Chance favours only those whose spirit has been prepared already, those unprepared cannot see the hand stretched out to them by fortune.

Louis Pasteur

Chapter 6

Randomness in Nature and as a Source of Efficiency in Algorithmics

6.1 Aims

In this chapter we aim to introduce one of the miracles of computer science. This marvel is based on a successful application of randomness in the design of efficient algorithms for apparently intractable problems.

To perform and to understand this magic one has to deal with the meaning of the notion of "randomness" first. Hence, we follow the history of science which is full of controversial discussions about the existence of true randomness in order to get a feeling for what can be viewed as true randomness and what is only apparent randomness.

J. Hromkovič, *Algorithmic Adventures*, DOI 10.1007/978-3-540-85986-4_6,

Taking that generated view about randomness we directly move to algorithmics in order to apply it for solving computationally hard problems. Here we see a miracle. Can you imagine, at first glance, that systems with a randomized control can achieve their goals billions of times faster than any deterministic system can? Do you believe that one has to pay for this huge speed-up by risking making an error with probability[1] smaller than $1/10^{19}$ only? Is this really a serious risk? If one had started such a randomized system every second since the Big Bang (i.e. between 10^{18} to 10^{19} times), then one may have expected none of these many runs to make an error. In another words the probability of making at least one error in these more than 10^{18} experimental runs of this randomized system is smaller than the probability of reaching the correct goal in all these attempts.

Here we call attention to the fact that in practice randomized algorithms (algorithms with a random control) with very small error probability can be even more reliable than their best deterministic counterparts. What do we mean by this? Theoretically all deterministic algorithms may not err. But the nature of the story is that deterministic programs are not absolutely reliable because during their runs on a computer a hardware error may occur and then they may produce wrong results. Clearly, the probability of the occurrence of a hardware error grows proportionally with the running time of the program. Therefore a fast randomized algorithm can be more reliable than a slow deterministic algorithm. For instance, if a randomized algorithm computes a result in 10 seconds with an error probability $1/10^{30}$, then it is more reliable than a deterministic algorithm that computes the result in one week. Hence, moving from common (deterministic) algorithms and systems to randomized ones is not necessarily related to decreasing reliability. And losing an apparently absolute reliability may not hurt too much. Randomization, here we mean building controlling systems by using a clever combination of deterministic strategies with random decisions, became a new paradigm of algorithmics. Efficient randomized algorithms making one mistake in billions of

[1] The risk of computing a wrong result is called the error probability.

applications on average are very practical and enable us to solve problems considered intractable for deterministic computations.

After presenting a magical example to convince ourselves about the power and the usefulness of randomness we do not leave the reader in her or his amazement. We will try to give the reader at least a feel for why such unbelievably elegant and wonderful ways for solving problems exist. We get a deeper view into the nature of the success of randomized systems and recognize some paradigms for the design of randomized algorithms. And, maybe, at the very end we can even encourage you to use the obtained knowledge in your everyday life and so to become a magician.

6.2 What Is Randomness and Does There Exist True Randomness?

In the first chapter we realized that the creation of notions is fundamental for the foundation and the development of scientific disciplines. The notion of **randomness** is one of the few most fundamental and most discussed notions of science. Philosophers, mathematicians, physicists, biologists and chemists discussed the existence of randomness and its possible role for thousands of years. Mathematicians created probability theory as an instrument for investigating random events. Computer scientists study why and when it is profitable to use randomness and design randomized algorithms. Engineers apply the obtained knowledge to build efficient advantageous randomized systems.

The notion of randomness is strongly related to another two fundamental notions: **determinism** and **nondeterminism**. The deterministic view of the world is based on the paradigm of causality. Every event has a cause, i.e., every event is a consequence of a cause. Every cause has its effect (result) and this result is a cause of further effects, etc. Following the principle of causality, if one knows the correct state of the Universe and all natural laws, one can completely predict the future. The whole development is

6.3 The Abundance of Witnesses Is Support in Shortage or Why Randomness Can Be Useful

The aim of this section is to present a task that can be solved efficiently by applying randomness despite the known fact that this task cannot be solved efficiently in any deterministic way. Here we design a randomized system for solving the mentioned task in such a simple way that anybody can use it in order to get a first convincing experience of the extraordinary power of randomized control.

What is the exact meaning of a **randomized** system or randomized algorithm (program)? In fact one allows two different basic transparent descriptions. Each deterministic program executes on a given input an unambiguously given sequence of computing steps. We say that the program together with its input unambiguously determines a unique computation. A randomized program (system) can have many different computations on the same input, and which one is going to be executed is decided at random. In what follows we present two possibilities for the performance of the random choice from possible computations.

1. The program works in the deterministic way except for a few special places, in which it may flip a coin. Depending on the result of coin flipping (head or tail), the program takes one of the possible two ways to continue in the work on the input.

   To insert this possibility of random choice into our programming language TRANSPARENT, it is sufficient to add the following instruction:

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

   In this way the program continues to work in the i-th row if "head" was the result of flipping coin and it continues in the j-th row if "tail" was the result of flipping coin.

2. The randomized algorithm has at the beginning a choice of several deterministic strategies. The program randomly chooses one of these strategies and applies it on the given input. The

rest of the computation is completely deterministic. The random choice is reduced to the first step of computation. For each new problem instance the algorithm chooses a new strategy at random.

Since the second way of modelling randomized systems is simpler and more transparent than the first one, we use it for the presentation of our exemplary application of the power of randomness.

Let us consider the following scenario.

We have two computers R_I and R_{II} (Fig. 6.1) that are very far apart. The task is to manage the same database on the computers. At the beginning both have a database with the same content. In the meantime the contents of these databases dynamically develop in such a way that one tries to perform all changes simultaneously on both databases with the aim of getting the same database, with complete information about the database subject (for instance, genome sequences), in both locations. After some time, one wants to check whether this process is successful, i.e., whether R_I and R_{II} contain the same data.

We idealize this verification task in the sense of a mathematical abstraction. We consider the contents of the databases of R_I and R_{II} as sequences of bits. Hence, we assume that the computer R_I has a sequence

$$x = x_1 x_2 x_3 \dots x_{n-1} x_n$$

of n bits and the computer R_{II} has the n-bits sequence

$$y = y_1 y_2 y_3 \dots y_{n-1} y_n.$$

The task is to use a communication channel (network) between R_I and R_{II} in order to verify whether $x = y$ (Fig. 6.1).

To solve this task one has to design a communication strategy, called a **communication protocol** in what follows. The **complexity of communication** and so the complexity of solving the verification problem is measured in the number of bits exchanged between R_I and R_{II} through the communication channel.

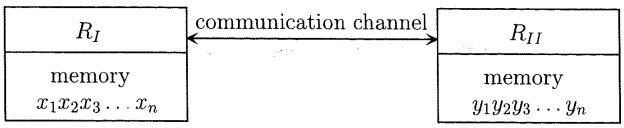

Fig. 6.1

Unfortunately, the provable fact is that any communication protocol for solving this task cannot do better for most possible inputs x and y than to exchange n bits. This means that the naive protocol based on sending all n bits (the whole x) of R_I to R_{II} and asking R_{II} to perform the comparison of x and y bit by bit is optimal. No multiple message exchange and no clever computation of R_I and R_{II} can help to decrease the communication complexity. If n is very large, for instance $n = 10^{16}$ (this corresponds to the memory size of 250000 DVDs), then the communication cost is too high. Moreover to reliably transfer 10^{16} bits without flipping or losing any particular bit using current Internet technologies is an almost unrealistic undertaking.

In what follows we design a randomized communication protocol that solves our "equality problem" within

$$4 \cdot \lceil \log_2(n) \rceil$$

communication bits[7]. We see that one can save an exponential complexity in this way. For instance, for $n = 10^{16}$ one needs to send only 256 bits instead of 10^{16} bits!

For the presentation of the randomized communication strategy it is more transparent to speak about a comparison of two numbers instead of the comparison of two bit sequences. Therefore we consider the sequences $x = x_1 \ldots x_n$ and $y = y_1 \ldots y_n$ of bits as binary representations of the integers.

[7] For a real number a, $\lceil a \rceil$ is the next larger integer. For instance, for $a = 4.15$, $\lceil a \rceil = 5$.

$$\mathbf{Number}(\boldsymbol{x}) = \sum_{i=1}^{n} 2^{n-i} \cdot x_i$$

and

$$\mathbf{Number}(\boldsymbol{y}) = \sum_{i=1}^{n} 2^{n-i} \cdot y_i$$

If these formulas do not mean anything to you anymore, and you have forgotten this secondary-school subject, you do not need to worry about this. The only important thing is to know that

Number(x) is a natural number represented by x, and that

$$0 \leq \text{Number}(x) \leq 2^n - 1.$$

Analogously,

$$0 \leq \text{Number}(y) \leq 2^n - 1.$$

For $n = 10^{16}$, these numbers can be really huge. Already for $n = 10^6 = 1000000$, the numbers can be around

$$2^{1000000}$$

and consist of about 300000 decimal digits.

Clearly, x and y are identical if and only if Number(x) = Number(y). Hence, we may focus on comparing Number(x) with Number(y) in our randomized protocol.

Exercise 6.1 For those who want to understand everything in detail. The sequence 10110 represents the integer

$$\begin{aligned}\text{Number}(10110) &= 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 \\ &= 1 \cdot 16 + 0 \cdot 8 + 1 \cdot 4 + 1 \cdot 2 + 0 \cdot 1 \\ &= 16 + 4 + 2 = 22.\end{aligned}$$

What number is represented by the bit sequence 101001? What is the binary representation of the integer 133?

The randomized communication protocol for inputs $x_1 \dots x_n$ and $y_1 \dots y_n$ corresponds to a random choice of a collection of deterministic communication strategies. The size of this collection equals the number of primes smaller than n^2.

In what follows, for each positive integer $m \geq 2$, we denote by

$$\text{PRIM}(m) = \{p \text{ is a prime} \mid p \leq m\}$$

the set of all primes in the interval from 1 to m and by

$$\text{Prim}(m) = |\text{PRIM}(m)|$$

the number of primes in PRIM(m).

In what follows we denote by

$$r = a \bmod b$$

the residue of the division $a : b$. For instance $2 = 14 \bmod 3$, because $14 : 3 = 4$ and the remainder is $r = 14 - 3 \cdot 4 = 2$.

If one considers the division $a : b$ in the framework of integers only, and the result is c and the remainder is $r < b$, then one can write

$$a = b \cdot c + r \ .$$

In our example for $a = 14$ and $b = 3$, the result of the division is $c = 4$ and the remainder is $r = 2$. Hence, one can write

$$14 = 3 \cdot 4 + 2,$$

where $r = 2 < 3 = b$.

Now, we are ready to present a randomized communication protocol for the comparison of x and y (better to say for comparing Number(x) with Number (y)).

Randomized communication protocol WITNESS for identity verification.

Initial situation: The computer R_I has n bits $x = x_1x_2\dots x_n$ (i.e., an integer Number(x), $0 \leq$ Number$(x) \leq 2^n - 1$).
The computer R_{II} has n bits $y = y_1y_2\dots y_n$ (i.e., an integer Number(y), $0 \leq$ Number$(y) \leq 2^n - 1$).

Phase 1: R_I chooses a prime p from PRIM(n^2) at random. Every prime from PRIM(n^2) has the same probability $1/\text{Prim}(n^2)$ to be chosen.

Phase 2: R_I computes the integer

$$s = \text{Number}(x) \bmod p$$

(i.e., the remainder of the division Number(x) : p) and sends the binary representations of

$$s \text{ and } p$$

to R_{II}.

Phase 3: After reading s and p, R_{II} computes the number

$$q = \text{Number}(y) \bmod p.$$

If $q \neq s$, then R_{II} outputs "unequal".
If $q = s$, then R_{II} outputs "equal".

Before analyzing the communication complexity and the reliability of WITNESS, let us illustrate the work of the protocol for a concrete input.

Example 6.1 Let $x = 01111$ and $y = 10110$.
Hence,

$$\text{Number}(x) = 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 8 + 4 + 2 + 1 = 15,$$
$$\text{Number}(y) = 1 \cdot 2^4 + 1 \cdot 2^2 + 1 \cdot 2^1 = 16 + 4 + 2 = 22$$

and

$$n = 5.$$

Consequently $n^2 = 25$ and

$$\mathrm{PRIM}(n^2) = \{2, 3, 5, 7, 11, 13, 17, 19, 23\}.$$

In Phase 1 the communication protocol WITNESS has the random choice from the collection of 9 primes of PRIM(25) and in this way it chooses one of 9 possible deterministic protocols.

Assume that R_I chooses the prime 5. In Phase 2 computer R_{II} computes

$$s = 15 \bmod 5 = 0$$

and sends the integers $p = 5$ and $s = 0$ to R_{II}. Then R_{II} computes

$$q = 22 \bmod 5 = 2.$$

Since $2 = q \neq s = 0$, the computer R_{II} gives the correct answer

"x and y are unequal".

Assume now that R_I chooses the prime 7 from PRIM(25) at random. Then the computer R_I computes

$$s = 15 \bmod 7 = 1$$

and sends the integers $p = 7$ and $s = 1$ to R_{II}. Then R_{II} computes

$$q = 22 \bmod 7 = 1.$$

Since $q = s$, the computer R_{II} gives the wrong answer

"x and y are equal".

We immediately observe that WITNESS may err because some of the random choices of primes can lead to a wrong answer. □

Exercise 6.2 Does there exist a prime in PRIM(23) different from 7, whose choice results in a wrong answer of WITNESS for $x = 01111$ and $y = 10110$?

Exercise 6.3 Let $x = y = 100110$ be the inputs. Does there exist a choice of a prime from PRIM(6^2) such that WITNESS provides the wrong answer "$x \neq y$"?

Exercise 6.4 Consider $x = 10011011$ and $y = 01010101$ as inputs. Estimate for how many primes from PRIM(8^2) WITNESS computes the wrong answer "$x = y$". What is the number of primes from PRIM(64) for which WITNESS provides the right answer "$x \neq y$"?

First, we analyze the communication complexity of WITNESS. The natural numbers Number(x) and Number(y) have the length of their binary representations bounded by n and so they are from the interval from 0 to $2^n - 1$. To save communication bits the communication of WITNESS consists of sending two integers smaller than n^2, namely p and the remainder of the division Number(x) : p. A natural number smaller than n^2 can be binary represented by

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

bits. The symbols $\lceil$ and $\rceil$ are used to denote the rounding to the next larger integer. For instance, $\lceil 2.3 \rceil = 3$, $\lceil 7.001 \rceil = 8$ and $\lceil 9 \rceil = 9$.

Since in WITNESS two numbers p and s are communicated, R_I can use exactly[8] $2\lceil \log_2 n \rceil$ bits for the representation of each one. In this way the number of communication bits for inputs of length n is always exactly

$$4 \cdot \lceil \log_2 n \rceil .$$

Let us see what that means for $n = 10^{16}$. As already mentioned, the best deterministic protocol cannot avoid the necessity of communicating at least

$$10^{16} \text{ bits}$$

for some inputs. Our protocol WITNESS always works within

$$4 \cdot \lceil \log_2(10^{16}) \rceil \leq 4 \cdot 16 \cdot \lceil \log_2 10 \rceil = 256 \text{ communication bits.}$$

The gap between communicating 10^{16} bits and 256 bits is huge. Even if it is possible to communicate 10^{16} bits in a reliable way the costs of sending 256 bits and of sending 10^{16} bits are incomparable. For this unbelievably large saving of communication costs we pay

[8] If the binary representation of p or s is shorter than $2\lceil \log_2 n \rceil$ one can add a few 0's at the beginning of the representation in order to get exactly this length.

by losing the certainty of always getting the correct result. The question of our main interest now is:

How large is the degree of unreliability we have used to pay for the huge reduction of the communication complexity?

The degree of uncertainty of computing the correct result is called the **error probability** in algorithmics. More precisely, the **error probability for two input strings x and y** is the probability

$$\mathbf{Error}_{WITNESS}(x, y)$$

that WITNESS computes a wrong answer on inputs x and y. For different input pairs (x, y) (i.e., for different initial situations) the error probabilities may differ. Our aim is to show that the error probability is very small for all[9] possible inputs x and y.

"What is the error probability for a given x and y and how can one estimate it?"

The protocol WITNESS chooses a prime for PRIM(n^2) for input strings x and y of length n at random. These choices decide whether the protocol provides the correct answer or not. Therefore we partition the set PRIM(n^2) into two parts. All primes that provide the correct answer for the input instance (x, y) we call

$$\textbf{good for } (x, y) \textbf{ .}$$

The prime 5 is good for (01111, 10110) as we have seen in Example 6.1.

The other primes whose choices result in the wrong answer for the problem instance (x, y) we call

$$\textbf{bad for } (x, y) \textbf{ .}$$

The prime 7 is bad for (01111, 10110) as was shown in Example 6.1.

[9] In computer science we have very high requirements on the reliability of randomized algorithms and systems. We force a high probability of getting the correct result for every particular problem instance. This is in contrast to the so-called stochastic procedures, for which one only requires that they work correctly with high probability on a statistically significant subset of problem instances.

Since each of the $\text{Prim}(n^2)$ many $\text{PRIM}(n^2)$ has the same probability of being chosen, we have

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

i.e., the error probability is the ratio between the number of bad primes for (x,y) in the urn (in $\text{PRIM}(n^2)$) and the number $\text{Prim}(n^2)$ of all primes in the urn. This is a simple consideration and we can convince the reader about it in a transparent way. Assume an urn with 15 balls. Assume that exactly 3 of these 15 balls are white. Then the probability of choosing a white ball at random is exactly $\frac{3}{15} = \frac{1}{5}$, if each ball has the same probability of being chosen. In another words, 20% of the balls in the urn are white and so one can expect that in a long sequence of repeated random experiments (choosing a ball at random and putting it back) a white ball is the result of the random choice in 20% of cases. Similarly, flipping a fair coin many times you may expect the number of flipped heads is approximately the same as the number of flipped tails. Analogously, if WITNESS chooses a prime from the 15 primes in $\text{PRIM}(7^2)$ for a given (x,y) at random and there are two bad primes for (x,y) among these 15 primes, the error probability for (x,y) is $2/15$.

The situation is depicted in Fig. 6.2. Our task is to show, for any instance (x,y) of our identity problem, that the set of bad primes for (x,y) is very small with respect to the size of $\text{PRIM}(n^2)$.

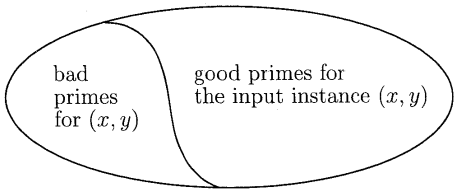

Fig. 6.2

How large is PRIM(m) for a positive integer m? One of the deepest and most important discoveries of mathematics[10] is the prime number theorem saying that

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

where $\ln m$ is the natural logarithm of m. The prime number theorem says that the primes are closely distributed among natural numbers, because one can find a prime approximately at each $(\ln m)$-th position. For our calculation we need only the known assertion

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

for all integers $m > 67$. Hence, we have

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

for all $n \geq 9$. Our next objective is to show that

for every problem instance (x,y), the number of bad primes for (x,y) is at most $n - 1$,

which is essentially smaller than $n^2/2 \ln n$.

Investigating the error probability of WITNESS, we distinguish two cases with respect to the relation between x and y.

Case 1 $x = y$ and so the right answer is "equal".

If $x = y$, then Number(x) =Number(y). For all primes p the equality

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

holds. Hence, it does not matter which p from PRIM(n^2) is chosen, $s = q$ must hold. In other words, if one divides two equal numbers by the same prime p, the residues of these two divisions must be the same. Hence, the protocol WITNESS computes the right answer "equal" for all primes in PRIM(n^2). Therefore

$$\text{Error}_{WITNESS}(x, y) = 0$$

[10] more precisely, of number theory

for all strings $x = y$ (for all problem instances (x, y)).

From the analysis of Case 1 we learned that a wrong answer can occur only for problem instances (x, y) with $x \neq y$.

Case 2 $x \neq y$ and so the right answer is "unequal". As we have already fixed for $(01111, 10110)$ in Example 6.1, the error probability can be different from 0 for problem instances (x, y) with $x \neq y$. The random choice of $p = 7$ results in the wrong answer "equal" for $x = 01111$ and $y = 10110$.

In what follows we investigate the properties of bad primes for a given (x, y) in order to prove a general upper bound $n - 1$ on the number of bad primes for every problem instance (x, y) with $x \neq y$ and $|x| = |y| = n$.

A prime p is bad for (x, y) if the residues of the divisions Number$(x) : p$ and Number$(y) : p$ equal each other, i.e., if

$$s = \text{ Number}(x) \bmod p = \text{ Number}(y) \bmod p.$$

The equality

$$s = \text{ Number}(x) \bmod p$$

means nothing other than

$$\text{Number}(x) = h_x \cdot p + s,$$

where h_x is the result of the division Number$(x) : p$ and $s < p$ is the remainder.

Analogously one can also write

$$\text{Number}(y) = h_y \cdot p + s,$$

where p is involved h_y times in Number(y) and $s < p$ is the remainder[11].

Assume Number(x) $\geq$Number(y) (in the opposite case when Number(y) $>$Number(x) the analysis can be finished in an analogous way). We compute the integer Number$(x)-$Number(y) as follows:

[11] For instance, for $x = 10110$ we have Number$(x) = 22$. For $p = 5$, Number$(x) = 22 = 4 \cdot 5 + 2$, where $h_x = 4$ and $s = 2$ is the remainder.

$$\begin{array}{rr} \text{Number}(x) & h_x \cdot p + s \\ -\text{Number}(y) & -h_y \cdot p - s \\ \hline \text{Dif}(x,y) & h_x \cdot p - h_y \cdot p \end{array}$$

In this way we obtain:

$$\text{Dif}(x,y) = \text{Number}(x) - \text{Number}(y) = h_x \cdot p - h_y \cdot p = (h_x - h_y) \cdot p.$$

The final conclusion is that the prime p divides the number

$$\text{Dif}(x,y) = \text{Number}(x) - \text{Number}(y).$$

What did we learn from this calculation?

A prime p is bad for (x,y) if and only if p divides Dif(x,y).

Why is this knowledge helpful? First of all, we have found a fast way of recognizing bad primes.

Example 6.2 Assume R_I has the string $x = 1001001$ and R_{II} has the string $y = 0101011$, both strings of length $n = 7$. The task is to estimate the bad primes for $(x,y) = (1001001, 0101011)$.

First we estimate

$$\begin{aligned} \text{PRIM}(n^2) &= \text{PRIM}(49) \\ &= \{2,3,5,7,11,13,17,19,23,29,31,37,41,43,47\} \end{aligned}$$

as the set of primes from which WITNESS chooses a prime at random. Following our new observation, the bad primes for $(1001001, 0101011)$ are exactly those primes that divide the difference

$$\begin{aligned} \text{Dif}(1001001, 0101011) &= \text{Number}(1001001) - \text{Number}(0101011) \\ &= 73 - 43 = 30. \end{aligned}$$

We immediately see that 2, 3, and 5 are the only primes in PRIM(49) that divide 30, and so 2, 3, and 5 are the only bad primes for $(1001001, 0101011)$. In this way we obtain

$$\text{Error}_{WITNESS}(1001001, 0101011) = \frac{3}{\text{Prim}(49)} = \frac{3}{15} = \frac{1}{5}.$$

□

Exercise 6.5 Find all bad primes for the following problem instances.

(i) $(01010, 11101)$
(ii) $(110110, 010101)$
(iii) $(11010010, 01101001)$.

Now, the remaining question is the following one:

> "How to use the knowledge that bad primes for (x, y) divides $\text{Dif}(x, y)$ to bound their number?"

Since both numbers $\text{Number}(x)$ and $\text{Number}(y)$ are smaller than 2^n, we have

$$\text{Dif}(x, y) = \text{Number}(x) - \text{Number}(y) < 2^n.$$

We show that a number smaller than 2^n cannot be divisible by more than $n - 1$ primes[12]. To do so we apply another famous theorem of mathematics, the so-called fundamental theorem of arithmetics. This theorem says that

> *each positive integer larger than 1 can be unambiguously expressed as a product of primes.*

For instance,

$$5940 = 2 \cdot 2 \cdot 3 \cdot 3 \cdot 3 \cdot 5 \cdot 11 = 2^2 \cdot 3^3 \cdot 5 \cdot 11$$

and so $2, 3, 5$, and 11 are the so-called prime factors of 5940. In other words, the number 5940 is divisible by exactly 4 primes $2, 3, 5$, and 11.

Let $p_1, p_2, p_3, \ldots, p_k$ be all prime factors of our number $\text{Dif}(x, y)$ and let $p_1 < p_2 < p_3 < \ldots < p_k$. In general, we see that $p_i > i$ for each i. Hence, we have

[12] Mathematicians would say a number smaller than 2^n can have at most $n-1$ prime factors.

$$\mathrm{Dif}(x,y) = p_1^{i_1} \cdot p_2^{i_2} \cdot p_3^{i_3} \cdot \ldots \cdot p_k^{i_k}$$

for $i_j \geq 1$ for all j from 1 to k, and we can write

$$\begin{aligned}\mathrm{Dif}(x,y) &\geq p_1 \cdot p_2 \cdot p_3 \cdot \ldots \cdot p_k \\ &> 1 \cdot 2 \cdot 3 \cdot \ldots \cdot k \\ &= k!\end{aligned}$$

Since $\mathrm{Dif}(x,y) < 2^n$ and $\mathrm{Dif}(x,y) > k!$, we obtain

$$2^n > k! \ .$$

Since $n! > 2^n$ for $n \geq 4$, k must be smaller than n and in this way we have obtained the stated aim that

$$k \leq n-1,$$

i.e., that the number of prime factors of $\mathrm{Dif}(x,y)$ is at most $n-1$ and so the number of bad primes for (x,y) is at most $n-1$.

Using this result we can upperbound the error probability of WITNESS on (x,y) for $x \neq y$ as follows. For every problem instance (x,y) of length $n \geq 9$,

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

Hence, the error probability of WITNESS for problem instances (x,y) with $x \neq y$ is at most $2\ln/n$, which is for $n = 10^{16}$ at most

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

In Example 6.2 we saw that the error probability can be relatively high, 1/5. This is because the analyzed *error probability decreases*

with growing n. Therefore, for small problem instances of a few thousand bits, one recommends to compare x and y in a deterministic way, since the costs for such a comparison are anyway small. Only for larger problem instances is it profitable to apply the randomized protocol WITNESS.

Exercise 6.6 What are the upper bounds on the error probability of WITNESS for input sizes $n = 10^3$ and $n = 10^4$? Up to which problem instance size would you recommend using the protocol WITNESS?

Exercise 6.7 Estimate the exact error probabilities of WITNESS for the following problem instances (x, y):

(i) $(00011011, 10101101)$
(ii) $(011000111, 000111111)$
(iii) $(0011101101, 0011110101)$.

Recently, we have seen a real miracle. A randomized system can efficiently and reliably perform tasks that are not tractable for deterministic systems. We know many hard problems that cannot be practically solved because the fastest known algorithms for them require an exponential complexity. Several such problems have to be solved daily and we are able to perform them only due to randomized algorithms. The whole area of e-commerce includes implementations of randomized algorithms in the software used. In fact, bad news for people searching for absolute reliability! But, for those who know the risk and can calculate it and reduce it, randomized systems provide what they are looking for.

Now it is time to at least partially reveal the nature behind the magic of randomization. We want to understand why randomization can cause such magnificent effects, and thus examine applications of randomization in algorithmics as a natural phenomenon.

We start our considerations by expressing doubts about the success of randomization presented above. We saw that the randomized protocol WITNESS works efficiently and with a high reliability. But is it really true that the best deterministic protocol for the identity problem cannot avoid sending almost n bits for most problem instances, and that for some problem instances the exchange of n bits is necessary? We did not prove it here. Why may one have

doubts with respect to these claims? Intuitively, these assertions look suspicious.

Let us go into detail and look at Fig. 6.2. We rename the good primes for (x, y) to **witnesses of the difference between x and y** (for short).We say that a prime p is a **witness of "$x \neq y$"** or a **witness for (x, y)** if

$$\text{Number}(x) \bmod p \neq \text{Number}(y) \bmod p.$$

A prime q is a non-witness of "$x \neq y$" if

$$\text{Number}(x) \bmod p = \text{Number}(y) \bmod p.$$

Thus, the good primes for (x, y) are witnesses for (x, y) and the bad primes are non-witnesses for (x, y). Using these new terms one can view the work of the protocol WITNESS as searching for a witness of "$x \neq y$". If WITNESS chooses a witness of "$x \neq y$" at random, then WITNESS proves the fact "$x \neq y$" and reliably provides the correct answer "unequal". If the protocol chooses a non-witness for (x, y) with $x \neq y$ at random, one cannot avoid the wrong answer "equal". The protocol WITNESS works correctly with high probability, because almost all **candidates for witnessing** in PRIM(n^2) are witnesses. The group of non-witnesses is so small relative to Prim(n^2) that the probability of choosing a non-witness at random is very small. And now one can present the following doubt.

> *If almost all elements of the set of the candidates for witnessing are witnesses for (x, y), why is it not possible to find a witness among them in an efficient, deterministic way and then to solve the task within a short communication? To fix a witness for (x, y) in a deterministic way means to exchange the randomness for a deterministic control and so to design an efficient deterministic communication protocol for the comparison of x and y.*

How to explain that this at first glance plausible idea does not work? Clearly, we can present a formal mathematical proof that there is no efficient deterministic communication protocol for this

task. But that is connected with two problems. First, despite the fact that this proof is not too hard for mathematicians and computer scientists, it requires knowledge that cannot be presupposed by a general audience. Secondly, understanding of the arguments of the proof does not necessarily help us to realize why the proposed idea does not work. You would only know that designing an efficient deterministic protocol for this task is impossible, but the main reason for that would remain hidden. Because of that we prefer to present only an idea in a way that increases our understanding of randomized control.

There is no efficient[13] deterministic algorithm for finding a witness for a given (x, y), because from the point of view of the computers R_I and R_{II}

> *the witnesses are "randomly" distributed among the candidates for witnessing.*

What does this mean? If you know x but only very little about y, you cannot compute a witness for (x, y), even when several attempts are allowed. This is the consequence of the fact that the witnesses are completely differently distributed in the candidate set PRIM(n^2) for distinct inputs of length n. There is no rule for computing a witness for a partially unknown input. Hence, from the point of view of R_I and R_{II} the soup in the pot containing all candidates (Fig. 6.3) looks as a chaos (a really chaotic mix of witnesses and non-witnesses) And this is the kernel of the success of randomness.

For several tasks one can solve the problem efficiently if a witness is available. In a real application, nobody provides a witness for us for free. But if one can build a set of candidates for witnessing in such a way that this set contains many witnesses, it is natural to search for witnesses at random. The chaotic distribution of witnesses is no obstacle for a random choice. But if the distribution of witnesses among candidates is really random, there does not exist any efficient deterministic procedure for finding a witness in this set.

[13] with respect to communication complexity

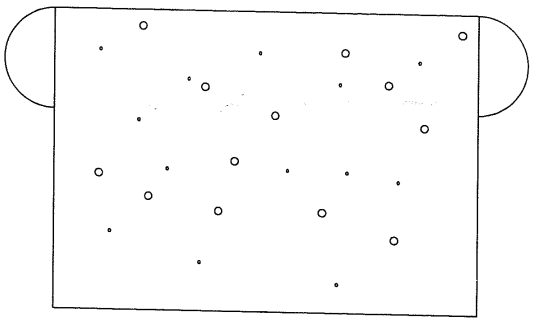

Fig. 6.3: A chaotic mix of witnesses and non-witnesses

Searching for efficient randomized algorithms one often applies the idea presented above. We try to find a kind of witness that fulfills the following requirements:

(i) If one has a witness for an input instance, then one can compute the correct result efficiently in a deterministic way.

(ii) If one has a candidate for witnessing, one can, for a given input, efficiently verify whether it is a witness or not.

(iii) There is an abundance of witnesses in the set of candidates.

Based on requirement (iii) this method for the design of randomized algorithms is called the method of the **abundance of witnesses**. Hence, to have many witnesses may be very useful. How one can even increase the abundance of witnesses is the topic of the next section.

6.4 What to Do When a Customer Requires a Very High Reliability?

The whole history of Man is strongly related to the search for certainty. Who are we? What are we looking for? What to do in order to get a guarantee for a "nice" or at least any future? Life and science educate us that the search for an absolute certainty is a nonsense that can even be dangerous. Striving for a non-existing

absolute certainty means to do without many things and activities and often means to run up a blind alley. Frustration and depression are typical consequences of such meaningless effort. But if one accepts uncertainty and with it also randomness as an inevitable part of life and learn to live with this, instead of frustration one discovers new possibilities as a reward for giving up non-existing ideals. The blind alleys are not blind anymore and the future is open for a variety of possibilities. That was exactly the case in the example presented in the previous section. The situation looked hopeless, because each protocol solving the task has communication costs that are too large. Exchanging the hypothetical absolute reliability of deterministic protocols for a practical reliability of randomized protocols, we found a good solution for our communication task. In several similar situations one is unable to find witnesses that have as high abundance as in our example. Sometimes only a fraction of the candidates for witnessing are witnesses. In such cases the error probability can grow to an extent that is not acceptable in the applications. The aim of this section is to show how one can master such situations.

Let us start with our example of the randomized communication protocol for the equality of two strings. For $n = 10^{16}$, the error probability is at most $0.37 \cdot 10^{-14}$. Assume we have a customer for whom the reliability of the protocol is extremely important and so he is asking for a still higher degree of guarantee for the correct answer. For instance, the customer is saying:

> *"I require the reduction of the error probability to such a small value, that if one applies the protocol t times for t equal to the product of the age of the Universe in seconds and the number of protons in the Universe, then the probability of getting at least one wrong answer in one of these t applications is smaller than one in a million."*

For sure, that is a very strange requirement. Can we satisfy this exaggerated request of the customer without any huge increase in communication complexity? The answer is "YES", and in order to show how to do it we present a simple consideration from probability theory.

Assume we have a fair die. Rolling a die is considered a random experiment with 6 possible outcomes $1, 2, 3, 4, 5$, and 6, in which the probability of each concrete outcome is the same, i.e. $1/6$.

Assume that the only "bad" result for us is "1" and all other outcomes of the experiment are welcome. Hence, the probability of getting an undesirable outcome is $1/6$. Next, we change the experiment and the definition of an undesirable outcome. We consider rolling the die five times and the only undesirable outcome of this new experiment is to get "1" in all five rolls. In other words, we are satisfied if in at least one of these five simple experiments of rolling a die the outcome is different from "1". What is the probability of getting the undesirable outcome? To get the answer, we have to estimate the probability of rolling "1" five times, one behind another. Because all five die rolls are considered as independent[14] experiments, one calculates as follows. One "1" is rolled with probability $1/6$. Two "1" are rolled one behind another with probability

$$\frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}.$$

The probability of rolling five "1" one behind each other is exactly

$$\frac{1}{6} \cdot \frac{1}{6} \cdot \frac{1}{6} \cdot \frac{1}{6} \cdot \frac{1}{6} = \frac{1}{6^5} = \frac{1}{7776}.$$

What does this consideration have in common with the error probability of randomized communication protocols? A lot. One application of the randomized protocol can be viewed as a random experiment, in which the undesirable outcome corresponds to the random choice of a non-witness for a given input. Our analysis of the error probability of the randomized protocol shows the probability of getting this undesirable outcome, and so the wrong answer is at most

$$\frac{2 \ln n}{n}.$$

Now, we adapt the idea with the repeated die rolls. If one chooses 10 primes at random, then the only undesirable outcome is that all

[14] The exact meaning of the term independent is presented in [Hro04b].

10 primes are non-witnesses for the given input (x, y). If at least one prime is a witness for (x, y), one gets the correct answer with certainty. This consideration results in the following randomized communication protocol.

WITNESS (10)

Initial situation: Computer R_I has n bits $x = x_1 \dots x_n$ and computer R_{II} has n bits $y = y_1 \dots y_n$.

Phase 1: R_I chooses 10 primes

$$p_1, p_2 \dots p_{10} \text{ from PRIM}(n^2) \text{ at random.}$$

Phase 2: For each i from 1 to 10 R_I computes

$$s_i = \text{Number}(x) \bmod p_i$$

and sends the binary representations of

$$p_1, p_2 \dots p_{10}, s_1, s_2 \dots s_{10}$$

to R_{II}.

Phase 3: After reaching the 20 numbers

$$p_1, p_2 \dots p_{10}, s_1, s_2 \dots s_{10},$$

the computer R_{II} computes

$$q_i = \text{Number}(y) \bmod p_i$$

for all $i = 1, 2 \dots 10$.
If there exists at least one i from $1, 2, \dots, 10$ such that $q_i \neq s_i$, then R_{II} knows with certainty that $x \neq y$ and outputs "unequal".
If $q_i = s_j$ for all 10 j from $1, 2, \dots, 10$, then either $x = y$ or $x \neq y$ and none of the 10 primes chosen is a witness of "$x \neq y$". In this case the protocol outputs "$x = y$".

What do we observe? If $x = y$ then WITNESS(10) outputs the correct result "equal" with certainty, exactly as WITNESS did too. If $x \neq y$, then WITNESS(10) can provide the wrong output only if none of the 10 primes chosen is a witness for (x, y). It suffices that at least one of these 10 primes is a witness for (x, y), say p_4, and R_{II} knows that $x \neq y$ because $s_4 \neq p_4$ and so p_4 witnesses the difference between x and y. If the probability of choosing a non-witness for (x, y) in one attempt is at most $2 \ln /n$, then the probability of choosing 10 non-witnesses at random one behind the other is at most

$$\left(\frac{2 \ln n}{n}\right)^{10} = \frac{2^{10} \cdot (\ln n)^{10}}{n^{10}}.$$

For $n = 10^{16}$ this probability is at most

$$\frac{0.4714}{10^{141}}.$$

The product of the age of the Universe in seconds and the number of protons in the Universe is smaller than

$$10^{99}.$$

We omit presenting a mathematical calculation that shows that getting a wrong output in 10^{99} applications of WITNESS(10) is smaller than 1 in 10^{30}.

In this way we have reduced the error probability strongly below any reasonable limits and so we fulfill all reliability requirements that have ever been formulated in practice. And we paid very little for this wonderful gain. Computing with 10 primes instead of with one increases the communication costs by the multiplicative factor 10. Hence, the communication complexity of WITNESS(10) is

$$40 \cdot \lceil \log_2 n \rceil .$$

These communication costs are negligible. For instance, for $n = 10^{16}$, WITNESS(10) communicates

2560 bits

only.

Since WITNESS(10) can be viewed as 10 repetitions of WITNESS, we say in the formal terms of algorithmics that

> *the complexity grows linearly with the number of repetitions (attempts to find a witness), while the error probability is reduced with an exponential speed-up in the number of repetitions.*

In fact, this situation belongs among the most favorable situations that one can have when searching for a solution to an algorithmic task.

Our randomized protocol WITNESS(10) is in fact more reliable than anything we could associate with the notion of reliability. There are two main reasons for that. First, already the protocol WITNESS provides a high degree of reliability. Secondly, the method of "repeated experiment" (of repeated random search for a witness) can essentially reduce the error probability even in cases when the probability of choosing a witness at random is small. The following exercises provide the reader with the possibility of applying the ideas presented above and so deepen one's imagination of the power of randomized computation.

Exercise 6.8 Let A be an algorithmic task for which every known algorithm has complexity at least 2^n. Let A be a decision problem for which only answers YES and NO are possible and so the task is to decide whether a problem instance x has a special property or not. Assume one has a kind of witness for x of size n with the following properties:

(i) If z is a witness for x, then one can verify in $10 \cdot n^2$ operations that x has the desired property. If z is not a witness for x, then the algorithm is unable to recognize whether x has the desired property or not. The fact that z is not a witness for x is recognized by the algorithm in time $10 \cdot n^2$, too.
(ii) At least half of the witness candidates are witnesses, i.e. the probability of choosing a witness at random is at least $1/2$.

The tasks are as follows:

1. Assume a randomized algorithm that performs 10 choices from the set of witness candidates at random in order to get a witness for the given input. Bound the error probability of this algorithm and analyze its computational complexity.

2. A customer is asking for an algorithm that solves the decision problem with error probability at most 1 in 10^9. How many repetitions of the original algorithm are necessary to reduce the error probability below this limit? What is the complexity of the resulting algorithm?

Exercise 6.9 Solve Exercise 6.8 if the assumption (i) is satisfied, and instead of (ii) we have the following assumptions:

(i) The portion of witnesses in the set of candidates is exactly 1/6.
(ii) The portion of witnesses in the set of candidates is only 1 in n, where n is the size of the problem instance (input).

6.5 What Are Our Main Discoveries Here?

If, for some tasks, one strongly requires to compute the correct results with absolute certainty, one can walk up a dead-end street. All algorithms that assume a theoretically absolute reliability demand so much computer work that they are intractable (not practicable). We know several computing tasks for which the best known algorithms need more time than the age of the Universe and more energy than the energy of the whole known Universe. To solve such problems is beyond the limits of physically doable. And then we discover that there is a magic way to solve these hard tasks. We do without the absolute reliability that does not exist anyway in practice and force a high probability of computing the correct result for each problem instance instead. We are allowed to require such a small error probability that one can speak about exchanging a hypothetical absolute reliability for practical certainty of getting the right result. Due to this fictitious reduction of our reliability requirements, we can save a huge amount of computer work. There exist problems for which a clever use of randomness causes a big jump from a physically undoable amount of work to a few seconds work on a common PC.

We learned two important paradigms for the design of randomized systems. The method of abundance of witnesses touches upon the deepest roots of the computational power of randomness. A witness for a problem instance is a piece of additional information

that helps us to efficiently solve the problem instance that cannot be solved efficiently without it. The method promises successful applications if one can find a kind of witness, such that there is an abundance of witnesses in the set of candidates for witnessing. The idea is then to use repeated random sampling (choice) from the set of candidates in order to get a witness efficiently with high probability, although one does not know any efficient way of computing a witness using deterministic algorithms. The question of whether there are problems that can be solved efficiently by applying randomization, but which cannot be solved efficiently in a deterministic way, can be expressed as follows:

> *If all kinds of sets of candidates for witnessing for a hard problem have the witnesses randomly distributed in the candidate set then the problem is efficiently solvable using a randomized algorithm, but not efficiently solvable using deterministic algorithms.*

The second approach we applied for the design of randomized algorithms is the *paradigm of increasing success probability (of a fast reduction of error probability) by repeating several random computations on the same problem instance.* This paradigm is often called the *paradigm of amplification* (of the success probability). Repeating WITNESS 10 times results in the protocol WITNESS(10), whose error probability tends to 0 with an exponential speed in the number of repetitions, but whose complexity grows only linearly in the number of repetitions. The situation is not always so favorable, but usually we are able to satisfy the reliability requirements of customers without paying too much with additional computer work.

The textbook "The Design and Analysis of Randomized Algorithms" [Hro04b, Hro05] provides a transparent introduction to methods for designing efficient randomized systems. On one side it builds the intuition for the subject in small steps, and on the other side it consistently uses the rigorous language of mathematics in order to provide complete argumentation for all claims related to the quality of designed algorithms in terms of efficiency and error

probability. The most exhaustive presentation of this topic is available in Motwani and Raghavan [MR95]. But this book is written for specialists and so it is not easily read by beginners. A delicacy for gourmets is the story about designing algorithms for primality testing by Dietzfelbinger [Die04], where the method of abundance of witnesses is applied several times. Further applications for designing randomized communication protocols are available in [KN97, Hro97]. Here, the mathematically provable advantages of randomness over determinism are presented. Unfortunately, because many highly nontrivial mathematical arguments are considered, these books are not accessible to a broad audience.

Solutions to Some Exercises

Exercise 6.2 The prime 7 is the only prime in PRIM(25) whose choice in the first step of WITNESS results in a wrong output "equal" for the input $x = 01111$ and $y = 10010$.

Exercise 6.5 (i) Let $x = 01010$ and $y = 11101$. Hence, $n = 5$ and we consider primes from PRIM(5^2). The integers represented by x and y are:

$$\text{Number}(01010) = 2^3 + 2^1 = 8 + 2 = 10$$
$$\text{Number}(11101) = 2^4 + 2^3 + 2^2 + 2^0 = 16 + 8 + 4 + 1 = 29.$$

To find the bad primes from PRIM(25) for x and y, we do not need to test all primes from PRIM(25). We know that every bad prime for x and y divides the difference

$$\text{Number}(11101) - \text{Number}(01010) = 29 - 10 = 19$$

The number 19 is a prime and so 19 is the only prime in PRIM(25) that 19 divides. Hence, 19 is the only bad prime for 01010 and 11101.

Exercise 6.7 (i) To calculate the error probability of WITNESS for $x = 00011011$ and $y = 10101101$, we need to estimate the cardinality of PRIM(8^2).

$$\text{PRIM}(64) = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61\}$$

and so $|\text{PRIM}(64)| = 18$. The integers represented by x and y are:

$$\text{Number}(x) = 2^4 + 2^3 + 2^1 + 2^0 = 16 + 8 + 2 + 1 = 27$$
$$\text{Number}(y) = 2^7 + 2^5 + 2^3 + 2^2 + 2^0 = 128 + 32 + 8 + 4 + 1 = 173.$$

Hence, $\text{Number}(y) - \text{Number}(x) = 173 - 27 = 146$. The factorization of the number 146 is:

$$146 = 2 \cdot 73 \, .$$

Thus, the integers 2 and 73 are the only primes that divide the difference

$$\text{Number}(y) - \text{Number}(x) \ .$$

The prime 73 does not belong to PRIM(64) and so the prime 2 is the only bad prime for (x, y) in PRIM(64). Hence, the error probability of the protocol WITNESS for $(01010, 11101)$ is exactly $1/18$.

Exercise 6.8 The algorithm halts when it finds a witness for x. In the worst case, the algorithm performs 10 attempts and each attempt costs at most $10 \cdot n^2$ operations for checking whether the chosen candidate is a witness for x. Therefore the worst-case complexity of the randomized algorithm is $20 \cdot 10 \cdot n^2$. If the input x does not have the property we are searching for, the algorithm provides the right answer "NO" with certainty. If x has the property, then the algorithm can output the wrong answer "NO" only when it chooses 20 times a non-witness for x. We assume (assumption (ii) of the exercise) that the probability of choosing a non-witness at random in an attempt is at most $1/2$. Hence, the probability of choosing no witness in 20 attempts is at most

$$\left(\frac{1}{2}\right)^{20} = \frac{1}{2^{20}} = \frac{1}{1048576} \leq 0.000001 \ .$$

In this way we determined that the error probability is smaller than 1 in one million.

How many attempts are sufficient in order to reduce the error probability below $1/10^9$? We know that $2^{10} = 1024 > 1000$. Hence,

$$2^{30} = (2^{10})^3 > 1000^3 = 10^9 \ .$$

Thus, 30 attempts suffices for getting an error probability below 1 in one million.