Enumerating bit sequences with isolated zero-bits via the Fibonacci recurrence

devblogs.microsoft.com/oldnewthing/20140512-00

May 12, 2014

Raymond Chen

Today's Little Program enumerates bit sequences of a particular length subject to the constraint that you cannot have consecutive 0-bits. This may sound kind of arbitrary, but it is important in magnetic media storage, because you cannot go too long without a flux reversal (traditionally represented by a 1); otherwise, the read head's clock starts to drift and gets out of sync with the data. (The read head uses flux reversals both for signaling and for clock synchronization.)

Let's say that an *allowable* bit sequence is one that contains no consecutive 0-bits.

The recurrence for enumerating these types of constrained bit sequence is the Fibonacci recurrence:

> $F(n) = F(n-1) + F(n-2)$

The way to see this is to study the last digit in an allowable bit sequence.

If the last digit is a 1, then if you delete it, you have an allowable bit sequence that is one digit shorter. Conversely, you can take any allowable bit sequence of length $n - 1$ and tack a 1 onto it, and you'll have an allowable sequence.

If the last digit is a 0, then if you delete it, you also have an allowable bit sequence that is one digit shorter, but not all allowable bit sequences of length $n - 1$ can be reached in this way. Allowable bit sequences that end in 0 cannot be reached because adding another 0 would result in two consecutive 0-bits, which is disallowed. Therefore, the last digit of the truncated bit sequence must be 1, and what you really have is an allowable bit sequence of length $n - 2$, followed by a hard-coded 1.

Okay, now that we understand the enumerative justification for the recurrence, we can proceed to write the code that generates it.

```
function GCR(n, f) {
 if (n == 0) { f(""); return; }
 if (n < 0) { return; }
 GCR(n-1, function(s) { f(s + "1"); });
 GCR(n-2, function(s) { f(s + "10"); });
}
GCR(8, console.log);
```

The test run calculates all 8-bit allowable sequences. But wait, there's a bug: It shows only the sequences that begin with a 1.

If you're Steve Wozniak, then this bug is a feature, because the Apple II floppy drive also required the first bit to be set, so our bug exactly matches the hardware requirements.

But let's fix our bug. Where did it come from?

Our recursive step missed a base case: The single-digit bit sequence 0 is allowable, but we rejected it because we thought that it needed to be separated from the null string by a 1, to protect against the null string ending in 0. But the null string doesn't end in zero, so this protection was unnecessary.

Repairing our base case:

```
function GCR(n, f) {
 if (n == 0) { f(""); return; }
 if (n == 1) { f("0"); f("1"); return; }
 GCR(n-1, function(s) { f(s + "1"); });
 GCR(n-2, function(s) { f(s + "10"); });
}
```

Raymond Chen

Follow