

Re: A Fun Recursive Prime Count Estimator

Source: <http://sci.tech-archive.net/Archive/sci.math/2006-05/msg03659.html>

- *From:* quasi <quasi@xxxxxxxx>
 - *Date:* Sat, 20 May 2006 03:09:51 -0400
-

On 19 May 2006 18:34:42 -0700, gjedwards@xxxxxxxx wrote:

An non-mathematician messes about with primes....and anticipates flames...yes, yes, I know for this must just be a reworking (probably not even that) of some other argument so please refer me to the appropriate place – a cursory search hasn't helped, but as a non-mathematician I'm not even sure what words to 'Google'....

Here's a recursive estimator for number of primes less than N. I like it because it everything follows from a single definition: 2 is prime.

For some N (picked out of the number line by throwing a dart*), the probability that it's divisible by 2 is 1/2 of course, by 3 its 1/3, etc.

For any two primes the probabilities that a randomly select N is divisible by them are independent, i.e. if N happens to be divisible by 2 then it doesn't influence your guess as to whether it might be divisible by 3, 5, 7, etc. For example, take out all multiples of 2, and still 1/3 of the remaining numbers are divisible by 3, etc.

If a number is *not* divisible by 2 (say) then the probability that it's divisible by a multiple of 2 (4,6, etc) is zero. Likewise 3, etc.

Therefore, the probability that N is *not* divisible by any number up to it's square root (i.e. it is prime) is:

(in plain text I don't know how you write that big pi-like symbol that means 'multiply all terms between this indices' – the multiply version of the Summation sign, so I'll use M to mean multiply terms for all n between 2 and root(N)

$$p(N) = M (1 - (p(n) * 1/n))$$

In case I screwed this up here's the algorithm in C (returns 'probability' that N is prime):

Re: A Fun Recursive Prime Count Estimator

```
double EstimatedPrimeProb(
double N
)
{
int i;
double prob = 1;

double rootN = sqrt(N);
if(N<=2)
return 1;
else
{
for(i=2; i<=rootN; i++)
{
prob = prob*( 1 - (EstimatedPrimeProb(i)/i ) );
}
return prob;
}
}
```

The key is that we only define the prime probability of a single number (2) to be 1. For any other number, N, we know that to get a probability we need to multiply out $(1-1/p)$ for all primes up to its root, but we're saying that we don't know what the primes actually are, so we multiply out for all integers between 2 and root N, i.e. $(1-1/n)$, but modifying $(1/n)$ by the probability that n is prime (which itself is estimated by the same function according to $p(n)$).

Adding up the probabilities up to N gives a great estimate (try the C code) of the prime count and moreover the error doesn't seem to be 'biased'.

Flame away with how I've restated Theorem [insert here], but my simple brain enjoyed the logic!

Best,

Gareth Edwards
Macclesfield
England

Your recursion works quite well.

My guess is you've recovered the essence of the counting method which underlies the Prime Number Theorem. Even if that's the case, and I'm not really sure, it was a very insightful discovery on your part.

Bravo.

quasi

Re: A Fun Recursive Prime Count Estimator

