Algorithms and Probability

Week 10

G09 - mkilic

30.IV.2026

Overview

1. **Primzahltest**
2. **Exercise: Probabilistic Method**
3. **Randomisiertes Quicksort & Quickselect**

Roadmap

1. Graphentheorie
   - Zusammenhang
   - Kreise
   - Matchings
   - Färbungen
2. W'keitstheorie
   - Bedingte W'keit
   - Unabhängigkeit
   - (mehrere) Zufallsvariablen
   - Diskrete Verteilungen
   - Abschätzen von W'keiten
   - Randomisierte Algorithmen
3. Algorithmen
   - Lange-Bunte Pfade
   - MaxFlow
   - MinCut
   - Kleinster umschliessender Kreis
   - Konvexe Hülle

Primzahltest

Primzahltest

Problem: Gegeben $n \in \mathbb{N}$, ist n prim?

Viele Anwendungsbereiche z.B. Kyrpographie, Hashing, PRNGs usw.
Wir werden 3 Herangehensweisen sehen um Primalität zu testen.

0 - Brute Force

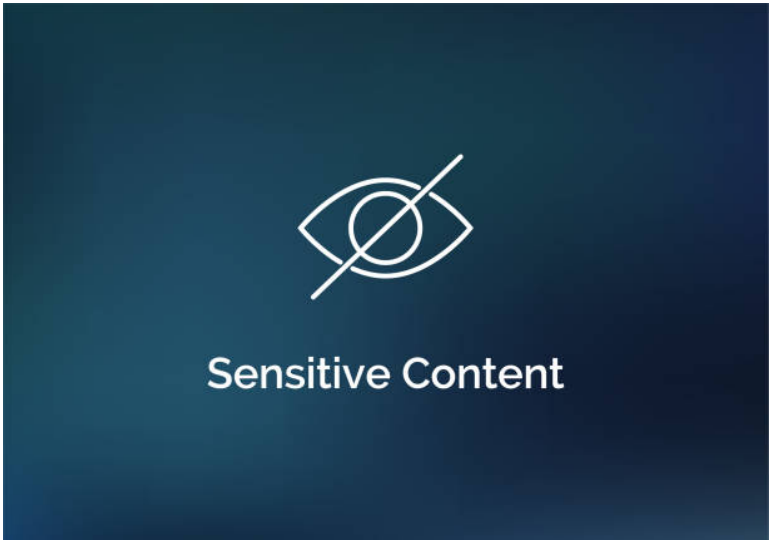
```
1     for all a ≤ √n:
2         if (n % a == 0):
3             return false
4     return true
5
```

naiv_prim_test(n)

Falls der Algorithmus einen Teiler a entdeckt, dann wissen wir, dass n prim ist. Wir sagen a ist ein **Zertifikat** [1] für Zusammengesetztheit von n. Sonst sucht unser Algorithmus weiter. Das ist noch nicht randomisiert. Ausgabe ist immer korrekt aber Algorithmus ist **zu langsam**.

[1] (Manchmal auch: *Zeuge/Witness*)

Some Discrete Math



Some Discrete Math

Definition 5.16. $\mathbb{Z}_m^* \stackrel{\text{def}}{=} \{a \in \mathbb{Z}_m \mid \gcd(a, m) = 1\}$.

Definition 5.17. The *Euler function* $\varphi : \mathbb{Z}^+ \to \mathbb{Z}^+$ is defined as the cardinality of $\mathbb{Z}_m^*$:

$$\varphi(m) = |\mathbb{Z}_m^*|.$$

Example 5.29. $\mathbb{Z}_{18}^* = \{1, 5, 7, 11, 13, 17\}$. Hence $\varphi(18) = 6$.

If p is a prime, then $\mathbb{Z}_p^* = \{1, \ldots, p-1\} = \mathbb{Z}_p \setminus \{0\}$, and $\varphi(p) = p - 1$.

Lemma 5.12. *If the prime factorization of m is $m = \prod_{i=1}^{r} p_i^{e_i}$, then*[11]

$$\varphi(m) = \prod_{i=1}^{r} (p_i - 1) p_i^{e_i - 1}.$$

Discrete Maths Script Page 102-103

1 - Euklid-Primzahltest

n prim $\Rightarrow ggT(a, n) = 1 \quad \forall a \in [n-1]$

```
1    choose a ∈ [n − 1] i.i.d.:
2    if ggT(a,n) > 1:
3        return false
4    else:
5        return true
6
```

euklid_prim_test(n)

1 - Euklid-Primzahltest

n prim $\Rightarrow ggT(a,n) = 1 \;\; \forall a \in [n-1]$

```
1    choose a ∈ [n − 1] i.i.d.:
2    if ggT(a,n) > 1:
3        return false
4    else:
5        return true
6
```

euklid_prim_test(n)

Zu viele 'false positives'. Falls n zusammengesetzt ist dann ist nur korrekt mit W'keit

$$\frac{|a \in [n-1] : ggT(a,n) = 1|}{n-1} = \frac{|\mathbb{Z}_n^*|}{n-1} = \frac{\varphi(n)}{n-1}$$

1 - Euklid-Primzahltest

n prim $\Rightarrow ggT(a,n) = 1 \ \ \forall a \in [n-1]$

```
choose a ∈ [n − 1] i.i.d.:
if ggT(a,n) > 1:
    return false
else:
    return true

```

euklid_prim_test(n)

Zu viele 'false positives'. Falls n zusammengesetzt ist dann ist nur korrekt mit W'keit

$$\frac{|a \in [n-1] : ggT(a,n) = 1|}{n-1} = \frac{|\mathbb{Z}_n^*|}{n-1} = \frac{\varphi(n)}{n-1}$$

Falls $n = pq$ für p und q prim dann ist diese W'keit $\frac{2}{\sqrt{n}}$ und das würde $O(\sqrt{n})$ Wiederholungen brauchen um Fehlerw'keit "genügend" zu reduzieren. Wie brute-force!

Kleiner Fermatscher Satz

Satz 2.77 (Kleiner fermatscher Satz). Ist $n \in \mathbb{N}$ prim, so gilt für alle Zahlen $0 < a < n$

$$a^{n-1} \equiv 1 \mod n.$$

A&W script page 155

Corollary 5.14 (Fermat, Euler). *For all $m \geq 2$ and all a with $\gcd(a, m) = 1$,*

$$a^{\varphi(m)} \equiv_m 1.$$

In particular, for every prime p and every a not divisible by p,

$$a^{p-1} \equiv_p 1.$$

discrete maths script page 104

a	$a^{14} \mod 15$	Result	Is Witness?
1	1	✅ Fermat holds	No
2	$16384 \mod 15 = 4$	❌ Violates Fermat	Yes
3	$4782969 \mod 15 = 9$	❌ Violates Fermat	Yes
4	1	✅ Fermat holds	No
5	$6103515625 \mod 15 = 10$	❌ Violates Fermat	Yes
6	$78364164096 \mod 15 = 6$	❌ Violates Fermat	Yes
7	$678223072849 \mod 15 = 13$	❌ Violates Fermat	Yes
8	$4398046511104 \mod 15 = 4$	❌ Violates Fermat	Yes
9	$22876792454961 \mod 15 = 6$	❌ Violates Fermat	Yes
10	$100000000000000 \mod 15 = 10$	❌ Violates Fermat	Yes
11	$289254654976 \mod 15 = 1$	✅ Fermat holds	No
12	$1283918464548864 \mod 15 = 9$	❌ Violates Fermat	Yes
13	$302875106592253 \mod 15 = 4$	❌ Violates Fermat	Yes
14	$11112006825558016 \mod 15 = 1$	✅ Fermat holds	No

Fermat - #witnesses: 10

a	$\gcd(a, 15)$	Witness?
1	1	❌
2	1	❌
3	3	✅
4	1	❌
5	5	✅
6	3	✅
7	1	❌
8	1	❌
9	3	✅
10	5	✅
11	1	❌
12	3	✅
13	1	❌
14	1	❌

GCD - #witnesses: 6

The witnesses are numbers that verify the compositeness of 15. Fermat can help us find more witnesses in this case, increasing the probability of us finding a witness randomly.

Miller-Rabin: Grundidee

n prim $\Rightarrow \mathbb{Z}_n^* = \{1, \ldots, n-1\}$ *Körper unter addition und Multiplikation modulo n*

Besonders gilt:

$$x^2 \equiv 1 \pmod{n} \quad \Rightarrow \quad x = 1 \text{ oder } x = n-1 \quad (\star)$$

Miller-Rabin: Zerlegung

Wir schreiben $n - 1 = d \cdot 2^k$ mit d ungerade.

Wenn n prim ist, gilt nach Fermat:

$$a^{n-1} \equiv 1 \pmod{n} \quad \text{für alle } a \in \{1, \ldots, n-1\}$$

Daraus folgt:

$$(a^d)^{2^k} \equiv 1 \pmod{n}$$

Miller-Rabin: Was passiert bei Primzahlen

Aufgrund $(\star)$ folgt aus folgender Aussage:

$$(a^d)^{2^k} \equiv 1 \pmod{n}$$

dass entweder

$$(a^d)^{2^{k-1}} \equiv 1 \pmod{n} \quad \text{oder} \quad (a^d)^{2^{k-1}} \equiv -1 \pmod{n}$$

Diese Idee wird rekursiv genutzt, um $a^d, a^{2d}, \ldots, a^{2^i d}$ zu prüfen.

Miller-Rabin: Iterativer Test

Man prüft iterativ:

$$(a^d)^{2^i} \equiv 1 \pmod{n} \quad \text{für alle } 0 \leq i \leq k$$

Oder: Für ein $0 \leq i < k$ gilt:

$$(a^d)^{2^i} \equiv n - 1 \pmod{n}$$

Wenn keine dieser Bedingungen erfüllt ist:

$$\Rightarrow a \text{ ist ein Zertifikat dafür, dass } n \text{ nicht prim ist.}$$

Miller-Rabin: Algorithmus

```
1  Miller-Rabin-Primzahltest(n)
2  1: if n = 2 then
3  2:     return 'Primzahl'
4  3: else if n gerade oder n = 1 then
5  4:     return 'keine Primzahl'
6  5: Wahle a ∈ {2,3,...,n-1} zufallig und
7  6: berechne k,d ∈ ℤ mit n-1 = d·2^k und d ungerade.
8  7: x ← a^d mod n
9  8: if x = 1 or x = n-1 then
10 9:     return 'Primzahl'
11 10: repeat k-1 mal
12 11:     x ← x^2 mod n
13 12:     if x = 1 then
14 13:         return 'keine Primzahl'
15 14:     if x = n-1 then
16 15:         return 'Primzahl'
17 16: return 'keine Primzahl'
```

Miller-Rabin-Primzahltest(n)

Miller-Rabin: Fazit

Laufzeit[2]**:** $O(\ln n)$
W'keiten:

Eingabe: n	**Ausgabe**
prim	immer prim
zusammengesetzt	mit 3/4 W'keit: "nicht prim"

Wir können die Fehlerwahrscheinlichkeit mit Felherreduktionsmethoden beliebig klein machen.

[2] Wegen binäre Exponentiation & Successive squaring

Exercise: Probabilistic Method

So far, we mainly used randomization to construct efficient algorithms. However, randomization can also be used to prove statements that don't involve randomness themselves. In fact, such proves can usually be phrased by describing a Monte-Carlo algorithm. Answer the two following questions. If you want to, you can describe your solution as a Monte-Carlo algorithm.

(a) You are given 1000 subsets $A_1, \ldots, A_{1000}$ of $\{1, \ldots, n\}$. Each subset has size at least 11. Can the numbers $1, \ldots, n$ be colored 'red' and 'blue', such that each set A_i contain at least one 'red' and at least one 'blue' number?

(b) You are given 10 points in the plane (i.e., $\mathbb{R}^2$). Can you place (filled) disks of radius 1 in the plane, such that (i) each of the 10 points is contained in a disk, and (ii) all disks are disjoint (i.e., their centers have distance at least 2)?
Remark: You may cover multiple points with the same disk. In particular, the task would be trivial if all points are very close to each other (then you only need one disk to cover all of them). Similarly, if all points are very far apart, you could use one disk per point without having to worry about disjointness.
Remark: a formal solution to this exercise would be very technical. Focus more on the ideas than on technical details

Randomisiertes Quicksort

```
1        if < r then
2            p ← Uniform({ℓ, ℓ + 1, ..., r})
3            t ← PARTITION(A, ℓ, r, p)
4            QUICKSORT(A, ℓ, t − 1)          \\Block I
5            QUICKSORT(A, t + 1, r)          \\Block II
6
```

Quicksort(A ℓ r)

Wir sind interessiert für Laufzeit. Sie ist von der Anzahl der Vergleiche bestimmt.

Randomisiertes Quicksort

Definiere: $X_{i,j} :=$ *Indikatorvariable für "a_i wird mit a_j verglichen"*

- Pivotelement wird mit allen anderen Elementen im selben Block verglichen. Sonst gibt es keine Vergleiche und das Pivotelement ist danach in keinem Block mehr.
- Damit das Paar (a_i, a_j) verglichen werden können, muss einer der beiden als Pivotelement gewählt werden. Jedes Paar wird maximal einmal verglichen.

Randomisiertes Quicksort

Definiere: $X_{i,j} :=$ *Indikatorvariable für "a_i wird mit a_j verglichen"*

- Pivotelement wird mit allen anderen Elementen im selben Block verglichen. Sonst gibt es keine Vergleiche und das Pivotelement ist danach in keinem Block mehr.
- Damit das Paar (a_i, a_j) verglichen werden können, muss einer der beiden als Pivotelement gewählt werden. Jedes Paar wird maximal einmal verglichen.

$$X = \sum_{1 \leq i < j \leq n} X_{i,j}$$

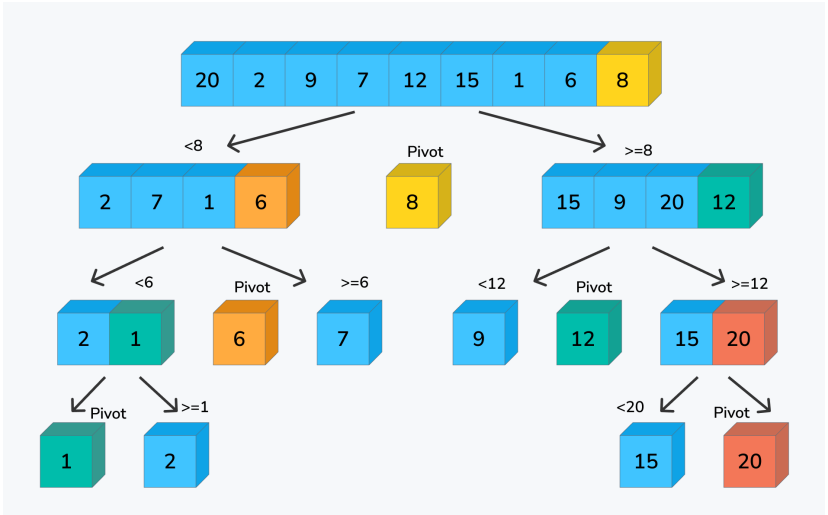
Für ein Paar (a_i, a_j) gibt es zwei Möglichkeiten:

1. a_i oder a_j wird als Pivot gewählt: $X_{i,j} = 1$ mit W'keit $\frac{2}{j-i+1}$
2. Einer von $a_{i+1}, \ldots a_{j-1}$ wird gewählt: $X_{i,j} = 0$

Also haben wir: $\mathbb{E}[X_{i,j}] = Pr[X_{i,j} = 1] = \frac{2}{j-i+1}$

$$\mathbb{E}[X] \overset{lin.}{=} \sum_{1 \leq i < j \leq n} \mathbb{E}[X_{i,j}] = \sum_{1 \leq i < j \leq n} \frac{2}{j-i+1} \cdots \leq 2n \ln(n)$$

Randomisiertes Quicksort



Quickselect

```
1    p ← Uniform({ℓ, ℓ+1, ..., r})    \\waehle Pivot zufaellig
2    t ← PARTITION(A, ℓ, r, p)
3    if t = ℓ + k − 1 then
4        return A[t]                    \\k-te kleinstes Element gefunden
5    else if t > ℓ + k − 1 then
6        return QUICKSELECT(A, ℓ, t − 1)        \\suche links
7    else
8        return QUICKSELECT(A, t + 1, r)          \\suche rechts
9
```

Quickselect(A ℓ r k)

Bei Quickselect suchen wir das k-te kleinste Element des Arrays A. Der Unterschied von Quicksort ist dass wir uns nicht für die beiden Blöcke interessieren, sondern nur für den Block, der unseres Element beinhaltet.

Setup: Total Cost of QuickSelect

[*this slide includes partly AI generated content*]

QuickSelect$(A, 1, n, k)$ makes a sequence of calls

$$\texttt{Partition}(A,\ \ell_i,\ r_i,\ p)$$

Each such call costs exactly $r_i - \ell_i$ comparisons.

The **total runtime** (measured in comparisons) is therefore:

$$T = \sum_{i=1}^{N} (r_i - \ell_i)$$

where N is the (random) number of recursive calls made.

Goal: Show $\mathbb{E}[T] = O(n)$.

Key Idea: Grouping Calls into Levels

[*this slide includes partly AI generated content*]

The size of the subarray changes from call to call. So we group the calls by subarray size.

Definition. Call i belongs to **level** j if:

$$\left(\tfrac{3}{4}\right)^j n \;<\; r_i - \ell_i + 1 \;\leq\; \left(\tfrac{3}{4}\right)^{j-1} n$$

- Level 1: subarray size $\leq n$
- Level 2: subarray size $\leq \frac{3}{4} n$
- Level 3: subarray size $\leq \left(\frac{3}{4}\right)^2 n$

$\vdots$

Every call falls into **exactly one** level. But it can be that we have to call the algorithm multiple times before we choose a nice pivot and actually reduce the size of the array by a factor of 3/4. Let N_j = number of calls at level j.

Why the Factor $\frac{3}{4}$?

[*this slide includes partly AI generated content*]

Claim: After each partition step, with probability $\geq \frac{1}{2}$, the subarray shrinks by a factor of at least $\frac{3}{4}$.

Why? The pivot is chosen *uniformly at random*. If it lands in the middle half of the current subarray (ranks $\frac{1}{4}$ to $\frac{3}{4}$), then the larger remaining piece has size $\leq \frac{3}{4} \cdot$ current size.

bottom 25%	middle 50%	top 25%
bad pivot	**good pivot**	bad pivot

Since the pivot is uniform:

$$P(\text{good pivot}) = \frac{1}{2} \qquad P(\text{bad pivot}) = \frac{1}{2}$$

Bounding the Total Cost

[*this slide includes partly AI generated content*]

Each call at level j costs at most $\left(\frac{3}{4}\right)^{j-1} n$ comparisons. Therefore:

$$T \leq \sum_{j=1}^{\infty} N_j \cdot \left(\frac{3}{4}\right)^{j-1} n$$

Taking expectations (by linearity):

$$\mathbb{E}[T] \leq n \cdot \sum_{j=1}^{\infty} \mathbb{E}[N_j] \cdot \left(\frac{3}{4}\right)^{j-1}$$

Note: We sum to ∞ because the termination level is random and unknown. For all levels beyond termination $N_j = 0$, so those terms vanish. The infinite sum is valid because $\frac{3}{4} < 1$ guarantees convergence.

Bounding $\mathbb{E}[N_j]$

[*this slide includes partly AI generated content*]

Question: How many calls land at level j before the subarray shrinks past it?

Think of each call as a **coin flip**:

$$\text{heads (good pivot)} \;\Rightarrow\; \text{advance to level } j+1 \quad \text{with prob. } \tfrac{1}{2}$$

$$\text{tails (bad pivot)} \;\Rightarrow\; \text{stay at level } j \quad \text{with prob. } \tfrac{1}{2}$$

N_j = number of flips needed to advance = **Geometric**$\left(\frac{1}{2}\right)$ random variable.

$$\mathbb{E}[N_j] = \frac{1}{1/2} = \mathbf{2} \qquad \text{for all } j \geq 1$$

Final Calculation: $\mathbb{E}[T] \leq 8n$

[*this slide includes partly AI generated content*]

Substituting $\mathbb{E}[N_j] \leq 2$ into the bound:

$$\begin{aligned}
\mathbb{E}[T] &\leq n \cdot \sum_{j=1}^{\infty} \mathbb{E}[N_j] \cdot \left(\tfrac{3}{4}\right)^{j-1} \\
&\leq n \cdot \sum_{j=1}^{\infty} 2 \cdot \left(\tfrac{3}{4}\right)^{j-1} \\
&= 2n \cdot \sum_{j=0}^{\infty} \left(\tfrac{3}{4}\right)^{j} \\
&= 2n \cdot \frac{1}{1-\frac{3}{4}} = 8n
\end{aligned}$$

Conclusion

[*this slide includes AI generated content*]

Theorem

The expected runtime of **QuickSelect**$(A, 1, n, k)$, measured in element comparisons, satisfies:

$$\mathbb{E}[T] \leq 8n = O(n)$$

for any k and any input array A with pairwise distinct elements.

Key ideas in the proof:

- Group recursive calls into geometric levels based on subarray size
- At each level, a random pivot is "good" with probability $\geq \frac{1}{2}$
- Expected calls per level ≤ 2 (geometric distribution)
- Costs form a convergent geometric series summing to $8n$

The End

♠