Code Jam 2008 - AMER Semifinal

Analysis: Code Sequence

The Problem

Normally when you count in base 2, each bit is worth a fixed amount: {..., 8, 4, 2, 1}. So when you count 0000, 0001, 0010, 0011, and so on, you get the sequence [0, 1, 2, 3, ...]. In this problem, we ask you to consider what would happen if the bits were worth hidden amounts, "bit values," specified by us. For example, if the bits were worth {..., 200, 10, 1}, then you could get the sequence [0, 1, 10, 11, 200, 201, ...].

What this problem asks you to do is find the next number in the generated sequence. To make things more difficult, we don't necessarily start at 0000 or 0001, so what may seem obvious:

```
[1, 10, 11, 200, 201, ?, ...]
```

...may not be. This could have come from the bit values {..., 200, 10, 1}, which would give **210**; but it could also have come from bit values {..., -1180, 1190, 990, 190, 1}. If we start counting from 10000, that gives:

```
[-1180, -1179, -990, -989, -190, -189, 0, 1, 10, 11, 200, 201, 1000]
```

The Solution

So how do we solve the problem? Let's have a look at the generated sequence above, and look for interesting features. One thing that may strike you is that every second number is 1 higher than the last, so let's look at the sequence of differences between adjacent numbers:

```
(1, 189, 1, 799, 1, 189, 1, 9, 1, 189, 1, 799)
```

The differences of 1 make a lot of sense: every *second* increase in a binary number just changes the lowest bit from 0 to 1. So if your sequence of differences looks like (x, a, x, b), or (a, x, b), then the next difference must be x. Figuring that out should let you solve the small input.

But how far can we take this? Let's eliminate all the times only the lowest bit was changed. This leaves us with the following sequence of differences:

```
(189, 799, 189, 9, 189, 799)
```

Every second number there is a 189. That makes sense too: every *fourth* increase in a binary number is the same, changing the lowest two bits from 01 to 10. Likewise, every *eighth* increase changes the lowest three bits from 011 to 100.

So where does this leave us? Looking for repetition; seeing if we can use it; and if not, eliminating it. For example, if our generated sequence is:

```
[0, 1, 3, 4, 7, 8, 17, 18, 21, 22, 24, 25]
```

Our sequence of differences is:

```
(1, 2, 1, 3, 1, 9, 1, 3, 1, 2, 1)
```

Since the sequence ends with the repeated number, 1, we don't know yet what comes next. Eliminating the 1s, we arrive at:

```
(2, 3, 9, 3, 2)
```

Aha! The next difference must be 3, and the answer is 28. If that hadn't worked, we would have kept going recursively until we found the answer.

But what if the sequence of differences is:

```
(1, 2, 1, 2, 1)
```

Here, we can't tell if the lowest bit was changing for a difference of 1 or a difference of 2, so we can't say for sure what's next. We (eventually) arrive at the same problem in all of the following sequences of differences:

```
(1, 2)
(3)
(4, 3, 4)
(5, 4, 5, 3, 5, 4, 5)
(6, 5, 6, 4, 6, 5, 6, 3, 6, 5, 6, 4, 6, 5, 6)
```

Here it's IMPOSSIBLE to find the answer: that (1, 2) could be (1, 2, 1) or (1, 2, 10, 2). (4, 3, 4) could have come from:

```
10001 10010 10011 10100 10101
(4, 3, 4, 3)
```

or from:

```
00100 00101 00110 00111 01000
(4, 3, 4, ?)
```

There's no way to tell where you are in the range [0, 1000000000], so you could be about to hit a whole new bit and you wouldn't know it.

Comments

When we add up the values of our bits, we work modulo 10007. I used negative numbers earlier because the concept is well-defined in mod space: "-1180" mod 10007 is 8827, which is to say that adding 8827 to anything higher than 1180 is the same as subtracting 1180 from it.

One of the nice things about Code Jam is that you can test your assumptions. When writing the input generator for this problem, I asserted that the difference must look like [x, a, x, b, x, c, ...] or [a, x, b, x, c, x, ...]. This let me catch a bug in my code: I wasn't taking the differences mod 10007! In a live contest, I would have had time to fix that and submit.

Finally, there's one limit that's a red herring: that the sequence can't go past 10^9. It isn't hard to convince yourself that the restriction is meaningless: any sequence of length 1000 can't affect more than 11 separate bits, so it doesn't matter if the highest bit is #13 or #32.