Basic Algorithms — Fall 2020 — Problem Set 4
Due: Nov 4, 11am

Refer to the Number Theory Primer (v0.13) posted on the course home page.

1. Exercise 1.9
2. Exercise 1.10
3. Exercise 1.11 (Prove this using Theorems 1.6 and 1.7)
4. Exercise 1.12 (Prove this using Theorem 1.8)
5. Exercise 2.7
6. Exercise 2.11
7. Exercise 2.25
8. Exercise 2.30
9. Exercise 3.3
10. Exercise 3.4
11. **(Finbonacci!)** ***[Food for thought: will not be graded]*** Recall the Fibonacci numbers: $F_0 = 0$, $F_1 = 1$, and $F_{k+2} = F_k + F_{k+1}$. Your task is to design an algorithm that, given n as input, computes the low-order base-10 digit of F_n. For example, on input $n = 7$, your algorithm should output 3, because $F_7 = 13$, whose low-order digit is 3. On input $n = 120$, your algorithm should output 0, because

$$F_{120} = 53583\,59254\,99096\,664087\,1840,$$

whose low-order digit is 0.

As you can see, F_{120} is very large: in fact, it is an 83-bit number, and so you would get integer overflow (and an incorrect result) if you tried to compute F_{120} directly on a typical 64-bit machine. Indeed, as we saw in Problem Set 1, the value F_n grows exponentially with n, and so this is not surprising. But luckily, you do not need to compute F_n. You only need to compute the low-order digit of F_n.

   (a) Give an algorithm that computes the low-order digit of F_n in time $O(n)$. All variables in your algorithm, including loop indices, should store values that are $O(n)$ in magnitude.

   (b) There is a faster algorithm that runs in time $O(\log n)$, again, using variables that store values that are $O(n)$ in magnitude. Can you find it?
   Hint:
$$\begin{pmatrix} F_{k+1} \\ F_{k+2} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_k \\ F_{k+1} \end{pmatrix}.$$

12. **(Euclid meets Finbonacci)** ***[Food for thought: will not be graded]*** Assume $a \geq b > 0$. In class, we showed that when we execute *Euclid*(a, b), the number of division steps performed by the algorithm is $O(\log b)$. This exercise develops an alternate proof of this fact (with a smaller constant in the big-O).

Let's introduce some notation. Suppose Euclid's algorithm performs ℓ division steps, where $\ell \geq 1$ (since $b > 0$). Let us set $r_{\ell+1} := a$ and $r_\ell := b$, and for $i = 1, \ldots, \ell$, let us define $q_1, \ldots, q_\ell \in \mathbb{Z}$ and $r_0, \ldots, r_{\ell-1} \in \mathbb{Z}$ using division with remainder, as follows:

$$r_{i+1} = r_i q_i + r_{i-1} \qquad (0 \leq r_{i-1} < r_i).$$

Note that these are precisely the division steps performed by Euclid: the first step divides $r_{\ell+1}$ by r_ℓ, obtaining the quotient q_ℓ and the remainder $r_{\ell-1}$, the second step divides r_ℓ by $r_{\ell-1}$, obtaining the quotient $q_{\ell-1}$ and remainder $r_{\ell-2}$, and so on. The last remainder computed is $r_0 = 0$.

Prove that $b \geq F_{\ell+1}$, where F_k is the kth Fibonacci number (recall: $F_0 = 0$, $F_1 = 1$, and $F_{k+2} = F_k + F_{k+1}$). From this, and Problem 7 on PS1, conclude that $\ell \leq \ln(b)/\ln(\phi) + 1$, where $\phi := (1 + \sqrt{5})/2 \approx 1.618$.