The Magic of Probability

Thomas Zeugmann

Hokkaido University
Laboratory for Algorithmics

http://www-alg.ist.hokudai.ac.jp/~thomas/COCR/

GCOE/IST Citizen Lecture No. 4

Introduction I

Let us start today's lecture by asking the following.

Question

Does probability help to design efficient algorithms?

Introduction I

Let us start today's lecture by asking the following.

Question

Does probability help to design efficient algorithms?

At first glance, this question may sound strange. We are very much used to design deterministic algorithms or to write deterministic programs.

Introduction II

Looking at all the usual algorithms we know, we can say that an algorithm is a computation method having the following properties.

(1) The instruction is a finite text.

(2) The computation is done step by step, where each step performs an elementary operation.

(3) In each step of the execution of the computation it is uniquely determined which elementary operation we have to perform.

(4) The next computation step depends only on the input and the intermediate results computed so far.

Introduction II

Looking at all the usual algorithms we know, we can say that an algorithm is a computation method having the following properties.

(1) The instruction is a finite text.

(2) The computation is done step by step, where each step performs an elementary operation.

(3) In each step of the execution of the computation it is uniquely determined which elementary operation we have to perform.

(4) The next computation step depends only on the input and the intermediate results computed so far.

Introduction II

Looking at all the usual algorithms we know, we can say that an algorithm is a computation method having the following properties.

(1) The instruction is a finite text.

(2) The computation is done step by step, where each step performs an elementary operation.

(3) In each step of the execution of the computation it is uniquely determined which elementary operation we have to perform.

(4) The next computation step depends only on the input and the intermediate results computed so far.

Introduction II

Looking at all the usual algorithms we know, we can say that an algorithm is a computation method having the following properties.

(1) The instruction is a finite text.

(2) The computation is done step by step, where each step performs an elementary operation.

(3) In each step of the execution of the computation it is uniquely determined which elementary operation we have to perform.

(4) The next computation step depends only on the input and the intermediate results computed so far.

Introduction II

Looking at all the usual algorithms we know, we can say that an algorithm is a computation method having the following properties.

(1) The instruction is a finite text.

(2) The computation is done step by step, where each step performs an elementary operation.

(3) In each step of the execution of the computation it is uniquely determined which elementary operation we have to perform.

(4) The next computation step depends only on the input and the intermediate results computed so far.

Introduction III

If probability should help, then we have to replace Property (3) above by allowing instructions of the following type:

```
Flip a coin.  If "head," goto i else goto j  .
```

Such a replacement directly implies that a program may have many *different* computations when run on the same input. So, on some runs, the program may *fail* to achieve its goal, and on some runs, it may *succeed*. However, we shall make sure that the program either succeeds or fails to achieve its goal.

But in general, we have no way to tell which of these two cases did actually happen.

Introduction III

If probability should help, then we have to replace Property (3) above by allowing instructions of the following type:

```
Flip a coin.  If "head," goto i else goto j  .
```

Such a replacement directly implies that a program may have many *different* computations when run on the same input. So, on some runs, the program may *fail* to achieve its goal, and on some runs, it may *succeed*. However, we shall make sure that the program either succeeds or fails to achieve its goal.

But in general, we have no way to tell which of these two cases did actually happen.

Introduction III

If probability should help, then we have to replace Property (3) above by allowing instructions of the following type:

```
Flip a coin.  If “head,” goto i else goto j  .
```

Such a replacement directly implies that a program may have many *different* computations when run on the same input. So, on some runs, the program may *fail* to achieve its goal, and on some runs, it may *succeed*. However, we shall make sure that the program either succeeds or fails to achieve its goal.

But in general, we have no way to tell which of these two cases did actually happen.

Introduction IV

Let us recall the basic definition of probability. Classically, one defines the probability of an event to be *the ratio of favorable cases to all cases*.

Example 1

Let us consider a fair die having the usual six possible outcomes 1, 2, 3, 4, 5, 6.
We consider the event that we throw an even number. Then the favorable cases are 2, 4, 6.
So, the probability to throw an even number is $3/6 = 1/2$.

It should be mentioned that this definition only works if all elementary events have the same probability. Therefore, we required our die to be fair.

Introduction IV

Let us recall the basic definition of probability. Classically, one defines the probability of an event to be *the ratio of favorable cases to all cases*.

Example 1

Let us consider a fair die having the usual six possible outcomes 1, 2, 3, 4, 5, 6.
We consider the event that we throw an even number. Then the favorable cases are 2, 4, 6.
So, the probability to throw an even number is $3/6 = 1/2$.

It should be mentioned that this definition only works if all elementary events have the same probability. Therefore, we required our die to be fair.

Introduction IV

Let us recall the basic definition of probability. Classically, one defines the probability of an event to be *the ratio of favorable cases to all cases*.

Example 1

Let us consider a fair die having the usual six possible outcomes 1, 2, 3, 4, 5, 6.
We consider the event that we throw an even number. Then the favorable cases are 2, 4, 6.
So, the probability to throw an even number is $3/6 = 1/2$.

It should be mentioned that this definition only works if all elementary events have the same probability. Therefore, we required our die to be fair.

Introduction V

So, in our lecture we shall ensure that each run of the probabilistic algorithm has the same probability. We shall achieve this goal by allowing only one random choice at the first stage of the algorithm. All other stages (or steps) must be deterministic.

Then our design of a probabilistic algorithm should ensure that the "overwhelming" number of runs is successful.

Now, it is time to present the problem we wish to study. This problem is taken from Juraj Hromkovič's book *Algorithmic Adventures, From Knowledge to Magic*.

Introduction V

So, in our lecture we shall ensure that each run of the probabilistic algorithm has the same probability. We shall achieve this goal by allowing only one random choice at the first stage of the algorithm. All other stages (or steps) must be deterministic.

Then our design of a probabilistic algorithm should ensure that the “overwhelming” number of runs is successful.

Now, it is time to present the problem we wish to study. This problem is taken from Juraj Hromkovič’s book *Algorithmic Adventures, From Knowledge to Magic*.

Introduction V

So, in our lecture we shall ensure that each run of the probabilistic algorithm has the same probability. We shall achieve this goal by allowing only one random choice at the first stage of the algorithm. All other stages (or steps) must be deterministic.

Then our design of a probabilistic algorithm should ensure that the "overwhelming" number of runs is successful.

Now, it is time to present the problem we wish to study. This problem is taken from Juraj Hromkovič's book *Algorithmic Adventures, From Knowledge to Magic*.

The Problem I

Suppose we have two computers C_1 and C_2 that are very far apart. Nevertheless, we want to have on both computers the same *huge* database. Initially, on both computers we have the same database. However, the database evolves over time and every new datum is sent to both computers.

So, the changes are to be performed *simultaneously* on both computers, e.g., incorporating newly discovered genome sequences into both databases.

From time to time we wish to check whether or not both computers do have the same database. In order to simplify the presentation, we consider the contents of the databases of C_1 and C_2 as a sequence of bits, i.e., computer C_1 has $x = x_1x_2\cdots x_{n-1}x_n$ and computer C_2 has $y = y_1y_2\cdots y_{n-1}y_n$.

The Problem I

Suppose we have two computers C_1 and C_2 that are very far apart. Nevertheless, we want to have on both computers the same *huge* database. Initially, on both computers we have the same database. However, the database evolves over time and every new datum is sent to both computers.

So, the changes are to be performed *simultaneously* on both computers, e.g., incorporating newly discovered genome sequences into both databases.

From time to time we wish to check whether or not both computers do have the same database. In order to simplify the presentation, we consider the contents of the databases of C_1 and C_2 as a sequence of bits, i.e., computer C_1 has $x = x_1x_2 \cdots x_{n-1}x_n$ and computer C_2 has $y = y_1y_2 \cdots y_{n-1}y_n$.

The Problem I

Suppose we have two computers C_1 and C_2 that are very far apart. Nevertheless, we want to have on both computers the same *huge* database. Initially, on both computers we have the same database. However, the database evolves over time and every new datum is sent to both computers.

So, the changes are to be performed *simultaneously* on both computers, e.g., incorporating newly discovered genome sequences into both databases.

From time to time we wish to check whether or not both computers do have the same database. In order to simplify the presentation, we consider the contents of the databases of C_1 and C_2 as a sequence of bits, i.e., computer C_1 has $x = x_1 x_2 \cdots x_{n-1} x_n$ and computer C_2 has $y = y_1 y_2 \cdots y_{n-1} y_n$.

The Problem II

Thus, by using a communication channel (a network) between C_1 and C_2, we have to verify whether or not $x = y$ (see the figure below).

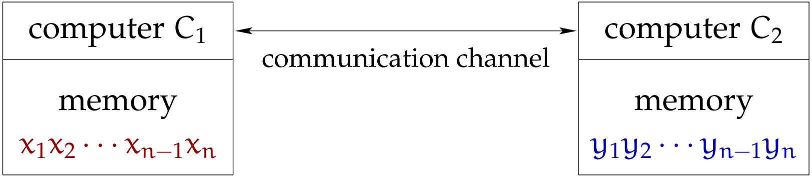


Figure 1: The verification task.

To solve this task one has to design a communication protocol.

The Problem III

We measure the complexity of the communication by counting the number of bits exchanged between C_1 and C_2.

The bad news are that any deterministic communication protocol cannot be better (on most inputs) than to exchange n bits.

This is, of course, also the trivial solution.

The Problem III

We measure the complexity of the communication by counting the number of bits exchanged between C_1 and C_2.
The bad news are that any deterministic communication protocol cannot be better (on most inputs) than to exchange n bits.
This is, of course, also the trivial solution.

Suppose that we have $n = 10^{16}$ (which is roughly the memory size of 25 000 DVDs). Exchanging such an amount of bits over the internet without making any error is almost unrealistic given current technology. And the communication complexity is too high. If we can transmit 10^7 many bits per second, it will take approxiamately 31 years.

The Problem IV

For the remaining part of the talk, it is advantageous to *interpret* the sequences $x = x_1 x_2 \cdots x_{n-1} x_n$ and $y = y_1 y_2 \cdots y_{n-1} y_n$, $x_i, y_i \in \{0, 1\}$, $i = 1, \ldots, n$, as numbers. That is

$$\begin{aligned} \text{num}(x) &= \sum_{i=1}^{n} 2^{n-i} \cdot x_i \, , \quad \text{and} \\ \text{num}(y) &= \sum_{i=1}^{n} 2^{n-i} \cdot y_i \, . \end{aligned}$$

Example 2

$$\begin{aligned} \text{num}(10001) &= 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 17 \, , \\ \text{num}(11111) &= 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 31 \, . \end{aligned}$$

The Problem IV

For the remaining part of the talk, it is advantageous to *interpret* the sequences $x = x_1x_2 \cdots x_{n-1}x_n$ and $y = y_1y_2 \cdots y_{n-1}y_n$, $x_i, y_i \in \{0, 1\}$, $i = 1, \ldots, n$, as numbers. That is

$$\begin{aligned} \mathrm{num}(x) &= \sum_{i=1}^{n} 2^{n-i} \cdot x_i \,, \quad \text{and} \\ \mathrm{num}(y) &= \sum_{i=1}^{n} 2^{n-i} \cdot y_i \,. \end{aligned}$$

Example 2

$$\begin{aligned} \mathrm{num}(10001) &= 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 17 \,, \\ \mathrm{num}(11111) &= 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 31 \,. \end{aligned}$$

The Problem V

Obviously, we have $x = y$ if and only if $\mathrm{num}(x) = \mathrm{num}(y)$. We should also note that

$$\begin{aligned} 0 &\leqslant \mathrm{num}(x) \leqslant 2^n - 1\,, \quad \text{and} \\ 0 &\leqslant \mathrm{num}(y) \leqslant 2^n - 1\,. \end{aligned}$$

So, these numbers are *huge*.

We need the following notations. For every positive integer $m \geqslant 2$ we set

$$\begin{aligned} \mathrm{PRIM}(m) &= \{p \mid p \text{ is a prime and } p \leqslant m\}\,, \quad \text{and} \\ \mathrm{Prim}(m) &= |\mathrm{PRIM}(m)|\,, \end{aligned}$$

where $|\mathrm{PRIM}(m)|$ denotes the number of elements in the set $\mathrm{PRIM}(m)$.

The Problem V

Obviously, we have $x = y$ if and only if $\text{num}(x) = \text{num}(y)$. We should also note that

$$\begin{aligned} 0 &\leqslant \text{num}(x) \leqslant 2^n - 1\,, \quad \text{and} \\ 0 &\leqslant \text{num}(y) \leqslant 2^n - 1\,. \end{aligned}$$

So, these numbers are *huge*.

We need the following notations. For every positive integer $m \geqslant 2$ we set

$$\begin{aligned} \text{PRIM}(m) &= \{p \mid p \text{ is a prime and } p \leqslant m\}\,, \quad \text{and} \\ \text{Prim}(m) &= |\text{PRIM}(m)|\,, \end{aligned}$$

where $|\text{PRIM}(m)|$ denotes the number of elements in the set $\text{PRIM}(m)$.

The Problem VI

In the following we denote by $r = a \bmod b$ the remainder of the division $a : b$.

Example 3

Let $a = 17$ and $b = 5$; then we can write

$$17 = 3 \cdot 5 + 2\,,$$

and therefore we obtain $2 = 17 \bmod 5$.

Now we are in the position to present our *randomized communication protocol* for the comparison of $\mathrm{num}(x)$ and $\mathrm{num}(y)$.

The Problem VI

In the following we denote by $r = a \bmod b$ the remainder of the division $a : b$.

Example 3

Let $a = 17$ and $b = 5$; then we can write

$$17 = 3 \cdot 5 + 2\,,$$

and therefore we obtain $2 = 17 \bmod 5$.

Now we are in the position to present our *randomized communication protocol* for the comparison of $\mathrm{num}(x)$ and $\mathrm{num}(y)$.

The RCP WITNESS I

The randomized communication protocol WITNESS

given: Computer C_1 and n bits $x_1 x_2 \cdots x_n$
Computer C_2 and n bits $y_1 y_2 \cdots y_n$

Phase 1: C_1 chooses uniformly at random a prime p from $PRIM(n^2)$.

Phase 2: C_1 computes the integer

$$s = num(x) \bmod p$$

and sends s and p in binary representation to C_2.

Phase 3: After reading s and p, the computer C_2 computes

$$q = num(y) \bmod p \ .$$

If $q \neq s$, then C_2 outputs "not equal."
If $q = s$, then C_2 outputs "equal."

The RCP WITNESS II

Example 4

Let $x = 01111$ and $y = 10110$. Hence, $n = 5$ and

$$\begin{aligned} \mathrm{num}(x) &= 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 15\,, \\ \mathrm{num}(y) &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 22\,. \end{aligned}$$

The RCP WITNESS II

Example 4

Let $x = 01111$ and $y = 10110$. Hence, $n = 5$ and

$$\begin{aligned} \text{num}(x) &= 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 15\,, \\ \text{num}(y) &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 22\,. \end{aligned}$$

Also, $n^2 = 25$ and $\text{PRIM}(25) = \{2, 3, 5, 7, 11, 13, 17, 19, 23\}$.
Assume that in Phase 1 the prime 5 has been chosen uniformly at random.
Then in Phase 2 the computer C_1 computes $0 = 15 \bmod 5$ and sends 0 and 101 to C_2.
In Phase 3, the computer C_2 computes $2 = 22 \bmod 5$ and thus outputs "not equal."

The RCP WITNESS III

Note that in our example the output of C_2 is for all primes from $\mathrm{PRIM}(25)$ "not equal" except for $p = 7$; here we obtain "equal."

Next, we analyze the communication cost of RCP WITNESS. As already stated, we have $\mathrm{num}(x), \mathrm{num}(y) \in \{0, 2^n - 1\}$.
We have to send two numbers s and p which are by construction less than n^2. For representing such numbers in binary we need

$$\lceil \log_2 n^2 \rceil \leqslant 2 \cdot \lceil \log_2 n \rceil$$

many bits. Thus, we have to send at most $4 \cdot \lceil \log_2 n \rceil$ many bits.

What does this mean for $n = 10^{16}$? The best deterministic protocol must send 10^{16} many bits.
Our RCP WITNESS works within

$$4 \cdot \lceil \log_2 10^{16} \rceil \leqslant 4 \cdot 16 \lceil \log_2 10 \rceil = 256$$

many communication bits.

The RCP WITNESS III

Note that in our example the output of C_2 is for all primes from PRIM(25) "not equal" except for $p = 7$; here we obtain "equal."

Next, we analyze the communication cost of RCP WITNESS. As already stated, we have $\text{num}(x), \text{num}(y) \in \{0, 2^n - 1\}$.

We have to send two numbers s and p which are by construction less than n^2. For representing such numbers in binary we need

$$\lceil \log_2 n^2 \rceil \leqslant 2 \cdot \lceil \log_2 n \rceil$$

many bits. Thus, we have to send at most $4 \cdot \lceil \log_2 n \rceil$ many bits.

What does this mean for $n = 10^{16}$? The best deterministic protocol must send 10^{16} many bits.

Our RCP WITNESS works within

$$4 \cdot \lceil \log_2 10^{16} \rceil \leqslant 4 \cdot 16 \lceil \log_2 10 \rceil = 256$$

many communication bits.

The RCP WITNESS III

Note that in our example the output of C_2 is for all primes from $\mathrm{PRIM}(25)$ "not equal" except for $p = 7$; here we obtain "equal."

Next, we analyze the communication cost of RCP WITNESS. As already stated, we have $\mathrm{num}(x), \mathrm{num}(y) \in \{0, 2^n - 1\}$.

We have to send two numbers s and p which are by construction less than n^2. For representing such numbers in binary we need

$$\lceil \log_2 n^2 \rceil \leqslant 2 \cdot \lceil \log_2 n \rceil$$

many bits. Thus, we have to send at most $4 \cdot \lceil \log_2 n \rceil$ many bits.

What does this mean for $n = 10^{16}$? The best deterministic protocol must send 10^{16} many bits.

Our RCP WITNESS works within

$$4 \cdot \lceil \log_2 10^{16} \rceil \leqslant 4 \cdot 16 \lceil \log_2 10 \rceil = 256$$

many communication bits.

The RCP WITNESS IV

Clearly, 256 many communication bits and 10^{16} many communication bits are incomparable in terms of the communication cost.
For this *unbelievable* large saving of communication cost we pay by losing the certainty of always getting the correct result.

Thus, the remaining task is to ask the following.

Question

How large is the degree of unreliability?

To answer this question, we have to compute the so-called *error probability*.

The RCP WITNESS IV

Clearly, 256 many communication bits and 10^{16} many communication bits are incomparable in terms of the communication cost.
For this *unbelievable* large saving of communication cost we pay by losing the certainty of always getting the correct result.

Thus, the remaining task is to ask the following.

Question

How large is the degree of unreliability?

To answer this question, we have to compute the so-called *error probability*.

The RCP WITNESS IV

Clearly, 256 many communication bits and 10^{16} many communication bits are incomparable in terms of the communication cost.
For this *unbelievable* large saving of communication cost we pay by losing the certainty of always getting the correct result.

Thus, the remaining task is to ask the following.

Question

How large is the degree of unreliability?

To answer this question, we have to compute the so-called *error probability*.

The RCP WITNESS V

We say that a prime is good for (x, y) if it leads to the correct output in the RCP WITNESS.
Otherwise, we say that a prime is bad for (x, y).

In our example, 7 was bad for $(01111, 10110)$ and all other primes in $\mathrm{PRIM}(25)$ were good for $(01111, 10110)$.

Since each prime from $\mathrm{PRIM}(n^2)$ has the same probability to be chosen, that means we have to estimate

$$\mathrm{Error}_{RCPW}(x, y) = \frac{\text{the number of bad primes for } (x, y)}{\mathrm{Prim}(n^2)} .$$

The RCP WITNESS V

We say that a prime is good for (x, y) if it leads to the correct output in the RCP WITNESS.
Otherwise, we say that a prime is bad for (x, y).

In our example, 7 was bad for $(01111, 10110)$ and all other primes in $\mathrm{PRIM}(25)$ were good for $(01111, 10110)$.

Since each prime from $\mathrm{PRIM}(n^2)$ has the same probability to be chosen, that means we have to estimate

$$\mathrm{Error}_{RCPW}(x, y) = \frac{\text{the number of bad primes for } (x, y)}{\mathrm{Prim}(n^2)} .$$

The RCP WITNESS VI

So, we have to show that $\text{Error}_{RCPW}(x, y)$ is small for every instance (x, y) of our identity problem (see Figure 2).

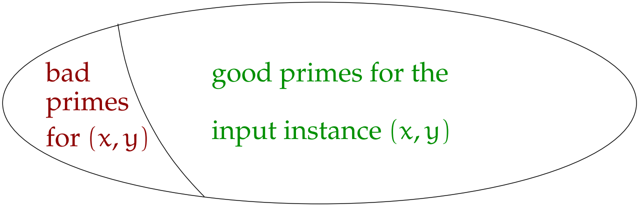


Figure 2: The error probability.

So, we need a bit more mathematics at this point.

The RCP WITNESS VI

So, we have to show that $\mathrm{Error}_{RCPW}(x, y)$ is small for every instance (x, y) of our identity problem (see Figure 2).

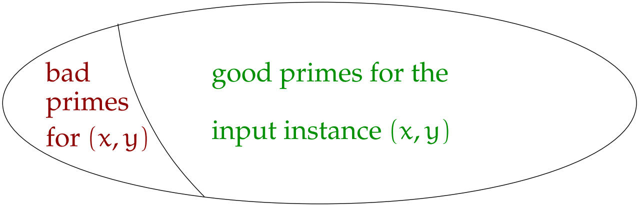


Figure 2: The error probability.

So, we need a bit more mathematics at this point.

The RCP WITNESS VII

One of the deepest and most important discoveries in number theory is that for all $m > 67$ we have

$$\mathrm{Prim}(m) > \frac{m}{\ln m} ,$$

where $\ln m$ denotes the *natural* logarithm of m. This is known as the Prime Number Theorem and it was only proved in 1896 independently by Hadamard and de la Vallée Poussin.

Therefore, for all $n \geqslant 9$ we know that

$$\mathrm{Prim}(n^2) > \frac{n^2}{2 \cdot \ln n} .$$

Now, it is best to make the following case distinction.

The RCP WITNESS VII

One of the deepest and most important discoveries in number theory is that for all $m > 67$ we have

$$\text{Prim}(m) > \frac{m}{\ln m} \,,$$

where $\ln m$ denotes the *natural* logarithm of m. This is known as the Prime Number Theorem and it was only proved in 1896 independently by Hadamard and de la Vallée Poussin.

Therefore, for all $n \geqslant 9$ we know that

$$\text{Prim}(n^2) > \frac{n^2}{2 \cdot \ln n} \,.$$

Now, it is best to make the following case distinction.

The RCP WITNESS VII

One of the deepest and most important discoveries in number theory is that for all $m > 67$ we have

$$\text{Prim}(m) > \frac{m}{\ln m} ,$$

where $\ln m$ denotes the *natural* logarithm of m. This is known as the Prime Number Theorem and it was only proved in 1896 independently by Hadamard and de la Vallée Poussin.

Therefore, for all $n \geqslant 9$ we know that

$$\text{Prim}(n^2) > \frac{n^2}{2 \cdot \ln n} .$$

Now, it is best to make the following case distinction.

The RCP WITNESS VIII

Case 1. $x = y$

Since $x = y$, we conclude that $\text{num}(x) = \text{num}(y)$, and hence

$$s = \text{num}(x) \bmod p \ = \ \text{num}(y) \bmod p = q$$

for all $p \in \text{PRIM}(n^2)$.

That is, in this case we have $\text{Error}_{RCPW}(x, y) = 0$ for all strings x and y such that $x = y$.

So, our RCP WITNESS can *only* make an error if $x \neq y$.

The RCP WITNESS VIII

Case 1. $x = y$

Since $x = y$, we conclude that $\text{num}(x) = \text{num}(y)$, and hence

$$s = \text{num}(x) \bmod p \; = \; \text{num}(y) \bmod p = q$$

for all $p \in \text{PRIM}(n^2)$.

That is, in this case we have $\text{Error}_{RCPW}(x, y) = 0$ for all strings x and y such that $x = y$.

So, our RCP WITNESS can *only* make an error if $x \neq y$.

The RCP WITNESS IX

Case 2. $x \neq y$

Let p be a bad prime for (x, y). Then we have

$$s = \text{num}(x) \bmod p = \text{num}(y) \bmod p = q$$

Thus, $s = q$ and we can write

$$\begin{aligned} \text{num}(x) &= h_x \cdot p + s \\ \text{num}(y) &= h_y \cdot p + s\,. \end{aligned}$$

Without loss of generality we assume $\text{num}(x) \geqslant \text{num}(y)$ and by subtracting the latter two equation, we obtain

$$\text{diff}(x, y) = \text{num}(x) - \text{num}(y) = (h_x - h_y) \cdot p\,,$$

that is, a bad prime must divide $\text{diff}(x, y)$.

The RCP WITNESS X

We know that $\text{diff}(x, y) < 2^n$ and that every prime $p_i > i$, where p_i is the ith prime number dividing $\text{diff}(x, y)$, $i = 1, \ldots, k$. Thus, we obtain

$$\begin{aligned} \text{diff}(x, y) &\geqslant p_1 \cdot p_2 \cdot \ldots \cdot p_k > 1 \cdot 2 \cdot \ldots \cdot k \\ &= k!\,. \end{aligned}$$

Hence, we arrive at the condition $2^n > k!$. Finally, $n! > 2^n$ for $n \geqslant 4$ and thus $k < n$ must hold.

So, we have seen that at most $k \leqslant n - 1$ primes from $\text{PRIM}(n^2)$ could be bad.

This allows us to upperbound the error probability.

$$\begin{aligned} \text{Error}_{RCPW}(x, y) &= \frac{\text{the number of bad primes for } (x, y)}{\text{Prim}(n^2)} \\ &\leqslant \frac{n-1}{n^2/\ln n^2} \leqslant \frac{2 \cdot \ln n}{n}\,. \end{aligned}$$

The RCP WITNESS X

We know that $\mathrm{diff}(x, y) < 2^n$ and that every prime $p_i > i$, where p_i is the ith prime number dividing $\mathrm{diff}(x, y)$, $i = 1, \ldots, k$. Thus, we obtain

$$\begin{aligned}\mathrm{diff}(x, y) &\geqslant p_1 \cdot p_2 \cdot \ldots \cdot p_k > 1 \cdot 2 \cdot \ldots \cdot k \\ &= k!\,.\end{aligned}$$

Hence, we arrive at the condition $2^n > k!$. Finally, $n! > 2^n$ for $n \geqslant 4$ and thus $k < n$ must hold.

So, we have seen that at most $k \leqslant n - 1$ primes from $\mathrm{PRIM}(n^2)$ could be bad.

This allows us to upperbound the error probability.

$$\begin{aligned}\mathrm{Error}_{RCPW}(x, y) &= \frac{\text{the number of bad primes for } (x, y)}{\mathrm{Prim}(n^2)} \\ &\leqslant \frac{n-1}{n^2/\ln n^2} \leqslant \frac{2 \cdot \ln n}{n}\,.\end{aligned}$$

The RCP WITNESS X

We know that $\text{diff}(x, y) < 2^n$ and that every prime $p_i > i$, where p_i is the ith prime number dividing $\text{diff}(x, y)$, $i = 1, \ldots, k$. Thus, we obtain

$$\begin{aligned} \text{diff}(x, y) &\geqslant p_1 \cdot p_2 \cdot \ldots \cdot p_k > 1 \cdot 2 \cdot \ldots \cdot k \\ &= k!\,. \end{aligned}$$

Hence, we arrive at the condition $2^n > k!$. Finally, $n! > 2^n$ for $n \geqslant 4$ and thus $k < n$ must hold.

So, we have seen that at most $k \leqslant n - 1$ primes from $\text{PRIM}(n^2)$ could be bad.

This allows us to upperbound the error probability.

$$\begin{aligned} \text{Error}_{RCPW}(x, y) &= \frac{\text{the number of bad primes for } (x, y)}{\text{Prim}(n^2)} \\ &\leqslant \frac{n - 1}{n^2 / \ln n^2} \leqslant \frac{2 \cdot \ln n}{n}\,. \end{aligned}$$

Concluding Remarks I

So, in our example the error probability to output "equal" for sequences x and y with $x \neq y$ is upper bounded by $(2 \cdot \ln n)/n$. For $n = 10^{16}$, this yields

$$\frac{0.36841}{10^{14}},$$

which is amazingly small.

Even better, if we look into the future and consider even bigger databases, then the error probability will be even smaller.

Concluding Remarks I

So, in our example the error probability to output "equal" for sequences x and y with $x \neq y$ is upper bounded by $(2 \cdot \ln n)/n$. For $n = 10^{16}$, this yields

$$\frac{0.36841}{10^{14}},$$

which is amazingly small.

Even better, if we look into the future and consider even bigger databases, then the error probability will be even smaller.

Concluding Remarks II

The results obtained allow for a further improvement. As we have seen, if $x = y$ then our RCP WITNESS *is correct* with certainty. All uncertainty is in case that $x \neq y$. Here we may wrongly output "equal." However, if we have found a prime $p \in \mathrm{PRIMES}(n^2)$ *witnessing* that $x \neq y$ then the result is again certainly correct.

Now, if we chose ℓ many primes independently at random from $\mathrm{PRIMES}(n^2)$ instead of just one, then the probability not to find a witness for $x \neq y$ is

$$\left(\frac{2 \cdot \ln n}{n}\right)^{\ell} .$$

For our $n = 10^{16}$ this gives for $\ell = 10$ the upperbound

$$\frac{0.4714}{10^{114}} \quad \text{to still make an error .}$$

Concluding Remarks II

The results obtained allow for a further improvement. As we have seen, if $x = y$ then our RCP WITNESS *is correct* with certainty. All uncertainty is in case that $x \neq y$. Here we may wrongly output "equal." However, if we have found a prime $p \in \text{PRIMES}(n^2)$ *witnessing* that $x \neq y$ then the result is again certainly correct.

Now, if we chose ℓ many primes independently at random from $\text{PRIMES}(n^2)$ instead of just one, then the probability not to find a witness for $x \neq y$ is

$$\left(\frac{2 \cdot \ln n}{n}\right)^{\ell} .$$

For our $n = 10^{16}$ this gives for $\ell = 10$ the upperbound

$$\frac{0.4714}{10^{114}} \quad \text{to still make an error} .$$

Concluding Remarks II

The results obtained allow for a further improvement. As we have seen, if $x = y$ then our RCP WITNESS *is correct* with certainty. All uncertainty is in case that $x \neq y$. Here we may wrongly output "equal." However, if we have found a prime $p \in \text{PRIMES}(n^2)$ *witnessing* that $x \neq y$ then the result is again certainly correct.

Now, if we chose ℓ many primes independently at random from $\text{PRIMES}(n^2)$ instead of just one, then the probability not to find a witness for $x \neq y$ is

$$\left(\frac{2 \cdot \ln n}{n}\right)^{\ell} .$$

For our $n = 10^{16}$ this gives for $\ell = 10$ the upperbound

$$\frac{0.4714}{10^{114}} \quad \text{to still make an error .}$$

Concluding Remarks III

What we have to pay in terms of communication costs for this further improvement?

Concluding Remarks III

What we have to pay in terms of communication costs for this further improvement?

In fact, not much. Instead of sending two numbers s and p in Phase 2, now we have to communicate 20 numbers $s_1, \ldots, s_{10}$ and $p_1, \ldots, p_{10}$. That is, instead of 256 bits we now have to communicate 2560 many bits.

Concluding Remarks III

What we have to pay in terms of communication costs for this further improvement?

In fact, not much. Instead of sending two numbers s and p in Phase 2, now we have to communicate 20 numbers $s_1, \ldots, s_{10}$ and $p_1, \ldots, p_{10}$. That is, instead of 256 bits we now have to communicate 2560 many bits.

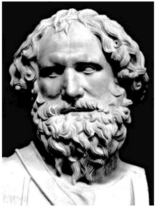
"There are things, that seem unbelievable to most people who have not studied mathematics."

Concluding Remarks III

What we have to pay in terms of communication costs for this further improvement?

In fact, not much. Instead of sending two numbers s and p in Phase 2, now we have to communicate 20 numbers $s_1, \ldots, s_{10}$ and $p_1, \ldots, p_{10}$. That is, instead of 256 bits we now have to communicate 2560 many bits.

"There are things, that seem unbelievable to most people who have not studied mathematics."



Archimedes (287 - 212 bef. Chr.)

Concluding Remarks IV

They call it

M A G I C

Concluding Remarks IV

They call it

MAGIC

We call it

SCIENCE

Thank you!