

$$r^{r-\ell} \prod_{j=0}^{\ell} (r-j),$$

and the proportion of pairs having the stated property (i.e., the above number divided by r^{r+1}) is

$$r^{-\ell-1} \prod_{j=0}^{\ell} (r-j) = \prod_{j=1}^{\ell} \left(1 - \frac{j}{r}\right).$$

The proposition states that the log of this is less than $-\lambda$ (where $\ell = 1 + [\sqrt{2\lambda r}]$). To prove the proposition, then, we take the log of the product on the right, and use the fact that $\log(1-x) < -x$ for $0 < x < 1$ (geometrically, this is simply the fact that the logarithm curve remains under the line which is tangent to it at the point $(1, 0)$). Using the formula for the sum of the first ℓ integers, we have:

$$\log\left(\prod_{j=1}^{\ell} \left(1 - \frac{j}{r}\right)\right) < \sum_{j=1}^{\ell} -\frac{j}{r} = \frac{-\ell(\ell+1)}{2r} < \frac{-\ell^2}{2r} < \frac{-(\sqrt{2\lambda r})^2}{2r} = -\lambda,$$

as required. This completes the proof of the proposition.

The significance of Proposition V.2.1 is that it gives an estimate for the probable length of time of the rho method, *provided that* we assume that our polynomial behaves like an average map from $\mathbf{Z}/r\mathbf{Z}$ to itself. Before explaining this estimate, we make a slight refinement of the rho method in the interest of efficiency.

Recall that the rho method works by successively computing $x_k = f(x_{k-1})$ and comparing x_k with the earlier x_j until we find a pair satisfying $\text{g.c.d.}(x_k - x_j, n) = r > 1$. But as k becomes large, it becomes very time-consuming to have to compute $\text{g.c.d.}(x_k - x_j, n)$ for each $j < k$. We now describe a way to carry out the algorithm so as to make only one g.c.d. computation for each k . First, observe that, once there is a k_0 and j_0 such that $x_{k_0} \equiv x_{j_0} \pmod{r}$ for some divisor $r|n$, we then have the same relation $x_k \equiv x_j \pmod{r}$ for any pair of indices j, k having the same difference $k - j = k_0 - j_0$. To see this, simply set $k = k_0 + m$, $j = j_0 + m$, and apply the polynomial f to both sides of the congruence $x_{k_0} \equiv x_{j_0} \pmod{r}$ repeatedly, i.e., m times.

We now describe how the rho algorithm works. We successively compute the x_k , and for each k we proceed as follows. Suppose k is an $(h+1)$ -bit integer, i.e., $2^h \leq k < 2^{h+1}$. Let j be the largest h -bit integer: $j = 2^h - 1$. We compare x_k with this particular x_j , i.e., we compute $\text{g.c.d.}(x_k - x_j, n)$. If this g.c.d. gives a nontrivial factor of n , we stop; otherwise we move on to $k + 1$.

This modified approach has the advantage that we compute only one g.c.d. for each k . It has the disadvantage that we probably will not detect the first time there is a k_0 such that $\text{g.c.d.}(x_{k_0} - x_{j_0}, n) = r > 1$ for some $j_0 < k_0$.