- This problemset has *six* questions.
- To get the credit for questions marked as SPOJ, you must get them accepted on `http://www.spoj.com/AOS`, but you **don't** have to send any explanation!
- For other questions, either send the solutions to `gawry1+aos@gmail.com`, or leave them in the envelope attached to the doors of my office (room 321).

1. Prove the following: a polynomial $f(x)$ of degree d over the integers modulo a prime p cannot have more than d different roots. For instance, $f(x) = x^3 + 2x + 2$ has just two roots $\{2, 3\}$ modulo 7.

> **Solution:** See `http://en.wikipedia.org/wiki/Lagrange's_theorem_(number_theory)`.

(SPOJ) 2. Given q and r, compute the signature (as defined during the lecture) of a string S. The string consists of letters `a` and `b`, and we treat them as digits 0 and 1, i.e., the signature of a string `baba` is $(r^3 + r) \pmod q$.

> **Solution:** The simplest way is to use Horner's rule. Start with the fingerprint of the empty string (0), and then append letters one-by-one updating the current fingerprint.

(SPOJ) 3. We consider a different scheme for computing the signatures $\phi(S[1..n]) = (S[1] * r^{m-1} \texttt{ xor } S[2] * r^{m-2} \texttt{ xor } \dots \texttt{ xor } S[n] * r^0) \bmod 2^{32}$. The string consists of letters `a` and `b`, and we again treat them as digits 0 and 1. Given r and a string S, find a different S' with exactly the same signature.

> **Solution:** Let the binary expansion of r^{m-i+1} be $v_i = \langle b_i(31), b_i(30), \dots, b_i(0)\rangle$. If you look at the condition that the fingerprint of S and S' should be the same, it makes sense to define $d_i \in \{0, 1\}$ which tells us whether $S[i] = S'[i]$ or not. Then the condition that $S \neq S'$ becomes $d_i \neq 0$ for some i, and the condition that the fingerprints are the same translates into a system of 32 linear equations. The j-th equation is of the form $d_1 b_1(j) + d_2 b_2(j) + d_3 b_3(j) + \dots d_m b_m(j) = 0 \bmod 0$. So, we have 32 linear equations over m variables. Because the equations, and we are operating modulo 2 (which is a field), we can use Gauss elimination (see `http://en.wikipedia.org/wiki/Gaussian_elimination`). This might be a little bit too slow, but one can observe that we never need to use more than 33 variables: the rest can be just set to 0. This is because we are really looking for a dependent set of vectors in a vector space of dimension 32 (if you don't know what a vector space is, just ignore this sentence). So, construct the system of 32 equations in at most 33 variables $d_1, d_2, \dots, d_{33}$, and solve it using Gauss elimination.

4. Compute the values of the π function for the word ababbabaabaab.

> **Solution:** The values are (starting from $i = 1, 2, 3, \ldots, 13$): $0, 0, 1, 2, 0, 1, 2, 3, 1, 2, 3, 1, 2$.

5. Consider a modification of the failure function π known as the strong failure function π'. It is defined as follows: for each $i = 1, 2, \ldots, |w| - 1$ we choose $\pi'[i]$ to be the longest proper border of $w[1..i]$ such that $w[\pi'[i] + 1] \neq w[i + 1]$. If there is no such border, $\pi'[i] = -1$.

   (a) Compute the values of the π' function for the word ababbabaabaab.

   > **Solution:**
   > The values are (starting from $i = 1, 2, 3, \ldots, 12$): $0, -1, 0, 2, -1, 0, -1, 3, 0, -1, 3, 0$.

   (b) Show how to (quickly) compute the values of π' given the values of π.

   > **Solution:**
   >
   > COMPUTE-π'-FROM-$\pi(\pi)$
   >
   > $\pi'[0] \leftarrow -1$
   > **for** $i \leftarrow 1$ **to** $n - 1$
   >   **do if** $\pi[i + 1] = \pi[i] + 1$
   >   **then** $\pi'[i] \leftarrow \pi'[\pi[i]]$
   >   **else** $\pi'[i] \leftarrow \pi[i]$

6. Consider a simplification of the Boyer-Moore algorithm, where we use only the bad character rule. Show an infinite family of instances on which such modification has quadratic running time.

> **Solution:** Many solutions are possible. The simplest seems to be $t =$ aaa...aaa and $p =$ baaa...aaaa. Notice that choosing $p =$ aaa...aaa would work if we were interested in generating all occurrences, but if we want just the leftmost, it's better to make sure that p doesn't occur in t.