473: Notes on Day 5 slides:

Slide 3 Asymptotic Analysis Example:

C < 1. The infinite series converges to 1/(1-c), so Theta(1)

C = 1 f(n) = n, so it is Theta(n)

C > 1 f(n) = (c^(n+1) – 1) / (c – 1). The limit of f(n) divided by c^n as n→infinity is c/(c-1),

so f(n) is Theta(c^n).

Slide 7: A Creative O(log N) Algorithm?

The file **fib3.py** has the solution, and **fib3-incomplete.py** has it partially completed.

```
#Solution for matrix_power:
def matrix_power(m, n):
   result = identity_matrix
   power = x
   while n > 0:
      if n % 2 == 1:
         result = matrix_multiply(result, power)
      power = matrix_multiply(power, power)
#     print "In loop:", n, power, result
      n = n /2
   return result
```

Slide 8: Reconsider our Fibonacci algorithms

Note on recursive: The length of F(n) is .694n, which is O(n).
Note on last one: The # of bits in the numbers in the matrix at most doubles with each matrix multiplication.

Thus we get AN UPPER BOUND OF M(1) + M(2) + M(4) + M(8) + … + M(F(n))

If a = 2, we get (Maple notation):
> **sum((2^i)^2), i=0..log[2](n));**

> **simplify(sum((2^i)^log[2](3), i=0..log[2](n)));**

Slide 12: Modular Addition and Multiplication

0 to N – 1, 0 to 2(N-1), Theta(n).

0 to $(N-1)^2$, 2n Theta(n^2)

Slide 14: Modular Exponentiation

To do better, we can use an algorithm like our previous recursive exponentiation algorithm