

Homework 5

Recall the Pollard “tail-chasing” factoring algorithm that we discussed in the lecture:

First be sure that n is not prime or a power of a prime.

To find a factor of n , choose random congruence classes a and $x_0 \pmod{n}$, set $y_0 = x_0$, and then for $i = 1, 2, \dots$ compute

$$x_i = f(x_{i-1}) \pmod{n}, \quad y_i = f(f(y_{i-1})) \pmod{n},$$

where $f(x) = x^2 + a$. Repeat until

$$\gcd(y_i - x_i, n) \neq 1,$$

in which case the gcd is a factor of n . If you eventually get $y_i \equiv x_i \pmod{n}$, or if it takes too many rounds to find an answer, try again with a different a and x_0 . “Too many” means more than $C \cdot \sqrt[4]{n}$ for a modest constant C , such as $C = 3$.

1. (a) Use Pollard’s algorithm to find a factor p of $n = 5083$ (it is best to use a calculator for this).

(b) Define $q = n/p$, so $n = pq$. Use Miller’s test (with a couple of tries if the number passes) to decide whether or not each of p and q is prime. If one or both is not prime, use Pollard again to factor further, continuing until you find the complete prime factorization of 5083.

Remark: for a number as small as 5083, factoring by trial division might be easier, but the point of the assignment is to make sure you understand how to use the Pollard algorithm.

2. Pollard’s method is based on assuming that n has a factorization $n = pq$ with p relatively prime to q , so the Chinese remainder theorem applies. Prove that every n that is not prime or a power of a prime has such a factorization.

3. (a) Devise an algorithm to test whether n is the k -th power of an integer, using $O(\log n)$ arithmetic operations (on integers with approximately the same number of digits as n or fewer). Hint: use a binary search for the alleged k -th root of n . What is the maximum number of digits the k -th root could have, given the number of digits of n ?

(b) To rule out the possibility that n is a power, we can use the algorithm in Problem 3 for $k = 2, 3, \dots$. How large is the largest k that needs to be tried? Give a big- O estimate for the total number of arithmetic operations needed to check that n is not a power of a smaller integer.

Chapter 2.7: Problems 3(b), 14, 17, 24(a)