Introduction

Analysis of Algorithms

Andreas Klappenecker

Mysterious Billboard

Recall Euler's Number e

$$e = \exp(1) = \sum_{k=0}^{\infty} \frac{1}{k!}$$

$$\approx 2.7182818284\ldots$$

Billboard Question

So the billboard question essentially asked: Given that

$$e = 2.7182818284\ldots$$

$$\text{Is } 2718281828 \text{ prime?}$$

$$\text{Is } 7182818284 \text{ prime?}$$

The first affirmative answer gives the name of the website

Strategy

1. Compute the digits of e
2. i := 0
3. while true do {
4. Extract 10 digit number p at position i
5. return p if p is prime
6. i := i+1
7. }

What needs to be solved?

Essentially, two questions need to be solved:

- How can we create the digits of e?
- How can we test whether an integer is prime?

Generating the Digits

Extracting Digits of e

We can extract the digits of e in base 10 by

```
d[0] = floor(e);                (equals 2)

e1 = 10*(e-d[0]);

d[1] = floor(e1);               (equals 7)

e2 = 10*(e1-d[1]);

d[2] = floor(e2);               (equals 1)
```

Unfortunately, e is a transcendental number, so there is **no pattern** to the generation of the digits in base 10.

Initial idea: Use rational approximation to e instead

Bounds on e

For all positive integers n, we have

$$\left(1+\frac{1}{n}\right)^{n} \leq e \leq \left(1+\frac{1}{n}\right)^{n+1}$$

Proof

For any t in the range $1 \leq t \leq 1 + 1/n$, we have

$$\frac{1}{1+\frac{1}{n}} \leq \frac{1}{t} \leq 1.$$

Hence,

$$\int_1^{1+1/n} \frac{1}{1+\frac{1}{n}}\,dt \leq \int_1^{1+1/n} \frac{1}{t}\,dt \leq \int_1^{1+1/n} 1\,dt.$$

Thus,

$$\frac{1}{n+1} \leq \ln\left(1+\frac{1}{n}\right) \leq \frac{1}{n}$$

Proof

Exponentiating

$$\frac{1}{n+1} \leq \ln\left(1+\frac{1}{n}\right) \leq \frac{1}{n}$$

yields

$$e^{1/(n+1)} \leq 1+\frac{1}{n} \leq e^{1/n}.$$

Therefore, we can conclude that

$$\left(1+\frac{1}{n}\right)^{n} \leq e \leq \left(1+\frac{1}{n}\right)^{n+1}.$$

Approximating e

Since

$$\left(1+\frac{1}{n}\right)^n \leq e \leq \left(1+\frac{1}{n}\right)^n \left(1+\frac{1}{n}\right),$$

the term

$$\left(1+\frac{1}{n}\right)^n$$

approximates e to k digits, when choosing $n = 10^{k+1}$.

Drawbacks

- We need rational arithmetic with long rationals
- Too much coding unless a library is used.
- Perhaps we can find a better solution by choosing a better data structure.
- Ideally, we would like to use integer arithmetic, machine size words if possible.

Generating the Digits
Version 2

Idea

- e is a transcendental number => no pattern when generating its digits in the usual number representation
- Can we find a better data structure?

Mixed-Radix Representations

Let $(a_1; a_2, a_3, a_3 \ldots)$

represent the number

$$a_1 + a_2 \frac{1}{2!} + a_3 \frac{1}{3!} + a_4 \frac{1}{4!} + \cdots$$

We can restrict a_2 to {0,1}, a_3 to {0,1,2}, and so on.

Why?

Mixed-Radix Representations

We can rewrite the number $(a_1; a_2, a_3, a_3 \ldots)$

as

$$a_1 + a_2 \frac{1}{2!} + a_3 \frac{1}{3!} + a_4 \frac{1}{4!} + \cdots$$
$$= a_1 + \frac{1}{2}\left(a_2 + \frac{1}{3}\left(a_3 + \frac{1}{4}\left(a_4 + \cdots\right)\right)\right)\right)$$

Computing the Digits of the Number e

- The number e is given in the form:

$$\begin{aligned} e &= \sum_{k=0}^{\infty} \frac{1}{k!} \\ &= 1 + \tfrac{1}{1}\left(1 + \tfrac{1}{2}\left(1 + \tfrac{1}{3}\left(1 + \cdots\right)\right)\right) \end{aligned}$$

- Thus, in mixed radix representation, e = (2;1,1,1,1,...) where the digit 2 is due to the fact that both k=0 and k=1 contribute to the integral part. Remember: 0!=1 and 1!=1.

Mixed Radix Representations

- In mixed radix representation $(a_0; a_1, a_2, a_3, \ldots)$

  a_0 is the integer part and $(0; a_1, a_2, a_3, \ldots)$ the fractional part.

- 10 times the number is $(10a_0; 10a_1, 10a_2, 10a_3, \ldots)$, but the representation is not regular anymore. The first few digits might exceed their bound. Remember that the ith digit is supposed to be i or less.

- Renormalize the representation to make it regular again

- The algorithm given for base 10 now becomes feasible; this is known as the spigot algorithm.

Spigot Algorithm for e

Goal: Compute the first n decimal digits of e.

1. Initialize: Let the first digit be 2 and initialize an array A of length $n + 2$ to $(1, 1, 1, . . . , 1)$.
2. Repeat $n - 1$ times:

   Multiply each entry of A by 10.

   Take the fractional part: Starting from the right, reduce the ith entry of A modulo $i + 1$, carrying the quotient one place left.

   Output the next digit: The final quotient is the next digit of e.

Suppose that $n \geq \lceil 10e \rceil = 28$.

Let $e_n = \displaystyle\sum_{k=0}^{n} \frac{1}{k!}$ be the truncated version of e.

$$\text{Then } e - e_n = \frac{1}{(n+1)!} \sum_{k=n+1}^{\infty} \frac{(n+1)!}{k!}$$

$$< \frac{1}{(n+1)!} \sum_{k=0}^{\infty} \frac{1}{2^k} < \frac{2}{(n+1)!}$$

$$\text{Since } e^n = \sum_{k=0}^{\infty} \frac{n^k}{k!} > \frac{n^n}{n!},$$

$$e - e_n < \frac{2}{(n+1)!} < \frac{1}{n!} < \left(\frac{e}{n}\right)^n < \left(\frac{1}{10}\right)^n.$$

So the first n digits of e and e_n are the same for $n \geq 28$.

```
def edigits(n):
  res = '2.'
  ten = 10
  mdigits = [1 for _ in range(2,n+4)] # n+2 multi-radix digits, initialized to 1
  for k in range(1,n): # create additional n-1 base-10 digits
    carry, sum, remainders = 0,0, []
    for modulus, mdigit in zip(range(n+3, 1, -1), mdigits):
         # this loop iterates over array of n+2 pairs (n+3,1), (n+2,1),..., (2, 1)
      sum = carry + ten * mdigit
      carry, remainder = divmod(sum, modulus)
      remainders.append(remainder)
    mdigits = remainders
    res += str(sum // 2)
  return res
```

How should we choose n?
(Heuristic argument)

Normal Numbers

The number e is conjectured to be a normal number, meaning that any sequence of n decimal digits are equally likely to occur.

A proof does not exist, but the conjecture has been verified for millions of digits of e.

Probability to be Prime

Let pi(x)=# of primes less than or equal to x.

Pr[number with <= 10 digits is prime]

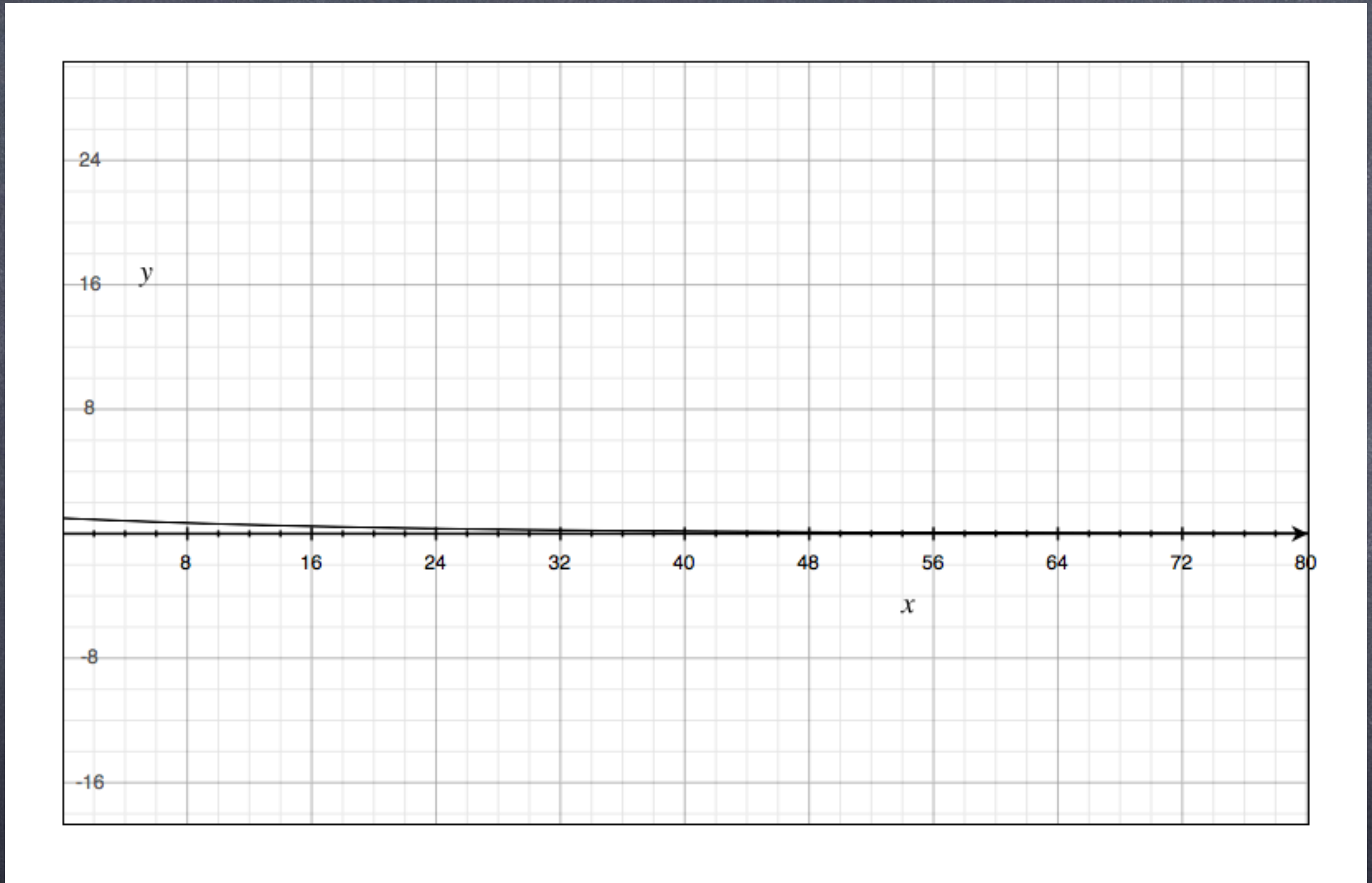
= pi(99999 99999)/99999 99999

= 0.045 (roughly)

Thus, the probability that **none** of the first k 10-digits numbers in e are prime is roughly 0.955^k

This probability rapidly approaches 0 for k->∞, so we need to compute just a few digits of e to find the first 10-digit prime number in e.

Pr[None of the first k numbers is a prime]



Number of Digits

In short, if we generate a few hundred digits of e, then we should be fine.

State of Affairs

We have sketched two solutions to the question of generating the digits of e

- A straightforward solution using rational approximation

- An elegant solution using the mixed-radix representation of e that led to the spigot algorithm

We made it plausible that not too many digits of e are needed to solve the problem

Testing Primality

How do we check Primality?

The second question concerning the testing of primality is simpler.

If a number x is not prime, then it has a divisor d in the range 2<= d <= sqrt(x).

Trial divisions are fast enough here!

Simply check whether any number d in the range 2 <= d < 100 000 divides a 10-digit chunk of e.

```
def isprime(n):
    bound = math.sqrt(n)
    d = 2
    while(d<= bound):
      if( n % d == 0):
        return False
      d = d+1
    return True
```

Answer

```
edigits(109)
'2.718281828459045235360287471352662497757247093699959574966967627724076630353547594571382178525166427427466391'
```

What was it all about?

The billboard was an ad paid for by Google.
The website

http://www.7427466391.com
contained another challenge and then asked people to submit their resume.

Google's obsession with e is well-known, since they pledged in their IPO filing to raise e billion dollars, rather than the usual round-number amount of money.

Summary

- Rational approximation to e and primality test by trial division
- Spigot algorithm for e and primality test by trial division