

CSCE 411
Design and Analysis of Algorithms

Andreas Klappenecker

Motivation

In 2004, a mysterious billboard showed up

- in the Silicon Valley, CA
- in Cambridge, MA
- in Seattle, WA
- in Austin, TX

and perhaps a few other places.

Remarkably, the puzzle on the billboard was immediately discussed worldwide in numerous blogs.

Motivation



Recall Euler's Number e

$$e = \exp(1) = \sum_{k=0}^{\infty} \frac{1}{k!}$$
$$\approx 2.7182818284\dots$$

Billboard Question

So the billboard question essentially asked: Given that

$$e = 2.7182818284 \dots$$

Is 2718281828 prime?

Is 7182818284 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

Extracting Digits of e

We can extract the digits of e in base 10 by

Extracting Digits of e

We can extract the digits of e in base 10 by

$d[0] = \text{floor}(e);$ (equals 2)

Extracting Digits of e

We can extract the digits of e in base 10 by

$d[0] = \text{floor}(e);$ (equals 2)

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

Extracting Digits of e

We can extract the digits of e in base 10 by

$$d[0] = \text{floor}(e); \quad (\text{equals } 2)$$

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

$$d[1] = \text{floor}(e1); \quad (\text{equals } 7)$$

Extracting Digits of e

We can extract the digits of e in base 10 by

$$d[0] = \text{floor}(e); \quad (\text{equals } 2)$$

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

$$d[1] = \text{floor}(e1); \quad (\text{equals } 7)$$

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

Extracting Digits of e

We can extract the digits of e in base 10 by

$$d[0] = \text{floor}(e); \quad (\text{equals } 2)$$

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

$$d[1] = \text{floor}(e1); \quad (\text{equals } 7)$$

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

$$d[2] = \text{floor}(e2); \quad (\text{equals } 1)$$

Extracting Digits of e

We can extract the digits of e in base 10 by

$$d[0] = \text{floor}(e); \quad (\text{equals } 2)$$

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

$$d[1] = \text{floor}(e1); \quad (\text{equals } 7)$$

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

$$d[2] = \text{floor}(e2); \quad (\text{equals } 1)$$

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

Extracting Digits of e

We can extract the digits of e in base 10 by

$$d[0] = \text{floor}(e); \quad (\text{equals } 2)$$

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

$$d[1] = \text{floor}(e1); \quad (\text{equals } 7)$$

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

$$d[2] = \text{floor}(e2); \quad (\text{equals } 1)$$

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

Some Bounds on $e = \exp(1)$

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,

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

Exponentiating

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

yields

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

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

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

the term

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

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

Drawbacks

- The rational approximation converges too slow.

Drawbacks

- The rational approximation converges too slow.
- We need rational arithmetic with long rationals

Drawbacks

- The rational approximation converges too slow.
- We need rational arithmetic with long rationals
- Too much coding unless a library is used.

Drawbacks

- The rational approximation converges too slow.
- 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.

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 Representation

$$a_0 + \frac{1}{2} \left(a_1 + \frac{1}{3} \left(a_2 + \frac{1}{4} \left(a_3 + \frac{1}{5} \left(a_4 + \frac{1}{6} (a_5 + \dots) \right) \right) \right) \right) \right)$$

The digits a_i are nonnegative integers.

The base of this representation is $(1/2, 1/3, 1/4, \dots)$.

The representation is called **regular** if

$a_i \leq i$ for $i \geq 1$.

Number is written as $(a_0, a_1, a_2, a_3, \dots)$

Computing the Digits of the Number e

- Second approach:

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

- In mixed radix representation $e = (2; 1, 1, 1, 1, \dots)$ 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 $(a_0; a_1, a_2, a_3, \dots)$ a_0 is the integer part and $(0; a_1, a_2, a_3, \dots)$ the fractional part.
- 10 times the number is $(10a_0; 10a_1, 10a_2, 10a_3, \dots)$, but the representation is not regular anymore. The first few digits might exceed their bound. Remember that the i th 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.

$$e = 2 + \left[\frac{1}{2} \left(1 + \frac{1}{3} \left(1 + \frac{1}{4} \left(1 + \frac{1}{5} \left(1 + \frac{1}{6} \left(1 + \dots \right) \right) \right) \right) \right) \right)$$

$$= 2 + \frac{1}{10} \left[\frac{1}{2} \left(10 + \frac{1}{3} \left(10 + \frac{1}{4} \left(10 + \frac{1}{5} \left(10 + \frac{1}{6} \left(10 + \dots \right) \right) \right) \right) \right) \right)$$

$$= 2 + \frac{1}{10} \left[7 + \frac{1}{2} \left(0 + \frac{1}{3} \left(1 + \frac{1}{4} \left(0 + \frac{1}{5} \left(1 + \frac{1}{6} \left(5 + \dots \right) \right) \right) \right) \right) \right)$$

$$= 2 \cdot 7 + \frac{1}{100} \left[\frac{1}{2} \left(0 + \frac{1}{3} \left(10 + \frac{1}{4} \left(0 + \frac{1}{5} \right. \right. \right. \right. \right. \\ \left. \left. \left. \left. \left(10 + \frac{1}{6} \left(50 + \dots \right) \right) \right) \right) \right) \right) \right]$$

$$= 2 \cdot 7 + \frac{1}{100} \left[1 + \frac{1}{2} \left(1 + \frac{1}{3} \left(1 + \frac{1}{4} \left(3 + \frac{1}{5} \right. \right. \right. \right. \right. \\ \left. \left. \left. \left. \left(4 + \frac{1}{6} \left(2 + \dots \right) \right) \right) \right) \right) \right) \right]$$

Spigot Algorithm

- `#define N (1000) /* compute N-1 digits of e, by brainwagon@gmail.com */`
- `main(i, j, q) {`
- `int A[N]; printf("2.");`
- `for (j = 0; j < N; j++)`
- `A[j] = 1;` here the *i*th digit is represented by $A[i-1]$, as the integral part is omitted
- `for (i = 0; i < N - 2; i++) {`
- `q = 0;`
- `for (j = N - 1; j >= 0;) {`
- `A[j] = 10 * A[j] + q;`
- `q = A[j] / (j + 2);` compute the amount that needs to be carried over to the next digit
- `A[j] %= (j + 2);` we divide by $j+2$, as regularity means here that $A[j] \leq j+1$
- `A[j] %= (j + 2);` keep only the remainder so that the digit is regular
- `j--;`
- `}`
- `putchar(q + 48);`
- `}`
- `}`

Revisiting the Question

For mathematicians, the previous algorithm is natural, but it might be a challenge for computer scientists and computer engineers to come up with such a solution.

Could we get away with a simpler approach?

After all, the billboard only asks for the **first** prime in the 10-digit numbers occurring in e .

Generating the Digits

Version 3

Probability to be Prime

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

$\Pr[\text{number with } \leq 10 \text{ digits is prime}]$

$$= \pi(99999\ 99999) / 99999\ 99999$$

$$= 0.045 \text{ (roughly)}$$

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 $k \rightarrow \infty$, so we need to compute just a few digits of e to find the first 10-digit prime number in e .

Google it!

Since we will likely need just few digits of Euler's number e , there is no need to reinvent the wheel.

We can simply

- google e or
- use the GNU bc calculator

to obtain a few hundred digits of e .

State of Affairs

We have provided three 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
- A crafty solution that provides enough digits of e to solve the problem at hand.

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 $2 \leq d \leq \sqrt{x}$.

Trial divisions are fast enough here!

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

A Simple Script

<http://discuss.fogcreek.com/joelonsoftware/default.asp?cmd=show&ixPost=160966&ixReplies=23>

```
#!/bin/sh
echo "scale=1000; e(1)" | bc -l | \
perl -0777 -ne '
s/[^0-9]//g;
for $i (0..length($_)-10) {
    $j=substr($_,$i,10);
    $j +=0;
    print "$i\t$j\n" if is_p($j);
}
```

```
sub is_p {
    my $n = shift;
    return 0 if $n <= 1;
    return 1 if $n <= 3;
    for (2 .. sqrt($n)) {
        return 0 unless $n % $_;
    }
    return 1;
}
```

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
- A simple crafty solution