Algorithms Activity 10: Intro to Dynamic Programming

Learning objective: Students will apply memoization techniques to speed up overlapping recursion.

Model 1: Fibonaccis

Here are three functions to compute Fibonacci numbers, implemented in Python. You may assume that they are all correct (at least they are supposed to be; if not XXX).

```
def fib1(n):
    if n <= 1:
        return n
    else:
        return fib1(n-1) + fib1(n-2)



def fib2(n):
    fibs = [0] * (n+1)
    fibs[1] = 1

    for i in range(2, n+1):
        fibs[i] = fibs[i-1] + fibs[i-2]

    return fibs[n]



fibtable = [0,1]

def fib3(n):

    while len(fibtable) < n+1:
        fibtable.append(-1)

    if fibtable[n] == -1:
        fibtable[n] = fib3(n-1) + fib3(n-2)

    return fibtable[n]
```

1 Which of the three implementations corresponds most directly to the recurrence defining Fibonacci numbers?

2 Draw the call tree for `fib1(5)`.

3 It turns out that `fib1` is extremely slow; it takes exponential time. Explain why it is slow. (You do not have to prove that it takes exponential time.)

4 Trace the execution of `fib2(5)` and explain how it works.

5 Which does more work, `fib2(5)` or `fib1(5)`? Why?

6 In terms of Θ, how long does `fib2(n)` take?[1]

[1] Assuming that each addition takes constant time, which is actually a big fat lie.

7 Suppose we switch the direction of the `for` loop in `fib2`, so `i` loops from `n` down to `2`. Would it still work? Why or why not?

© 2017 Brent A. Yorgey. This work is licensed under a Creative Commons Attribution 4.0 International License.

8 Trace the execution of `fib3(5)` and explain how it works.

9 In terms of Θ, how long does `fib3(n)` take?

10 Fill in this statement: `fib3` is just like `fib1` except that

____.

11 Fill in this statement: `fib2` is just like `fib3` except that

____.

12 Why don't we do something akin to `fib2` or `fib3` for merge sort?

13 Consider the following recursive definition of $Q(n)$ for $n \geq 0$:

$$Q(0) = 0$$
$$Q(1) = Q(2) = 1$$
$$Q(n) = \max \begin{cases} Q(n-3)^2 \\ Q(n-1) + Q(n-2) \end{cases}$$

(Note that there are *three* base cases.) Using pseudocode, write an algorithm to calculate $Q(n)$ efficiently.

© 2017 Brent A. Yorgey. This work is licensed under a Creative Commons Attribution 4.0 International License.