Brian Sampson
I pledge my honor that I have abided by the Stevens Honor System.
CS385 HA2

ALGORITHM *Mystery*(n)
//Input: A nonnegative integer n
$S \leftarrow 0$
for $i \leftarrow 1$ **to** n **do**
$S \leftarrow S + i * i$
return S

a. What does this algorithm compute?
b. What is its basic operation?
c. How many times is the basic operation executed?
d. What is the efficiency class of this algorithm?
e. Suggest an improvement, or a better algorithm altogether, and indicate its efficiency class. If you cannot do it, try to prove that, in fact, it cannot be done.

4a. This algorithm computes the sum of squares from 1 to n.
4b. Its multiplication. Below shows how.

$$1^2 + 2^2 + \ldots + n^2 = S$$

$$S = \sum_{n=1}^{n} (n^2)$$

4c. This loop is iterated for n intervals
4d. $\theta(n)$
4e. A better way of doing this algorithm is by using the sum of squares of n Natural Numbers Formula. This formula is $\frac{(n(n+1)(2n+1))}{6}$. This would make the algorithm $\theta(1)$.

1. Solve the following recurrence relations.
a. $x(n) = x(n-1) + 5$ for $n > 1$, $x(1) = 0$
b. $x(n) = 3x(n-1)$ for $n > 1$, $x(1) = 4$
c. $x(n) = x(n-1) + n$ for $n > 0$, $x(0) = 0$
d. $x(n) = x(n/2) + n$ for $n > 1$, $x(1) = 1$ (solve for $n = 2^k$)
e. $x(n) = x(n/3) + 1$ for $n > 1$, $x(1) = 1$ (solve for $n = 3^k$)

1a.
$x(n) = x(x-1) + 5$

Step 1: $x(n-1) = [x(n-2) + 5] + 5 = x(n-2) + 10$
Step 2: $x(n-2) = [[x(n-3) + 5] + 5] + 5 = x(n-3) + 15$
Step 3: $x(n) = x(n-i) + 5 * i$
Step 4: $n - i = 1$
$i = n - 1$
Step 5: $x(n) = x(n-(n-1)) + 5(n-1)$
$x(1) = 0$
$x(n) = x(1) + 5n - 5 \Rightarrow x(n) = 5n - 5$

1b.

$x(n) = 3x(n-1)$

Step 1: $x(n-1) = 3^2 * x(n-2)$

Step 2: $x(n-2) = 3^3 * x(n-3)$

Step 3: $x(n) = 3^i * x(n-i)$

Step 4: $n - i = 1$

$i = n - 1$

Step 5: $3^{(n-1)} * x(n-(n-1))$

$x(1) = 4$

$3^{(n-1)} * x(1)$

$x(n) = 3^{(n-1)} * 4$

$\theta(3^n)$

1c.

$x(n) = x(n-1) + n$

Step 1: $x(n-1) = x(n-2) + 2n$

Step 2: $x(n-2) = x(n-3) + 3n$

Step 3: $x(n) = x(n-i) + in$

Step 4: $n - i = 0$

$i = n$

Step 5: $x(n) = x(n-n) + n^2$

$x(0) = 0$

$x(n) = n^2$

$\theta(n^2)$

1d.

$x(n) = x(\frac{n}{2}) + n$

Step 0: $n = 2^k$

$x(2^k) = x(2^{k-1}) + 2^k$

Step 1: $x(\frac{k}{2}) = x(\frac{k}{4}) + \frac{k}{2}$

Step 2: $x(\frac{k}{4}) = x(\frac{k}{8}) + \frac{k}{4} + \frac{k}{2}$

Step 3: $x(2^k) = x(2^{k-i}) + 2^k(1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \ldots)$

$x(2^k) = x(2^{k-i}) + 2 * 2^k$

Step 4: $2^{k-i} = 2^0$

$k - i = 0$

$i = k$

$n = 2^k$

$k = lg(n)$

Step 5: $x(2^k) = x(2^0) + 2 * 2^k$

$x(n) = x(1) + 2 * n$

$x(n) = 1 + 2 * n$

$\theta(n)$

1e.

$x(n) = x(\frac{n}{3}) + 1$

Step 0: $n = 3^k$

$x(3^k) = x(3^{k-1}) + 1$

Step 1: $x(3^{k-1}) = x(3^{k-2}) + 2$

$x(3^k) = x(3^{k-2}) + 2$

Step 2: $x(3^{k-2}) = x(3^{k-3}) + 3$

$x(3^k) = x(3^{k-3}) + 3$

Step 3: $x(3^k) = x(3^{k-i}) + i$

Step 4: $3^{k-i} = 3^0$

$k - i = 0$

$k = i$

$x(3^k) = x(3^{i-i}) + i$

$x(3^k) = x(3^0) + k$

$x(3^k) = x(1) + k$

Step 5: $n = 3^k$

$k = log_3(n)$

$x(n) = x(1) + log_3(n)$

$x(1) = 1$

$x(n) = x(1) + log_3(n)$

$x(n) = 1 + log_3(n)$

$\theta(log_3(n))$

ALGORITHM $S(n)$

//Input: A positive integer n
//Output: The sum of the first n cubes
if $n = 1$ **return** 1
else return $S(n-1) + n * n * n$

a. Set up and solve a recurrence relation for the number of times the algorithm's basic operation is executed.

b. How does this algorithm compare with the straightforward nonrecursive algorithm for computing this sum?

3a.

$x(n) = x(n-1) + 2$

$x(n-1) = x(n-2) + 4$
$x(n-2) = x(n-3) + 6$
$x(n) = x(n-i) + 2i$
$n - i = 1$
$i = n - 1$
$x(n) = x(n-(n-1)) + 2(n-1)$
$x(n) = x(1) + 2n - 2$
$x(n) = 2n - 1$
$\theta(n)$

3b.

The sum of cubes can be written as $\frac{n^2(n+1)^2}{4}$, which will run in $\theta(1)$. This is always faster than the recursive or iterative algorithm for this function.

However, let's compare all three methods together (Recursive, Iterative, and Formula).

	Time	Space
Recursive	$\theta(n)$	$\theta(n)$
Iterative	$\theta(n)$	$\theta(1)$

Equation	$\theta(1)$	$\theta(1)$

As we can see, the equation will always be faster and less space then the recursive method. The iterative method will take up less space and be the same time complexity as the recursive method.