CS 70 Discrete Mathematics for CS
Spring 2008 David Wagner

# HW 7

## Due Thursday, March 13th

**1. (5 pts.) Practice with erasure codes**
Alice sends you a message using an erasure code using the modulus $q = 7$. You know that her original message consists of $n = 3$ packets $m_1, m_2, m_3$ such that $P(i) \equiv m_i \pmod 7$. She sent you the encoded packets $c_1, \ldots, c_7$ given by $c_i \equiv P(i) \pmod 7$, but unfortunately only $c_1$, $c_2$, and $c_7$ arrived ($c_3$, $c_4$, $c_5$ and $c_6$ were lost). In particular, you receive $c_1 = 6$, $c_2 = 6$, $c_7 = 3$. Find the polynomial $P(x)$ and the original message $m_1, m_2, m_3$.

*Hint:* You shouldn't have to do any difficult calculations to solve this; you should already have done all the calculations that are needed for this on a prior homework.

**2. (15 pts.) Guess the polynomial**
Luqman challenges Min to think of a polynomial in one variable—$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$—whose coefficients are all nonnegative integers. Luqman claims that he will ask for just two evaluations of the polynomial, and then he will tell Min its coefficients. Here's how.

1. Luqman asks Min to evaluate $P(1)$. He calls that value $r$.
2. Luqman asks Min to evaluate $P(r+1)$. After receiving the answer $y$ and giving it some thought, Luqman reads off the coefficients of $P(x)$.

Here's an example. Luqman asks for $P(1)$, and Min says 9. Luqman then asks for $P(10)$. Min says 342. Luqman correctly identifies $P(x)$ as $3x^2 + 4x + 2$.

(a) Luqman asks for $P(1)$, and Min says 15. Luqman asks for $P(16)$, and Min says 450. What's $P(x)$?

(b) Describe a general method that Luqman could use to recover the polynomial $P(x)$ from Min's two answers $r$ and $y$. Your method should be reasonably efficient.

(c) Demonstrate that knowing $r = P(1)$ and $y = P(r)$ does *not* provide you with enough information to uniquely determine $P(x)$.

(d) David would like to replicate the trick, but he is impatient—he does not want to have to wait for Min's first answer before choosing his second query to Min. In other words, David would like to find some $z \in \mathbb{N}$ such that he can ask Min for $P(1)$ and $P(z)$, and this will be enough information to uniquely determine $P(x)$. Can he do it? Why or why not?

**3. (10 pts.) Fingerprinting without polynomials**
In class you saw one way to compute a fast fingerprint on a $n$-bit message $s$, based on expressing the message $s$ as a polynomial and evaluating that polynomial at a random point.

In this problem we will examine an alternative approach to fingerprinting. Given a $n$-bit message $s$, we consider $s$ as a number in the range $0 \leq s < 2^n$. Let $B = 2^{2n+100}$. We pick a random number $r$; however, this time $r$ is chosen at random from the set of all prime numbers less than $B$. Finally, the fingerprint is $G(s,r) = s \bmod r$. In other words, the fingerprint of a message is obtained by picking a random prime $r$ (of appropriate size), reducing the message modulo $r$, and using the result as our fingerprint.

(a) Show that if $s, s'$ are two different $n$-bit messages with $s \neq s'$, then $G(s,r) = G(s',r)$ happens with probability at most $1/2^{100}$, for sufficiently large values of $n$.

*Hint:* Count the number of unlucky values of $r$. You may use the following theorem from number theory, without proving it yourself:

**Fact:** For sufficiently large $n$, there are at least $2^{n+100}$ different prime numbers less than $B$.

(b) An antivirus company proposes that we use this idea to protect files on our hard disk against attack from viruses. We'll pick a single random prime $r$, and store $r$ secretly in a safe place. Then, for each of the files $s_1, \ldots, s_k$ on our hard disk, we'll store their fingerprints $G(s_1,r), \ldots, G(s_k,r)$ on the hard disk as well. When the computer boots up, the first thing it will do is check the files on the hard disk against their stored fingerprints using its secret prime $r$.

Assume a virus can examine and modify everything stored on the hard disk. Assume that the secret prime $r$ is stored in a safe place where no virus can learn it (e.g., on a smartcard). The evil virus writer would like try to inject specially constructed 'errors' into our files, with the hope of avoiding detection. The antivirus company hopes the fingerprinting scheme will be secure against this sort of attack.

What do you think of the antivirus company's proposal?

**4. (20 pts.) String matching**

In this problem, we use the polynomial-based fingerprinting method outlined in class. Let $q$ be prime. Given a $n$-bit bitstring $s = \langle s_0, s_1, \ldots, s_{n-1} \rangle$, define the polynomial $\tilde{s}(x) = s_{n-1}x^{n-1} + \cdots + s_1 x + s_0$. Then, pick a random value $r$ from the set $\{0,1,2,\ldots,q-1\}$ and compute $\tilde{s}(r) \bmod q$. In other words, the fingerprint of the bitstring $s$ is given by $F(s,r) = \tilde{s}(r) \bmod q$.

Let $n, m$ be large natural numbers with $n < m$. Suppose we are given a $m$-bit bitstring $t = \langle t_0, t_1, \ldots, t_{m-1} \rangle$. Let $t@i$ denote the $n$-bit substring $\langle t_i, t_{i+1}, \ldots, t_{i+n-1} \rangle$ of $t$ that starts at position $i$. For example, if $m = 6$, $n = 4$, and $t = \langle 0,1,1,0,1,0 \rangle$, then $t@0 = \langle 0,1,1,0 \rangle$ and $t@1 = \langle 1,1,0,1 \rangle$.

(a) Show how to efficiently compute $F(t@i, r)$ from $F(t@(i+1), r)$.

(b) Suppose we are given a $m$-bit string $t$ and a $n$-bit string $u$, and we want to test whether $u$ appears as a substring of $t$. Consider the following naive algorithm:

```
NaiveStringMatch($t,u$):
1. For $i = m - n$ down to 0, do:
2.     If $t@i = u$, then return $i$.
3. Return "No match."
```

Argue that this naive algorithm has a $O(mn)$ worst-case running time, if we count each bit operation as one unit of time.

(c) Consider the following algorithm (sketch):

```
FasterStringMatch($t,u$):
0. Pick a random $r$ from $\{0,1,\ldots,q-1\}$. Compute $F(u,r)$ and $F(t@(m-n),r)$.
1. For $i = m - n$ down to 0, do:
2.     If $F(t@i,r) = F(u,r)$, then do:
3.         If $t@i = u$, then return $i$.
4. Return "No match."
```

(Here $q$ is a prime that is hard-coded into the algorithm and chosen to be large enough so that the probability of a false match in the fingerprint is sufficiently small.)

Describe how to implement FasterStringMatch() so that, if we ignore the cost of executing step 3, the total running time will be $O(m(\lg q)^2)$ bit operations.

(d) We say that there is a “false match” in the $i$-th iteration of the loop if FASTERSTRINGMATCH$()$ executes step 3 but doesn’t immediately return (i.e., because $F(t@i,r) = F(u,r)$ but $t@i \neq u$). If false matches are rare, then intuitively the cost of executing step 3 will be negligible in practice compared to the overall cost, so we’d like to choose the prime $q$ to make false matches rare.

How large must $q$ be to ensure that the chances of a false match in any one iteration of the loop will be at most $1/10^6$? Your answer may depend upon $n$.

**5. (0+5 pts.) Optional bonus problem (more challenging!)**
(This is an *optional* problem, worth 5 points of extra credit, for those who like a more difficult challenge.)

Suppose that the poker site from HW6 Question 1 catches on to your tricks, and decides to change the modulus $m$ to some secret value. Now you do not know $m$, $a$, or $b$, but you do know, say, $x_0, \ldots, x_{99}$. Describe an efficient procedure that will have a good chance at predicting future values from the generator (i.e., predicting $x_{100}, x_{101}, \ldots$).