Donald Knuth - Art of Computer Programming - Personal Solutions

Morgan Bruce

June 14, 2022

Contents

Chapter 1

Basic Concepts

1.1 Algorithms

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. Try to use the minimum number of replacements.

Solution. Use the sequence $t \leftarrow d$, $d \leftarrow a$, $a \leftarrow b$, $b \leftarrow c$, $c \leftarrow t$. This is the minimum number of replacements since every replacement correclty places a new variable in its spot, except for the assignment of d to a dummy variable, which is required to not lose information of the value of d. □

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

Proof. Suppose $m < n$. Then the remainder of m/n must simply be $r \leftarrow m$, as the quotient is necessarily 0. Applying the next steps of the algorithm, we assign $m \leftarrow n$, $n \leftarrow r$. Thus after one step of the algorithm, $m \geq n$, and it suffices to check that the ordering remains after another step of the algorithm.

Suppose $m \geq n$. Then the remainder of m/n is some integer r such that $0 \leq r < n$. Applying the assignments, we have $m \leftarrow n$, $n \leftarrow r$, and thus since before assignments we had $r < n$, after assignments we have that $n < m$. So the ordering of m and n is preserved at the beginning of step E1 after the algorithm is applied. □

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.

Solution. □

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

Solution. Working through Euclid's algorithm by hand, the remainder of 6099 / 2166 is 1767. The remainder of 2166 / 1767 is 399. The remainder of 1767 / 399 is 171. The remainder of 399 / 171 is 57. The remainder of 171 / 57 is 0. Thus the greatest common divisor of 2166 and 6099 is 57. Using the script at `code/chapter01/euclid.py` confirms this answer. □

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.

Solution. The procedure fails on finiteness, as once step 18 is reached, it loops back to step 3, with no exit condition. The procedure fails on definiteness, as the steps of reading and parsing chapters are very open to what best suits each person (what material to reread, what to skip, which questions to tackle, etc). The procedure fails on output, as it is impossible to measure precisely what a person gains by reading and studying the textbook (we have a vague notion of "improved computer science knowledge," but we can't quantitatively measure this).

Comparing to Algorithm E, the procedure is more loosely defined and is open to modification based on what suits a person's needs. It does not have precise conditions for moving between steps. Additionally, the network is more complex, as it involves moving between points based on personal understanding, rather than a precise step-by-step condition. □

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

Solution. We compute the number of times E1 is computed for $m \in [5]$.

($n = 1$) We have that $1 = 0 \cdot 5 + 1$. Taking $m \leftarrow 5$, $n \leftarrow 1$, we have $5 = 5 \cdot 1 + 0$, taking 2 steps.

($n = 2$) We have that $2 = 0 \cdot 5 + 2$. Taking $m \leftarrow 5$, $n \leftarrow 2$, we have $5 = 2 \cdot 2 + 1$. Taking $m \leftarrow 2$, $n \leftarrow 1$, we have $2 = 2 \cdot 1 + 0$, taking 3 steps.

($n = 3$) We have that $3 = 0 \cdot 5 + 3$. Taking $m \leftarrow 5$, $n \leftarrow 3$, we have $5 = 3 \cdot 1 + 2$. Taking $m \leftarrow 3$, $n \leftarrow 2$, we have $3 = 2 \cdot 1 + 1$. Taking $m \leftarrow 2$, $n \leftarrow 1$, we have $2 = 1 \cdot 2 + 0$, taking 4 steps.

($n = 4$) We have that $4 = 0 \cdot 5 + 4$. Taking $m \leftarrow 5$, $n \leftarrow 4$, we have $5 = 1 \cdot 4 + 1$. Taking $m \leftarrow 4$, $n \leftarrow 1$, we have $4 = 4 \cdot 1 + 0$, taking 3 steps.

($n = 5$) We have that $5 = 1 \cdot 5 + 0$, taking 1 step.

So, we have that $T_5 = (2 + 3 + 4 + 3 + 1)/5 = 2.6$. □

7. [*HM21*] Let U_m be the average number of times that step E1 is executed in Algorithm E, if m is known and n is allowed to range over all positive integers. Show that U_m is well defined. Is U_m in any way related to T_m?

Solution. □

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 $r \leftarrow |m - n|$, $n \leftarrow \min(m, n)$.]

Solution. □

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_1 might be the set of all states of the matchine, 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.")

Solution. □

1.2 Mathematical Preliminaries

1.2.1 Mathematical Induction

1. [*05*] Explain how to modify the idea of proof by mathematical induction, in case we want to prove some statement $P(n)$ for all *nonnegative* integers - that is, for $n = 0, 1, 2, \ldots$ instead of for $n = 1, 2, 3, \ldots$.

Solution. For the initial base case, instead of setting $k \leftarrow 1$, set $k \leftarrow 0$. Then, provide a proof for $P(0)$. From there, continue the induction as normal, except showing that "if all of $P(0)$, ..., $P(k)$ are true, then $P(k+1)$ is true." □

2. [*15*] There must be something wrong with the following proof. What is it?

Theorem. Let a be any positive number. For all positive integers n we have $a^{n-1} = 1$.

Proof. If $n = 1$, $a^{n-1} = a^{1-1} = a^0 = 1$. And by induction, assuming that the theorem is true for $1, 2, \ldots, n$, we have

$$a^{(n+1)-1} = a^n = \frac{a^{n-1} \times a^{n-1}}{b^{n-1}} = \frac{1 \times 1}{1} = 1, \qquad \text{where } b = a^{(n-2)/(n-1)};$$

so the theorem is true for $n + 1$ as well.

Solution. If $b = a^{(n-2)/(n-1)}$, then when proving the case for $n + 1 = 2$, we have a division by zero. Thus the assumption that the theorem is true for $1, 2, \ldots, n$ does not hold □

3. [*18*] The following proof by induction seems correct, but for some reason the equation for $n = 6$ gives $\frac{1}{2} + \frac{1}{6} + \frac{1}{12} + \frac{1}{20} + \frac{1}{30} = \frac{5}{6}$ on the left-hand side, and $\frac{3}{2} - \frac{1}{6} = \frac{4}{3}$ on the right-hand side. Can you find the mistake?

Theorem.

$$\frac{1}{1 \times 2} + \frac{1}{2 \times 3} + \cdots + \frac{1}{(n-1) \times n} = \frac{3}{2} - \frac{1}{n}.$$

Proof. We use induction on n. For $n = 1$, clearly $3/2 - 1/n = 1/(1 \times 2)$; and, assuming that the theorem is true for n,

$$\frac{1}{1 \times 2} + \cdots + \frac{1}{(n-1) \times n} + \frac{1}{n \times (n+1)} = \frac{3}{2} - \frac{1}{n} + \left(\frac{1}{n} - \frac{1}{n+1}\right) = \frac{3}{2} - \frac{1}{n+1}.$$

Solution. The base case of $n = 1$ is not $\frac{1}{1\times2} = \frac{3}{2} - \frac{1}{1}$, as on the left we are assuming $n = 2$ and on the right assuming $n = 1$. It would have to be $\frac{1}{0\times1} = \frac{3}{2} - \frac{1}{1}$, which is clearly invalid, or starting at $n = 2$, it would have to be $\frac{1}{1\times2} = \frac{3}{2} - \frac{1}{2}$, which is clearly not correct. □

4. [*20*] Prove that, in addition to Eq. (3), Fibonacci numbers satisfy $F_n \geq \phi^{n-2}$ for all positive integers n.

Proof. Let $n = 1$, then $1 \geq \phi^{-1} \approx \frac{1}{1.618}$. Let $n = 2$, then $2 \geq \phi^0 = 1$. Suppose $F_n \geq \phi^{n-2}$ for all $1, \ldots, n$. Then

$$\begin{aligned} F_{n+1} &= F_n + F_{n-1} && \text{by definition of Fibonacci numbers} \\ &\geq \phi^{n-2} + \phi^{n-3} && \text{by inductive hypothesis} \\ &= \phi^{n-1} && \text{by (5).} \end{aligned}$$

This proves the inductive case, and thus $F_n \geq \phi^{n-2}$ for all positive integers n. □

5. [*21*] A *prime number* is an integer > 1 that has no positive integer divisors other than 1 and itself. Using this definition and mathematical induction, prove that every integer $n > 1$ may be written as a product of one or more prime numbers, namely in the form $n = p_1 \ldots p_k$ for some $k \geq 1$, where each

of $p_1, \ldots, p_k$ is prime. (A prime number is considered to be the "product" of a single prime, namely itself.)

Proof.