# Chapter 7
# Randomization

## Algorithm Theory
## WS 2017/18

## Fabian Kuhn

# Randomization

## 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:** nodes

- $n$ processes, 1 resource
  (e.g., communication channel, shared database, …)
- There are time slots $1,2,3, \ldots$
- In each time slot, only one client (process) 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

n processes
slots 1, 2, 3, ...

## 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

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

**Events:**

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

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

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

$$\mathcal{S}_{x,t} = \mathcal{A}_{x,t} \cap \left( \bigcap_{y \neq x} \overline{\mathcal{A}_{y,t}} \right)$$

$\mathcal{A}_{x,t}$, $\overline{\mathcal{A}_{y,t}}$ mutually independent

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

choose $p$ s.t. $\mathbb{P}(\mathcal{S}_{x,t})$ is maximized

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

# Fixing $p$

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

$$\underline{\boldsymbol{p = \frac{1}{n}}} \quad \Longrightarrow \quad \underline{\mathbb{P}(\mathcal{S}_{x,t}) = \frac{1}{n}\overbrace{\left(1-\frac{1}{n}\right)^{n-1}}}. \longrightarrow \frac{1}{e\cdot n}$$

- **Asymptotics:**

$$\text{For } n \geq 2\text{:} \quad \frac{1}{4} \leq \overbrace{\left(1-\frac{1}{n}\right)^{n}}^{\to \frac{1}{e}} < \frac{1}{e} < \overbrace{\left(1-\frac{1}{n}\right)^{n-1}}^{\to \frac{1}{e}} \leq \frac{1}{2}$$

- **Success probability:**

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

# Time Until First Success

$q := \mathbb{P}(S_{x,t}) = \frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1}$

**Random Variable $T_{\cancel{i}\,x}$:**

- $T_{\cancel{i}\,x} = t$ if proc. $\cancel{i}\,x$ is successful in slot $t$ for the first time

- **Distribution:**

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

- $T_i$ is geometrically distributed with parameter

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

- **Expected time** until first success:

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

# Time Until First Success

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

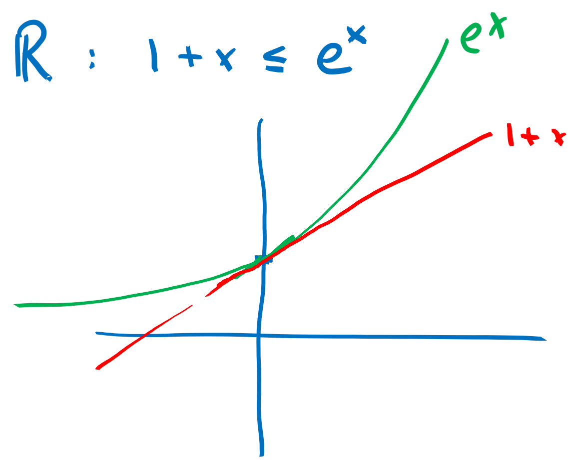
$$\mathcal{F}_{x,t} = \bigcap_{t'=1}^{t} \overline{\mathcal{S}_{x,t'}}$$

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

$$\frac{1}{en} < q \le \frac{1}{2n}$$

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

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




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

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

$$< e^{-1/en}$$

# Time Until First Success

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

$e^{-\frac{en \cdot c \cdot \ln n}{en}}$

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

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

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

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

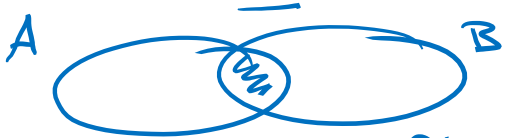
with prob. $\geq 1 - \frac{1}{n^c}$

for any const. $c > 0$

$c$ can only affect the hidden const.

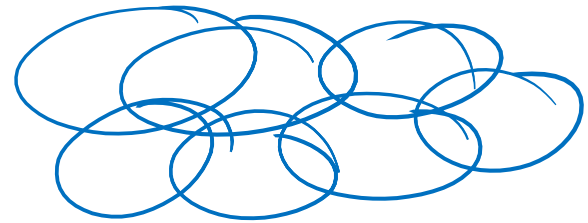
# Time Until All Processes Succeed

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




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

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



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

$$\mathbb{P}\left(\bigcup_{x}^{k} \mathcal{E}_x\right) \leq \sum_{x}^{k} \mathbb{P}(\mathcal{E}_x)$$

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

$< e^{-t/en}$

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

union bound

# 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 $t = \lceil en \cdot (c+1) \ln n \rceil$

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

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

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

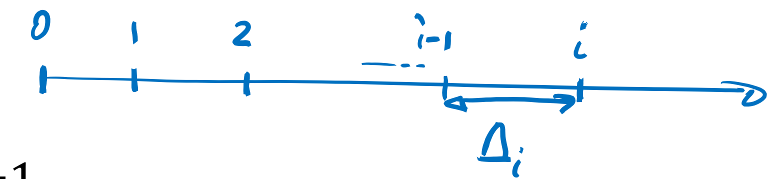
# Expected Time Until All Processes Succeed

**Claim:** In expectation, the time until all processes succeed at least once is $\Theta(n \log n)$.

Proof:

$T_0 = 0$

- **Random variables $T_i$:** time until exactly $0 \le i \le n$ different processes have succeeded
- **Goal:** Compute $\mathbb{E}[T_n]$



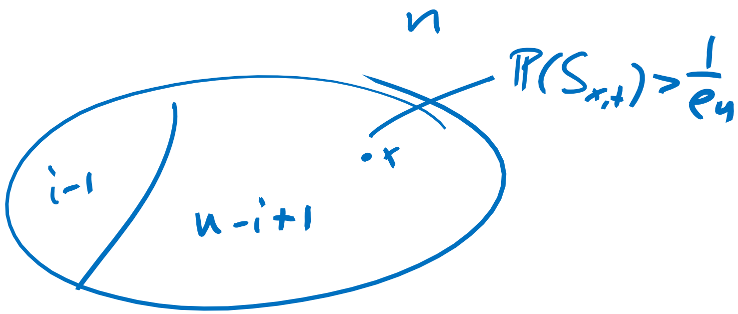

- Random variable $\Delta_i \coloneqq T_i - T_{i-1}$
  - $\Delta_i$ measures the number of rounds needed for the $i^{th}$ process to succeed after exactly $i-1$ processes have succeeded
- We can express $T_n$ as a function of the $\Delta_i$ random variables:

$$\boldsymbol{T_n = \Delta_1 + \Delta_2 + \cdots + \Delta_n}$$

$$T_1 - T_0 + T_2 - T_1 + T_3 - T_2 + \ldots + T_n - T_{n-1} = T_n - T_0$$

# Expected Time Until All Processes Succeed

**Claim:** In expectation, the time until all processes succeed at least once is $\Theta(n \log n)$.




**Distribution of $\Delta_i$?**

- Recall that $\frac{\mathbf{1}}{\boldsymbol{en}} < \mathbb{P}(\boldsymbol{\mathcal{S}_{x,t}}) \leq \frac{\mathbf{1}}{\mathbf{2}\boldsymbol{n}}$
- Event $\boldsymbol{\mathcal{S}_t}$: some new process is successful in round $t$
- Assume that exactly $i-1$ processes have been successful so far

$$\boldsymbol{q_i} \coloneqq \mathbb{P}(\boldsymbol{\mathcal{S}_t} \mid \text{"}\mathbf{exactly}\ \boldsymbol{i}-\mathbf{1}\ \mathbf{succ.\,proc.\,before\ round}\ \boldsymbol{t}\text{"})$$

$$\frac{n-i+1}{en} < q_i \leq \frac{n-i+1}{2n}$$

# Expected Time Until All Processes Succeed

**Claim:** In expectation, the time until all processes succeed at least once is $\Theta(n \log n)$.

**Distribution of $\Delta_i$?**

- $q_i := \mathbb{P}(\mathcal{S}_t \mid \text{"exactly } i-1 \text{ succ. proc. before round } t\text{"})$
- $\Delta_i$ is geometrically distributed with parameter $q_i$

$$\frac{n-i+1}{en} < q_i \leq \frac{n-i+1}{2n}$$

$$\mathbb{E}[\Delta_i] = \frac{1}{q_i} \qquad \mathbb{E}[\Delta_i] < \frac{en}{n-i+1}$$

$$\mathbb{E}[\Delta_i] \geq \frac{2n}{n-i+1}$$

# Expected Time Until All Processes Succeed

**Claim:** In expectation, the time until all processes succeed at least once is $\Theta(n \log n)$.

$E[\Delta_i] < \frac{en}{n-i+1}$

- Recall we need $\mathbb{E}[T_n]$, where $T_n = \Delta_1 + \Delta_2 + \cdots + \Delta_n$

$$E[T_n] = E[\Delta_1 + \Delta_2 + \ldots + \Delta_n] \overset{\text{lin. of exp.}}{=} \sum_{i=1}^{n} E[\Delta_i]$$

$$< en \cdot \sum_{i=1}^{n} \frac{1}{n-i+1} = en \cdot \underbrace{\sum_{j=1}^{n} \frac{1}{j}}_{\text{harmonic series}} = en \cdot H(n) = en(\ln n + \Theta(1))$$

Aloha channel

$H(n) = \ln(n) + \Theta(1)$

$E[T_n] < en \ln n + \Theta(n)$

$E[T_n] \geq 2n \ln n + \Theta(n)$

# Primality Testing

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

**Simple primality test:**

$a \cdot b = n$

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**

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

size of input: $\Theta(\log n)$

time is 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 $x^2 \equiv 1 \pmod{p}$ are $x \equiv \pm 1 \pmod{p}$

$\mathbb{Z}_p^* = \{1, \dots, p-1\}$

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

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

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

$p$ has to be a factor of $x+1$ or $x-1$

$x+1 \equiv 0 \pmod{p}$

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

not true if $p$ is not prime

$p = 15$

$x = 4 \qquad x^2 \equiv 1 \pmod{15}$

- 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 $a \in \mathbb{Z}_p^*$,

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

**Proof:** recall $x^2 \equiv 1 \pmod{p} \iff x \equiv \pm 1 \pmod{p}$

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

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

$$a^{\frac{p-1}{2}} \begin{cases} +1 \pmod{p} \\ -1 \pmod{p} \; ✓ \end{cases}$$

$\frac{p-1}{2} = d \iff a^d \equiv 1 \pmod{p}$ ✓

$\frac{p-1}{2}$ even $\Rightarrow a^{\frac{p-1}{4}} = \begin{cases} +1 \quad \ldots \\ -1 \quad ✓ \end{cases}$

$\frac{p-1}{2} = 2^{s-1} \cdot d$

# 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 $a \in \{1, \dots, n-1\}$,

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

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

$$a^d \not\equiv 1 \pmod{n} \;\; \textbf{and} \;\; a^{2^r d} \not\equiv -1 \pmod{n} \;\; \text{for all } 0 \leq r < s,$$

we can conclude that $n$ is not a prime.

- For every odd composite $n > 2$, at least $^3/_4$ of all possible $a$ satisfy the above condition

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. $x \coloneqq a^d \bmod n$;
5. **if** $x = 1$ **or** $x = n - 1$ **then return probably prime;**
6. **for** $r \coloneqq 1$ **to** $s - 1$ **do**
7. $\quad x \coloneqq x^2 \bmod n$;
8. $\quad$ **if** $x = n - 1$ **then return probably prime;**
9. **return composite;**

# Analysis

**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 $\underline{k}$ times, it fails to detect a composite number $n$ with probability at most $\underline{4^{-k}}$.

# Running Time

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

- **Example:** $d = 22 = \overbrace{\underbrace{101}_{5}1}^{11}0_2$

10110

$$x^{22} = (x^{11})^2 = \left((x^5)^2 \cdot x\right)^2 = \left(((x^2)^2 \cdot x)^2 \cdot x\right)^2$$

# Running Time

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

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. $x := a^d \bmod n$;
5. **if** $x = 1$ **or** $x = n - 1$ **then return probably prime;**
6. **for** $r := 1$ **to** $s - 1$ **do**
7. $\quad x := x^2 \bmod n$;
8. $\quad$ **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 $a \in \{1, \dots, O(\log^2 n)\}$

- It has long not been proven whether a deterministic, polynomial-time algorithm exists

- In 2002, Agrawal, Kayal, and Saxena gave an $\tilde{O}(\log^{12} n)$-time deterministic algorithm
  – Has been improved to $\tilde{O}(\log^6 n)$

- In practice, the randomized Miller-Rabin test is still the fastest algorithm