ECE374 SP23 HW8

Contributors

Zhirong Chen (zhirong4)

Ziyuan Chen (ziyuanc3)

Problem 2

Until 2002, it was an open question whether the decision problem "Is integer n a composite number, in other words, not prime?" could be computed in time polynomial in the size of its input. Why doesn't the following algorithm suffice to prove it is in P, since it runs in $O(n)$ time?

$\text{PrimalityTesting}(n)$
    $\text{composite} \leftarrow \textbf{false}$
    for $i \leftarrow 2$ to $n-1$
        if $n \bmod i = 0$
            $\text{composite} \leftarrow \textbf{true}$
    return composite

Solution

This is more or less a word game. The algorithm, despite its correctness, is not polynomial in the ***size*** of its input -- in other words, the ***number of digits*** (in any base a).

We are looking for an algorithm that runs in $O(s)$ time for an input of size $s = \log n$, but our intuitive PrimalityTesting algorithm actually has $O(n) = O(a^s)$ complexity!

As pointed out by the well-known paper on *the AKS Primality Test* (that gives an authentic polynomial algorithm),

> Let PRIMES denote the set of all prime numbers. The definition of prime numbers already gives a way of determining if a number n is in PRIMES: try dividing n by every number $m \leq \sqrt{n}$ -- if any m divides n then it is composite, otherwise it is prime.
> .....The test, however, is inefficient: it takes $\Omega(\sqrt{n})$ steps to determine if n is prime. An efficient test should need only a polynomial (in the ***size*** of the input = $\lceil \log n \rceil$) number of steps.

Agrawal, M.; Kayal, N.; and Saxena, N. "Primes is in P." *Ann. Math*. **160**, 781-793, 2004.