CSC 148: Introduction to Computer Science

Week 6

More complex recursion

- Nested list mutation
- Recursion efficiency

University of Toronto Mississauga,

Department of Mathematical and Computational Sciences

Nested list mutation

- Worksheet …
  - Add 1 to every element stored in nested list

- Also really good review for a classic memory-related error!

Recursion and redundancy

- Remember recursion:
  - Calculating Fibonacci numbers
    - if n < 2, fib(n) = 1
    - fib(n) = fib(n-1) + fib(n-2)
- Write a recursive program for this..

```
def fib(n: int) -> int:
    """
    Returns the n-th fibonacci number.
    """
    pass
```

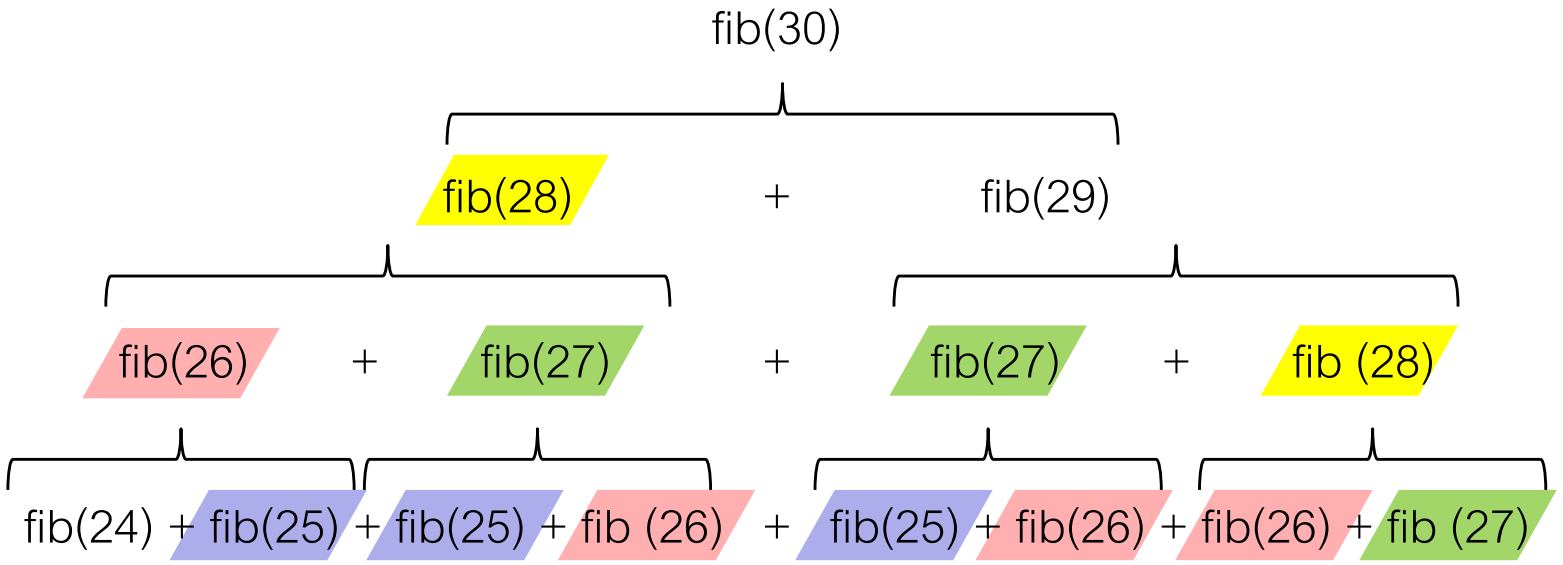
Recursion and redundancy

- Remember recursion:
  - Calculating Fibonacci numbers
    - if n < 2, fib(n) = 1
    - fib(n) = fib(n-1) + fib(n-2)
- Write a recursive program for this..

```
def fib(n: int) -> int:
    """
    Returns the n-th fibonacci number.
    """
    if n < 2:
        return 1
    else:
        return fib(n-1) + fib(n-2)
```

Redundancy

- Unnecessary repeated calculations => inefficient!
- Let's expand the recursion: fib(n) = fib(n-1) + fib(n-2)



How could we avoid calculating items we already calculated?

Solution? Memoize

- Keep track of already calculated values

```
def fib_memo(n: int,  seen: dict[int, int]) -> int:
    """
    Returns the <n>-th fibonacci number, reasonably quickly, without redundancy.
    Parameter <seen> is a dictionary of already-seen results
    """
    if n not in seen:
        seen[n] = (n  if  n < 2
                   else  fib_memo(n-2,  seen)  +  fib_memo(n-1,  seen) )
    return seen[n]
```

Running out of stack space

- Some programming languages have better support for recursion than others; python may run out of space on its stack for recursive function calls ...

- For example, recursively traversing a **very** long list ...

Recursive vs iterative

- Any recursive function can be written iteratively
  - May need to use a stack too, potentially
- Recursive functions are not more efficient than the iterative equivalent
  - Could be the same, with compiler support..
- Why ever use recursion then?
  - If the nature of the problem is recursive, writing it iteratively can be

    a) more time consuming, and/or

    b) less readable

Recursive functions are not more efficient than their iterative equivalent

But .. Recursion is a powerful technique for naturally recursive problems