CS 812
Spring 2024
Homework #1

Due in class Monday, February 26, 2024

Rules for Homework.

i. You will not be asked to do routine problems. Please get in the habit of working out examples; it is a good way to beef up your notes.

ii. Everyone must do his or her own work. Use of any sources other than class notes and recommended texts must be accompanied by a citation. In any case, there should be significant "value added" by the student's work.

iii. Starred problems have research potential, as their difficulty is unknown (at least to me).

iv. Unless stated otherwise, "complexity" means bit complexity, assuming standard algorithms for basic arithmetic.

1. Traditional school arithmetic courses usually included a pencil-and-paper method for square roots, reminiscent of long division. See the article "Arithmetic," by C.C. MacDuffee, *Encyclopedia Britannica*, 1984. (A link is on our web site, next to this homework.)

   a) Suppose $n > 1$ is an integer. Estimate the number of single-digit operations needed to compute m digits of $\sqrt{n}$ by this method.

   b) As MacDuffee explains, the method can be generalized to cube roots and beyond, but for fifth roots and higher, it is "cumbersome and seldom used." Do you agree? Give cost analysis to confirm or refute MacDuffee's conclusion.

   Historical note: The gloriously named Cyrus Colton MacDuffee was Professor of Mathematics at the University of Wisconsin.

2. Suppose you are given a large positive integer Q, written in binary notation. How would you decide, efficiently, whether Q is nontrivially a binomial coefficient? That is, is $Q = \binom{n}{k}$ for some k, n with $1 < k < n - 1$?

   Continuation: Can you list all the (n, k) pairs in polynomial time?

3. Let p be an odd prime. In class, we showed how to solve $x^2 \equiv a \pmod{p^k}$, using an algorithm that solved a linear congruence at each step. This problem explores an alternative method, for which there is only one inversion, at the very end.

   a) Find the Newton iteration for solving $f(x) = 0$, where $f(x) = a - 1/x^2$. Note that it has the form
   $$x' = x - P(a, x), \tag{1}$$
   where P is a polynomial in $\mathbf{Q}[A, X]$, with coefficient denominators prime to p.

   b) Prove quadratic convergence: if $ax^2 \equiv 1 \pmod{p^i}$ then $a(x')^2 \equiv 1 \pmod{p^{2i}}$. Thus, when a is a square mod p^k, we can find $\sqrt{a}$ by doing $\log k$ iterations of (1) and inverting the result.

   c) (*) We have two examples of rational functions for which the Newton iteration function is a polynomial. (The other one is $a - 1/x$, which we used to reduce reciprocal to multiplication.) Can you characterize all such functions?

4. In his work on undecidable propositions of arithmetic, Kurt Gödel used the following lemma to encode a sequence of natural numbers as a pair of numbers:

   Let
   $$\beta(x_1, x_2, x_3) = x_1 \bmod x_2(x_3 + 1) + 1.$$
   Then for any sequence $a_0, a_1, \ldots, a_{k-1}$ of natural numbers, there are b, c for which
   $$\beta(b, c, i) = a_i, i = 0, \ldots, k - 1$$
   Gödel's proof: Take $\ell = \max\{k, a_1, \ldots, a_{k-1}\}$ and then $c = \ell!$. The existence of b is a consequence of the Chinese remainder theorem, since the moduli $cj + 1$ are pairwise relatively prime for $j = 1, \ldots, k$.

   a) Find the bit complexity of the "decoding" function β.

   b) Suppose that $a_i \leq A$, $i = 0, \ldots, k - 1$. As a function of k and A, how many bits are needed for the pair (b, c)? Is Gödel's encoding function, which takes $a_0, \ldots, a_{k-1}$ and produces the pair (b, c), computable in polynomial time?

   c) (*) Investigate the complexity of alternative sequence representation methods. This is rather open-ended, so here are some ideas. Are there other ways to make pairwise relatively prime moduli that don't already rely on sequences? (This would rule out using the first k primes as moduli.) Another direction is to study the alternative to the β function given by R. Smullyan, Theory of Formal Systems, p. 71.

   Another interesting question is the extent to which Gödel's sequence representation is uniquely decodable. That is, if I know (b, c), do I know the a_i's? Related to this is the following: Suppose I am given
   $$\prod_{j=1}^{k} (j\ \ell! + 1).$$
   Can I recover k and ℓ?