Homework 8: Core solutions

Review exercises for Chapter 5 page 183 problems 25, 26a-26b, 29.
Section 8.1 on page 252 problems 8, 9, 10, 13.
Section 8.2 on page 264 problems 1, 2, 4, 5, 7a-7b, 9.

25. *Find a formula for a_n, given $a_0 = a_1 = 1$ and $a_n = 3a_{n-1} - 2a_{n-2} + 5$ for $n \geq 2$. Solution:* The difference between this recurrence relation and the previous ones (from the last homework) is the extra term "+5". First, let's guess a *particular solution*, p_n that satisfies the inhomogeneous recurrence relation $a_n = ra_{n-1} + sa_{n-2} + f(n)$ above (with $r = 3, s = -2$, and $f(n) = 5$), ignoring initial conditions. We guess that $p_n = c$ for some constant c, since $f(n)$ is constant. We plug in our guess p_n into the inhomogeneous recurrence and solve for c. We have

$$\begin{aligned} p_n &= 3p_{n-1} - 2p_{n-2} + 5 \\ c &= 3c - 2c + 5 \\ c - c &= 5 \\ 0 &= 5 \end{aligned}$$

Oops! Looks like our guess was not a good one. Let's try a different guess. Let's try $p_n = a + bn$ since, at least, $f(n) = 5$ is linear (it's even constant, but that obviously didn't work). With this choice of p_n we plug into the inhomogeneous recurrence and try to solve for the constants a and b. We have

$$\begin{aligned} p_n &= 3p_{n-1} - 2p_{n-2} + 5 \\ a + bn &= 3(a + b(n-1)) - 2(a + b(n-2)) + 5 \\ a + bn &= 3a - 2a + 3bn - 2bn - 3b + 4b + 5 \\ 0 &= b + 5 \end{aligned}$$

So $b = -5$ and a is undetermined (so just set $a = 0$, if we pick a different value it will work itself out when we find c_1, c_2 later). Thus, a particular solution to the inhomogeneous recurrence, ignoring initial conditions, is $p_n = -5n$.

Now, by Theorem 5.3.2 on page 172 we have that the solution to the recurrence is the sum $p_n + q_n$ where p_n is as above and q_n is a general solution to the homogeneous recurrence relation $a_n = 3a_{n-1} - 2a_{n-2}$, ignoring initial conditions (meaning we haven't solved for c_1, c_2 yet).

We solve $a_n = 3a_{n-1} - 2a_{n-2}$ in the usual way. The characteristic polynomial is $x^2 - 3x + 2 = (x-2)(x-1)$, which has roots $x = 1, 2$. So the general solution to the homogeneous recurrence is $a_n = c_1 + c_2 2^n$.

Finally, we have that the general solution to the inhomogeneous recurrence we started with is $a_n = p_n + q_n = -5n + c_1 + c_2 2^n$. We solve for c_1, c_2 in the usual way, by taking into account

the initial values $a_0 = 1, a_1 = 1$. We have

$$a_0 = 1 \implies c_1 + c_2 = 1$$
$$a_1 = 1 \implies -5 + c_1 + 2c_2 = 1.$$

Solving the system of two linear equations in the two variables c_1, c_2 we immediately obtain $c_2 = 5$ and $c_1 = -4$. Thus, the solution is

$$a_n = -5n - 4 + 5 \cdot 2^n, \ n \geq 0.$$

Just because it's the first time we've tried this, lets check. The first few terms of the sequence, starting at $n = 0$ of course, are $1, 1, 6, 21, 56, \ldots$, and this satisfies $a_0 = a_1 = 1$ and $a_2 = 3(1) - 2(1) + 5 = 6$ ✓ and $a_3 = 3(6) - 2(1) + 5 = 21$ ✓ and $a_4 = 3(21) - 2(6) + 5 = 56$ ✓. We did it! (or at least, we checked the first few values) (to make extra sure we could also check that the solution we found works by plugging it into the recurrence, but I'm sick of this problem, um, I mean, ahem "checking that way is left to the reader")

□

29. *Let a and b be any positive integers and define the recurrence $a_1 = a, a_2 = b$, and $a_n = a_{n-1} + a_{n-2}$ for $n \geq 3$. Assume that the sequence $\frac{a_2}{a_1}, \frac{a_3}{a_2}, \frac{a_4}{a_3}, \ldots$ has a limit k. What is the value of k? Solution:* What to try? The recurrence relation given has characteristic polynomial $x^2 - x - 1$, which has roots $x = -1 \pm \sqrt{2}$ using the quadratic formula. So, the general solution is $a_n = c_1(-1 + \sqrt{2})^n + c_2(-1 - \sqrt{2})^n$. For now, we are not overly concerned with the particular values of c_1, c_2. Note that, since $|-1 + \sqrt{2}| < 1$ that, in the limit, $a_n \approx c_2(-1 - \sqrt{2}^n)$. Therefore, $a_{n+1}/a_n \approx \frac{c_2(-1-\sqrt{2})^{n+1}}{c_2(-1-\sqrt{2})^n} = -1 - \sqrt{2}$. In other words, the limit of a_{n+1}/a_n is $-1 - \sqrt{2}$ as n tends to infinity. (Cool!) (Notice, we never cared at all about what the values of a, b, c_1, c_2 were)

□

8. *Describe an algorithm that, upon input of n real numbers, $a_1, a_2, \ldots, a_n$, outputs their average. Solution:*

Input: $a_1, a_2, \ldots, a_n$.
Procedure:
STEP 1: Initialize $S = 0$.
STEP 2: For $i = 1..n$,

replace S with $S + a_i$.
STEP 3: replace S with S/n.
Output: S.

□

9. *Describe an algorithm that outputs the maximum of the numbers $a_1, \ldots, a_n$. Solution:*

Input: $a_1, a_2, \ldots, a_n$.
Procedure:
STEP 1: Initialize $S = a_1$.
STEP 2: For $i = 1..n$,
IF $a_i \leq a_{i+1}$, replace S with a_{i+1}.
Output: S.

□

10. *Describe an algorithm that determines how many of the numbers $a_1, \ldots, a_n$ are equal to x. Solution:*

Input: $a_1, a_2, \ldots, a_n$, and x.
Procedure:
STEP 1: Initialize $S = 0$.
STEP 2: For $i = 1..n$,
IF $a_i = x$, replace S with $S + 1$.
Output: S.

□

11. *Describe an algorithm that gives all solutions x in the range $0 \leq x < n$ to the congruence*

$$ax \equiv b (mod\ n)$$

for given values of a, b, n, which should all be integer inputs. If there are no solutions, the algorithm should output the words NO SOLUTION. *Solution:*

It may be best to give some examples before trying to tackle this problem. First, some questions to get us started.

What are the solutions to $x \equiv 3$ *(mod* 7*)?* Recall, $x \equiv 3 \pmod 7$ means that the integer x has a remainder of 3 after dividing by 7. There is only one x value that is in the range $0 \leq x < 7$, namely $x = 3$.

What are the solutions to $2x \equiv 3$ *(mod* 7*)?* The statement $2x \equiv 3 \pmod 7$ means "$2x$ has a remainder of 3 after dividing by 7". Lets just check the x-values in the range $x = 0, \ldots, 6$ that make that true.

$$\begin{array}{lll} x = 0 & 0 \equiv 3 \pmod 7 & \text{FALSE} \\ x = 1 & 2 \equiv 3 \pmod 7 & \text{FALSE} \\ x = 2 & 4 \equiv 3 \pmod 7 & \text{FALSE} \\ x = 3 & 6 \equiv 3 \pmod 7 & \text{FALSE} \\ x = 4 & 8 \equiv 3 \pmod 7 & \text{FALSE} \\ x = 5 & 10 \equiv 3 \pmod 7 & \text{FALSE} \\ x = 6 & 12 \equiv 3 \pmod 7 & \text{FALSE} \end{array}$$

Ok, so in this case, our algorithm should output NO SOLUTION.

Let's try another case. Set $a = 2$, $b = 4$ and $n = 7$. We are going to look for solutions to the equivalence $ax \equiv b \pmod 7$,

$$2x \equiv 4 \pmod 7.$$

Again, we look in the range $x = 0, \ldots, 6$ (since $n = 7$).

$$\begin{array}{lll} x = 0 & 0 \equiv 4 \pmod 7 & \text{FALSE} \\ x = 1 & 2 \equiv 4 \pmod 7 & \text{FALSE} \\ x = 2 & 4 \equiv 4 \pmod 7 & \text{TRUE} \\ x = 3 & 6 \equiv 4 \pmod 7 & \text{FALSE} \\ x = 4 & 8 \equiv 4 \pmod 7 & \text{FALSE} \\ x = 5 & 10 \equiv 4 \pmod 7 & \text{FALSE} \\ x = 6 & 12 \equiv 4 \pmod 7 & \text{FALSE} \end{array}$$

Alright, so in this case, the output of our algorithm should be $x = 4$.

Let's do another case. Let's try to make a lot of output this time. Set $a = 7$, $b = 0$ and $n = 7$.

Then,

$$\begin{array}{lll} x=0 & 0\equiv 0 \pmod 7 & \text{TRUE} \\ x=1 & 7\equiv 0 \pmod 7 & \text{TRUE} \\ x=2 & 14\equiv 0 \pmod 7 & \text{TRUE} \\ x=3 & 21\equiv 0 \pmod 7 & \text{TRUE} \\ x=4 & 28\equiv 0 \pmod 7 & \text{TRUE} \\ x=5 & 35\equiv 0 \pmod 7 & \text{TRUE} \\ x=6 & 42\equiv 0 \pmod 7 & \text{TRUE} \end{array}$$

In this case, the output should be $x = 0, 1, 2, 3, 4, 5, 6, 7$.

Ok, last example. Set $a = 4$, $b = 0$, and $n = 8$. Now lets find the solutions x in the range $x = 0, \ldots, 7$ (since $n = 8$ now the range is bigger, $0 \leq x < n$, then it was when n was equal to 7).

$$\begin{array}{lll} x=0 & 0\equiv 0 \pmod 7 & \text{TRUE} \\ x=1 & 4\equiv 0 \pmod 7 & \text{FALSE} \\ x=2 & 8\equiv 0 \pmod 7 & \text{TRUE} \\ x=3 & 12\equiv 0 \pmod 7 & \text{FALSE} \\ x=4 & 16\equiv 0 \pmod 7 & \text{TRUE} \\ x=5 & 20\equiv 0 \pmod 7 & \text{FALSE} \\ x=6 & 24\equiv 0 \pmod 7 & \text{TRUE} \end{array}$$

In this case, the output should be $x = 0, 2, 4, 6$.

Ok, we are ready to try to describe the algorithm.

Recall, $ax \equiv b \pmod n$ if and only if $ax - b \equiv 0 \pmod n$. In other words, x satisfies the equivalence $ax \equiv b \pmod n$ exactly when $ax - b$ is divisible by n. Now, in order to ensure that our algorithm is a *finite* process, we can't check all the possible multiples of 7 to see if $ax - b$ is one of them, so we will have to write a sub-routine to deal with checking if $ax - b$ is divisible by 7.

There is some issue when $ax - b$ is negative. But for now, we will avoid discussing it.

Sub-routine $DIV(x, a, b, n)$**: Input:** x, a, b, n with $ax - b \geq 0$.
Procedure:
STEP 1: Initialize $S = 0$.
STEP 2: For $k = 0..\lceil \frac{ax-b}{n} \rceil$,

IF $ax - b = kn$, replace S with 1.

Output: S.

The algorithm $DIV(x, a, b, n)$ outputs 1 if $ax - b$ is divisible by n and outputs 0 otherwise. This is because if the output of the algorithm is 1, then for some k in the range $0 \leq k \leq \lceil \frac{ax-b}{n} \rceil$ the equality $ax - b = kn$ holds.

If we knew that $ax - b$ were non-negative for all x in the range $0 \leq x < n$, the algorithm we would want is

Input: a, b, n.

Procedure:

STEP 1: Initialize $S = \{\}$.

STEP 2: For $x = 0..n - 1$,

IF $DIV(x, a, b, n) = 1$, replace S with $S \cup \{x\}$.

Output: S.

We can adjust the algorithm to accommodate for the restriction $ax - b$ by replacing b with its remainder modulo n. That is, $ax - b$ is divisible by 7 if and only if $ax - (b + 7k)$ is divisible by 7 for all integers k (*very* easy to prove directly). To that end, define the subroutine

Sub-routine $REM(b, n)$**:**

Input: b, n.

Procedure:

STEP 1: Initialize $r = 0$.

STEP 2: For $x = 0..n - 1$,

IF $DIV(x, 1, b, n) = 1$, replace r with x.

Output: r where $0 \leq r < n$ and $r \equiv b \pmod{n}$.

Finally, the desired algorithm is

Input: a, b, n.

Procedure:

STEP 1: Initialize $S = \{\}$.

STEP 2:

IF $ax - b \geq 0$

For $x = 0..n - 1$,

IF $DIV(x, a, b, n) = 1$, replace S with $S \cup \{x\}$.

IF $ax - b$¡ 0

Replace b with $REM(b, n)$,

For $x = 0..n - 1$,

IF $DIV(x, a, b, n) = 1$, replace S with $S \cup \{x\}$.
Output: S.

□

1. *Consider the distance algorithm used to calculate*

$$\sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

using $3n$ *operations:* n *subtractions,* n *squarings,* $n - 1$ *additions, and 1 square root. Assume that addition and subtraction require the same amount of time, that addition is twice as fast as multiplication, and that multiplication is ten times as fast as the square root operation. Find the complexity function. Solution:* Considering the time it takes to perform the subtraction operation as a 'unit', the complexities of the + and − operation are both 1. The complexity of ^{2} is 2, and the complexity of $\sqrt{\ }$ is 10. The complexity of the algorithm is thus $n+2n+n-1+10 = 4n + 9 = O(n)$. (by the way, the original algorithm was also $O(n)$, so really no time is lost with the extra assumptions)

□

2. *Show that the number of operations required to multiply an* n*-digit number by a single-digit number does not exceed* $3n - 2$*. Count* $+$ *and* $\cdot$ *as equivalent operations. Solution:* Consider the following algorithm for multiplying an n-digit number by a single-digit number.

Input: $(a_n a_{n-1} \ldots a_1)_{10}, x$.
Procedure:
STEP 1: Initialize $m_1 = r_1 = \cdots = m_n = r_n = m_{n+1} = r_{n+1} = 0$.
STEP 2: For $i = 1..n$,
replace m_i with the ones digit of $x \cdot a_i + r_i$,
IF $x \cdot a_i \geq 10$ replace r_{i+1} with 1,
STEP 3: Replace m_{n+1} with r_{n+1}.
Output: $(m_{n+1} m_n m_{n-1} \ldots m_1)_{10}$.

There are a total of $2n$ multiplications and n additions in this algorithm. Thus, the complexity is bounded by $3n$. But we need to get rid of two operations to meet the book's requirement of a complexity bounded by $3n - 2$. It isn't hard to see how to change the algorithm above to get the desired result.

Input: $(a_n a_{n-1} \dots a_1)_{10}, x$.
Procedure:
STEP 1: Initialize $m_1 = r_1 = \cdots = m_n = r_n = m_{n+1} = r_{n+1} = 0$.
STEP 2: Replace m_1 with the ones digit of $x \cdot a_1$.
STEP 3: For $i = 2..n$,
Replace m_i with the ones digit of $x \cdot a_i + r_i$,
IF $x \cdot a_i \geq 10$ replace r_{i+1} with 1,
STEP 4: Replace m_{n+1} with r_{n+1}.
Output: $(m_{n+1} m_n m_{n-1} \dots m_1)_{10}$.

In Step 2 there is one operation. In each iteration of Step 3 there are three operations, and there are $n-1$ iterations, for a total of $3(n-1)$ operations in Step 3. Thus, the total number of operations is $1 + 3(n-1) = 1 + 3n - 3 = 3n - 2$ ✓. □

By the way ...

If we do not allow the operation "$\cdot$", then we replace $x \cdot a_i + r_i$ with $\underbrace{a_i + a_i + \cdots + a_i}_{x \text{ times}} + r_i$. With this replacement, the algorithm now uses $x + 1$ of the "+" operations in the i-th iteration of Step 2. Thus, the complexity is bounded by $\sum_{i=1}^{n}(x+1) = n(x+1) = O(n)$. Unfortunately, this is not good enough for the problem; we are supposed to show the complexity is bounded by $f(n) = 3n - 2$ and $n(x+1)$ is bigger if $x \geq 3$, which is possible.

If we instead replace $x \cdot a_i + r_i$ with $\underbrace{x + x + \cdots + x}_{a_i \text{ times}} + r_i$, then the complexity of the i-th iteration of Step 2 is bounded by $a_i + 1$, and the complexity of the algorithm is $\sum_{i=1}^{n}(a_i + 1) \leq \max\{a_1, \dots, a_n\} \cdot n + n = O(n)$. But, again, this complexity is not good enough unless the maximum of the digits in the n-digit number is 2.

Even though neither are "good enough", they are both good enough from an asymptotic standpoint (they all are $O(n)$).

Question: What is the best complexity you can get if you only allow the operation +? What is the best complexity you can get if the cost of $\cdot$ is some constant times the cost of +?

3. *Let x be a real number and n a positive integer. Consider the following two algorithms for calculating x^{2^n}.*

Algorithm A:
Input: x, n.
Procedure:
STEP 1: Initialize $a = 1$.

STEP 2: For $i = 1..2^n$,
replace a with xa.
Output: a.

Algorithm B:
Input: x, n.
Procedure:
STEP 1: Initialize $a = x$.
STEP 2: For $i = 1..n$,
replace a with a^2.
Output: a.

Find complexity functions for both algorithms and explain why ***B*** *is more efficient. Assume that* $\cdot$ *is the basic operation. Solution:* The complexity of **A** is 2^n while the complexity of **B** is only n. Since, $n \prec 2^n$, the complexity of **B** is much (*much MUCH*) better.

□

7a. *Show* $f(n) = 5n$ *is* $O(g)$ *where* $g(n) = n^3$. *Solution:* We need to find $c > 0$ and $n_0 \in \mathbb{Z}$ such that

$$\forall n \geq n_0 \quad f(n) \leq cg(n).$$

Setting $c = 1$ and $n_0 = 5$ works, since if $n \geq 5$ then $5n \leq n^2 \leq n^3$.

□

7b. *Show* $f(n) = 17n^4 + 8n^3 + 5n^2 + 6n + 1$ *is* $O(g)$ *where* $g = n^4$ *Solution:* Set $c = 85$ and $n_0 = 0$. Then, if $n \geq 0$ we have

$$17n^4 + 8n^3 + 5n^2 + 6n + 1 \leq 17n^4 + 17n^4 + 17n^4 + 17n^4 + 17n^4 = 85n^4.$$

□

9. *If* $f, g, h : \mathbb{N} \to \mathbb{R}$, $f = O(h)$ *and* $g = O(h)$, *show that* $f + g = O(h)$. *Solution:* By definition, since $f = O(h)$ there exists $c_1 > 0$ and $n_0 \in \mathbb{Z}$ such that if $n \geq n_0$ then $f(n) \leq c_1 h(n)$. Similarly, there exists $c_2 > 0$ and $m_0 \in \mathbb{Z}$ such that if $n \geq m_0$ then $g(n) \leq c_2 h(n)$. Set $c = c_1 + c_2$ and $N_0 = \max\{m_0, n_0\}$. Then, if $n \geq N_0$ then $f(n) \leq c_1 h(n)$ and $g(n) \leq c_2 h(n)$, so $f(n) + g(n) \leq (c_1 + c_2)h(n)$, as desired.

□