



# **Chapter 7**

# **Randomization**

**Algorithm Theory**  
**WS 2016/17**

**Fabian Kuhn**

## Randomized Algorithm:

- An algorithm that uses (or can use) **random coin flips** in order to make decisions

**We will see:** **randomization** can be a **powerful tool** to

- Make algorithms **faster**
- Make algorithms **simpler**
- Make the analysis simpler
  - Sometimes it's also the opposite...
- Allow to **solve problems (efficiently)** that cannot be solved (efficiently) without randomization
  - True in some computational models (e.g., for distributed algorithms)
  - Not clear in the standard sequential model

# Contention Resolution

A simple starter example (from distributed computing)

- Allows to introduce important concepts
- ... and to repeat some basic probability theory

## Setting:

- <sup>nodes / machines</sup>  $n$  processes, 1 resource  
(e.g., shared database, communication channel, ...)
- There are time slots 1,2,3, ...
- In each time slot, only one client can access the resource
- All clients need to regularly access the resource
- If client  $i$  tries to access the resource in slot  $t$ :
  - Successful iff no other client tries to access the resource in slot  $t$

## Algorithm Ideas:

- Accessing the resource deterministically seems hard
  - need to make sure that processes access the resource at different times
  - or at least: often only a single process tries to access the resource
- **Randomized solution:**  
In each time slot, each process tries with **probability  $p$** .

## Analysis:

- How large should  $p$  be?
- How long does it take until some process  $i$  succeeds?
- How long does it take until all processes succeed?
- What are the probabilistic guarantees?

# Analysis

$i \in \{1, \dots, n\}$   
 $t = 1, 2, \dots$

$A, B, C$  indep  
 $\mathbb{P}(A \cap B \cap C) = \mathbb{P}(A) \cdot \mathbb{P}(B) \cdot \mathbb{P}(C)$



## Events:

- $\mathcal{A}_{i,t}$ : process  $i$  **tries to access** the resource in time slot  $t$ 
  - Complementary event:  $\overline{\mathcal{A}_{i,t}}$

$$\mathbb{P}(\mathcal{A}_{i,t}) = p, \quad \mathbb{P}(\overline{\mathcal{A}_{i,t}}) = 1 - p$$

- $\mathcal{S}_{i,t}$ : process  $i$  is **successful** in time slot  $t$

$$\mathcal{S}_{i,t} = \mathcal{A}_{i,t} \cap \left( \bigcap_{j \neq i} \overline{\mathcal{A}_{j,t}} \right) \quad \mathcal{A}_{i,t}, \overline{\mathcal{A}_{j,t}} \text{ indep.}$$

- **Success probability** (for process  $i$ ):

$$\mathbb{P}(\mathcal{S}_{i,t}) = \mathbb{P}(\mathcal{A}_{i,t}) \cdot \prod_{j \neq i} \mathbb{P}(\overline{\mathcal{A}_{j,t}}) = p(1-p)^{n-1}$$

choose  $p$  to maximize  $\mathbb{P}(\mathcal{S}_{i,t})$

Fixing  $p$   $\lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x$

- $\mathbb{P}(\mathcal{S}_{i,t}) = p(1 - p)^{n-1}$  is maximized for

$$\underline{\underline{p = \frac{1}{n}}} \quad \Rightarrow \quad \mathbb{P}(\mathcal{S}_{i,t}) = \underbrace{\frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1}}.$$

- **Asymptotics:**

$$\text{For } n \geq 2: \quad \underbrace{\frac{1}{4}} \leq \underbrace{\left(1 - \frac{1}{n}\right)^n}_{\rightarrow \frac{1}{e}} < \frac{1}{e} < \underbrace{\left(1 - \frac{1}{n}\right)^{n-1}}_{\rightarrow \frac{1}{e}} \leq \underline{\underline{\frac{1}{2}}}$$

- **Success probability:**

$$\underline{\underline{\frac{1}{en}}} < \underline{\underline{\mathbb{P}(\mathcal{S}_{i,t})}} \leq \underline{\underline{\frac{1}{2n}}}$$

# Time Until First Success $q := \mathbb{P}(\mathcal{S}_{i,t}) > \frac{1}{en}$

Random Variable  $T_i$ : time until first success of proc.  $i$

- $T_i = t$  if proc.  $i$  is successful in slot  $t$  for the first time

- **Distribution:**

$$\mathbb{P}(T_i=1)=q, \mathbb{P}(T_i=2)=(1-q)q, \mathbb{P}(T_i=t)=(1-q)^{t-1} \cdot q$$

- $T_i$  is **geometrically distributed** with parameter

$$\underline{q} = \mathbb{P}(\mathcal{S}_{i,t}) = \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} > \frac{1}{en}.$$

- **Expected time** until first success:

$$\mathbb{E}[T_i] = \frac{1}{\underline{q}} < \underline{en}$$

# Time Until First Success

**Failure Event  $\mathcal{F}_{i,t}$ :** Process  $i$  does not succeed in time slots  $1, \dots, t$

$$\mathcal{F}_{i,t} = \bigcap_{t'=1}^t \overline{\mathcal{S}_{i,t'}}$$

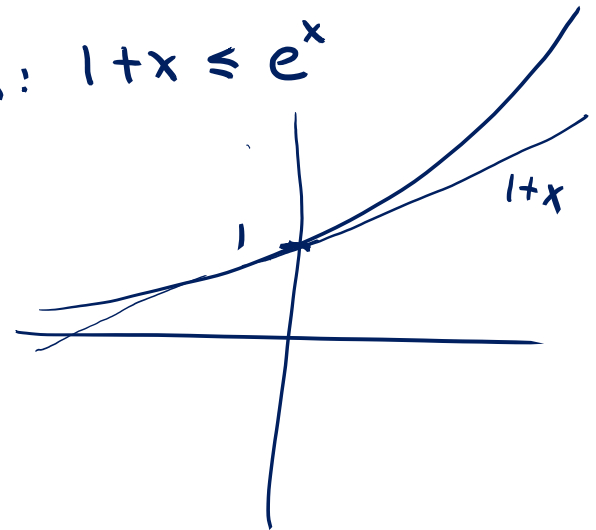
- The events  $\mathcal{S}_{i,t}$  are independent for different  $t$ :

$$\mathbb{P}(\mathcal{F}_{i,t}) = \mathbb{P}\left(\bigcap_{r=1}^t \overline{\mathcal{S}_{i,r}}\right) = \prod_{r=1}^t \mathbb{P}(\overline{\mathcal{S}_{i,r}}) = (1 - \mathbb{P}(\mathcal{S}_{i,r}))^t$$

$$\forall x \in \mathbb{R}: 1+x \leq e^x$$

- We know that  $\mathbb{P}(\mathcal{S}_{i,r}) > 1/en$ :

$$\mathbb{P}(\mathcal{F}_{i,t}) < \underbrace{\left(1 - \frac{1}{en}\right)^t}_{< e^{-1/en}} < \underline{\underline{e^{-t/en}}}$$





# Time Until First Success

No success by time  $t$ :  $\mathbb{P}(\mathcal{F}_{i,t}) < e^{-t/en}$   $e^{c \ln n} = (e^{\ln n})^c$

$t = \underline{\underline{en}}$ :  $\mathbb{P}(\mathcal{F}_{i,t}) < \underline{\underline{1/e}}$

- Generally if  $t = \underline{\underline{\Theta(n)}}$ : constant success probability

$t \geq \underline{\underline{en}} \cdot \underline{\underline{c}} \cdot \ln n$ :  $\mathbb{P}(\mathcal{F}_{i,t}) < 1/e^{c \cdot \ln n} = \underline{\underline{1/n^c}}$


- For success probability  $1 - 1/n^c$ , we need  $t = \Theta(n \log n)$ .
- We say that  $i$  succeeds with high probability in  $O(n \log n)$  time.

with prob.  $1 - \frac{1}{n^c}$   
for every const.  $c$

|  
hidden const.  
depends on  $c$

# Time Until All Processes Succeed $\mathcal{F}_{i,t}$

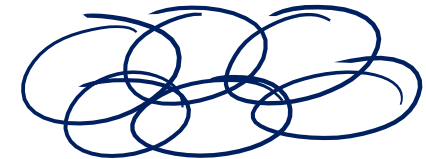
**Event  $\mathcal{F}_t$ :** some process has not succeeded by time  $t$



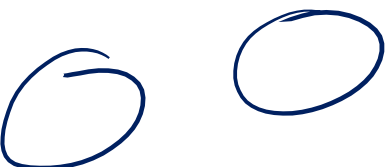
$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) - \mathbb{P}(A \cap B) \leq \mathbb{P}(A) + \mathbb{P}(B)$$

$$\mathcal{F}_t = \bigcup_{i=1}^n \mathcal{F}_{i,t}$$

$$\mathbb{P}(\mathcal{F}_{i,t}) < e^{-t/en}$$



**Union Bound:** For events  $\mathcal{E}_1, \dots, \mathcal{E}_k$ ,



$$\mathbb{P}\left(\bigcup_i \mathcal{E}_i\right) \leq \sum_i \mathbb{P}(\mathcal{E}_i)$$

$$\mathbb{P}(\overline{\mathcal{F}_{i,t}} \cap \overline{\mathcal{F}_{j,t}}) = 0$$

Probability that not all processes have succeeded by time  $t$ :

$$\mathbb{P}(\mathcal{F}_t) = \mathbb{P}\left(\bigcup_{i=1}^n \mathcal{F}_{i,t}\right) \leq \sum_{i=1}^n \mathbb{P}(\mathcal{F}_{i,t}) < n \cdot e^{-t/en}.$$

# Time Until All Processes Succeed

**Claim:** With high probability, all processes succeed in the first  
 $O(n \log n)$  time slots.

Proof:

- $\mathbb{P}(\mathcal{F}_t) < n \cdot e^{-t/en}$
- Set  $t = \lceil en \cdot (c + 1) \ln n \rceil$

$$\mathbb{P}(\mathcal{F}_t) < n e^{-(c+1) \ln n} = n \left( e^{\ln n} \right)^{-(c+1)} = n \cdot \frac{1}{n^{c+1}} = \frac{1}{n^c}$$

$$\mathbb{P}(\overline{\mathcal{F}_t}) > 1 - \frac{1}{n^c}$$

Remark:  $\Theta(n \log n)$  time slots are necessary for all processes to succeed with reasonable probability

# Primality Testing

**Problem:** Given a natural number  $n \geq 2$ , is  $n$  a prime number?

## Simple primality test:

1. **if**  $n$  is even **then**
2.     **return** ( $n = 2$ )
3. **for**  $i := 1$  **to**  $\lfloor \sqrt{n}/2 \rfloor$  **do**
4.     **if**  $2i + 1$  divides  $n$  **then**
5.         **return false**
6. **return true**

input size:  $O(\log n)$

time:  $O(\sqrt{n})$

exp. in size of input

- **Running time:**  $O(\sqrt{n})$

# A Better Algorithm?



- How can we test primality efficiently?
- We need a little bit of basic number theory...

**Square Roots of Unity:** In  $\mathbb{Z}_p^*$ , where  $p$  is a prime, the only solutions of the equation  $\underline{x^2} \equiv \underline{1} \pmod{p}$  are  $x \equiv \pm 1 \pmod{p}$

$$\mathbb{Z}_p^* = \{1, \dots, p-1\} \quad x^2 \equiv 1 \pmod{p}$$

$$x^2 - 1 \equiv 0 \pmod{p}$$

$$(x+1)(x-1) \equiv 0 \pmod{p} \iff (x+1)(x-1) = \overset{\text{integer}}{c} \cdot p$$

not true if  $p$  is not a prime

$\overset{p}{\varphi}$   
one of the factors has to be a  $0 \pmod{p}$

- If we find an  $x \not\equiv \pm 1 \pmod{n}$  such that  $x^2 \equiv 1 \pmod{n}$ , we can conclude that  $n$  is not a prime.

# Algorithm Idea

**Claim:** Let  $p > 2$  be a prime number such that  $p - 1 = 2^s d$  for an integer  $s \geq 1$  and some odd integer  $d \geq 3$ . Then for all  $a \in \mathbb{Z}_p^*$ ,

$a^d \equiv 1 \pmod{p}$  or  $a^{2^r d} \equiv -1 \pmod{p}$  for some  $0 \leq r < s$ .

**Proof:** recall  $x^2 \equiv 1 \pmod{p} \iff x \in \{-1, +1\} \pmod{p}$

- **Fermat's Little Theorem:** Given a prime number  $p$ ,

$$\forall a \in \mathbb{Z}_p^*: \quad \underline{a^{p-1} \equiv 1 \pmod{p}}$$

$$\begin{array}{l}
 a^{\frac{p-1}{2}} \equiv \begin{cases} +1 \pmod{p} \\ -1 \pmod{p} \end{cases} \quad \checkmark \\
 \begin{array}{l} \xrightarrow{\quad} \frac{p-1}{2} = d \quad \checkmark \\ \searrow \frac{p-1}{2} \text{ even} \Rightarrow a^{\frac{p-1}{4}} = \begin{cases} +1 \\ -1 \end{cases} \end{array}
 \end{array}$$

# Primality Test


**We have:** If  $n$  is an odd prime and  $n - 1 = 2^s d$  for an integer  $s \geq 1$  and an odd integer  $d \geq 3$ . Then for all  $a \in \{1, \dots, n - 1\}$ ,

➔  $a^d \equiv 1 \pmod{n}$  or  $a^{2^r d} \equiv -1 \pmod{n}$  for some  $0 \leq r < s$ .

**Idea:** If we find an  $a \in \{1, \dots, n - 1\}$  such that

➔  $a^d \not\equiv 1 \pmod{n}$  and  $a^{2^r d} \not\equiv -1 \pmod{n}$  for all  $0 \leq r < s$ ,  
we can conclude that  $n$  is not a prime.

- For every odd composite  $n > 2$ , at least  $\frac{3}{4}$  of all possible  $a$  satisfy the above condition

*Handwritten note:*  $1, \dots, n-1$  possible  $a$   


- How can we find such a *witness*  $a$  efficiently?

# Miller-Rabin Primality Test

- Given a natural number  $n \geq 2$ , is  $n$  a prime number?

## Miller-Rabin Test:

1. if  $n$  is even then return ( $n = 2$ )
2. compute  $s, d$  such that  $n - 1 = 2^s d$ ;
3. choose  $a \in \{2, \dots, n - 2\}$  uniformly at random;
4.  $x := a^d \bmod n$ ;
5. if  $x = 1$  or  $x = n - 1$  then return probably prime;
6. for  $r := 1$  to  $s - 1$  do
7.      $x := x^2 \bmod n$ ;
8.     if  $x = n - 1$  then return probably prime;
9. return composite;

if  $n$  is a prime

$P(\text{test succ.}) = 1$

if  $n$  is composite  
 $P(\text{test succ.}) \geq 3/4$



## Theorem:

- If  $n$  is prime, the Miller-Rabin test always returns **true**. *prime*
- If  $n$  is composite, the Miller-Rabin test returns **false** with probability at least  $3/4$ . *composite*

## Proof:

- If  $n$  is prime, the test works for all values of  $a$
- If  $n$  is composite, we need to pick a good witness  $a$

**Corollary:** If the Miller-Rabin test is repeated  $k$  times, it fails to detect a composite number  $n$  with probability at most  $4^{-k}$ .



## Cost of Modular Arithmetic:

- Representation of a number  $x \in \mathbb{Z}_n$ :  $O(\log n)$  bits
- Cost of adding two numbers  $x + y \bmod n$ :  $O(\log n)$
- Cost of multiplying two numbers  $x \cdot y \bmod n$ : naively  $O(\log^2 n)$ 
  - It's like multiplying degree  $O(\log n)$  polynomials
  - use FFT to compute  $z = x \cdot y$   $O(\log n \cdot \log \log n \cdot \log \log \log n)$

# Running Time

Cost of exponentiation  $x^d \bmod n$ :

- Can be done using  $O(\log d)$  multiplications
- Base-2 representation of  $d$ :  $d = \sum_{i=0}^{\lfloor \log d \rfloor} d_i 2^i$

- **Fast exponentiation:**

1.  $y := 1$ ;
2. **for**  $i := \lfloor \log d \rfloor$  **to** 0 **do**
3.      $y := y^2 \bmod n$ ;
4.     **if**  $d_i = 1$  **then**  $y := y \cdot x \bmod n$ ;
5. **return**  $y$ ;

$$1011 = 1010 + 1$$

- **Example:**  $d = 22 = \overbrace{10110}_2$

$$\begin{aligned} x^{22} &= (x^{11})^2 = (x^{10} \cdot x)^2 = ((x^5)^2 \cdot x)^2 = ((x^4 \cdot x)^2 \cdot x)^2 \\ &= (((x^2)^2 \cdot x)^2 \cdot x)^2 \end{aligned}$$

# Running Time

**Theorem:** One iteration of the Miller-Rabin test can be implemented with running time  $O(\log^2 n \cdot \log \log n \cdot \log \log \log n)$ .

1. **if**  $n$  is even **then return** ( $n = 2$ )
2. compute  $s, d$  such that  $n - 1 = 2^s d$ ;
3. choose  $a \in \{2, \dots, n - 2\}$  uniformly at random;
4.  $x := a^d \bmod n$ ;  $O(\log n)$  mult.
5. **if**  $x = 1$  **or**  $x = n - 1$  **then return probably prime**;
6. **for**  $r := 1$  **to**  $s - 1$  **do**  $O(\log n)$  iter.
  7.  $x := x^2 \bmod n$ ;  $O(1)$  mult.
  8. **if**  $x = n - 1$  **then return probably prime**;
9. **return composite**;

# Deterministic Primality Test

- If a conjecture called the generalized Riemann hypothesis (GRH) is true, the Miller-Rabin test can be turned into a polynomial-time, deterministic algorithm
  - It is then sufficient to try all  $a \in \{1, \dots, \underline{O(\log^2 n)}\}$
- It has long not been proven whether a deterministic, polynomial-time algorithm exists
- In 2002, Agrawal, Kayal, and Saxena gave an  $\underline{\tilde{O}(\log^{12} n)}$ -time deterministic algorithm
  - Has been improved to  $\underline{\tilde{O}(\log^6 n)}$
- In practice, the randomized Miller-Rabin test is still the fastest algorithm