CS 385- HA2: Recurrence Relations
Breona Pizzuta
9/30/24
“I pledge my honor that I have abided by the Stevens Honor System”

Do the following exercises in the Levitin textbook Download Levitin textbook:

p. 67, #4 a, b, c, d, e (1 point each)

4. Consider the following algorithm.

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.

a) This algorithm computes the sum of square numbers within n nonnegative integers.
b) The basic operation is multiplication
c) The basic operation is executed n times.
d) The efficiency class of this algorithm is Θ(n)
e) A more efficient algorithm would be using the formula for the sum of squares, $\sum_{i=1}^{n} i^2 = \frac{n+(n+1)(2n+1)}{6}$. This algorithm has the improved efficiency class of Θ(1).

p. 76, #1 a, b, c, d, e (5 points each)

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 $n = 3^k$)

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
Step 1: replace n with n-1 x(n-1) = x(n-1-1) +5 x(n)= x(n-1-1)+5+5 x(n)= x(n-2) +10 Step 2: replace n with n-2 x(n-2)= x(n-1-2)+5 +5 +5 x(n)= x(n-3) +15 Step 3: General form x(n)= x(n-i) + 5(i) Step 4: Plug in given condition x(1)= 0 n-i= 1 i= n-1 Step 5: Plug in with initial x(n) = x(n- (n-1)) + 5(n-1) x(n)= x(n-n-1) +5n-5 x(n) = x(1) + 5n-5 x(n) = 5n-5	Step 1: replace n with n-1 x(n-1) = 3x(n-1-1) x(n)= 3(3x(n-2)) x(n)= 9x(n-2) Step 2: replace n with n-2 x(n-2)= 3x(n-2-1) x(n)= 9((3x(n-3)) x(n)= 27x(n-3) Step 3: General form x(n)= $3^{i}x(n - i)$ Step 4: Plug in given condition x(1)= 4 n-i=1 i= n-1 Step 5: Plug in with initial x(n)= $3^{(n-1)}x(n - (n - 1)$ x(n)= $3^{(n-1)}x(n - n + 1)$ x(n)= $3^{(n-1)}x(1)$ x(n)= $3^{(n-1)} * 4$	Step 1: replace n with n-1 x(n-1)= x(n-1-1) + (n-1) x(n)= x(n-2) +(n-1)+n Step 2: replace n with n-2 x(n-2)= x(n-3) +(n-2) x(n)= x(n-3)+(n-2)+ (n-1)+n Step 3: General form x(n)= x(n-i) + (n-i+1) +(n-i+2) +...+n Step 4: Plug in given condition x(0)=0 n-i=0 i=n Step 5: Plug in with initial x(n)= x(n-n)+ (n-(n+1)+(n-(n+2) +...+n x(n)= x(0)+ 1 + 2 +...+n x(n)= $\frac{n(n+1)}{2}$

d) $x(n)= x(n/2) +n$ for $n>1$, $x(1)=1$ (solve for $n=2^k$)

Step 0: Substitute

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

Step 1: replace 2^k with 2^{k-1}

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

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

Step 2: replace 2^k with 2^{k-2}

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

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

Step 3: General Form

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

Step 4: Plug in given condition

$x(1)=1$

$2^{k-i}= 1$

$k-i=0$

$i=k$

Step 5: Plug in with initial

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

$x(2^k)= x(2^0) + 2^1 + 2^2 + \dots + 2^k$

$x(2^k)= x(1) + 2^1 + 2^2 + \dots + 2^k$

$x(2^k)= 2 * 2^k - 1$

$x(n)= 2n-1$

e) $x(n)= x(1/3)+1$ for $n>1$, $x(1)=1$ (solve for $n= 3^k$)

Step 0: Substitute

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

Step 1: replace 3^k with 3^{k-1}

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

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

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

Step 2: replace 3^k with 3^{k-2}

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

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

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

Step 3: General Form

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

Step 4: Plug in given condition

$x(1)=1$

$3^{k-1} = 1$

$k-i=0$

$i=k$

Step 5: Plug in with initial

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

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

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

$\log_3 n= k$

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

p. 76-77, #3 a (5 points), b (5 points)

3. Consider the following recursive algorithm for computing the sum of the first n cubes: $S(n) = 1^3 + 2^3 + \cdots + n^3$.

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?

a) S(n)= S(n-1)+2 for n>1, S(1)=0

Step 1: replace n with n-1
S(n-1)=S(n-1-1)+2
S(n)= S(n-2)+2+2

Step 2: replace n with n-2
S(n-2)=S(n-3)+2
S(n)= S(n-3)+2+2+2
S(n)= S(n-3)+6

Step 3: General form
S(n)= S(n-i) + 2i

Step 4: Plug in given condition
S(1)=0
n-i=1
i= n-1

Step 5: Plug in with initial
S(n)= S(n-(n-1))+ 2(n-1)
S(n)= S(1)+ 2n-2
S(n)= 0+ 2n-2
S(n)=2n-2

b) Loop that runs n-1 times and uses two basic operations per iteration so 2n-2 basic operations total. This is the same as the recursive operations. However, recursive will use more memory than non recursive because it creates a new stack frame for every iteration.