Algorithm Design 21/22

Hands On 1 - Universal Hash Family

Federico Ramacciotti

1 Problem

Prove that the family $\mathcal{H}$ of functions is universal, given $m > 1$ and $p \in [m+1, 2m)$ prime, $\mathcal{H} = \{h_{ab} = ((ax+b) \mod p) \mod m, a \in [1, p-1], b \in [0, p-1]\}$ that is, for any $k_1 \neq k_2$, it holds that $|\{h \in H \; s.t. \; h(k_1) = h(k_2)\}| = \frac{|\mathcal{H}|}{m}$.
Hint: consider $r = (ak_1 + b) \mod p$ and $s = (ak_2 + b) \mod p$ where $k_1, k_2 \in [0, p-1]$.

2 Solution

Consider k_1, k_2 distinct keys in $[0, p-1]$. Given an hash function $h \in \mathcal{H}$, define

$$r = (ak_1 + b) \mod p$$

$$s = (ak_2 + b) \mod p$$

We can see that $r - s \equiv a(k_1 - k_2) \mod p$ and therefore, since p is prime and $k_1 \neq k_2$, we get that $r \neq s$. This implies that, computing any hash function $h \in \mathcal{H}$, distinct inputs k_1, k_2 give distinct values of r and s modulo p. Moreover, any $p(p-1)$ choice for the pair (a, b) generates a distinct pair (r, s) with $r \neq s$.

To show this, derive a and b from r and s:

$$\begin{aligned} b &= (r - ak_1) \mod p \\ a &= (r - s)(k_2 - k_1)^{-1} \mod p \end{aligned}$$

Since there are $p(p-1)$ possible pairs (r, s) with $r \neq s$, we can see that there is a one-to-one correspondence between the pairs (a, b) and (r, s).

Therefore, all this implies that the probability of $k_1 = k_2$ is equal to the probability that $r \equiv s \mod m$ with r, s randomly chosen distinct values modulo p:

$$Pr[k_1 = k_2] = Pr[r \equiv s \mod m]$$

For a given r we have $p-1$ values to choose s and the number of values such that $s \equiv r \mod m$ is $\frac{p-1}{m}$. This means that, given $|\mathcal{H}| = p(p-1)$, the number of bad hash functions (i.e. the ones that give collisions), is $p\frac{p-1}{m} = \frac{|\mathcal{H}|}{m}$ and therefore

$$Pr\big[\{h \in \mathcal{H} \mid h(k_1) = h(k_2)\}\big] = \frac{\text{\# bad choices}}{\text{\# all choices}} = \frac{\frac{|H|}{m}}{|H|} = \frac{1}{m}$$

So, we have proven that $\mathcal{H}$ is a Universal Hash Family.