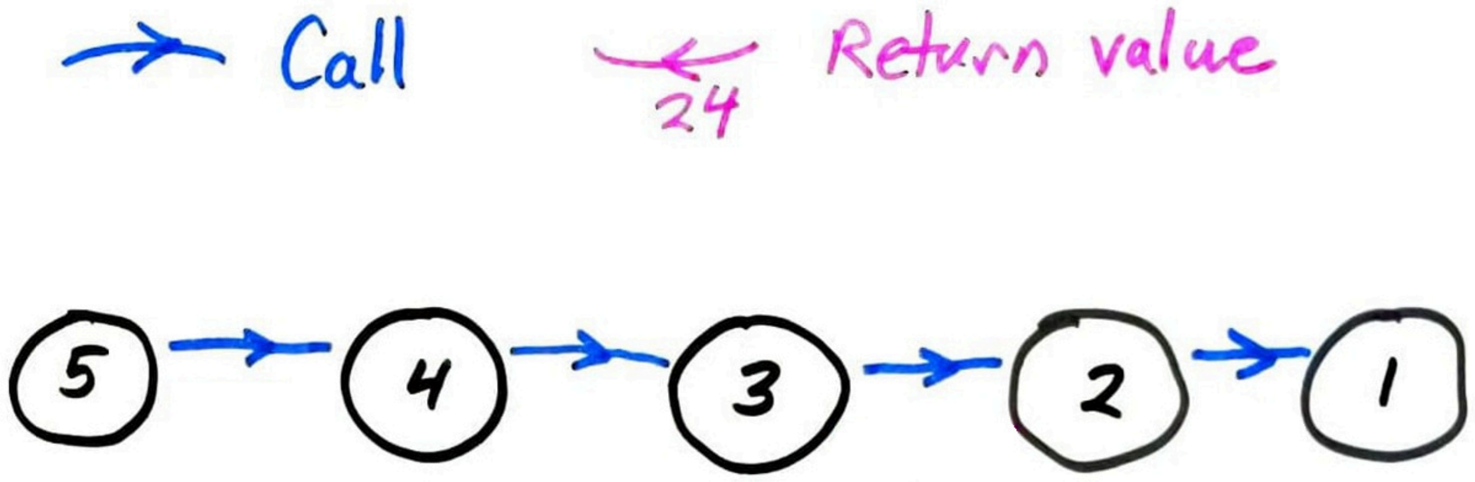
$\longrightarrow$ Call

$$\begin{tikzcd}
5 \arrow[r] & 4 \arrow[r] & 3 \arrow[r] & 2
\end{tikzcd}$$

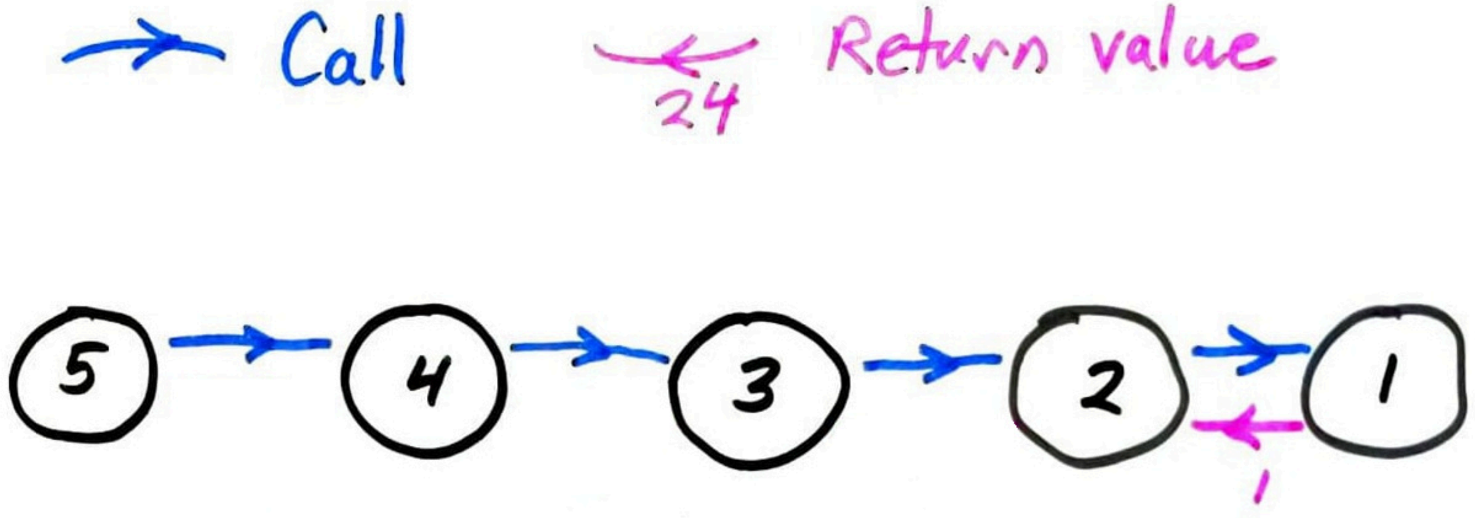
FACT CALL GRAPH



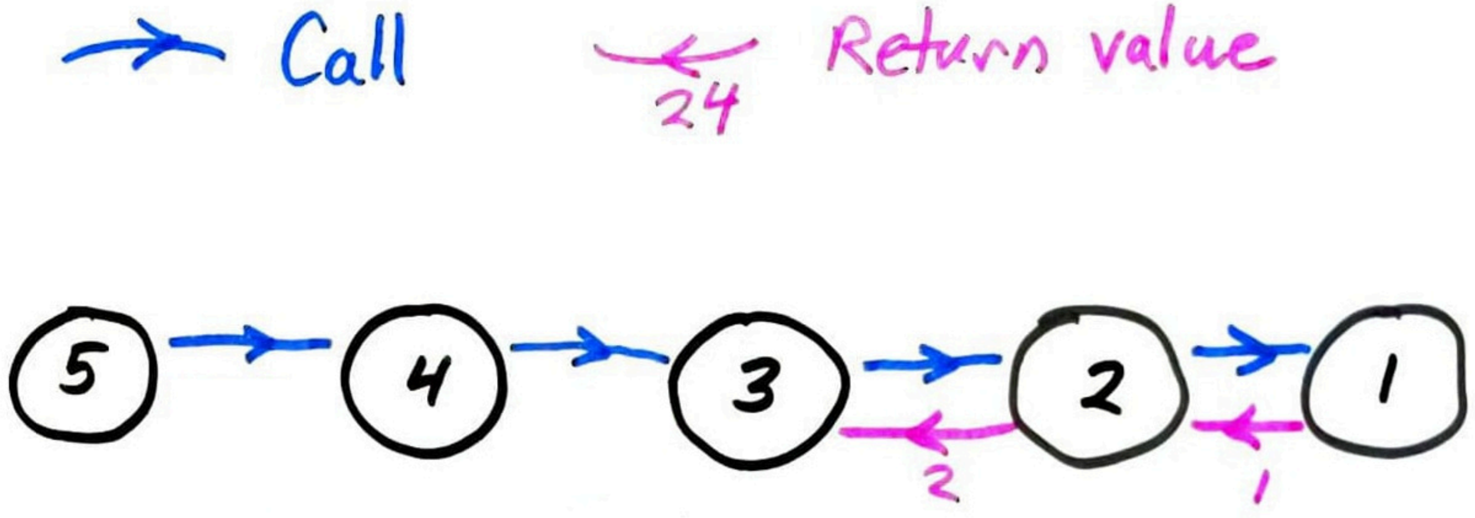
FACT CALL GRAPH



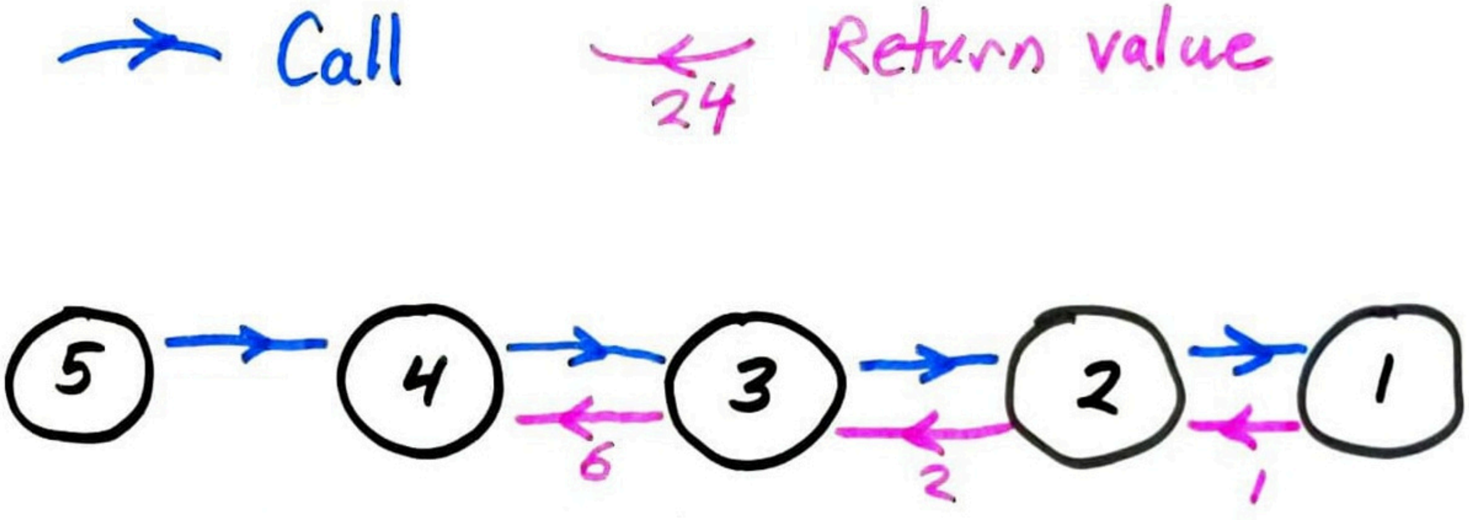
FACT CALL GRAPH



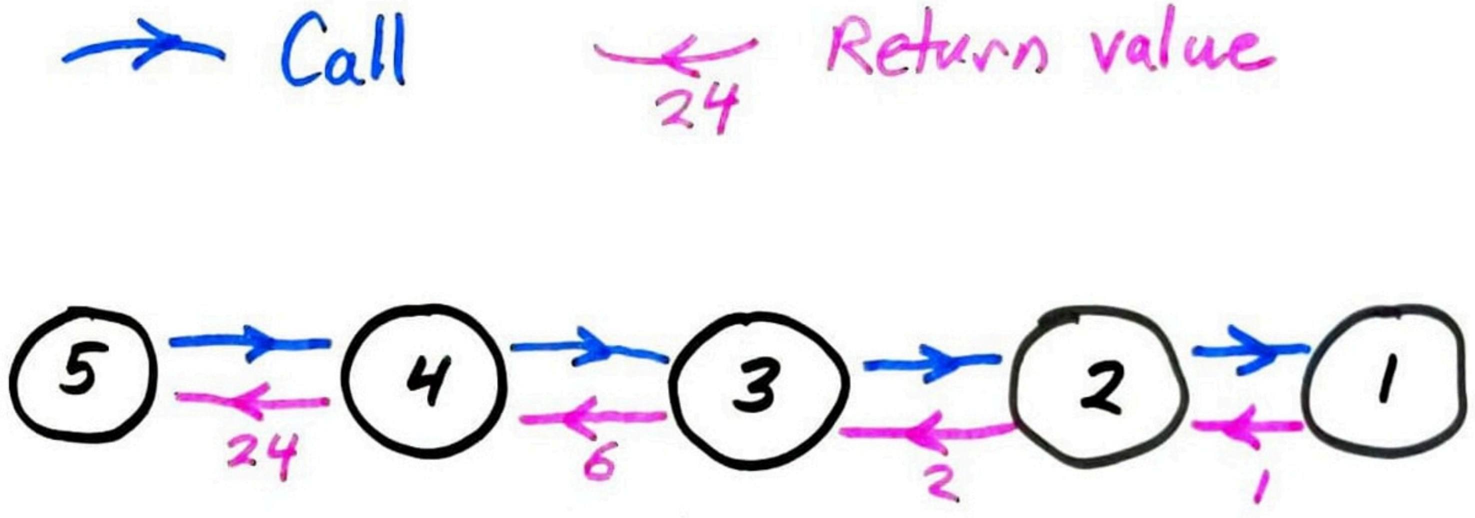
FACT CALL GRAPH



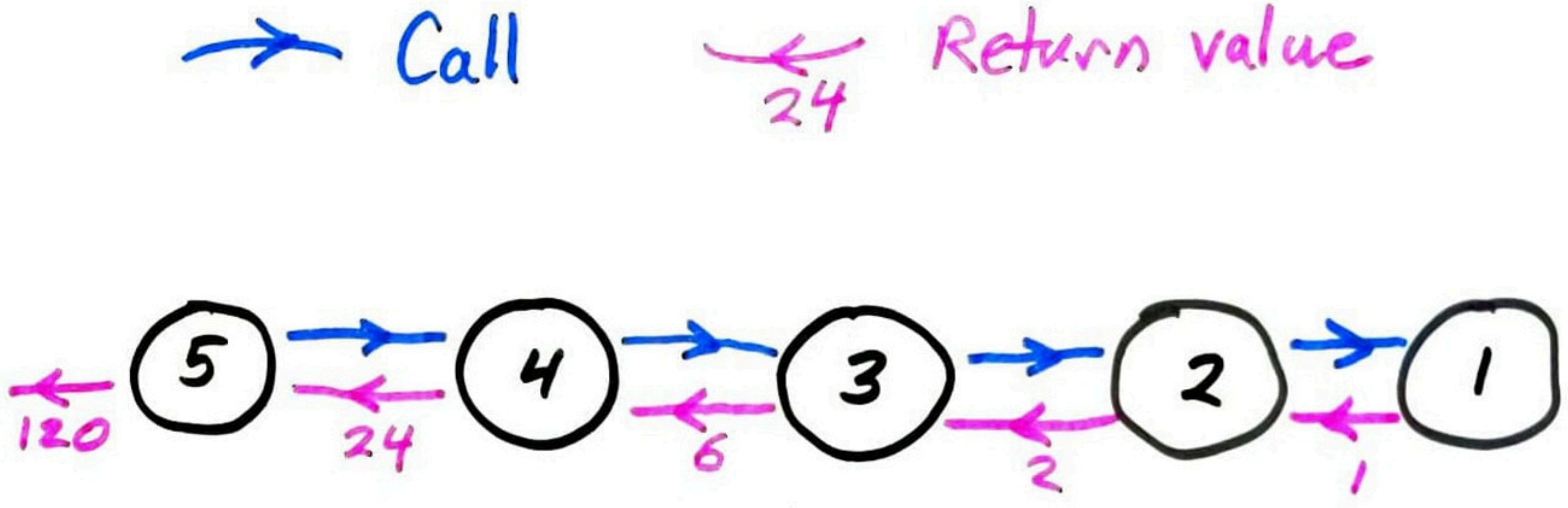
FACT CALL GRAPH



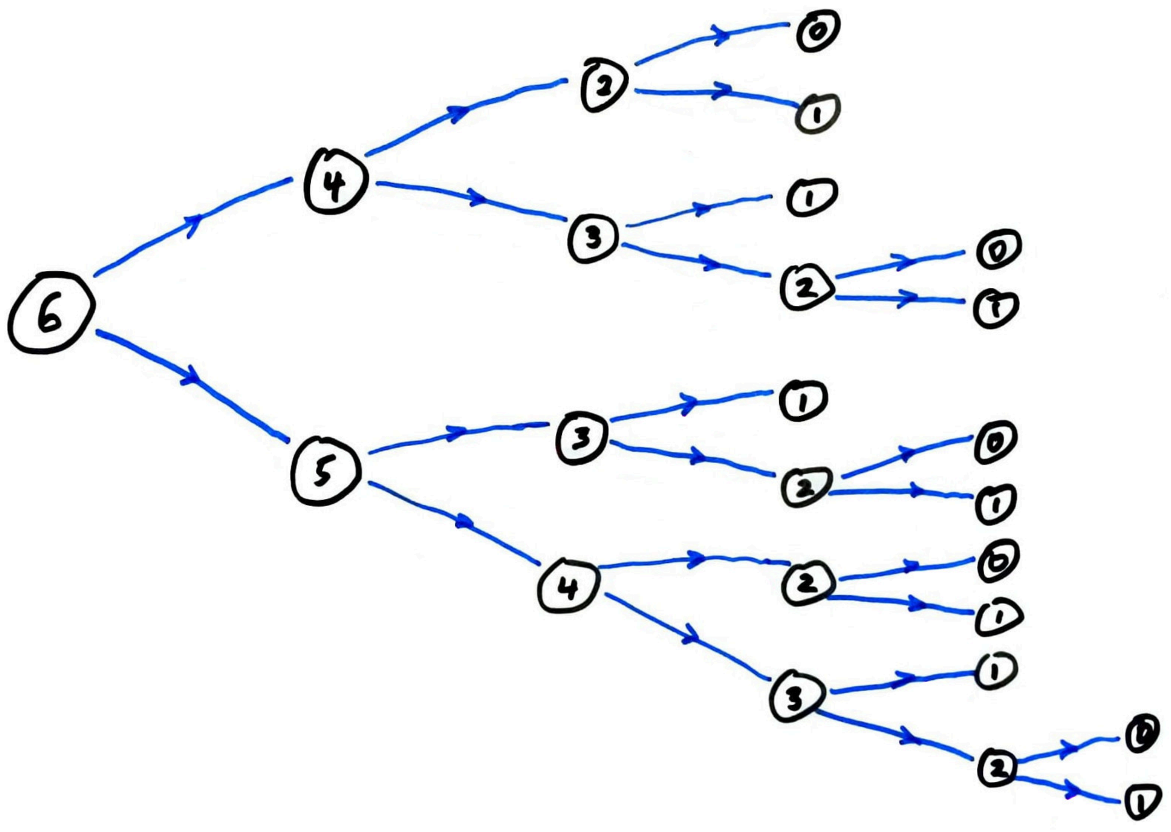
FACT CALL GRAPH



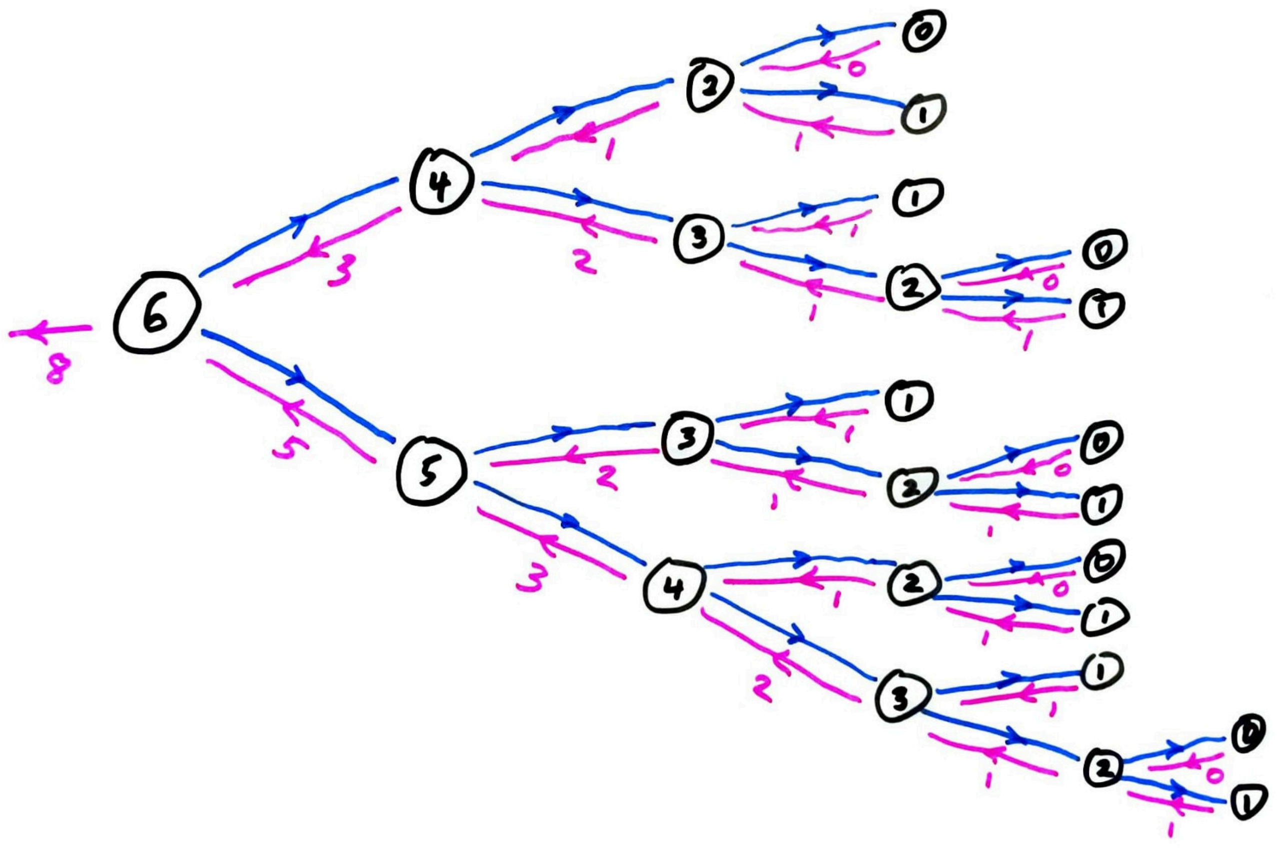
FACT CALL GRAPH



FIB CALL GRAPH



FIB CALL GRAPH



MEMOIZATION

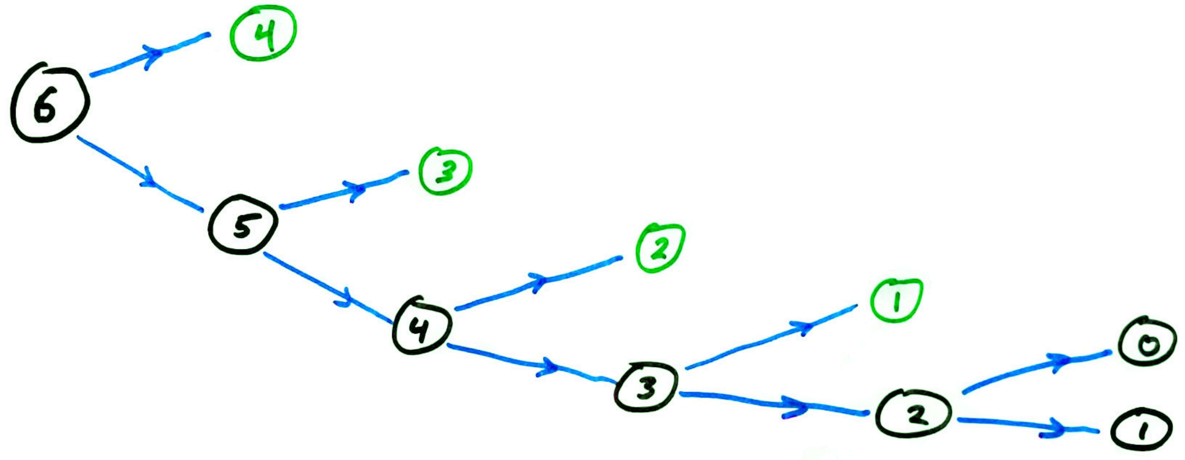
`fib` computes the same terms over and over again.

Instead, let's store all previously computed results, and use the stored ones whenever possible.

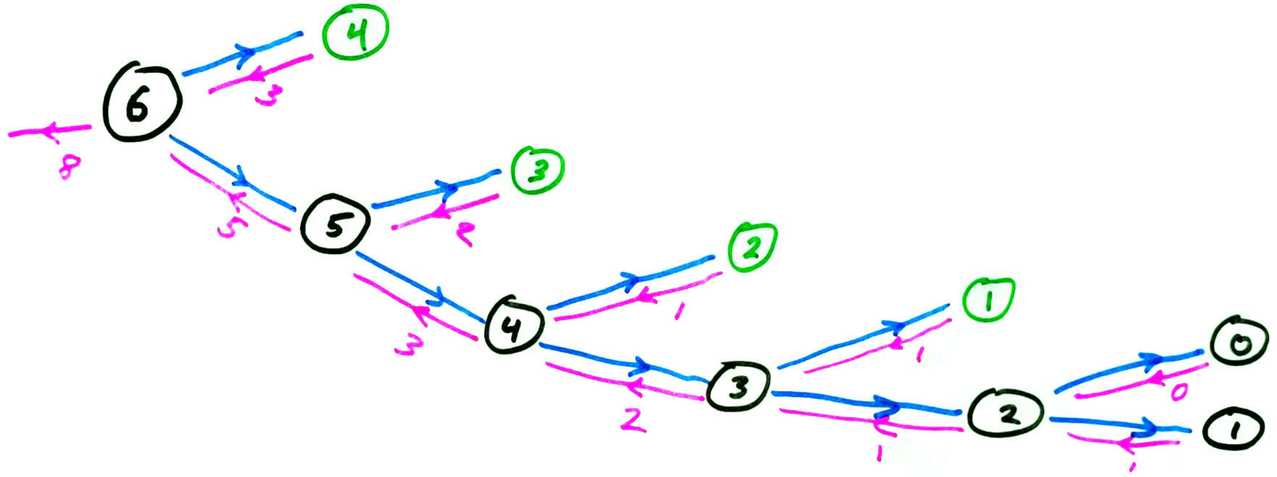
This is called **memoization**. It only works for **pure functions**, i.e. those which always produce the same return value for any given argument values.

`math.sin(...)` is pure; `random.random()` is not.

MEMOIZED FIB CALL GRAPH



MEMOIZED FIB CALL GRAPH



FIBONACCI TIMING SUMMARY

	n=35	n=450
recursive	1.9s	> age of universe
memoized recursive	<0.001s	0.003s
iterative	<0.001s	0.001s

Measured on my old office PC (2015, Intel i7-6700K) with Python 3.8.5

MEMOIZATION SUMMARY

Recursive pure functions with multiple self-calls often benefit from memoization.

Memoization does not alleviate recursion depth limits.

Memoization trades running time for memory consumption.

REFERENCES

- Algorithms by Jeff Erickson, available as a free PDF, discusses some examples of recursion in Chapter 1.
- Lutz discusses recursive functions in Chapter 19 (pages 555-559 in the print edition).
- Intro to Python for Computer Science and Data Science by Deitel and Deitel discusses recursion in Chapter 11. The online version of this text is freely available to UIC students, faculty, and staff. (You will first need to log in with you UIC email.)
- The open textbook Think Python, 2ed, by Allen B. Downey discusses recursion in Sections 5.8 to 5.10.
- Computer Science: An Overview by Brookshear and Brylow discusses recursion in Section 5.5. (This book is often an optional text for MCS 260.)

REVISION HISTORY

- 2022-02-09 Last year's lecture on this topic finalized
- 2023-02-06 Updated version for spring 2023