

The math of CRC functions

Source: <http://sci.tech-archive.net/Archive/sci.math/2006-03/msg00043.html>

- *From:* Peter Seibel <peter@xxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 01 Mar 2006 05:53:52 GMT
-

I'm writing some software that needs to compute hashes of strings of text such that the chance of hash collisions is as small as possible while the hashes can be computed as (time) efficiently as possible. I'm thinking CRC functions are a good answer. However I don't understand the mathematics behind CRCs well enough to be able to figure certain things out. (I did find what seems a good tutorial on some of the basics at[1].) I'm hoping someone here will have a few minutes to help me out or point me to something more I can read about the math behind CRCs that doesn't require first obtaining a degree in mathematics.

For starters, under what circumstances will a CRC function return uniformly distributed results? From what I understand if the inputs given are uniformly distributed so will the outputs. But what if, for instance, the inputs are all strings of ASCII text and thus the distribution of bytes making up the string are not uniform?

How does the choice of polynomial affect things? I understand that you need to select the polynomial carefully in order to ensure certain characteristics such as the ability to detect all 1-bit errors and so on. But what if all I care about is minimizing the chance of hash collisions? I'm thinking I'm going to need to pick my own polynomial instead of just using one of the standard ones because I may need to use a non-standard size CRC: CRC-48, CRC-96, or CRC-128 and haven't been able to find any standard algorithms whose polys I can nick. Also, I don't even know enough to know whether a polynomial chosen for its good error-detecting characteristics is necessarily going to be good for what I need which is maximizing the uniformity of the distribution of hash values when hashing ASCII text.

Finally, is there anything to be said about this strategy: because we're worried about a 32-bit CRC having too many collisions (c.f. the birthday paradox) one of my co-developers suggested that we compute the CRC-32 of each piece of text in the normal way and then again on the string reversed. If these two hash values are completely independent (and uniformly distributed) then I suppose that gives us 64-bits hash goodness with a correspondingly lower probability of collision. However my very vague intuition is that that's just hinky and there's likely some subtle mathematical reason why the two hashes

The math of CRC functions

won't be completely independent and thus we will have less than 64-bits worth of hash. But proving it one way or the other is far beyond my mathematical abilities.

Thanks in advance for any help you can give--hopefully these are at least well-formed questions and maybe some of them are even intriguing; it's been a long time since I was considered good at math so I have no way of knowing.

-Peter

[1] <http://www.ross.net/crc/download/crc_v3.txt>

--

Peter Seibel * peter@xxxxxxxxxxxxxxxx

Gigamonkeys Consulting * <http://www.gigamonkeys.com/>

Practical Common Lisp * <http://www.gigamonkeys.com/book/>

.