DSC 40B - Discussion 06

Problem 1.

The *Fibonacci sequence* is a sequence of numbers where the next number is determined by adding the two previous numbers. The sequence is famous because the ratio of consecutive elements converges to the golden ratio.

The formal definition of the Fibonacci sequence is recursive:

$$x_n = \begin{cases} 1 & \text{if } n \in \{0, 1\} \\ x_{n-1} + x_{n-2} & \text{otherwise} \end{cases}$$

a) As a warmup, write out the first six elements of the sequence, starting with $n = 0$

Solution: 1,1,2,3,5,8

b) Write a recursive function which takes in n and computes the nth Fibonacci number. How large can n be before your function starts to take more than a second or so to finish?

Solution:

```
def fibonacci(n):
    '''Compute the nth Fibonacci number'''
    if (n==0) or (n==1):
        return 1
    return fibonacci(n-1) + fibonacci(n-2)
```

c) Draw the recursion tree that results from running your function with $n = 5$. Are there overlapping subproblems?

d) Your recursive code for computing the Fibonacci sequence is, in a sense, a *backtracking* algorithm. Modify it by adding a cache to create a top-down dynamic programming solution for computing the nth Fibonacci number.

Solution:

```
def fibonacci_dp(n, cache=None):
    if cache is None:
        cache = [None]*(n+1)

    if(cache[n] is not None):
        return cache[n]

    if (n==0) or (n ==1):
        cache[n] = 1
        return 1

    fib_n = fibonacci_dp(n-1, cache) + fibonacci_dp(n-2, cache)
    cache[n] = fib_n
```

```
    return fib_n
```

e) Create a bottom-up, iterative version of your function.

Solution:

```
def fibonacci_bottom_up(n):
    fib_seq = [None]*(n+1)

    fib_seq[0] = 1

    if(n >= 1):
        fib_seq[1] = 1

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

    return fib_seq[n]
```