Exercises from Section 1.1

Tord M. Johnson

April 25, 2020

1. [*10*] The text showed how to interchange the values of variables m and n, using the replacement notation, by setting $t \leftarrow m$, $m \leftarrow n$, $n \leftarrow t$. Show how the values of *four* variables (a, b, c, d) can be rearranged to (b, c, d, a) by a sequence of replacements. In other words, the new value of a is to be the original value of b, etc. Try to use the minimum number of replacements.

The sequence of replacements is as shown below.

$$\begin{aligned} t &\leftarrow a \\ a &\leftarrow b \\ b &\leftarrow c \\ c &\leftarrow d \\ d &\leftarrow t \end{aligned}$$

2. [*15*] Prove that m is always greater than n at the beginning of step E1, except possibly the first time this step occurs.

We will prove that $m > n$ holds as a precondition to step E1, except possibly the first time this step occurs.

Proposition. *$m > n$ holds as a precondition to step E1, except possibly the first time this step occurs.*

Proof. Initially, before the first time step E1 occurs, either $m > n$ or $m \leq n$.

First, we consider the case when $m > n$. After E1, we have $0 \leq r < n$. If $r = 0$, the algorithm terminates by step E2, proving one variant of the first case. Otherwise, if $r > 0$, after the assignment of step E3, we have $m > n$ as a precondition to step E1, concluding the proof for the first case.

Second, we consider the case wehn $m \leq n$. After E1, we have $0 \leq r < n$. If $m = n$, then $r = 0$ and the algorithm terminates by step E2, proving one variant of the second case. Otherwise, if $m < n$, after the assignment of step E3, we have $m > n$ as a precondition to step E1, concluding the proof for the second case.

Therefore, having proved all possible cases, $m > n$ holds as a precondition to step E1, except possibly the first time this step occurs. □

3. [*20*] Change Algorithm E (for the sake of efficiency) so that all trivial replacement operations such as "$m \leftarrow n$" are avoided. Write this new algorithm in the style of Algorithm E, and call it Algorithm F.

Algorithm F (*Efficient Euclid's algorithm*). Given two positive integers m and n, find their *greatest common divisor*, that is, the largest positive integer that evenly divides both m and n. For the sake of efficiency, all trivial replacement operations are avoided.

F1. [Find first remainder.] Divide m by n and let m be the remainder. (We will have $0 \leq m < n$.)

F2. [Is first remainder zero?] If $m = 0$, the algorithm terminates; n is the answer.

F3. [Find second remainder.] Divide n by m and let n be the remainder. (We will have $0 \leq n < m$.)

F4. [Is second remainder zero?] If $n = 0$, the algorithm terminates; m is the answer.

F5. [Repeat.] Go back to step F1. ■

4. [*16*] What is the greatest common divisor of 2166 and 6099?

We will determine the greatest common divisor of 2166 and 6099 using iterations i of Algorithm E.

i	m_i	n_i	r_i
1	2166	6099	2166
2	6099	2166	1767
3	2166	1767	399
4	1767	399	171
5	399	171	57
6	171	57	0

Therefore, the greatest common divisor of 2166 and 6099 is 57.

► **5.** [*12*] Show that the "Procedure for Reading This Set of Books" that appears after the preface actually fails to be a genuine algorithm on at least three of our five counts! Also mention some differences in format between it and Algorithm E.

The "Procedure for Reading This Set of Books" that appears after the preface fails to be a genuine algorithm on at least three of five counts.

The procedure lacks *finiteness*, as it will not always terminate after a finite number of steps. In particular, when we reach step 18, we are to go back to step 3, causing us to repeat the procedure an infinite number of times.

The procedure lacks *definiteness*, as some steps are not precisely defined. As an example, consider step 10, which instructs us to report errors to the author.

For the procedure, the *output* is not clearly defined.

The procedure also lacks *effectiveness*, as some steps are not sufficiently basic and in principle can't be done exactly and in a finite length of time. Consider steps 12 and 13 which instruct us to work on exercises in the book. In the case of exercises rated 50 or higher, research problems not yet solved satisfactorily, work on such exercises is clearly not bounded in time to be considered effective.

There are also differences in format. In particular: it is not headed by a label specifying it as an algorithm and with an identifying letter, such as **Algorithm A**; it is not accompanied by a brief description; step numbers are not preceded by the (missing) identifying letter, such as **A1**; each step is not introduced with a brief description in square brackets; parenthetical remarks do not appear to be limited to asserting invariant conditions; and the procedure is not terminated by a square symbol, such as ■.

6. [*20*] What is T_5, the average number of times step E1 is performed when $n = 5$?

We want to determine T_5, the average number of times step E1 is performed when $n = 5$. It is sufficient to consider only those cases where $1 \leq m \leq n$. We count the steps s for each case below.

s	m_s	n_s	r_s
1	1	5	1
2	5	1	0

s	m_s	n_s	r_s
1	2	5	2
2	5	2	1
3	2	1	0

s	m_s	n_s	r_s
1	3	5	3
2	5	3	2
3	3	2	1
4	2	1	0

s	m_s	n_s	r_s
1	4	5	4
2	5	4	1
3	4	1	0

s	m_s	n_s	r_s
1	5	5	0

The average number of steps is $(2+3+4+3+1)/5 = 13/5 = 2.6$. That is, $T_5 = 2.6$.

▶ **7.** [*M21*] Suppose that m is known and that n is allowed to range over all positive integers; let U_m be the average number of times that step E1 is executed in Algorithm E. Show that U_m is well defined. Is U_m in any way related to T_m?

Assume that the value of m is known but n is allowed to range over all positive integers. We want to know the *average* number of times, U_m that step E1 of Algorithm E will be performed.

U_m is well defined. For $m < n$, the first iteration of the algorithm simply switches m and n (constituting one performance of step E1); thereafter, on average, T_n additional steps are required. Otherwise, if $m \geq n$, we only have a finite number of cases to consider for $1 \leq n \leq m$, which as $n \to \infty$, will contribute to the average.

Therefore, $U_m = T_n + 1$.

8. [*M25*] Give an "effective" formal algorithm for computing the greatest common divisor of positive integers m and n, by specifying θ_j, ϕ_j, a_j, b_j as in Eqs. (3). Let the input be represented by the string $a^m b^n$, that is, m a's followed by n b's. Try to make your solution as simple as possible. [*Hint:* Use Algorithm E, but instead of division in step E1, set $r \leftarrow |m-n|$, $n \leftarrow \min(m,n)$.]

First, we develop an algorithm that doesn't rely on division to determine the greatest common divisor.

Algorithm G (*Euclid's algorithm without division*). Given two positive integers m and n, find their *greatest common divisor*, without division.

G1. [Find difference.] Find the difference $|m-n|$ and let r be this difference. Also set $n \leftarrow \min(m,n)$. (We will have $0 \leq r < n$.)

G2. [Is it zero?] If $r = 0$, the algorithm terminates with answer n.

G3. [Reduce.] Set $m \leftarrow n$, $n \leftarrow r$, and go back to step G1. ■

Next, we express this algorithm formally, without any initial consideration of whether it is "effective" or not.

Let Q be the set of all ordered pairs (m,n); all ordered triples $(m,n,1)$; all ordered quadruples $(m,n,r,2)$, and $(m,n,r,3)$; and all singletons (n); where m and n are positive integers and r is a nonnegative integer. Let I be the subset of all pairs (m,n), and let Ω be the subset of all singletons (n). Let f be defined as follows:

$$\begin{aligned} f((m,n)) &= (m,n,1) \\ f((m,n,1)) &= (m,\min(m,n),|m-n|,2) \\ f((m,n,r,2)) &= (n) \text{ if } r=0,\ (m,n,r,3) \text{ otherwise} \\ f((m,n,r,3)) &= (n,r,1) \\ f((n)) &= (n) \end{aligned}$$

Lastly, we redefine this algorithm formally to ensure it is also "effective."

Let $A = \{a, b, c\}$ be our alphabet, and $N = 5$. Given input $a^m b^n$, our algorithm will terminate with $a^{\gcd(m,n)}$. Let θ_j, ϕ_j, a_j, and b_j be specified as follows:

j	θ_j	ϕ_j	a_j	b_j	Note
0	ab	ϵ	2	1	c^r
1	ϵ	c	0	0	
2	a	b	3	2	$b^{\min(m,n)}$
3	c	a	4	3	a^r
4	b	b	5	0	repeat if $r > 0$
5	ϵ	ϵ	5	5	$a^{\gcd(m,n)}$

(Note that here, ϵ denotes the empty string.)

► **9.** [*M30*] Suppose that $C_1 = (Q_1, I_1, \Omega_1, f_1)$ and $C_2 = (Q_2, I_2, \Omega_2, f_2)$ are computational methods. For example, C_1 might stand for Algorithm E as in Eqs. (2) except that m and n are restricted in magnitude, and C_2 might stand for a computer program implementation of Algorithm E. (Thus Q_2 might be the set of all states of the machine, i.e., all possible configurations of its memory and registers; f_2 might be the definition of single machine actions; and I_2 might be the set of initial states, each including the program that determines the greatest common divisor as well as the particular values of m and n.)

Formulate a set-theoretic definition for the concept "C_2 is a representation of C_1" or "C_2 simulates C_1." This is to mean intuitively that any computation sequence of C_1 is mimicked by C_2, except that C_2 might take more steps in which to do the computation and it might retain more information in its states. (We thereby obtain a rigorous interpretation of the statement, "Program X is an implementation of Algorithm Y.")

Let $C_1 = (Q_1, I_1, \Omega_1, f_1)$ and $C_2 = (Q_2, I_2, \Omega_2, f_2)$ be computational methods. We say "C_2 is a representation of C_1" (or alternatively, "$C2$ simulates C_1") when the following properties hold:

1. There exists a function $t_{Q_2} : Q_2 \to Q_1$ that translates every state of C_2 to one of C_1. t_{Q_2} is not necessarily injective or surjective, as C_2 could very well enter additional states of computation not entered by C_1.
2. t_{Q_2} is bijective over $I_2 \to I_1$, such that $t_{Q_2}^{-1}$ is well defined for all $i_1 \in I_1$. If C_2 accepts additional input not accepted by C_1, suitable assignments for such additional input may be established in the defintion of f_2 over I_2.
3. $\forall q_2 \in Q_2 \exists k \in \mathbb{Z}^+ (f_1(t_{Q_2}(q_2)) = t_{Q_2}(f_2^k(q_2)))$, where $f_2^k(q_2) = f_2(q_2)$ if $k = 1$, or $f_2^{k-1}(f_2(q_2))$ otherwise. That is, k is the number of iterations of f_2, and could be specified as a function $t_{f_2} : Q_2 \to \mathbb{Z}^+$.
4. t_{Q_2} is total over $\Omega_2 \to \Omega_1$, such that t_{Q_2} covers the full domain of Ω_2. That is to say, $\forall \omega_2 \in \Omega_2 (\omega_2 \in \Omega_2 \iff t_{Q_2}(\omega_2) \in \Omega_1)$. t_{Q_2} is not necessarily injective or surjective over $\Omega_2 \to \Omega_1$, as C_2 could very well provide additional output not computed by C_1.

This can be demonstrated as follows. Suppose $C_1 = (Q_1, I_1, \Omega_1, f_1)$ where: $I_1 = \{(m, n)\}$; $\Omega_1 = \{(n)\}$; $Q_1 = I_1 \cup \{(m, n, r, 1), (m, n, r, 2), (m, n, p, 3)\} \cup \Omega_1$ for positive integers m, n, p, and nonnegative integer r; and f_1 is defined as follows:

$$\begin{aligned} f_1((m,n)) &= (m, n, 0, 1) \\ f_1((m,n,r,1)) &= (m, n, m \bmod n, 2) \\ f_1((m,n,r,2)) &= (n) \text{ if } r = 0,\ (m, n, r, 3) \text{ otherwise} \\ f_1((m,n,p,3)) &= (n, p, p, 1) \\ f_1((n)) &= (n) \end{aligned}$$

And suppose $C_2 = (Q_2, I_2, \Omega_2, f_2)$ where: $I_2 = \{(m, n)\}$; $\Omega_1 = \{(m, n, d)\}$;
$Q_2 = I_1 \cup \{(m, n, a, b, 1), (m, n, a, b, r, 2), (m, n, a, b, r, 3), (m, n, a, b, r, 4), (m, n, a, b, 5)\} \cup \Omega_2$ for

positive integers a, b, m, n, and nonnegative integer r; and f_2 is defined as follows:

$$\begin{aligned}
f_2((m,n)) &= (m,n,m,n,1)\\
f_2((m,n,a,b,1)) &= (m,n,a,b,a \bmod b,2)\\
f_2((m,n,a,b,r,2)) &= (m,n,b) \text{ if } r=0,\ (m,n,a,b,r,3) \text{ otherwise}\\
f_2((m,n,a,b,r,3)) &= (m,n,b,b,r,4)\\
f_2((m,n,a,b,r,4)) &= (m,n,a,r,5)\\
f_2((m,n,a,b,5)) &= f_2(m,n,a,b,1) = (m,n,a,b,a \bmod b,2)\\
f_2((m,n,d)) &= (m,n,d)
\end{aligned}$$

We then let $t_{Q_2}((m,n)) = (m,n)$ so that $t_{Q_2}^{-1}((m,n)) = (m,n)$, and $t_{Q_2}((m,n,d)) = (d)$. We additionally define:

$$\begin{aligned}
t_{Q_2}((m,n,a,b,1)) &= (a,b,0,1)\\
t_{Q_2}((m,n,a,b,r,2)) &= (a,b,r,2)\\
t_{Q_2}((m,n,a,b,r,3)) &= (m,n,a,b,r,3)\\
t_{Q_2}((m,n,a,b,r,4)) &= (a,b,b,1)\\
t_{Q_2}((m,n,a,b,5)) &= (a,b,b,1)
\end{aligned}$$

and $t_{f_2}((m,n,a,b,r,3)) = t_{f_2}((m,n,a,b,r,4)) = 2$ or $t_{f_2}(q) = 1$ otherwise. Observe that $f_1(t_{Q_2}(q_2)) = t_{Q_2}(f_2^k(q_2))$ as required:

q_2	$t_{Q_2}(q_2)$	$f_1(t_{Q_2}(q_2))$ $= t_{Q_2}(f_2^k(q_2))$	k	$f_2^k(q_2)$	Case
(m,n)	(m,n)	$(m,n,0,1)$	1	$(m,n,m,n,1)$	
$(m,n,a,b,1)$	$(a,b,0,1)$	$(a,b,a \bmod b,2)$	1	$(m,n,a,b,a \bmod b,2)$	
$(m,n,a,b,r,2)$	$(a,b,r,2)$	(b)	1	(m,n,b)	$r=0$
$(m,n,a,b,r,2)$	$(a,b,r,2)$	$(a,b,r,3)$	1	$(m,n,a,b,r,3)$	$r>0$
$(m,n,a,b,r,3)$	$(a,b,r,3)$	$(b,r,r,1)$	2	$(m,n,b,r,5)$	
$(m,n,a,b,r,4)$	$(a,b,b,1)$	$(a,b,a \bmod b,2)$	2	$(m,n,a,b,a \bmod b,2)$	
$(m,n,a,b,r,5)$	$(a,b,b,1)$	$(a,b,a \bmod b,2)$	1	$(m,n,a,b,a \bmod b,2)$	
(m,n,d)	(d)	(d)	1	(m,n,d)	

Hence, "C_2 is a a representation of C_1."

For an alternative approach to simulation, see R. W. Floyd and R. Beigel, *The Language of Machines* (Computer Science Press, 1994), Section 3.3.