

CS 1050B: Constructing Proofs

Problem Set 3 :

Due Monday, Oct 9th, after the class

1. Problem 1 : Rosen 7.1: 8

Find the solution to each of these recurrence relations with the given initial conditions. Use an iterative approach such as that used in Example 5.

Answer:

In the iterative approach, we write a_n in terms of a_{n-1} , then write a_{n-1} in terms of a_{n-2} (using the recurrence relation with $n-1$ plugged in for n), and so on. When we reach the end of this procedure, we use the given initial value of a_0 . This will give us an explicit formula for the answer or it will give us a finite series, which we then sum to obtain an explicit formula for the answer.

a) $a_n = -a_{n-1}, a_0 = 5$
 $a_n = -a_{n-1} = (-1)^2 a_{n-2} = \dots = (-1)^n a_{n-n} = (-1)^n a_0 = 5 \cdot (-1)^n$

b) $a_n = a_{n-1} + 3, a_0 = 1$
 $a_n = 3n + 1$

c) $a_n = a_{n-1} - n, a_0 = 4$

$$\begin{aligned} a_n &= -n + a_{n-1} \\ &= -n + (-(n-1) + a_{n-2}) = -(n + (n-1)) + a_{n-2} \\ &\quad \vdots \\ &= -(n + (n-1) + (n-2) + \dots + (n - (n-1))) + a_{n-n} \\ &= -\frac{n(n+1)}{2} = \frac{-n^2 - n + 8}{2} \end{aligned}$$

d) $a_n = 2a_{n-1} - 3, a_0 = -1$

$$\begin{aligned} a_n &= -3 + 2a_{n-1} \\ &= -3 + 2(-3 + 2a_{n-2}) = -3 + 2(-3) + 4a_{n-2} \\ &\quad \vdots \\ &= -3(1 + 2 + \dots + 2^{n-1}) + 2^n a_{n-n} \quad (\text{Remember this } \sum_{n=0}^k 2^n = 1 + 2 + \dots + 2^n = 2^{n+1} - 1) \\ &= -2^{n+2} + 3 \end{aligned}$$

e) $a_n = (n+1)a_{n-1}, a_0 = 2$
 $a_n = (n+1)a_{n-1} = (n+1)na_{n-2} = (n+1)n(n-1)a_{n-3} = \dots = (n+1)! \cdot a_0 = 2(n+1)!$

f) $a_n = 2na_{n-1}, a_0 = 3$

$$\begin{aligned} a_n &= 2na_{n-1} \\ &= 2n(2(n-1)a_{n-2}) = 2^2(n(n-1))a_{n-2} \\ &\quad \vdots \\ &= 2^n(n-1)(n-2)(n-3)\cdots 1 \cdot a_0 \\ &= 3 \cdot 2^n! \end{aligned}$$

g) $a_n = -a_{n-1} + n - 1, a_0 = 7$

$$\begin{aligned} a_n &= -a_{n-1} + (n-1) \\ &= -(-a_{n-2} + (n-2)) + (n-1) \\ &= -(-(-a_{n-3} + (n-3))) - (n-2) + (n-1) \\ &= (-1)^n a_{n-n} + (-1)^{n-1}(n-n) + (-1)^{n-2}(n-(n-1)) + \cdots + (-1)^1(n-2) + (-1)^0(n-1) \\ &= (-1)^n a_0 + \sum_{k=1}^n (-1)^{k-1}(n-k) \end{aligned}$$

Now let $b_n = \sum_{k=1}^n (-1)^{k-1}(n-k)$ and simplify it by even and odd case. When n is even, we have

$$\begin{aligned} b_n &= -(n-n) + (n-(n-1)) + \cdots - (n-2) + (n-1) \\ &= 0 + 1 - 2 + 3 + \cdots - (n-2) + (n-1) \\ &= \frac{n}{2} \end{aligned}$$

When n is odd, we have

$$\begin{aligned} b_n &= (n-n) - (n-(n-1)) + \cdots - (n-2) + (n-1) \\ &= 0 - 1 + 2 - 3 + \cdots - (n-2) + (n-1) \\ &= \frac{n-1}{2} \end{aligned}$$

Therefore, we have

$$\begin{aligned} a_n &= (-1)^n a_0 + b_n \\ &= (-1)^n \cdot 7 + \begin{cases} \frac{n}{2} & n \text{ is even} \\ \frac{n-1}{2} & n \text{ is odd} \end{cases} \\ &= (-1)^n \cdot 7 + \lfloor \frac{n}{2} \rfloor \end{aligned}$$

Another way to write this down is $a_n = (-1)^n \cdot 7 + \frac{2n-1+(-1)^n}{4}$

2. Problem 2 : Rosen 7.3: 8

Suppose that $f(n) = 2f(n/2) + 3$ when n is even, and $f(1) = 5$. Find

Answer:

- a) $f(2)$
 $f(2) = 2f(1) + 3 = 2 \cdot 5 + 3 = 13$
- b) $f(8)$
 $f(8) = 2f(4) + 3 = 2(2f(2) + 3) + 3 = 61$
- c) $f(64) = 509$
- d) $f(1024) = 8189$

3. Problem 3 : Rosen 7.3: 20

- a) Set up a divide-and-conquer recurrence relation for the number of modular multiplications required to compute $a^n \bmod m$, where a , m , and n are positive integers, using the recursive algorithms from Example 3 in Section 4.4.
- b) Use the recurrence relation you found in part (a) to construct a big- O estimate for the number of modular multiplications used to compute $a^n \bmod m$ using the recursive algorithm.

Answer:

- a) We compute $a^n \bmod m$, when n is even, by first computing $y := a^{n/2} \bmod m$ recursively and then doing one modular multiplication, namely $y \cdot y$. When n is odd, we first compute $y := a^{(n-1)/2} \bmod m$ recursively and then do two multiplications namely $y \cdot y \cdot a$. So if $f(n)$ is the number of multiplications required, assuming the worst, then we have essentially $f(n) = f(n/2) + 2$.
- b) By the Master Theorem, with $a = 1$, $b = 2$, $c = 2$, and $d = 0$, we see that $f(n)$ is $O(n^0 \log n) = O(\log n)$.