Exercise 2, Discrete Mathematics for Bioinformatics

Sascha Meiers, Martin Seeger

Winter term 2011/2012

2.1 Modulo Arithmetic

a) We show that $\langle a \rangle \subset \langle d \rangle$.

Since $d = \gcd(a, n)$, there is a $k \in \mathbb{N}$ such that $a = kd$. Hence, if $v \in \langle a \rangle$, i.e. $v = ai \mod n$, then $v = dki \mod n$ which implies that $v \in \langle d \rangle$.

b) We show that $\langle a \rangle \supset \langle d \rangle$.

Any element v of $\langle d \rangle$ can be written as $v = di \mod n$ (*). On the other hand, $v \in \langle a \rangle$ iff $v = aj \mod n$.

We now use Bezout's lemma to find x, y, such that $ax + ny = d$. This is inserted into (*) to yield

$$v = di \mod n = (ax + ny)i \mod n = axi \mod n.$$

In other words, $v \in \langle a \rangle$. □

2.2 Hashing

Let x, y be character strings both of length n. Now we can interprete their characters as numbers in radix 2^p, leading to a hash function

$$h(x) = \sum_{i=0}^{n} x_i 2^{p \cdot i} \bmod 2^p - 1$$

If y is nothing else than a permutation of the characters in x, then especially their sum of the digits is equal, i.e.

$$\sum_{i=0}^{n} x_i = \sum_{i=0}^{n} y_i$$

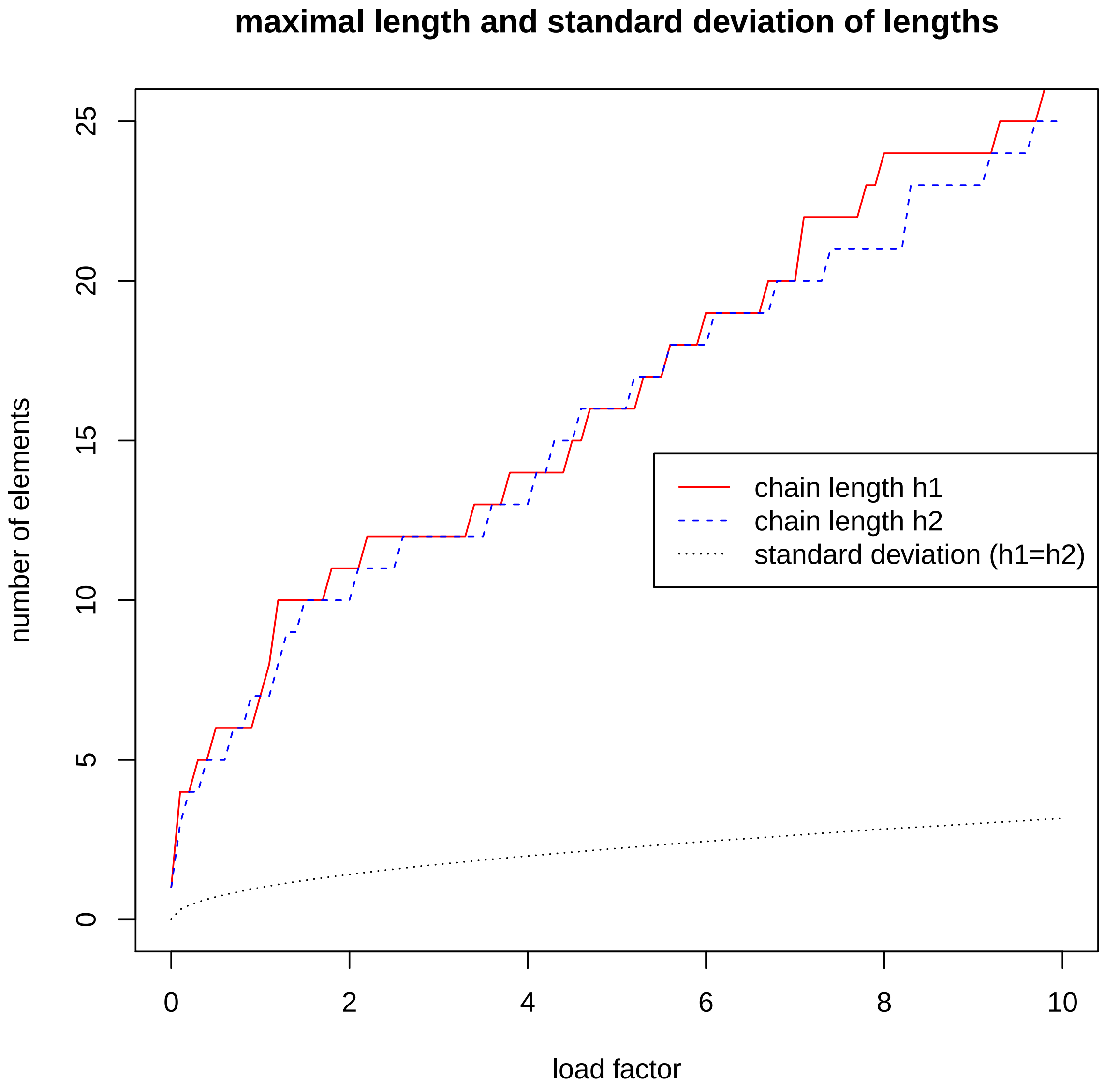
Proof: $h(x) = h(y)$

$$\begin{aligned}
h(x) &= \sum_{i=0}^{n} x_i 2^{p\cdot i} \bmod 2^p - 1 && (1)\\
&= \sum_{i=0}^{n} \big(x_i 2^{p\cdot i} \bmod 2^p - 1\big) \bmod 2^p - 1 && (2)\\
&= \sum_{i=0}^{n} (x_i \bmod 2^p - 1) \left(\underbrace{2^p \bmod 2^p - 1}_{1}\right)^i \bmod 2^p - 1 && (3)\\
&= \sum_{i=0}^{n} x_i \bmod 2^p - 1 && (4)\\
&= \sum_{i=0}^{n} y_i \bmod 2^p - 1 && (5)\\
&= h(y) && (6)
\end{aligned}$$

2.3 Hashing

We choose two different hash functions h_1 and h_2 and simulate hashing with chaining for random integer variables. We take a fixed table size of $p = 48049$, which is a prime number, and analyze the distribution of the inserted elements in the hash table. Therefore we measure the maximum chain length as well as the standard deviation of the chains' lengths for load factors from zero up to ten. The average chain length is equal to the load factor. The figures show the measured values for both hash functions

$$h_1(x) = x \bmod p \qquad h_2(x) = p\left(\frac{\sqrt{5}-1}{2}x \bmod 1\right)$$



Obviously both hash functions distribute both with nearly the same quality. The maximum length is usually more than twice as large as the load factor, which is the lower bound. Of course these results mainly depend on the input data, which here comes from Python's random command (uniformely distributed numbers $\in \{0, .., 2^{20}\}$)

2.4 Expected value