Randomisierte Algorithmen

Wintersemester 2010/11

Beate Bollig
Informatik LS2

„Das Gewebe dieser Welt ist aus
Notwendigkeit und Zufall gebildet;
die Vernunft des Menschen stellt sich zwischen beide
und weiß sie zu beherrschen.
Sie behandelt das Notwendige als den Grund ihres Daseins;
Das Zufällige weiß sie zu lenken, zu leiten und zu nutzen.“

Johann Wolfgang von Goethe

Randomisierter Algorithmus

Verhalten hängt in jedem Schritt von einem Zufallsexperiment („Münzwurf“) ab

- → variiert von einer Ausführung zur nächsten, selbst bei gleicher Eingabe
- → Laufzeit und/oder Korrektheit können nur noch mit gewissen Wahrscheinlichkeiten garantiert werden
- → Laufzeit, Speicherplatz, berechnetes Ergebnis (für festgehaltene Eingabe) Zufallsvariablen

Analyse randomisierter Algorithmen:
„erwartete“ Werte eines Leistungsmaßes, welche für jede Eingabe gültig sind

Vorteile randomisierter Algorithmen

- Effizienz
- Einfachheit

Bemerkung:

Aus praktischer Sicht können randomisierte Algorithmen sogar „zuverlässiger“ als entsprechende deterministische Algorithmen sein.

??? Schneller Algorithmus zuverlässiger als langsamer Algorithmus im Hinblick auf Auftreten eines Hardwarefehlers

Beispiel:

- deterministischer Algorithmus 3 Tage
- randomisierter Algorithmus liefert korrektes Ergebnis mit Wahrscheinlichkeit $1 - 10^{-30}$ in 10 Sekunden

Warnung:

Bei Verwendung randomisierter Algorithmen in kritischen Bereichen immer bedenken, dass verwendete Zufallszahlen von Pseudo-Zufallszahlen-Generatoren bereitgestellt werden, also die Annahme der „idealen Zufälligkeit“ nicht unbedingt gilt.

Frage:

Wie kann „Zufall“ vorteilhaft in einem randomisierten Algorithmus eingesetzt werden?

→ Entwurfsparadigmen randomisierter Algorithmen

- Vereitelung gegnerischer Angriffe
  (foiling an adversary)
  Problem als Spiel zwischen Algorithmenentwerfer und (bösartigem) Gegenspieler
  Aufgrund Randomisierung nicht alle Schritte des Algorithmus vorhersagbar
  → Probleme worst case Eingabe zu konstruieren

- Signaturen/Fingerabdrücke
  (fingerprinting and hashing)
  Repräsentation eines großen Objekts durch einen kleinen „Fingerabdruck“ mittels einer „zufälligen“ Abbildung
- Zufällige Umordnung (random reordering)
  Zufälliges Umordnen der Eingabe verhindert schlechte Eingabe mit hoher Wahrscheinlichkeit
- Lastbalancierung (load balancing)
  Auswahl zwischen verschiedenen Ressourcen mittels Randomisierung zwecks gleichmäßiger Verteilung der Last

- Zufallsstichproben (random sampling)
  Kleine, zufällige Stichprobe repräsentiert sehr große Menge, d.h. hat ähnliche Eigenschaften
- Überfluß an Zeugen (abundance of witnesses)
  In einem großen Suchraum liegen relativ viele Elemente mit einer bestimmten Eigenschaft, so dass zufällig gewähltes Element mit gewisser Wahrscheinlichkeit „Zeuge“ ist.

Frage: Warum können wir Zeugen nicht auch deterministisch schnell finden?

Für verschiedene Eingaben können Zeugen anders zwischen „Zeugenkandidaten“ verteilt sein.

- Schnell konvergierende Markov-Ketten
  (rapidly mixing Markov chains)
  Stochastischer Prozeß zur Erzeugung einer guten (gleichförmigen) Stichprobe aus einem großen Raum
- Isolierung und Aufbrechen von Symmetrien
  (isolation and symmetry breaking)
  Beim verteilten oder parallelen Rechnen müssen mehrere Prozessoren oft gemeinsame Entscheidungen treffen. Randomisierung hilft Deadlocks zu vermeiden und Symmetrien aufzubrechen.

- Probabilistische Methode und Existenzbeweise
  (probabilistic methods and existence proofs)
  Beweis der Existenz eines spezifischen kombinatorischen Objekts (z.B. Graph mit bestimmten Eigenschaften), indem gezeigt wird, dass zufällig gewähltes Objekt aus geeignet definiertem Universum die gewünschte Eigenschaft mit positiver Wahrscheinlichkeit hat

Manche Ideen sind alt . . .

J. Shallit (1992)
Randomized algorithms in “primitive” cultures or what is the oracle complexity of a dead chicken
ACH SIGACT News 23 (4), 77-80.

- de Leeuw, Moore, Shannon, Shapiro (1955)
  Computability by probabilistic machines
- Rabin (1963)
  Probabilistic automata
- Gill (1977)
  Computational complexity of probabilistic Turing machines
- Solovay, Strassen (1977)
  Fast Monte Carlo test for primality
- Rabin (1980)
  Probabilistic algorithms for testing primality

Wahrscheinlichkeitstheorie ist Basisinstrument zur Modellierung und Analyse randomisierter Algorithmen

Hier: Beschränkung auf endliche Wahrscheinlichkeitsräume

Modellierung eines Zufallsexperiments

- Festlegung der Menge S aller möglichen Resultate (elementare Ereignisse)
- Ereignis $\hat{=}$ Teilmenge von S
- Festlegung der Wahrscheinlichkeiten der elementaren Ereignisse (Summe der Wahrscheinlichkeiten ist 1)
- Wahrscheinlichkeit des Ereignis A $\hat{=}$ Summe der Wahrscheinlichkeiten der elementaren Ereignisse in A

Teilinformation über Ausgang eines Experiments

→ bedingte Wahrscheinlichkeit

- Teilinformation $\hat{=}$ Modellierung eines Ereignisses B, das mit Sicherheit vorkommt
- Wahrscheinlichkeit von A bei gegebenem B: Wahrscheinlichkeit von $A \cap B$ in neuem Wahrscheinlichkeitsraum, B Menge der elementaren Ereignisse
- Ereignisse A und B in einem Wahrscheinlichkeitsraum unabhängig, wenn Wahrscheinlichkeit von A gleich Wahrscheinlichkeit von A bei gegebenem B; Wahrscheinlichkeit von $A \cap B$ entspricht Produkt der Wahrscheinlichkeit von A und B

Sicheres Auftreten von B bewirkt keine Änderung der Wahrscheinlichkeit von A

Grundlegende Instrumente

zur Analyse von Zufallsexperimenten

- Zufallsvariable: Funktion von S nach $\mathbb{R}$
- Erwartungswert einer Zufallsvariablen:
  durch die Wahrscheinlichkeiten gewichteter
  Durchschnittswert der Zufallsvariablen

Anforderung an randomisierte Algorithmen

für jede Eingabe „mit großer Wahrscheinlichkeit“ korrekt und effizient arbeiten

→ Wahrscheinlichkeitsräume für Algorithmus A und Eingabe X:
Menge $S_{A,X}$ aller Berechnungen von A auf X
Zufallsvariable, die jedem korrekten Lauf 1 zuordnet (sonst 0)

→ Erwartungswert dieser Zufallsvariablen $\hat{=}$ Wahrscheinlichkeit Berechnung korrekter Ausgabe
Gegenwahrscheinlichkeit $\hat{=}$ Fehlerwahrscheinlichkeit von A auf X

Sichtweise auf randomisierte Algorithmen

- Wahrscheinlichkeitsverteilung über Menge deterministischer Algorithmen
- nichtdeterministische Algorithmen mit Wahrscheinlichkeitsverteilung über jede nichtdeterministische Verzweigung

Verallg.

Definition:

Ein (diskreter) Wahrscheinlichkeitsraum ist ein Paar (Ω, Pr), wobei Ω eine nichtleere abzählbare Menge und $Pr : \Omega \longrightarrow \mathbb{R}_{\geq 0}$ ein Wahrscheinlichkeitsmaß, d.h. eine Funktion mit

$$\sum_{\omega \in \Omega} Pr(\omega) = 1 \qquad \text{ist.}$$

- Elemente in Ω: Elementarereignisse
- Menge $E \subseteq \Omega$: Ereignis mit $Pr(E) := \sum_{\omega \in E} Pr(\omega)$
- Ω endlich und $Pr(\omega) = \frac{1}{|\Omega|}$ für alle $\omega \in \Omega$
  $\longrightarrow$ Pr Gleichverteilung auf Ω
- $\Omega^+ \mathrel{\hat{=}}$ Menge mit positiver Wahrscheinlichkeit, also
  $\Omega^+ := \{\omega \in \Omega \mid Pr(\omega) > 0\}$

Beobachtung:

(i) Für jedes Ereignis $E \subseteq \Omega$ gilt: $0 \leq Pr(E) \leq 1$

(ii) Für eine abzählbare Menge $\{E_i \mid i \in I\}$ von paarweise disjunkten Ereignissen gilt: $Pr\left(\bigcup_{i \in I} E_i\right) = \sum_{i \in I} Pr\left(E_i\right)$

Das Sekretärinnenproblem (Variante)

Aufgabe: Prof W. muss neue Sekretärin einstellen

n Bewerberinnen

Einzelgespräche, nach Gespräch sofort Entscheidung über Einstellung

Ziel: beste Sekretärin einstellen

Entlassung einer bereits eingestellten Sekretärin

$\longrightarrow$ Kosten b

Annahme: Bewerberinnen nach Qualifikation vollständig geordnet, keine Bewerberin mit gleicher Qualifikation

$\longrightarrow$ Güte der Bewerberinnen Werte aus $\{1, \ldots, n\}$

Algorithmus: W-Sekretärin (n)

Eingabe: Bewerberinnen $s_1, \ldots, s_n$

Ausgabe: beste Bewerberin aus $s_1, \ldots, s_n$

Bewerberin wird mit ihrer Güte identifiziert

```
best ← 0
for i ← 1, ..., n do
    Führe Gespräch mit Bewerberin s_i
    if s_i besser als Bewerberin best then
        best ← s_i
        stelle s_i ein
    end if
end for
```

Frage:

Wie hoch sind die Kosten?

Worst case Analyse

$s_1, \ldots, s_n$ Permutation über $\{1, \ldots, n\}$
obere Schranke: $(n-1)b$

Satz:

Im worst case erzeugt W-Sekretärin Kosten $(n-1)b$.

Eingabe $1, 2, \ldots, n$

Frage: Welche Kosten entstehen „typischerweise“?

$\longrightarrow$ probabilistische Analyse

Probabilistische Analyse

Eingabe wird „zufällig“ gewählt

Frage: Was heißt hier zufällig?

Annahmen über Verteilung der Eingabe müssen wohlbegründet sein

Hier: Bewerberinnen erscheinen in zufälliger Reihenfolge ihrer Qualifikation, d.h. für $\pi \in S_n$ ist Wahrscheinlichkeit, dass Bewerberinnen in dieser Reihenfolge erscheinen, $\frac{1}{n!}$

Aufgabe: Analyse der erwarteten Kosten

Zufallsvariable

Definition:

Eine Zufallsvariable X ist eine Funktion $X : \Omega \to \mathbb{R}$.

$Pr(X = x)$ bezeichnet Wahrscheinlichkeit des Ereignisses $\{\omega \in \Omega \mid X(\omega) = x\}$

Der Erwartungswert $E[X]$ von X ist definiert als

$$E[X] = \sum_{x \in X(\Omega)} x \cdot Pr(X = x)$$

(sofern diese Summe endlich ist oder konvergiert)

Hier: Zufallsvariable Y bezeichnet Kosten von W-Sekretärin
X bezeichnet # Einstellungen

also $Y = b \cdot (X - 1)$

Ziel: Berechnung $E[Y]$

Beobachtung:

Für beliebige Zufallsvariablen $X, Y, X_1, \ldots, X_n$ (unter der Voraussetzung, dass alle Erwartungswerte definiert sind) gilt:

(i) $X \leq Y$ (d.h. $\forall \omega \in \Omega : X(\omega) \leq Y(\omega)$) $\longrightarrow$ $E[X] \leq E[Y]$

(ii) $E(\alpha X + \beta Y) = \alpha E[X] + \beta E[Y]$

(iii) $E(X_1 + \ldots + X_n) = E[X_1] + \cdots + E[X_n]$

Linearität des Erwartungswertes

Indikatorvariable

(iv) Ist $X \in \{0, 1\}$ (d.h. $\forall \omega \in \Omega : X(\omega) \in \{0, 1\}$), so ist $E[X] = Pr(X = 1)$

Hier: Indikatorvariable X_i mit

$$X_i := \begin{cases} 1, & \text{wenn Bewerberin } s_i \text{ eingestellt wird} \\ 0, & \text{sonst} \end{cases}$$

$$X = \sum_{i=1}^{n} X_i$$

$$E[Y] = E[b(X-1)] = bE[X] - b = b\sum_{i=1}^{n} E[X_i] - b$$

Jetzt: $Pr(X_i = 1)$ bestimmen

$\uparrow$

Wahrscheinlichkeit, dass $s_i = \max\{s_1, \ldots, s_i\}$

$\uparrow$

$$\frac{(i-1)!\binom{n}{i}(n-i)!}{n!} = \frac{1}{i}$$

$$E[X] = \sum_{i=1}^{n} E[X_i] = \sum_{i=1}^{n} \frac{1}{i} = H_n$$

harmonische Zahl der Ordnung n

$$\ln(n+1) \leq H_n \leq \ln n + 1$$

Satz:

Erscheinen die Bewerberinnen in einer zufälligen Reihenfolge, so hat W-Sekretärin erwartete Kosten von $bH_n - b = \mathcal{O}(b \ln n)$

Rückwärtsanalyse oft hilfreich

neues Zufallsexperiment erzeugt Permutation „rückwärts“:
s_i wird gleichverteilt aus $\{1, \ldots, n\} \setminus \{s_{i+1}, \ldots, s_n\}$,
$i = n, n-1, \ldots, 1$, gewählt; jede Permutation
Wahrscheinlichkeit $\frac{1}{n!}$; $Pr(s_i = \max\{s_1, \ldots, s_i\}) = \frac{1}{i}$

Randomisierter Algorithmus

Oft keine vernünftige Information über Wahrscheinlichkeitsverteilung der Eingabe

Hier: Verteilung wird „erzwungen“

Algorithmus: Rand-Sekretärin (n)

1. Wähle zufällige Permutation $\pi \in S_n$.
2. Wende W-Sekretärin an, um Bewerberin in Reihenfolge $\pi(1), \ldots, \pi(n)$ zu befragen und einzustellen.

Beobachtung:

Algorithmus Rand-Sekretärin kann für gleiche Eingabe in unterschiedlichen Läufen unterschiedliche Kosten verursachen.

Zufall steckt im Algorithmus, nicht in der Eingabe

Beobachtung:

Rand-Sekretärin erzwingt Gleichverteilung der Eingabe

$\longrightarrow$

Satz:

Erwartete Kosten von Rand-Sekretärin betragen
$b \cdot H_n - b = \mathcal{O}(b \ln n)$

Anmerkung:

Erwartete Kosten von $\mathcal{O}(b \ln n)$ gelten für jede Eingabe.

Kap. 1

Einführung, Motivation; Entwurfsparadigmen und Beispielalgorithmen

- Matrixmultiplikation
- Quicksort
- Spielbaumauswertung
- 2SAT
- Speisende Philosophen

Kap. 2

Klassifikation randomisierter Algorithmen, randomisierte Komplexitätsklassen, probability amplification

Matrixmultiplikation (Signaturmethode)

Eingabe: 3 $n \times n$ Matrizen $A, B, C \in \mathbb{F}^{n \times n}$

Frage: Gilt $A \cdot B = C$?

Körper mit neutralen Elementen $0, 1$

z.B. $\mathbb{Z}_p$

Einfache deterministische Lösung

- ausmultiplizieren
- bester bekannter Multiplikationsalgorithmus
  Coppersmith, Winograd (1990)
  $\mathcal{O}\left(n^{2,376\ldots}\right)$ Schritte

Algorithmus: Freivalds (1977)

1. Wähle zufällig gleichverteilt $\gamma \in \{0,1\}^n$
2. Berechne
   (a) $x := B \cdot \gamma$
   (b) $y := A \cdot x$
   (c) $z := C \cdot \gamma$
3. Antworte ja genau dann, wenn $y = z$

Fingerabdruck von C

Fingerabdruck von $A \cdot B$

Laufzeit $\mathcal{O}\left(n^2\right)$

Beobachtung:

- $A \cdot B = C \longrightarrow$ immer richtiges Ergebnis
- $A \cdot B \neq C \longrightarrow$ Fehlerwahrscheinlichkeit $\leq \frac{1}{2}$ (noch z. z.)

Satz:

Ist $A \cdot B \neq C$ und γ ein Vektor unabhängiger Zufallsbits, dann ist die Fehlerwahrscheinlichkeit des Algorithmus

$$Pr(A \cdot B \cdot \gamma = C \cdot \gamma) \leq \frac{1}{2}.$$

Beweis:

- Sei $D = A \cdot B - C$, also $D \neq 0$.
  Sei $y = A \cdot B \cdot \gamma$ und $z = C \cdot \gamma$.
- Dann gilt $y = z$ genau dann, wenn $D \cdot \gamma = 0$.
- Sei d die erste Zeile von D.
  O.B.d.A. sei $d = (d_1, \ldots, d_n)$ nicht der Nullvektor und die Nichtnulleinträge von d seien $d_1, \ldots, d_k$.

Beweis (Fortsetzung):

- Erster Eintrag in $D \cdot \gamma$ ist $d \cdot \gamma$
- Untere Schranke für Wahrscheinlichkeit, dass $d \cdot \gamma \neq 0$ $\longrightarrow$ untere Schranke für Wahrscheinlichkeit, dass $D \cdot \gamma \neq 0$
- Suche obere Schranke für Wahrscheinlichkeit, dass $d \cdot \gamma = 0$
  - $\sum_{i=1}^{k} d_i \cdot \gamma_i = 0 \Leftrightarrow \gamma_k = \left(-\sum_{i=1}^{k-1} d_i \cdot \gamma_i\right) / d_k$
  - $\gamma_1, \ldots, \gamma_{k-1}$ bereits gewählt
    - $\rightarrow$ Gleichung für höchstens einen der beiden Werte für γ_k richtig
    - $\rightarrow$ Wahrscheinlichkeit höchstens $\frac{1}{2}$
- Mit Wahrscheinlichkeit mindestens $\frac{1}{2}$ ist $d \cdot \gamma \neq 0$ und damit $D \cdot \gamma \neq 0$.

$\blacksquare$

Verringerung der Fehlerwahrscheinlichkeit

- Ziehe Komponenten des Zufallsvektors aus größerer Menge $S \subseteq \mathbb{F}$
- Probability amplification
  Führe k unabhängige Wiederholungen des Algorithmus durch, antworte nur mit ja, wenn jeder Durchlauf Ergebnis ja liefert.
  $\longrightarrow$ Fehlerwahrscheinlichkeit $\leq (\frac{1}{2})^k$

Bemerkung:

Monte Carlo Algorithmus (Algorithmus gibt manchmal falsche Lösung aus) mit einseitigem Fehler (er irrt sich nur, wenn $A \cdot B \neq C$)

Quicksort (Zufallsstichproben)

Algorithmus:

Eingabe: Menge S von paarweise verschiedenen Zahlen

Aufgabe: Sortiere S in aufsteigender Ordnung

1. Wähle zufällig gleichverteilt $y \in S$.
2. $S_1 := \{x \in S \mid x < y\}$,
   $S_2 := \{x \in S \mid x > y\}$.
3. Sortiere rekursiv S_1 und S_2.

Ausgabe: Sortierte Version von S_1, y, sortierte Version von S_2

Beobachtung:

Quicksort liefert immer korrekte Ausgabe:
Las Vegas Algorithmus

Wahl von y hat Einfluß auf Laufzeit
(auch bei festgehaltener Eingabe)

- immer kleinster Wert als Pivotelement gewählt
  $\longrightarrow$ quadratische Laufzeit
- immer Median als Pivotelement gewählt
  $\longrightarrow$ Laufzeit $\Theta(n \log n)$

Laufzeit des Algorithmus ist Zufallsvariable

Jetzt: Bestimmung des Erwartungswertes der Laufzeit

- Indikatorvariable $X_{ij} := \begin{cases} 1, & \text{wenn } S'_i \text{ und } S'_j \text{ verglichen werden} \\ 0, & \text{sonst} \end{cases}$

  i-tes Element in sortierter Reihenfolge

  j-tes Element in sortierter Reihenfolge

- gesuchter Erwartungswert ist gleich

$$E\left[\sum_{i=1}^{n}\sum_{j>i} X_{ij}\right] = \sum_{i=1}^{n}\sum_{j>i} E\left[X_{ij}\right]$$

$=p_{ij}$

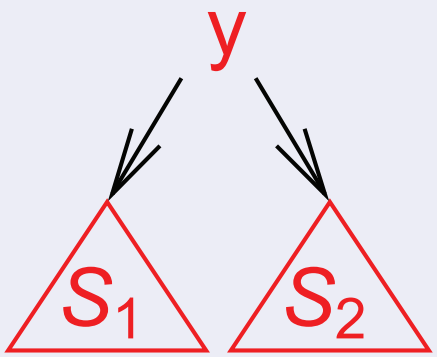
Wahrscheinlichkeit für Vergleich von S'_i und S'_j

Satz:

Der Erwartungswert der Laufzeit des randomisierten Quicksort-Algorithmus für n Elemente ist $\mathcal{O}(n \log n)$.

Beweis:

- Seien i und j ($1 \leq i < j \leq n$) beliebig aber fest.
- Betrachte den zur Ausführung des Algorithmus gehörigen Binärbaum, bei welchem die Knoten die jeweils gewählten Pivotelemente repräsentieren:



In-Order-Durchlauf ergibt sortierte Reihenfolge der Werte

Beobachtung:

Zwei Elemente werden genau dann miteinander verglichen, wenn das eine Vorgänger des anderen ist, also das andere in einem Unterbaum des einen ist.

Beweis (Fortsetzung):

- „schichtweiser“ Durchlauf durch den Binärbaum

  an Wurzel beginnen, absteigend zur nächsten Schicht, diese von links nach rechts durchlaufen

- Betrachte das erste Element S'_k, $i \leq k \leq j$, das auf diese Weise erreicht wird:
  - $k \notin \{i,j\}$: $\left.\begin{array}{l} S'_i \text{ liegt im linken Unterbaum} \\ S'_j \text{ liegt im rechten Unterbaum} \end{array}\right\} \rightarrow$ kein Vergleich von S'_i und S'_j
  - $k \in \{i,j\}$: S'_i und S'_j werden miteinander verglichen

$\longrightarrow p_{ij} = \frac{2}{j-i+1}$

Beweis (Fortsetzung):

$$\sum_{i=1}^{n}\sum_{j>i} E[X_{ij}] = \sum_{i=1}^{n}\sum_{j>i} p_{ij} = \sum_{i=1}^{n}\sum_{j>i} \frac{2}{j-i+1}$$

$$= \sum_{i=1}^{n}\sum_{k=2}^{n-i+1} \frac{2}{k}$$

$$\leq 2\sum_{i=1}^{n}\sum_{k=1}^{n} \frac{1}{k}$$

$$= 2nH_n$$

■

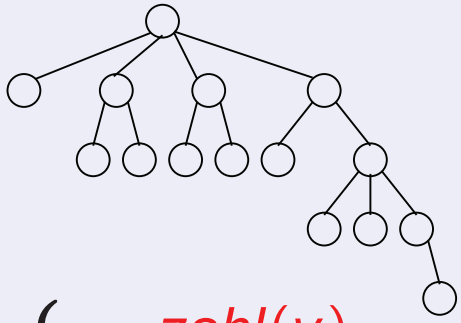
Bemerkung:

Keine Annahmen über Eingabeverteilung

Spielbaumauswertung (Vereitelung gegnerischer Angriffe)

Definition:

Ein Spielbaum ist ein gerichteter Baum mit Wurzel, in dem jedes Blatt v eine Bewertung $zahl(v)$ trägt.



Ziel:

Auswertung des Spielbaums, d.h. Berechnung von $wert(wurzel)$, wobei

$$wert(v) = \begin{cases} - \; zahl(v) & \text{falls } v \text{ Blatt} \\ - \; \min\{wert(v_1), \ldots, wert(v_r)\}, & \text{falls } v \text{ auf gerader Ebene und } v_1, \ldots, v_r \text{ alle Nachfolger von } v \\ - \; \max\{wert(v_1), \ldots, wert(v_r)\}, & \text{falls } v \text{ auf ungerader Ebene und } v_1, \ldots, v_r \text{ alle Nachfolger von } v \end{cases}$$

Jetzt:

Spezialfall, dass an Blättern nur Einsen und Nullen stehen

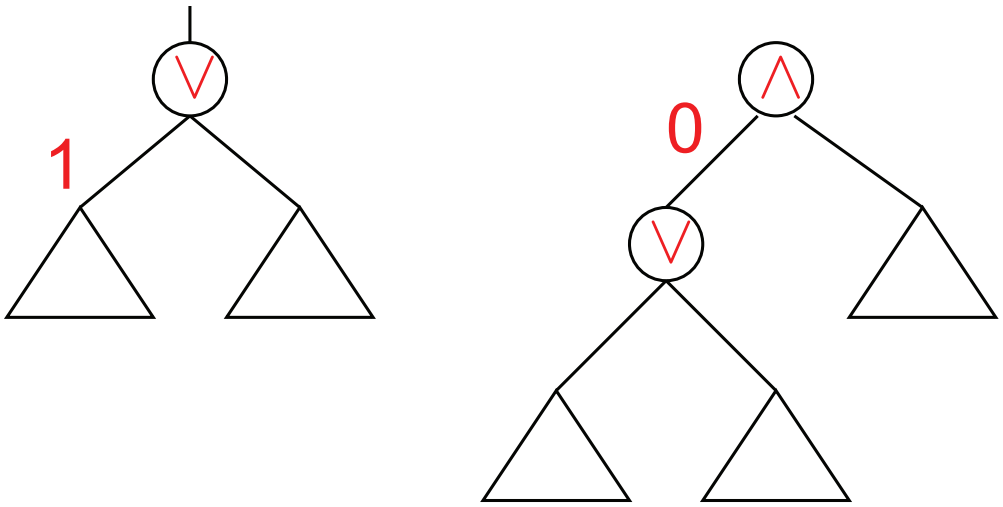
Definition:

$T_{d,k}$ bezeichnet die Menge der Bäume mit den folgenden Eigenschaften:

1. Jeder innere Knoten hat genau d Nachfolger.
2. Alle Blätter haben Entfernung $2k$ von der Wurzel.
3. Für alle Blätter gilt $zahl(v) \in \{0, 1\}$.

Beobachtung:

min $\longrightarrow \wedge$ max $\longrightarrow \vee$



Algorithmus:

Eingabe: $T_{2,k}$

Ausgabe: Auswertung des Spielbaums $T_{2,k}$ wobei möglichst wenig Blätter gelesen werden sollen

1. Auswertung eines $\wedge$-Knotens:
   (a) Wähle zufällig gleichverteilt einen der beiden Nachfolgerknoten und werte ihn rekursiv aus.
   (b) Falls der ausgewertete Nachfolger Wert 1 hat, werte auch den anderen Nachfolger aus; ansonsten gib 0 für den $\wedge$-Knoten aus.
2. Auswertung eines $\vee$-Knotens:
   Analog, wobei 1 und 0 vertauscht sind.

Satz:

Die erwartete Anzahl Blätter, die der Algorithmus zur Bestimmung des Wertes der Wurzel eines Baumes $T_{2,k}$ betrachten muss, beträgt höchstens 3^k.

(Erinnerung: $T_{2,k}$ hat $2^{2k} = 4^k$ Blätter)

$n^{0,793}$ für $n := 4^k$

Beweis: Induktion über k

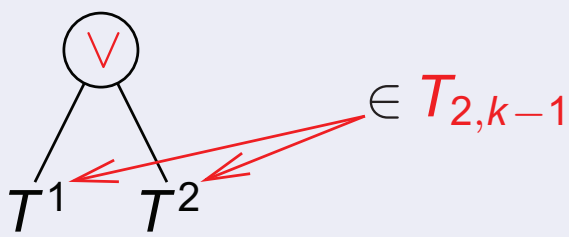
Induktionsanfang: $k = 0$

$T_{2,k}$ besteht aus einem Blatt.

Es reicht $3^0 = 1$ Blatt zu betrachten.

Induktionsschluss: (Erinnerung: per Definition Wurzel $\wedge$-Knoten)

„Vorüberlegung“:



- Wurzel $\vee$ Wert 0
- Wurzel $\vee$ Wert 1

Beweis (Fortsetzung):

- Wurzel $\vee$ Wert 0:

  T^1 und T^2 müssen ausgewertet werden

  $\xrightarrow{\text{Ind.vor.}}$ $2 \cdot 3^{k-1}$ Blätter

- Wurzel $\vee$ Wert 1:
  - mit Wahrscheinlichkeit mindestens $\frac{1}{2}$ wird Nachfolger mit Wert 1 ausgewählt
  - mit Wahrscheinlichkeit höchstens $\frac{1}{2}$ müssen beide Nachfolger betrachtet werden

  $\xrightarrow{\text{Ind.vor.}}$ $\frac{1}{2} \cdot 3^{k-1} + \frac{1}{2} \cdot 2 \cdot 3^{k-1} = \frac{3}{2} \cdot 3^{k-1}$ $\longleftarrow$ obere Schranke für Erwartungswert

Jetzt: Wurzel $\wedge$ eines $T_{2,k}$-Baumes untersuchen

Beweis (Fortsetzung):

- Wurzel $\wedge$ Wert 1:

  beide Unterbäume müssen betrachtet werden

  $\xrightarrow[\text{des Erwartungswertes}]{\text{Linearität}} 2 \cdot \frac{3}{2} \cdot 3^{k-1} = 3^k$

- Wurzel $\wedge$ Wert 0:
  - mit Wahrscheinlichkeit mindestens $\frac{1}{2}$ wird Nachfolger mit Wert 0 ausgewählt
  - mit Wahrscheinlichkeit höchstens $\frac{1}{2}$ müssen beide Nachfolger betrachtet werden

$\longrightarrow$ $\frac{1}{2} \cdot 2 \cdot 3^{k-1} + \frac{1}{2}(\frac{3}{2}3^{k-1} + 2 \cdot 3^{k-1}) = 2\frac{3}{4}3^{k-1} < 3^k$

$\longrightarrow$ insgesamt also Erwartungswert für betrachtete Blätter höchstens 3^k

■

Bemerkung:

- Las Vegas Algorithmus: immer korrekte Antwort, Laufzeit ($\hat{=}$ Anzahl betrachteter Blätter) hängt vom Zufall ab, keine Annahme über Verteilung der Eingaben
- Für jeden deterministischen Algorithmus existiert Instanz von $T_{2,k}$, so dass alle n Blätter betrachtet werden müssen.
- Jeder randomisierte Algorithmus muss im Durchschnitt $n^{0.793}$ Blätter auswerten.
  Saks, Wigderson (1986)

2SAT (Markov-Ketten)

Algorithmus:

Eingabe: Aussagenlogische Formel F in konjunktiver Form, wobei alle Klauseln genau 2 Literale enthalten

Frage: Existiert eine erfüllende Belegung für F?

1. Starte mit einer beliebigen Belegung a der Variablen von F.
2. Wiederhole r mal:
   (a) Falls es für a in jeder Klausel ein erfülltes Literal gibt, so antworte mit „Ja“, gib die aktuelle Belegung a aus und halte.
   (b) Wähle zufällig unerfüllte Klausel aus F, wähle zufällig Literal aus dieser Klausel und flippe den Wert der dazugehörigen Variable.
3. Gib aus „F vermutlich unerfüllbar“ und halte.

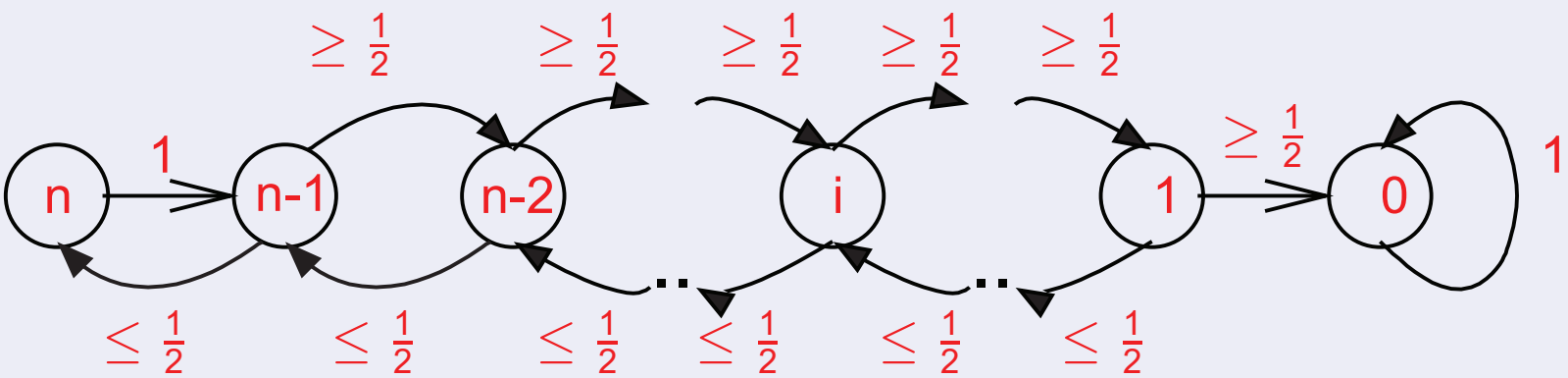
Satz:

Ist F erfüllbar, findet obiger Algorithmus für $r := 2n^2$ Wiederholungen, wobei n die Anzahl der Variablen über die F gegeben ist ist, mit Wahrscheinlichkeit mindestens $\frac{1}{2}$ eine erfüllende Belegung.

Beweis:

$H(a, b)$ bezeichne Hammingdistanz zwischen $a, b \in \{0, 1\}^n$.
a^* sei erfüllende Belegung für F
<u>*Idee :*</u> Verlauf von $H(a, a^*)$ als Random Walk

jeweils aktuelle Belegung des Algorithmus



Beweis (Fortsetzung):

- Wird Belegung einer Variablen geflippt, vergrößert oder verkleinert sich Hammingdistanz um 1.
- Wahrscheinlichkeit, dass Hammingdistanz sich verkleinert, ist mindestens $\frac{1}{2}$, denn a^* erfüllt die Klausel.

Frage: Wie groß ist die erwartete Anzahl von Schritten E_i bis 0 erreicht wird, wenn wir in i starten?

Beobachtung: $E_i \leq \frac{1}{2}E_{i-1} + \frac{1}{2}E_{i+1} + 1$

Hier: obere Schranke für E_i gesucht, vereinfachende Annahme außerdem

$$E_i = \frac{1}{2}E_{i-1} + \frac{1}{2}E_{i+1} + 1$$
$$E_0 = 0$$
$$E_n = 1 + E_{n-1}$$

(Vorauss.: $E_i < \infty$)

Beweis (Fortsetzung):

Behauptung: $E_i = i(2n - i)$

Beweis: $E_i = \frac{1}{2}E_{i-1} + \frac{1}{2}E_{i+1} + 1$

$$\Leftrightarrow \frac{1}{2}\underbrace{(E_i - E_{i-1})}_{D_i} = \frac{1}{2}(E_{i+1} - E_i) + 1$$

$$\frac{D_i}{2} = \frac{D_{i+1}}{2} + 1$$

$$\Leftrightarrow \quad D_i = D_{i+1} + 2$$

$$D_n = E_n - E_{n-1} = 1$$

$\longrightarrow$ $D_n = 1$, $D_{n-1} = 3$, $D_{n-2} = 5$, ..., $D_1 = 2(n-1) + 1$

$\longrightarrow$ $D_1 + D_2 + \ldots + D_i = (E_1 - E_0) + (E_2 - E_1) + \ldots + (E_i - E_{i-1})$

$= E_i - E_0 = E_i$

Also: $E_i = (2n - 1) + (2n - 3) + \ldots = 2n \cdot i - (1 + 3 + \ldots) = 2n \cdot i - i^2$

Beweis (Fortsetzung):

$E_i \leq n^2, 0 \leq i \leq n$

Definition: Markoff-Ungleichung

Sei $X \geq 0$ Zufallsvariable.
Dann gilt für alle $t > 0$: $Pr(X \geq t) \leq \frac{E[X]}{t}$

(siehe auch Folie 81)

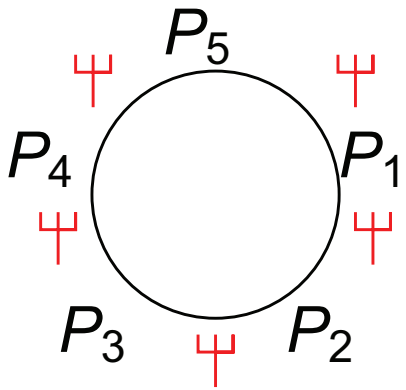
→ Mit Wahrscheinlichkeit mindestens $\frac{1}{2}$ findet der Algorithmus in höchstens $2n^2$ Schritten eine erfüllende Belegung für F.

■

Problem der speisenden Philosophen

(Aufbrechen von Symmetrien)

Eingabe: n an einem runden Tisch sitzende Philosophen mit je einer Gabel zwischen 2 beliebigen Philosophen



Um zu essen, muss ein Philosoph beide neben ihm liegenden Gabeln benutzen.

Gesucht: deadlock- und lockout-freies Protokoll

Gibt es (irgendwann) hungrigen Philosophen, dann wird (irgendwann) einer essen

Jeder hungrige Philosoph bekommt (irgendwann) etwas zu essen

Lehmann, Rabin (1981)

kein deterministischer Algorithmus, der sowohl
1) voll verteilt d.h. ohne zentralen Speicher/Prozess, auf den jeder andere Prozess zugreifen kann,

als auch
2) symmetrisch ist.

↑

- alle Prozesse führen das selbe Programm aus
- alle lokalen und gemeinsamen Variablen sind identisch initialisiert
- Prozesse kennen nicht ihre Identität
  → Prozessnummern können nicht verglichen werden

Algorithmus:

Lösung: randomisierte „Gabelaufnahmereihenfolge“

Protokoll: (für jeden Philosophen)

Wiederhole für immer folgendes:

1. Denke bis hungrig.
2. Wirf Münze um zu entscheiden, ob zuerst linke oder rechte Gabel aufgenommen werden soll.
3. Warte bis gewählte Gabel frei und nimm sie auf.
4. Falls andere Gabel nicht frei, lege gewählte Gabel wieder zurück und gehe nach 2.
5. Andere Gabel auch frei, dann nimm sie auf.
6. Iß.
7. Lege beide Gabeln nieder.

Anmerkung:

- Protokoll ist deadlock-frei, d.h. falls es hungrigen Philosophen gibt, wird mit Wahrscheinlichkeit 1 schließlich einer essen.
- Protokoll ist nicht lockout-frei, es gibt jedoch randomisierte Protokolle, die lockout-frei sind.

Kapitel 2 Klassifikation randomisierter Algorithmen

– Las Vegas und Monte Carlo Algorithmen

Beispiel:

Äquivalenztest für multilineare Ausdrücke
Test auf bipartites perfektes Matching

} MC-Algorithmen

– „randomisierte Komplexitätsklassen“:

ZPP, *RP*, *BPP*, *PP*

– Zusammenhang zwischen den Komplexitätsklassen

– probability amplification

Randomisierte…

- … **Algorithmen** sind Algorithmen, bei denen der i-te Schritt von einem Münzwurf abhängt.
  (Formal: unabhängige Zufallsvariable X_i mit $\Pr(X_i = 0) = \Pr(X_i = 1) = \frac{1}{2}$.)
- … **Turingmaschinen** haben 2 Übergangsfunktionen δ_0, δ_1; davon wird in jedem Schritt eine zufällig ausgewählt.
- … **Registermaschinen** verwenden den Befehl RGOTO i: Gehe mit Wkeit $\frac{1}{2}$ zu Programmschritt i.

Rechenzeit eines rand. Algos A

Durchschnitt =
Erwartungswert

$t_A(x)$: durchschnittliche Rechenzeit bei Eingabe x

$t_A(n) := \max\{t_A(x) \mid |x| \leq n\}$

(maximale durchschnittliche Rechenzeit)

über alle Eingaben
mit Länge $\leq n$

über die
Zufallsbits

Fehlerarten eines rand. Algos

1. **Fehlerfreie Algorithmen** (Las Vegas Algos)

- Erwartungswert der Rechenzeit ist beschränkt oder
- Ausgabe „? “ ist erlaubt.

Beispiel:

Quicksort, immer korrekt, erwartete Rechenzeit $\mathcal{O}(n \log n)$.

meistens
Correct

2. **Algos mit Fehlern** (Monte-Carlo-Algos)

- Rechenzeit ist unabhängig von den Zufallsbits beschränkt, das Ergebnis darf aber falsch sein.

Beispiel für einen Monte Carlo Algo

Äquivalenztest für multilineare Ausdrücke (ÄMA)

Eingabe: Multilineare Ausdrücke A_1, A_2 über einem endlichen Körper $\mathbb{F}_p$ (hier: p Primzahl)

Frage: Sind A_1 und A_2 äquivalent?

Multilineare Ausdrücke enthalten:

- Variablen $x_1, \ldots, x_n$, sowie Zahlen aus $\mathbb{F}_p$
- Verknüpfungen $+, -, \cdot$ in $\mathbb{F}_p$, sowie Klammern
- aber keine Quadrate von Variablen

Beispiel:

$(1 + x_1) \cdot (5 + 3 \cdot x_2 \cdot x_3) \cdot (4 + x_4) \bmod 127$

Beispiel:

Sind $(1 + x_1) \cdot (5 + 3 \cdot x_2 \cdot x_3) \cdot (4 + x_4)$ und
$20 + 60 \cdot x_1 \cdot x_2 \cdot x_3 \cdot x_4$ äquivalent $\bmod 127$?

1. Idee: ausmultiplizieren - i.A. exponentiell

2. Idee: Variablenbelegungen ausprobieren:
Z.B. $x_1 = x_2 = x_3 = x_4 = 0$: Ergebnisse 20 und 20.
Z.B. $x_1 = x_2 = x_3 = x_4 = 1$: Ergebnisse 80 und 80.

Besser: Zufällige Variablenbelegungen

Klar: Äquivalente Ausdrücke liefern immer dasselbe Ergebnis.

Frage: Können nichtäquivalente Ausdrücke für fast alle Eingaben gleich sein?

NEIN (wenn p hinreichend groß)

Satz: Schwartz-Zippel

Sei p eine Primzahl. Sei A ein multilinearer Ausdruck über $x_1, \ldots, x_n$ und $A \bmod p \neq 0$. Die Anzahl der Eingaben $x = (x_1, \ldots, x_n) \in \{0, \ldots p-1\}^n$ mit $A(x) \bmod p \neq 0$ ist mindestens $(p-1)^n$.

Algorithmus: Monte Carlo Algo für ÄMA ($p \geq 2n$)

Wähle zufällig $x = (x_1, \ldots, x_n) \in \{0, \ldots, p-1\}^n$.
Falls $(A_1(x) - A_2(x)) \bmod p = 0$
Ausgabe: „vermutlich äquivalent“
sonst Ausgabe: „nicht äquivalent“

Analyse:

- $A_1 = A_2$: Ausgabe „vermutlich äquivalent“ mit Wkeit 1
- $A_1 \neq A_2$: Ausgabe „nicht äquivalent“ mit Wkeit mindestens $\frac{(p-1)^n}{p^n} = \left(1 - \frac{1}{p}\right)^n \geq 1 - \frac{n}{p} \geq \frac{1}{2}$.

Beweis: Satz von Schwartz und Zippel

Induktion über n (Anzahl der Variablen)

$n = 1$: A ist lineare Funktion in einer Variablen und hat höchstens eine Nullstelle.

Satz (Schwartz und Zippel): Sei p eine Primzahl. Sei A ein multilinearer Ausdruck über $x_1, \ldots, x_n$ und $A \bmod p \neq 0$. Die Anzahl der Eingaben $x = (x_1, \ldots, x_n) \in \{0, \ldots p-1\}^n$ mit $A(x) \bmod p \neq 0$ ist mindestens $(p-1)^n$.

Beweis (Fortsetzung):

$n \geq 2$: Sei $a = (a_1, \ldots, a_n)$ mit $A(a) \bmod p \neq 0$.

$B(\omega) := A(a_1, \ldots, a_{n-1}, \omega)$ hat höchstens eine Nullstelle.

Für die mindestens $p - 1$ Werte ω mit $B(\omega) \neq 0$ hat $A(x_1, \ldots, x_{n-1}, \omega)$ nach I.V. mindestens $(p-1)^{n-1}$ „Nicht-Nullstellen“.

A hat $\geq (p-1)^{n-1} \cdot (p-1) = (p-1)^n$ „Nicht-Nullstellen“.

Satz (Schwartz und Zippel): Sei p eine Primzahl. Sei A ein multilinearer Ausdruck über $x_1, \ldots, x_n$ und $A \bmod p \neq 0$. Die Anzahl der Eingaben $x = (x_1, \ldots, x_n) \in \{0, \ldots p-1\}^n$ mit $A(x) \bmod p \neq 0$ ist mindestens $(p-1)^n$.

Zusammenfassung

- A_1, A_2 äquivalent

$\rightarrow$ Mit Wahrscheinlichkeit 1 Ausgabe „vermutlich äquivalent“.

- A_1, A_2 verschieden

$\rightarrow$ Mit Wahrscheinlichkeit $\geq \frac{1}{2}$ Ausgabe „nicht äquivalent“.

Wichtig: Wahrscheinlichkeit ergibt sich nur aus den Zufallsbits, nicht aus der Eingabe.

Einseitiger Fehler: Nur eine der möglichen Ausgaben kann falsch sein.

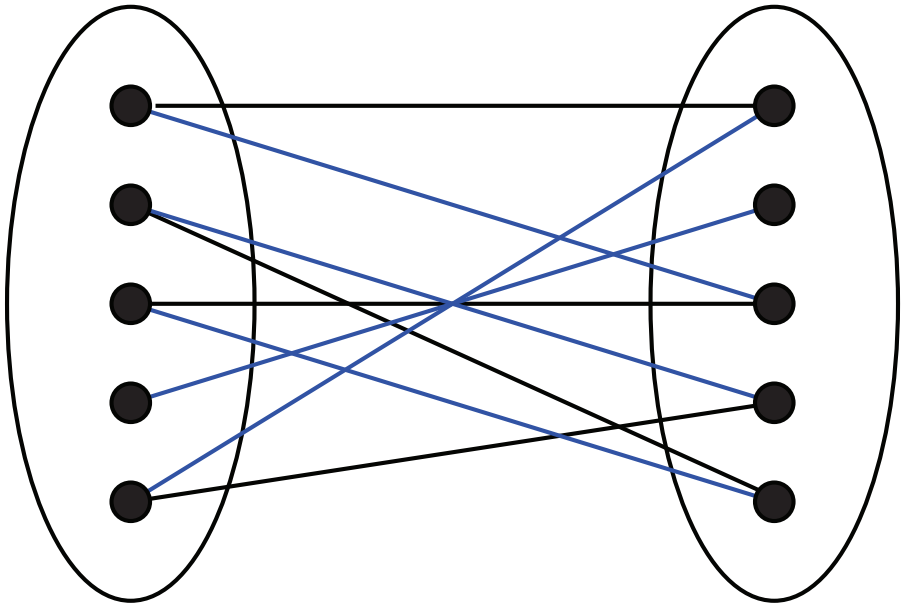
„false positives“

Anwendung des Algorithmus

Bipartites perfektes Matching (BPM)

Eingabe: Ein bipartiter Graph $G = (V_1, V_2, E)$ mit $|V_1| = |V_2| = n$

Frage: Enthält G ein perfektes Matching, d.h., gibt es $E' \subseteq E$ mit $|E'| = n$, so dass die Kanten in E' alle Knoten enthalten?



Adjazenzmatrix eines bipartiten Graphen

Knoten aus V_1

v_j

Knoten aus V_2

w_i

$$\begin{array}{ccc} & \vdots & \\ \cdots & 0/1 & \cdots \\ & \vdots & \end{array}$$

gibt an, ob v_j mit w_i verbunden ist

Tutte-Matrix eines bipartiten Graphen

Knoten aus V_1

v_j

Knoten aus V_2

w_i

$$\begin{array}{ccc} & \vdots & \\ \cdots & 0/x_{ij} & \cdots \\ & \vdots & \end{array}$$

Variable x_{ij}, falls w_i und v_j verbunden, 0 sonst

Kriterium für BPM

Satz: Edmonds

Sei A die Tutte-Matrix zum bipartiten Graphen G. Dann enthält G genau dann ein perfektes Matching, wenn $det(A) \neq 0$.

Anmerkung:

- $det(A)$ ist ein multilinearer Ausdruck in den Variablen $x_{11}, \ldots, x_{nn}$
- Es ist also zu testen, ob $det(A)$ gleich dem Ausdruck 0 ist.

Beweis: Satz von Edmonds

$$det(A) = \sum_{\pi \in S_n} sgn(\pi) x_{1,\pi(1)} x_{2,\pi(2)} \cdots x_{n,\pi(n)}$$

- Jeder Summand enthält verschiedene Mengen von Variablen
  $\Rightarrow$ Summanden heben sich nicht auf.
- Der Summand zu π ist genau dann ungleich 0, wenn die Kanten $\{1, \pi(1)\}, \ldots, \{n, \pi(n)\}$ vorhanden sind, es also ein BPM gibt.

Satz (Edmonds): Sei A die Tutte-Matrix zum bipartiten Graphen G. Dann enthält G genau dann ein perfektes Matching, wenn $det(A) \neq 0$.

Anwendung

- $det(A)$ ist ein multilinearer Ausdruck.
- Der Test, ob $det(A) \neq 0$ ist also mit dem beschriebenen Monte Carlo Algorithmus möglich, wähle dazu $p \geq 2n^2$.
- Resultat:
  - G enthält BPM $\Rightarrow$ Ausgabe „Ja“ mit Wahrscheinlichkeit $\geq \frac{1}{2}$.
  - G enthält kein BPM $\Rightarrow$ Ausgabe „Nein“ mit Wahrscheinlichkeit 1.

Komplexitätsklassen für Las Vegas Algos

EP(expected polynomial time):

Menge der Probleme mit einem fehlerfreien randomisierten Algorithmus mit erwarteter polynomieller Rechenzeit

ZPP($\varepsilon(n)$) (zero error probabilistic polynomial):

Menge der Probleme mit einem fehlerfreien randomisierten Algorithmus mit polynomieller Rechenzeit und Versagenswahrscheinlichkeit höchstens $\varepsilon(n) < 1$

gibt „?“ aus

Komplexitätsklassen für Monte Carlo Algos

BPP($\varepsilon(n)$) (bounded error probabilistic polynomial):

Menge der Probleme mit einem randomisierten Algorithmus mit polynomieller Rechenzeit und Fehlerwahrscheinlichkeit höchstens $\varepsilon(n) < \frac{1}{2}$

gibt etwas Falsches aus

Sonderfall: Entscheidungsprobleme

BPP($\varepsilon(n)$): zweiseitiger Fehler $\varepsilon(n) < \frac{1}{2}$:
$x \notin L$: Wahrscheinlichkeit $\geq 1 - \varepsilon(n)$ für verwerfen
$x \in L$: Wahrscheinlichkeit $\geq 1 - \varepsilon(n)$ für akzeptieren

RP($\varepsilon(n)$): (random polynomial)
einseitiger Fehler, bedeutet
$x \notin L$: Wahrscheinlichkeit 1 für verwerfen
$x \in L$: Wahrscheinlichkeit $\geq 1 - \varepsilon(n)$ für akzeptieren

D.h., Akzeptieren des Algorithmus ist garantiert richtig.

$BPM \in RP(\frac{1}{2})$

Erläuterung zu $RP(1/2)$

	Akzeptieren	Verwerfen
$x \notin L$	Mit Wkeit 0	Mit Wkeit 1
$x \in L$	Mit Wkeit $\geq \frac{1}{2}$	Mit Wkeit $\leq \frac{1}{2}$
	Akzeptieren garantiert richtig	Verwerfen ev. falsch

„false negatives“

Umgekehrter einseitiger Fehler

„false negatives“

RP($\varepsilon(n)$)
$x \notin L$: Wahrscheinlichkeit 1 für verwerfen
$x \in L$: Wahrscheinlichkeit $\geq 1 - \varepsilon(n)$ für akzeptieren

Das heißt, Akzeptieren ist garantiert richtig.

„false positives“

co-RP($\varepsilon(n)$)
$x \notin L$: Wahrscheinlichkeit $\geq 1 - \varepsilon(n)$ für verwerfen
$x \in L$: Wahrscheinlichkeit 1 für akzeptieren

Das heißt, Verwerfen ist garantiert richtig.

Wir haben gezeigt: ÄMA $\in$ *co-RP*$(\frac{1}{2})$.

Sind randomisierte Algorithmen sinnvoll?

Fehlerwahrscheinlichkeit oder Versagenswahrscheinlichkeit von 2^{-100} ist beispielsweise viel geringer als z.B. Wahrscheinlichkeit für Rechnerausfall.

Wichtig: Fehlerwahrscheinlichkeit bzw. Versagenswahrscheinlichkeit wird für alle Eingaben garantiert.

Gleich: Fehlerwahrscheinlichkeit bzw. Versagenswahrscheinlichkeit kann durch Wiederholung verkleinert werden.
„probability amplification“

Exkurs Markoff-Ungleichung

Satz:

Sei $X \geq 0$ eine Zufallsvariable. Dann gilt für alle $t > 0$:

$$Pr(X \geq t) \leq E(X)/t$$

Beweis:

$$\begin{aligned} E(X) &= \sum_x x \cdot Pr(X = x) \\ &\geq \sum_{x<t} x \cdot 0 + \sum_{x \geq t} t \cdot Pr(X = x) \\ &= t \cdot Pr(X \geq t). \end{aligned}$$

Zusammenhang zwischen den Komplexitätsklassen

Satz:

$EP = ZPP(1/2)$.

Beweis: $EP \subseteq ZPP(\frac{1}{2})$

Sei $L \in EP$ und A Algorithmus für L mit erwarteter Rechenzeit $p(n)$.
Modifiziere A so, dass nach Ablauf von $2p(n)$ Schritten abgebrochen wird und „?“ ausgegeben wird.

$\longrightarrow$ w.c. Rechenzeit: $2p(n)$, also polynomiell

$\longrightarrow$ $Pr(Ausgabe = ?) \leq \frac{1}{2}$ (Markoff-Ungleichung)

Beweis (Fortsetzung): $ZPP(\frac{1}{2}) \subseteq EP$

Sei $L \in ZPP(\frac{1}{2})$ und B Algo für L mit Rechenzeit $p(n)$ und Versagenswkeit $\leq \frac{1}{2}$.

Wiederhole B so oft, bis Ergebnis ungleich „?“.

$\longrightarrow$ Resultat korrekt

Anzahl an Wiederholungen ist geometrisch verteilte Zufallsvariable mit Parameter $\geq \frac{1}{2}$.

$\longrightarrow$ Erwartungswert höchstens 2

$\longrightarrow$ erw. Rechenzeit höchstens $2p(n)$

Exkurs Geometrische Verteilung

Motivation

Zufallsexperiment hat Ausgänge 1, 0 mit Wahrscheinlichkeiten $p \in [0,1]$ bzw. $1-p \in [0,1]$.
Wiederhole das Experiment so lange bis zum ersten Mal 1 auftaucht.

Pr(„beim $k$ten Versuch tritt 1 zum ersten Mal auf"): $(1-p)^{k-1} \cdot p$

$\in \mathbb{N}$

$X = k$

geometrisch verteilte Zufallsvariable mit Parameter p

Definition:

Geometrische Verteilung mit Parameter p ist Wkeitsmaß auf $\Omega = \mathbb{N}$ mit Zähldichte $f_p(k) := (1-p)^{k-1} \cdot p$.

(Es gilt $\sum_{k=1}^{\infty} f_p(k) = 1$ (Übung))

Eigenschaften geometrisch verteilter Zufallsvariablen

Sei X geometrisch verteilt zum Paramter p.

- Sei $n > 0$ und $k \geq 0$. Dann gilt:

  $\Pr(X = n + k \mid X > k) = \Pr(X = n)$

  Gedächtnislosigkeit

- $E[X] = \frac{1}{p}$

Verringern der Versagenswkeit

Satz:

$ZPP(1 - 1/p(n)) = ZPP(2^{-q(n)})$ für Polynome $p(n)$ und $q(n)$.

probability amplification

Beweis:

Wiederhole den Algo A mit Versagenswkeit $1 - 1/p(n)$ insgesamt $t(n)$-mal.
Neuer Algo versagt nur, wenn A in allen Wiederholungen versagt.

Wkeit dafür ist höchstens $\left(1 - \frac{1}{p(n)}\right)^{t(n)}$.

Beweis (Fortsetzung):

Wähle $t(n) = \lceil (\ln 2) \cdot p(n) \cdot q(n) \rceil$.

$\longrightarrow$ Versagenswkeit ist höchstens

$$\left(1 - \frac{1}{p(n)}\right)^{(\ln 2)p(n)q(n)} \leq e^{-(\ln 2)\cdot q(n)} = 2^{-q(n)}.$$

Hier benutzt:

$$(1 - 1/x)^x \leq e^{-1}.$$

$\longrightarrow$ $t(n)$ Polynom $\longrightarrow$ polynomielle Rechenzeit

Vereinfachung bei *ZPP*

Definition:

$ZPP := ZPP(\frac{1}{2})$

Konstante egal

ZPP^* : Menge der Probleme in $ZPP(\varepsilon(n))$ mit $\varepsilon(n) < 1$

ZPP : Probleme mit praktisch relevanten Algos, da Versagenswkeit exponentiell klein gemacht werden kann

ZPP^* praktisch irrelevant

Probability Amplification bei *RP*

Satz:

$RP(1 - 1/p(n)) = RP(2^{-q(n)})$ für Polynome $p(n)$ und $q(n)$.

Beweis:

Wiederhole Algo A mit einseitigem Fehler $1 - 1/p(n)$ insgesamt $t(n)$-mal.

Akzeptiere, wenn in mindestens einer Wiederholung akzeptiert wurde.

Wähle $t(n)$ wie bei ZPP.

Vereinfachung bei *RP*

Definition:

$RP := RP(\frac{1}{2})$

Konstante egal

RP^*: Menge der Probleme in $RP(\varepsilon(n))$ mit $\varepsilon(n) < 1$

RP: Probleme mit praktisch relevanten Algos, da Fehlerwkeit exponentiell klein gemacht werden kann

RP^* praktisch nicht relevant

Exkurs Chernoff-Ungleichung

Satz:

Sei $0 < p < 1$ und seien $X_1, \ldots, X_n$ unabhängige 0-1-Zufallsvariablen mit $\Pr(X_i = 1) = p$ und $\Pr(X_i = 0) = 1 - p$.

Sei $X = X_1 + \cdots + X_n$.

Dann ist $E(X) = np$ und für alle $0 < \delta < 1$ gilt $\Pr(X \leq (1 - \delta) \cdot E(X)) \leq e^{-E(X)\delta^2/2}$.

Probability Amplification bei *BPP*

Satz:

$BPP(\frac{1}{2} - \frac{1}{p(n)}) = BPP(2^{-q(n)})$ für Polynome $p(n)$ und $q(n)$, wenn die Komplexitätsklassen auf *eindeutig lösbare* Probleme eingeschränkt sind.

Beweis:

Wiederhole Algo A mit zweiseitigem Fehler $\frac{1}{2} - \frac{1}{p(n)}$ insgesamt $t(n)$-mal mit

$$t(n) = \lceil (2 \cdot \ln 2) \cdot p(n)^2 \cdot q(n) \rceil.$$

X_i : 0-1-Zufallsvar., die angibt, ob A in der i-ten Wiederholung erfolgreich ist

$$E(X_i) = \Pr(X_i = 1) \geq \frac{1}{2} + \frac{1}{p(n)} =: s(n).$$

Beweis (Fortsetzung):

$X = X_1 + \cdots + X_{t(n)}$: Anzahl der erfolgreichen Iterationen

$E(X) \geq t(n)s(n) > t(n)/2.$ $\qquad s(n) = \frac{1}{2} + \frac{1}{p(n)}$

Wkeit, dass Mehrheitsentscheidung falsch:

$$\Pr(X \leq \tfrac{t(n)}{2}) \leq \Pr(X \leq \tfrac{E(X)}{2s(n)})$$

$$= \Pr(X \leq (1 - \underbrace{(1 - \frac{1}{2s(n)})}_{\delta})E(X))$$

$$\overset{\text{Chernoff-Ungl.}}{\leq} e^{-E(X)\delta^2/2}$$

Abschätzung des Exponenten

Beweis (Fortsetzung):

$$-E(X) \cdot \delta^2/2$$

$$\leq -t(n) \cdot s(n) \cdot \delta^2/2$$

$$= -t(n)\left(\frac{1}{2} + \frac{1}{p(n)}\right)\frac{2}{(p(n)+2)^2}$$

$$= \cdots = -\frac{t(n)}{p(n)(p(n)+2)}$$

$$\leq -\frac{2(\ln 2)p(n)^2 q(n)}{2p(n)^2}$$

$$= -q(n)\ln 2$$

$$E(X) \geq t(n)s(n)$$

$$s(n) = \frac{1}{2} + \frac{1}{p(n)}$$

$$\delta = 1 - \frac{1}{2s(n)}$$

$$= \cdots = \frac{2}{p(n)+2}$$

Zusammenfassung

Beweis (Fortsetzung):

$$\Pr(X \leq \frac{t(n)}{2}) \leq e^{-E(X)\delta^2/2} \leq e^{-q(n)\ln 2} \leq 2^{-q(n)}$$

Wkeit, dass Mehrheitsentscheidung aus den Ergebnissen der $t(n)$ Iterationen falsch ist

Da $t(n)$ polynomiell, ist Gesamtrechenzeit polynomiell.

Vereinfachung *BPP*

Definition:

$BPP := BPP(\frac{1}{3})$

Jede Konstante $< \frac{1}{2}$ möglich.

PP : Menge der Probleme in $BPP(\varepsilon(n))$ mit $\varepsilon(n) < \frac{1}{2}$

Kein „*B*“, da $\varepsilon(n)$ exponentiell nah an $\frac{1}{2}$ möglich

BPP : Probleme mit praktisch relevanten Algos, da Fehlerwkeit exponentiell klein gemacht werden kann

PP (probabilistic polynomial) praktisch irrelevant

Zusammenhang zwischen Komplexitätsklassen

Satz:

$$
\begin{array}{ccc}
BPP & \subseteq & PP \\
\cup\!\mid & & \updownarrow\ ? \\
ZPP & \subseteq & ZPP^{*} \\
\cup\!\mid & & \\
P & &
\end{array}
$$

Hier: Komplexitätsklassen für Berechnungsprobleme

Komplementbildung

Definition:

Sei L ein Entscheidungsproblem. Das Komplement $\overline{L}$ von L ist das Problem, das aus L durch Umdrehen der Frage entsteht.

Beispiel:

Komplement des Primzahltests ist die Frage, ob die geg. Zahl zusammengesetzt ist.

Definition:

Sei C die Komplexitätsklasse. Dann ist $co - C = \{L \mid \overline{L} \in C\}$.

co-C ist nicht das Komplement von C.

Abschluss gegen Komplement

Definition:

Eine Komplexitätsklasse C von Entscheidungsproblemen heißt gegen Komplementbildung abgeschlossen, wenn $C = co - C$.

Bemerkung:

$P = co - P$, $ZPP = co - ZPP$, $ZPP^* = co - ZPP^*$, $BPP = co - BPP$, $PP = co - PP$,
d.h., diese Klassen sind gegen Komplementbildung abgeschlossen.

Beweis:

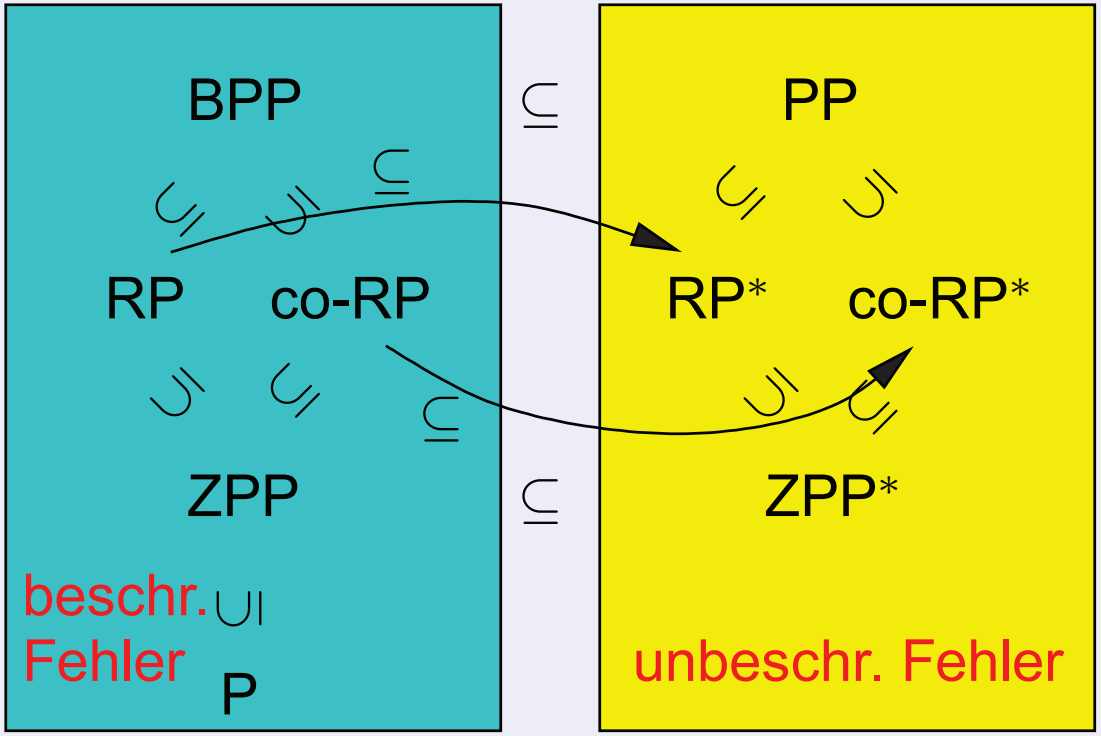
Folgt aus der Symmetrie von Akzeptieren und Verwerfen.

Beispiel:

Der Algorithmus für *ÄMA* zeigt, dass $ÄMA \in co - RP$ ist.

Vermutung: $RP \neq co - RP$

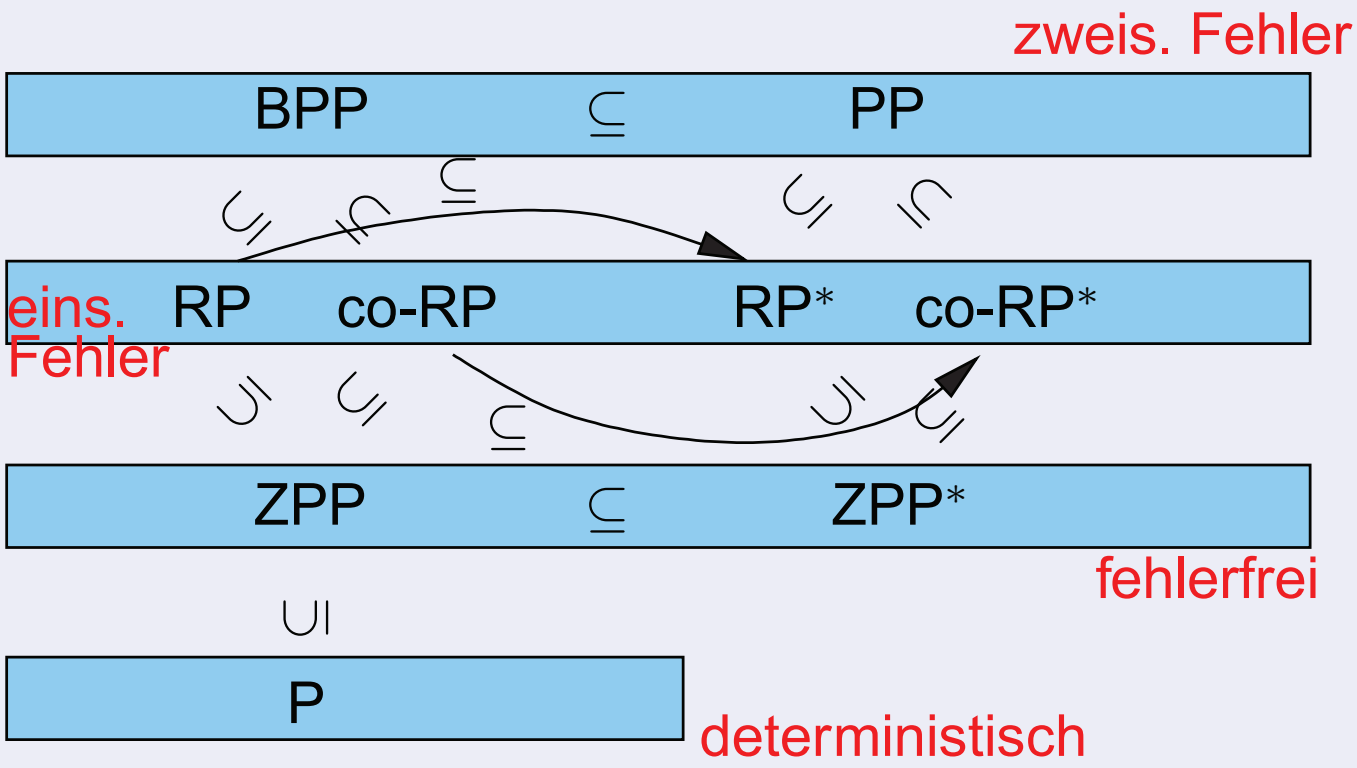
Satz:



Hier: Klassen für Entscheidungsprobleme

Satz:

(Fortsetzung)



Hier: Klassen für Entscheidungsprobleme

Fehlende Beweise

- $RP \subseteq BPP$

  $RP = RP(\frac{1}{3}) \subseteq BPP(\frac{1}{3}) = BPP$

- $ZPP \subseteq RP$ und $ZPP^* \subseteq RP^*$

  Ersetze Ausgabe „?“ durch Ausgabe „0“.

- $RP^* \subseteq PP$: etwas schwieriger

Idee für den Beweis $RP^* \subseteq PP$

Sei $L \in RP^*$ und A ein zugehöriger Algo.

- $x \notin L$: A akzeptiert mit Wkeit 0.
- $x \in L$: A akzeptiert mit Wkeit größer 0.

Ziel: Algo B mit

- $x \notin L$: B akzeptiert mit Wkeit kleiner $\frac{1}{2}$.
- $x \in L$: B akzeptiert mit Wkeit größer $\frac{1}{2}$.

Idee: Wkeiten „verschieben“, indem unabh. von der Eingabe mit Wkeit „knapp $\frac{1}{2}$“ akzeptiert wird und sonst A läuft

Beweis: $RP^* \subseteq PP$

Sei $L \in RP^*$ und A ein zugehöriger Algo.
Sei $p(n)$ die Rechenzeit von A.

$\rightarrow$ A verwendet höchstens $p(n)$ Zufallsbits.

$\rightarrow$ Alle Wkeiten Vielfaches von $2^{-p(n)}$, d.h.:
jede Wkeit$>$ 0 ist mindestens $2^{-p(n)}$.

Konstruiere **Algo** B**:**

- Akzeptiere mit Wkeit $\frac{1}{2} - \frac{1}{2 \cdot 2^{p(n)}}$
  (benötigt $p(n) + 1$ Zufallsbits).
- Ansonsten simuliere A.

Beweis (Fortsetzung): Zusammenfassung

$x \notin L \rightarrow B$ akzeptiert mit Wkeit $\frac{1}{2} - \frac{1}{2 \cdot 2^{p(n)}} < \frac{1}{2}$.

$x \in L \rightarrow B$ akzeptiert mindestens mit Wkeit

$$\underbrace{\left(\frac{1}{2} - \frac{1}{2 \cdot 2^{p(n)}}\right)}_{\text{Wkeit, vorab 1 zu berechnen}} + \underbrace{\left(\frac{1}{2} + \frac{1}{2 \cdot 2^{p(n)}}\right)}_{\text{Wkeit, A auszuführen}} \cdot \underbrace{\frac{1}{2^{p(n)}}}_{\text{Wkeit, dass A 1 berechnet}} > \frac{1}{2}.$$

Zusammenhang zwischen *ZPP* und *RP*

Satz:

$ZPP = RP \cap co - RP$.

Beweis:

$ZPP \subseteq RP$ und $ZPP \subseteq co - RP$ s.o.
$RP \cap co - RP \subseteq ZPP$:

Sei $L \in RP \cap co - RP$.
Sei A der RP-Algo für L und B der RP-Algo für $\overline{L}$.

Konstruiere ZPP-Algo C für L:

- Simuliere A. Falls A akzeptiert, akzeptiere.
- Simuliere B. Falls B akzeptiert, verwirf.
- Ansonsten gib „?“aus.

Zusammenhang zwischen ZPP^* und RP^*

Analog folgt: $ZPP^* = RP^* \cap co - RP^*$.

Primzahltest:

- Fermat-Test
- Miller-Rabin Test

Kap. 3

Anwendung:

- Generierung zufälliger Primzahlen
- Pattern Matching
- FBDD Äquivalenz

} Fingerprinting

Kapitel 3 Primzahltest

Idee: kleinen Satz von Fermat nutzen,
dabei sogenannte falsche Zeugen,
Fermatsche Pseudo-Primzahlen, insbesondere
Carmichaelsche Zahlen, umgehen

Einsatz: Maple, Mathematica

Kleiner Satz von Fermat
Sei p eine Primzahl und $a \in \mathbb{N}$ mit $a \not\equiv 0 \bmod p$. Dann gilt $a^{p-1} \equiv 1 \bmod p$.

Zahlentheorie

Erinnerung:

Proposition 3.1.13:

$\exists x, y \in \mathbb{Z} : 1 = nx + my \longrightarrow n, m$ teilerfremd

Beweis:

$d \mid n$ und $d \mid m \longrightarrow d \mid 1 \longrightarrow ggT(n, m) = 1$

Proposition 3.3.6:

(a) $m \mid a \cdot b$ und $ggT(m, a) = 1$

$\longrightarrow m \mid b$

Ist p Primzahl und teilt $a \cdot b$, teilt p die Zahl a oder b.

Erinnerung (Fortsetzung):

Satz: Chinesischer Restklassensatz

Sei $n = n_1 \cdot n_2 \cdot \ \cdots \ \cdot n_k$ mit $ggT(n_i, n_j) = 1$, $1 \leq i < j \leq k$.

Abbildung $\Phi : \mathbb{Z}_n \rightarrow \mathbb{Z}_{n_1} \times \ldots \times \mathbb{Z}_{n_k}$ mit
$a \mapsto (a \bmod n_1, a \bmod n_2, \ldots, a \bmod n_k)$ ist bijektiv, wobei gilt:

(a) $\Phi(a +_n b) = (a_1 +_{n_1} b_1, \ldots, a_k +_{n_k} b_k)$

(b) $\Phi(a \cdot_n b) = (a_1 \cdot_{n_1} b_1, \ldots, a_k \cdot_{n_k} b_k)$

(c) $\Phi(a^m \bmod n) = (a_1^m \bmod n_1, \ldots, a_k^m \bmod n_k)$.

Algebra

Erinnerung:

Lemma 4.1.6:

$(G, \circ, e)$ endliche Gruppe, $H \subseteq G$ mit

(i) $e \in H$

(ii) H abgeschlossen unter $\circ$

$\longrightarrow$ H Untergruppe von G

Proposition :

$(G, \circ, e)$ Gruppe, $a \in G$: $< a >:= \{a^i \mid i \in \mathbb{Z}\}$

$\uparrow$

kleinste Untergruppe von G, die a enthält

Erinnerung (Fortsetzung):

Definition:

$(G, \circ, e)$ Gruppe, $a \in G$

Ordnung $ord_G(a) = \begin{cases} |< a >|, & \text{wenn } |< a >| \text{ endlich} \\ \infty, & \text{sonst} \end{cases}$

Proposition 4.2.7:

$(G, \circ, e)$ Gruppe, $a \in G$ mit $ord_G(a) = m$, $m \geq 1$

(b) $a^i = a^j \Leftrightarrow m \mid j - i$

Beispiel:

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$2^{n-1} \bmod n$	1	0	1	2	1	0	4	2	1	8	1	2	4	0	1

keine Primzahl nach kleinem Satz von Fermat
2 Fermat Zeuge, dass 4 keine Primzahl

Definition 5.1.1:

$a, 1 \leq a < n$, heißt F-Zeuge für n, wenn $a^{n-1} \bmod n \neq 1$.

F-Zeuge bezeugt Zusammengesetztheit einer Zahl, liefert jedoch keinen Hinweis über mögliche Faktorisierung

2 ist F-Zeuge für alle zusammengesetzten Zahlen bis 340,

jedoch: $\left.\begin{array}{lcl} 341 & = & 11 \cdot 31 \\ 2^{340} & \equiv & 1 \bmod 341 \end{array}\right\}$ 2 F-Betrüger für 341

Definition 5.1.2:

n zusammengesetzte, ungerade Zahl a, $1 \leq a < n$, heißt F-Betrüger für n, wenn $a^{n-1} \bmod n = 1$

1 und $n - 1$ F-Betrüger für alle zusammengesetzten ungeraden n

Definition 3.3.7:

$m \geq 1$: $\mathbb{Z}_m^* := \{a \mid 1 \leq a < m, ggT(a,m) = 1\}$

$\varphi(m) := \mid \mathbb{Z}_m^* \mid$

Eulers φ-Funktion

Proposition 3.3.8:

(a) $1 \in \mathbb{Z}_m^*$

(b) $a, b \in \mathbb{Z}_m^* \rightarrow a \cdot_m b \in \mathbb{Z}_m^*$

(c) $a \in \mathbb{Z}_m^* \Leftrightarrow \exists b \in \mathbb{Z}_m^* : a \cdot_m b = 1$

Beweis:

(b) z.z. $\underbrace{a, b \in \mathbb{Z}_m^*}$ $\rightarrow a \cdot_m b \in \mathbb{Z}_m^*$

$$\rightarrow ggT(a, m) = ggT(b, m) = 1$$

$$\rightarrow ggT(a \cdot b, m) = 1$$

$$\rightarrow ggT(ab \bmod m, m) = 1 \checkmark$$

Proposition 3.1.7

(d) $ggT(n, m) = ggT(n + mx, m)$

(c) „$\Rightarrow$“ z.z. $\underbrace{a \in \mathbb{Z}_m^*}$ $\rightarrow \exists b \in \mathbb{Z}_m : a \cdot_m b = 1$

$$ggT(a, m) = 1 \rightarrow \exists x, y : 1 = a \cdot x + m \cdot y$$

$$\rightarrow ax \equiv 1 \bmod m$$

$$\rightarrow b = x \bmod m$$

„$\Leftarrow$“ $a \cdot_m b = 1 \rightarrow ab - 1 = mx \Leftrightarrow 1 = -mx + ab$

$$\xrightarrow[3.1.13]{\text{Propos.}} ggT(a, m) = 1$$

Lemma 5.1.3:

$n \geq 2$ und $n \in \mathbb{N}$

(a) Wenn $1 \leq a < n$ und $a^r \bmod n = 1$ für ein $r \geq 1$, gilt $a \in \mathbb{Z}_n^*$.

(b) Wenn $a^{n-1} \bmod n = 1$ für alle $a, 1 \leq a < n$, ist n Primzahl.

Beweis:

(a) Wenn $a^r \bmod n = 1$, ist $a \in \mathbb{Z}_n^*$
(a^{r-1} inverses Element Proposition 3.3.8 (c)).

(b) Aus (a) folgt $\mathbb{Z}_n^* = \{1, 2, \ldots, n-1\}$, also n Primzahl.

Lemma 5.1.3 (a)

→ für jede ungerade, zusammengesetzte Zahl n existiert F-Zeuge:

$$\{1, 2, \ldots, n-1\} - \mathbb{Z}_n^* = \{a \mid 1 \leq a < n,\ ggT(a, n) > 1\}$$

$$\#n - 1 - \varphi(n)$$

Aber: $\varphi(n)$ kann sehr groß sein

Beispiel:

$n = \underbrace{p \cdot q}_{\text{Primfaktorzerlegung von n}}$

$ggT(a, n) > 1$

$\Leftrightarrow \quad p \mid a$ oder $q \mid a$

$\# \; p + q - 2$

weitere F-Zeugen möglich

Beispiel: F-Zeugen und F-Betrüger für $n = 91 = 7 \cdot 13$

Vielfache von 7	7	14	21	28	35	42	49	56	63	70	77	84
Vielfache von 13	13	26	39	52	65	78						
F-Zeugen	2	5	6	8	11	15	18	19	20	24	31	32
in $\mathbb{Z}_{91}*$	33	34	37	41	44	45	46	47	50	54	57	58
	59	60	67	71	72	73	76	80	83	85	86	89
	1	3	4	9	10	12	16	17	22	23	25	27
F-Betrüger	29	30	36	38	40	43	48	51	53	55	61	62
	64	66	68	69	74	75	79	81	82	87	88	90

#F-Zeugen $>$ # F-Betrüger

Algorithmus 5.1.4: Fermat-Test

Eingabe: ungerade Zahl $n \geq 3$

1 a zufällig gleichverteilt aus $\{2, \ldots, n-2\}$
2 if $a^{n-1} \bmod n \neq 1$
3 then return 0;
4 else return 1;

Laufzeit: $\mathcal{O}\left(\log^3 n\right)$

$co - RP^*$-Algorithmus für Primzahltest

Frage: Fehlerwahrscheinlichkeit?

Satz 5.1.5:

Existiert für eine ungerade zusammengesetzte Zahl $n \geq 3$ mindestens ein F-Zeuge a in $\mathbb{Z}_n^*$, klassifiziert der Fermat Test n mit Wahrscheinlichkeit größer $\frac{1}{2}$ korrekt.

Beweis:

$L_n^F = \{a \mid 1 \leq a < n, a^{n-1} \bmod n = 1\} \subseteq \mathbb{Z}_n^*$

z.z. L_n^F Untergruppe von $\mathbb{Z}_n^*$

$\mathbb{Z}_n^*$ endlich $\xrightarrow[4.1.6]{Lemma}$ (i) $1 \in L_n^F$ ✓

(ii) L_n^F abgeschlossen unter $\cdot_n$ ✓

F-Zeuge in $\mathbb{Z}_n^*$ $\longrightarrow$ $L_n^F \subsetneq \mathbb{Z}_n^*$

Proposition 3.5.12:

Sei $\rho_1^{k_1} \cdot \rho_2^{k_2} \cdot \ldots \cdot \rho_r^{k_r}$ die Primfaktorzerlegung von n.

$\rho_i \neq \rho_j, i \neq j$

$$\varphi(n) = \prod_{1 \leq i \leq r} (\rho_i - 1)\rho_i^{k_i - 1} = n \prod_{1 \leq i \leq r} (1 - \tfrac{1}{\rho_i})$$

Beweis:

- $n = \rho_1$ ✓ $\quad n = 1$ ✓
- $n = \rho_1^{k_1}, \quad k_1 \geq 2$

$$\varphi(n) = \underbrace{\rho_1^{k_1} - 1}_{\{1,\ldots,n-1\}} - \underbrace{(\rho_1^{k_1-1} - 1)}_{\text{Vielfache von } \rho_1 \text{ in Menge } \{1,\ldots,\rho_1^{k_1}-1\}} = \rho_1^{k_1} - \rho_1^{k_1-1}$$

$$= \rho_1^{k_1}(1 - \tfrac{1}{\rho_1}) \checkmark$$

Beweis (Fortsetzung):

- $n = \rho_1^{k_1} \cdot \ldots \cdot \rho_r^{k_r}, \quad r \geq 2$

$$ggT(\rho_i^{k_i}, \rho_j^{k_j}) = 1, \; i \neq j$$

Erinnerung:

$\varphi(n) = \varphi(n_1) \cdot \varphi(n_2)$
für $n = n_1 \cdot n_2$
und $ggT(n_1, n_2) = 1$

$$\varphi(n) = \prod_{1 \leq i \leq r} \varphi(\rho_i^{k_i}) = \prod_{1 \leq i \leq r} \rho_i^{k_i}(1 - \tfrac{1}{\rho_i}) = n \prod_{1 \leq i \leq r} (1 - \tfrac{1}{\rho_i})$$

Probability amplification $\longrightarrow$ Fehlerwahrscheinlichkeit $< (\frac{1}{2})^l$

Leider: es gibt Zahlen n ohne F-Zeugen in $\mathbb{Z}_n^*$

Definition 5.1.7:

Ungerade zusammengesetzte Zahl n heißt Carmichaelsche Zahl, wenn $a^{n-1} \bmod n = 1, \forall a \in \mathbb{Z}_n^*$

Beispiel:

$561 = 3 \cdot 11 \cdot 17$

kleinste Carmichaelsche Zahl

Frage: Wieviele?

$\exists x_0$ mit $\forall x : x \geq x_0 : |\{n \mid n \leq x, n \text{ Carmichaelsche Zahl}\}| > x^{2/7}$

Fehlerwahrscheinlichkeit Carmichaelsche Zahl:

$$\frac{\varphi(n)-2}{n-3} > \frac{\varphi(n)}{n} \overset{\text{Propos. 3.5.12}}{=} \prod_{\rho \text{ mit } \rho|n} \left(1 - \frac{1}{\rho}\right)$$

Zahlen aus $\{2, \ldots, n-2\}$

Beispiel:

$n = 651693055693681 = 72931 \cdot 87517 \cdot 102103$

$\frac{\varphi(n)}{n} > 0.99996$

Probability amplification

Lemma 5.1.8:

n Carmichaelsche Zahl $\rightarrow$ n Produkt von mindestens drei verschiedenen Primzahlen

Beweis: Kontraposition (Ziel: F-Zeuge a in $\mathbb{Z}_n^*$)

1. Fall: $\underbrace{p^2}_{p \geq 3 \text{ ungerade}} \mid n,\ n = \underbrace{p^k}_{k \geq 2} \cdot m$ mit $p \nmid m$

$$m = 1 \quad \rightarrow \quad a = 1 + p$$

$$m \geq 3 \quad \rightarrow \quad 1 \leq a < p^2 m \text{ mit } \quad a \equiv 1 + p \bmod p^2$$

$$a \equiv 1 \qquad \bmod m$$

Existenz: Chinesischer Restklassensatz

- $a \in \mathbb{Z}_n^*$: $\quad p^2 \mid a - (1 + p) \quad \rightarrow p \nmid a$

$$ggT(a, m) = 1$$

$$\xrightarrow{3.3.6} \quad ggT(a, n) = 1 \checkmark$$

Beweis (Fortsetzung):

- a F-Zeuge (Kontraposition)

Annahme:

$$
\begin{aligned}
& a^{n-1} \equiv 1 \bmod n \\
\xrightarrow{p^2 | n} \quad & a^{n-1} \equiv 1 \bmod p^2 \\
& a^{n-1} \equiv (1+p)^{n-1} \\
& \qquad \equiv 1 + (n-1)p + \sum_{2 \le i \le n-1} \binom{n-1}{i} p^i \\
& \qquad \equiv 1 + (n-1)p \bmod p^2 \\
\rightarrow \quad & (n-1)p \equiv 0 \bmod p^2 \ \lightning \qquad p \nmid (n-1)
\end{aligned}
$$

✓

Beweis (Fortsetzung):

2. Fall:

$n = \underbrace{p \cdot q}_{\textit{Primzahlen}}$ mit $p > q$

Generator $\mathbb{Z}_p^*$

$1 \leq a < p \cdot q$ mit $a \equiv g \bmod p$

$a \equiv 1 \bmod q$

Existenz:

Chinesischer Restklassensatz

$\rightarrow a \in \mathbb{Z}_n^*$, denn $p \nmid a$ und $q \nmid a$

- a F-Zeuge (Kontraposition)

  $a^{n-1} \equiv 1 \bmod n$

  $g^{n-1} \equiv a^{n-1} \equiv 1 \bmod p$

Beweis (Fortsetzung):

$$g^{n-1} \equiv g^{p-1} \equiv 1 \bmod p$$

g Generator von $\mathbb{Z}_p^*$

$$\xrightarrow{4.2.7(b)} \quad (p-1) \mid (n-1) - (p-1)$$

$$\longrightarrow \quad (p-1) \mid (n-1)$$

$$\longrightarrow \quad (p-1) \mid \underbrace{p \cdot q - 1}_{=(p-1)\cdot q+(q-1)}$$

$$\longrightarrow \quad (p-1) \mid (q-1) \quad \lightning_{p>q}$$

✓

Nichttriviale Quadratwurzeln von 1

Definition 5.2.1:

a ist Quadratwurzel von 1 modulo n mit $1 \leq a < n$, wenn $a^2 \equiv 1 \bmod n$

Lemma 5.2.2:

p Primzahl und $1 \leq a < p$ mit $a^2 \equiv 1 \bmod p$,
dann $a = 1$ oder $a = p - 1$

triviale Quadratwurzeln

Beweis:

$(a^2 - 1) \bmod p = (a+1)(a-1) \bmod p = 0$
$\longrightarrow p \mid (a+1)$ oder $p \mid (a-1) \longrightarrow a = p-1$ oder $a = 1$ ✓

Beobachtung:

$n = p_1 \cdot \ldots \cdot p_r$, $p_i \neq p_j$ für $i \neq j$ und $p_i > 2,\ 1 \leq i \leq r$,
hat 2^r Quadratwurzeln von 1 modulo n:
$a,\ 0 \leq a < n$ mit $a \bmod p_j \in \{1, p_j - 1\}$, $1 \leq j \leq r$
($\#$: 2^r Chinesischer Restklassensatz)

Konsequenz:

Zufällige Suche von nichttrivialen Quadratwurzeln von 1 modulo n für Zahlen n mit wenigen Primfaktoren nicht vielversprechend

$$n - 1 = u \cdot 2^k, \quad k \geq 1$$

n ungerade

$$a^{n-1} \equiv ((a^u) \bmod n)^{2^k} \bmod n$$

also: Berechnung von $a^{n-1} \bmod n$
in $(k+1)$ Schritten möglich
$b_0 = a^u \bmod n$
$b_i = (b_{i-1})^2 \bmod n, \; 1 \leq i \leq k$

Beispiel:

$n = 325 = 5^2 \cdot 13$
$(n-1) = 324 = 81 \cdot 2^2$

a	$b_0 = a^{81} \bmod 325$	$b_1 = a^{162} \bmod 325$	$b_2 = a^{324} \bmod 325$
2	252	129	66
7	307	324	1
32	57	324	1
49	324	1	1
65	0	0	0
126	1	1	1
201	226	$\boxed{51}$	1
224	$\boxed{274}$	1	1

- 2 F-Zeuge für 325 in $\mathbb{Z}^*_{325}$
- 65 F-Zeuge nicht in $\mathbb{Z}^*_{325}$
- 7, 32, 49, 126, 201, 224 F-Betrüger
- 51, 274 nichttriviale Quadratwurzeln in $\mathbb{Z}^*_{325}$

Mögliche Folgen $b_0, \ldots, b_k$ (für a mit $1 \leq a < n-1$)

b_0	b_1	$\ldots$				$\ldots$	b_{k-1}	b_k	
1	1	$\ldots$	1	1	1	$\ldots$	1	1	keine Information, ob n Primzahl
$n-1$	1	$\ldots$	1	1	1	$\ldots$	1	1	
$*$	$*$	$\ldots$	$*$	$n-1$	1	$\ldots$	1	1	
$*$	$*$	$\ldots$	$*$	$*$	$*$	$\ldots$	$*$	$n-1$	a ist F-Zeuge
$*$	$*$	$\ldots$	$*$	$*$	$*$	$\ldots$	$*$	$*$	
$*$	$*$	$\ldots$	$*$	1	1	$\ldots$	1	1	Entdeckung von nichttrivialen Quadratwurzeln *modulo* n
$*$	$*$	$\ldots$	$*$	$*$	$*$	$\ldots$	$*$	1	

$b_0 \neq 1$ und $n-1$ taucht nicht auf $\rightarrow$ Wert für b_k unwichtig

Definition 5.2.3:

$n - 1 = u2^k, k \geq 1$

$ggT(u, 2) = 1$

$a, 1 \leq a < n$, A-Zeuge für n, wenn

- $a^u \not\equiv 1 \bmod n$ und
- $a^{u \cdot 2^i} \not\equiv n - 1 \bmod n$ für alle i mit $0 \leq i < k$

n zusammengesetzt:
a kein A-Zeuge für n, dann A-Betrüger für n.

Lemma 5.2.4:

a A-Zeuge für $n \longrightarrow n$ zusammengesetzt

- $\underbrace{\text{Artjukov (1966/67):}}_{\text{in russisch}}$ Vorschlag $b_i \equiv a^{u2^i} \bmod n$, $0 \leq i \leq k$, nutzen, um zu testen, ob n zusammengesetzt
- Miller (1976): unter Annahme Erweiterter Riemannscher Hypothese kleinster A-Zeuge für zusammengesetzte Zahl n in $\mathcal{O}\left(\ln^2 n\right)$
- Rabin (1980): Millers det. Suche für A-Zeuge $\rightarrow$ randomisierte Suche
- Bach (1990): Verbesserung auf $2\ln^2 n$

$\longrightarrow$ deterministischer Primzahltest mit $\mathcal{O}\left(\log^3 n\right)$ arithmetischen Operationen unter Erweiterter Riemannscher Hypothese

Algorithmus 5.2.5: Miller-Rabin Test

Eingabe: ungerade Zahl $n \geq 3$

1 Suche u und k mit $n - 1 = u \cdot 2^k$ und $ggT(u, 2) = 1$;
2 Wähle a zufällig gleichverteilt aus $\{2, \ldots, n-2\}$;
3 $b \leftarrow a^u \bmod n$;
4 if $b \in \{1, n-1\}$ then return 1;
5 repeat $k - 1$ mal
6 $\quad b \leftarrow b^2 \bmod n$;
7 $\quad$ if $b = n - 1$ then return 1;
8 $\quad$ if $b = 1$ then return 0;
9 return 0;

Laufzeit:

1 Höchstens $\log n$ Divisionen, um u und k zu bestimmen (schneller mit Shift-Operationen)

3 $a^u \bmod n$ mittels Schneller Exponentiation
$\mathcal{O}(\log n)$ arithmetische Operationen,
$\mathcal{O}\left(\log^3 n\right)$ $(\tilde{\mathcal{O}}(\log^2 n))$ Bit-Operationen

5-8 $k - 1 < \log n$ Schleifendurchläufe
Multiplikation $\bmod n$ teuerste Operation
$\rightarrow$ $\mathcal{O}(\log n)$ arithmetische Operationen,
$\mathcal{O}\left(\log^3 n\right)$ $(\tilde{\mathcal{O}}(\log^2 n))$ Bit-Operationen

Lemma 5.2.6:

Miller - Rabin Test ist co-RP-Algorithmus für Primzahltest.

Beweis:

z.z. Ausgabe 0 $\rightarrow$ A-Zeuge gefunden.

1. Fall: $a^{u2^i} \equiv 1 \bmod n$ für ein $i \in \{1, \ldots, k-1\}$
Zeile 8 und $a^{u2^j} \not\equiv n-1 \bmod n,\ j < i$
$\rightarrow$ a A-Zeuge

2. Fall: $a^{u2^i} \bmod n \notin \{1, n-1\}$ für alle $i \in \{0, \ldots, k-1\}$
Zeile 9 $\rightarrow$ a A-Zeuge

Fehlerwahrscheinlichkeit Miller-Rabin Test

(n zusammengesetzte, ungerade Zahl)

Erinnerung:

Analyse Fermat Test: L_n^F (echte) Untergruppe von $\mathbb{Z}_n^*$

F-Betrüger

Hier: Menge der A-Betrüger nicht notwendigerweise (echte) Untergruppe von $\mathbb{Z}_n^*$

Beispiel:

n=325, 7 und 32 A-Betrüger

$7 \cdot 32 \bmod 325 = 224$ A-Zeuge

$\rightarrow$ nicht abgeschlossen

Ausweg: Nachweis einer echten Untergruppe von $\mathbb{Z}_n^*$, die alle A-Betrüger enthält

$L_n^A \subseteq L_n^F \subseteq \mathbb{Z}_n^*$

1. Fall: $\mathbb{Z}_n^*$ enthält F-Zeugen, d. h. $L_n^F \subsetneq \mathbb{Z}_n^*$
Argumentation analog Beweis Satz 5.1.5 ✓

2. Fall: $L_n^F = \mathbb{Z}_n^* \rightarrow n$ Carmichaelsche Zahl
gesucht: echte Untergruppe B_n^A von $\mathbb{Z}_n^*$,
die alle A-Betrüger enthält, also $L_n^A \subseteq B_n^A$

$i_0 := \max i \geq 0 : \exists$ A-Betrüger a_0 mit $\boxed{a_0^{u \cdot 2^i} \equiv n - 1 \bmod n}$

i existiert, $\quad ggT(u, 2) = 1$

denn $(n-1)^u \equiv (-1)^u \equiv -1 \bmod n$

$0 \leq i_0 < k$, denn $a_0^{u \cdot 2^k} \mod n = a_0^{n-1} \mod n = 1$

Carmichaelsche Zahl

$B_n^A := \{a \mid 0 \leq a < n, a^{u \cdot 2^{i_0}} \bmod n \in \{1, n-1\}\}$

Lemma 5.3.1:

(a) $L_n^A \subseteq B_n^A$

(b) B_n^A Untergruppe von $\mathbb{Z}_n^*$

(c) $\mathbb{Z}_n^* - B_n^A \neq \emptyset$

$\longrightarrow$ Fehlerwahrscheinlichkeit Miller-Rabin Test höchstens $\frac{1}{2}$

Probability amplification $\longrightarrow$ Fehlerwahrscheinlichkeit höchstens $(\frac{1}{2})^l$

Beweis:

(a) a A-Betrüger

Fall 1: $a^u \bmod n = 1$

$\longrightarrow a^{u \cdot 2^{i_0}} \bmod n = 1 \longrightarrow a \in B_n^A$

Fall 2: $\exists\, i : a^{u \cdot 2^i} \bmod n = n - 1$

$\xrightarrow{\text{Def } i_0} 0 \leq i \leq i_0$

- $i = i_0 \rightarrow a \in B_n^A$ ✓
- $i < i_0 \rightarrow a^{u2^{i_0}} \bmod n = a^{u2^i \cdot 2^{i_0 - i}} \bmod n$

$= (a^{u2^i} \bmod n)^{2^{i_0 - i}} \bmod n$

$= 1 \rightarrow a \in B_n^A$ ✓

Beweis (Fortsetzung):

(b) $\xrightarrow{\text{Lemma 4.1.6}}$ reicht z.z.

(i) $1 \in B_n^A$, denn $1^{u \cdot 2^{i_0}} \bmod n = 1$

(ii) B_n^A abgeschlossen unter $\cdot$ mod n in $\mathbb{Z}_n^*$

$$a, b \in B_n^A : (ab)^{u2^{i_0}} \bmod n$$

$$= \underbrace{(\underbrace{(a^{u2^{i_0}} \bmod n)}_{\in\{1,n-1\}} \cdot \underbrace{(b^{u2^{i_0}} \bmod n)}_{\in\{1,n-1\}})}_{\in\{1,n-1\}} \bmod n$$

$$\rightarrow (ab) \bmod n \in B_n^A$$

Beweis (Fortsetzung):

(c) $n = n_1 \cdot n_2$

Carmichaelsche Zahl

$ggT(n_1, n_2) = 1$

$a_0^{u \cdot 2^{i_0}} \equiv -1 \bmod n$

A-Betrüger

$n - 1$

$$\left.\begin{aligned} a &\equiv a_0 \bmod n_1 \\ a &\equiv 1 \bmod n_2 \end{aligned}\right\} \xrightarrow{\text{Chinesischer Restklassensatz}} a \in \{0, \ldots, n-1\} \text{ eindeutig}$$

Lemma 5.1.8

Carmichaelsche Zahlen

- ungerade
- mindestens drei verschiedene Primfaktoren

Beweis (Fortsetzung):

z.z. $a \in \mathbb{Z}_n^* - B_n^A$

$$\boxed{\begin{aligned} a &\equiv a_0 \mod n_1 \\ a &\equiv 1 \mod n_2 \end{aligned}}$$

- $$\begin{aligned} a^{u \cdot 2^{i_0}} &\equiv (a \bmod n_1)^{u \cdot 2^{i_0}} \\ &\equiv -1 \bmod n_1 \end{aligned}$$

$\rightarrow a^{u \cdot 2^{i_0}} \not\equiv 1 \bmod n$

(denn: $a^{u \cdot 2^{i_0}} \equiv 1 \bmod n$

$\rightarrow a^{u \cdot 2^{i_0}} \equiv 1 \bmod n_1$)

$$\begin{aligned} a^{u \cdot 2^{i_0}} &\equiv (a \bmod n_2)^{u \cdot 2^{i_0}} \\ &\equiv 1^{u \cdot 2^{i_0}} \\ &\equiv 1 \bmod n_2 \end{aligned}$$

$\rightarrow a^{u \cdot 2^{i_0}} \not\equiv -1 \mod n$

$\longrightarrow a^{u \cdot 2^{i_0}} \bmod n \notin \{1, n-1\} \rightarrow a \notin B_n^A$

- $$\left.\begin{aligned} a^{u \cdot 2^{i_0+1}} &\bmod n_1 = 1 \\ a^{u \cdot 2^{i_0+1}} &\bmod n_2 = 1 \end{aligned}\right\} \xrightarrow{\text{Chines. Restklassensatz}} a^{u \cdot 2^{i_0+1}} \bmod n = 1$$

$\xrightarrow[\text{5.1.3 (a)}]{\text{Lemma}} a \in \mathbb{Z}_n^*$ ✓

Anwendung: Generierung zufälliger Primzahlen

Häufig vorkommende Aufgabe in der Kryptographie:

> Für gegebenes $l \in \mathbb{N}$ generiere zufällige Primzahl p der binären Länge l ($p \in [2^{l-1}, 2^l - 1]$).

Idee: zufällig gleichverteilte Zahl aus $[2^{l-1}, 2^l - 1]$ generieren und auf Primzahleigenschaft testen

Eingabelänge: $\lceil \log(l+1) \rceil$

Ausgabelänge: l

$\Big\}$ Komplexität eines Algorithmus in l messen

Algorithmus: Primzahl (l, k)

Eingabe: $l, k \in \mathbb{N}$ mit $l \geq 3$

1) $X :=$„noch nicht gefunden“;
   $J := 0$;

2) while $X :=$„noch nicht gefunden“ und $J < 2l^2$ do
   begin generiere zufällig gleichverteilte Bitfolge
   $a_1, a_2, \ldots, a_{l-2}$ und berechne
   $$n := 2^{l-1} + \sum_{i=1}^{l-2} a_i 2^i + 1;$$
   realisiere k unabhängige Läufe des
   Miller-Rabin Tests auf n
   if mindestens eine Ausgabe ist 0
   then $J := J + 1$
   else $X :=$„schon gefunden“; Output „n“
   end

3) if $J := 2l^2$ then Output „keine Primzahl gefunden“

Laufzeit:

- höchstens $2l^2$ Durchläufe der while-Schleife
- Miller-Rabin Test Laufzeit $\mathcal{O}\left(l^3\right)$ (Bitoperationen)
- pro Durchlauf der while-Schleife k Läufe des Miller-Rabin Tests

$\longrightarrow$ $\mathcal{O}\left(k \cdot l^5\right)$

Frage: Wie groß Wahrscheinlichkeit, dass generierte Zahl Primzahl ist?

Satz: Primzahlsatz (Hadamard/Vallée Poussin)

$\pi(n) \hat{=} \#$ Primzahlen in $\{1, \ldots, n\}$

$$\lim_{n \to \infty} \frac{\pi(n)}{n/\ln n} = 1$$

Hier: #Primzahlen im Intervall $[2^{l-1}+1, 2^l-1]$ gesucht

$\longrightarrow$ gesucht Abschätzung untere Schranke für $\pi(n) - \pi(n/2)$

$$\pi(n) = \frac{n}{\ln n}\,(1 \pm o(1)) \geq \frac{n}{\ln n}\,(1 - \varepsilon_n)$$

$$\pi(\tfrac{n}{2}) = \frac{n/2}{\ln(n/2)}\,(1 \pm o(1)) \leq \frac{n/2}{\ln n}\,(1 + \varepsilon'_n)$$

$$\pi(n) - \pi(\tfrac{n}{2}) \geq \frac{n}{\ln n} - \frac{n/2}{\ln n} - o(\frac{n}{\ln n})$$

$$= \frac{n}{2\ln n} - o(\frac{n}{\ln n})$$

→ Wkeit, dass Primzahl generiert wird, mindestens $1/\ln n - o(1/\ln n)$

↑ generierte Zahl aus $[2^{l-1}+1, 2^l - 1]$

n groß genug:
Wahrscheinlichkeit, dass Primzahl generiert wird, mindestens $\frac{1}{2 \ln n} > \frac{1}{2l}$

hier: $n = 2^l - 1$

$\longrightarrow$ Pr(Primzahl (l, k) = „Keine Primzahl gefunden“)

$$< ((1 - \frac{1}{2l}) \cdot \omega_k)^{2l^2}$$

$$< (1 - \frac{1}{2l})^{2l^2}$$

$$< (\frac{1}{e})^l$$

Wahrscheinlichkeit, dass Miller-Rabin Test in k unabhängigen Durchläufen zusammengesetzte Zahl richtig klassifiziert, also $\omega_k \geq 1 - \frac{1}{2^k}$

Frage: Wie groß ist die Wahrscheinlichkeit, dass zusammengesetzte Zahl als Primzahl ausgegeben wird?

Beobachtung: Primzahl (l, k) gibt zusammengesetzte Zahl nur dann als Primzahl aus, wenn

(i) alle vorher generierten Zahlen keine Primzahlen waren und als zusammengesetzte Zahlen identifiziert worden sind,

(ii) die Zahl selbst nicht als zusammengesetzte Zahl erkannt worden ist.

$$p_i \leq \left(\left(1 - \frac{1}{2l}\right) \cdot \omega_k\right)^{i-1} \cdot \left(1 - \frac{1}{2l}\right) \frac{1}{2^k}$$

Wahrscheinlichkeit, dass $i$te Zahl fälschlicherweise als Primzahl ausgegeben wird

Wahrscheinlichkeit, dass Primzahl (l, k) fälschlicherweise zusammengesetzte Zahl als Primzahl ausgibt, höchstens

$$\begin{aligned}
\sum_{i=1}^{2l^2} p_i &\leq \frac{1}{2^k} \cdot \sum_{i=1}^{2l^2} \left(1 - \frac{1}{2l}\right)^i \omega_k^{i-1} \leq \frac{1}{2^k} \cdot \sum_{i=1}^{2l^2} \left(1 - \frac{1}{2l}\right)^i \\
&\leq \frac{1}{2^k} \cdot \frac{1 - (1 - 1/(2l))^{2l^2+1}}{1 - (1 - 1/(2l))} \leq \frac{2l}{2^k}
\end{aligned}$$

Pattern Matching (Signaturmethode)

Eingabe: Sei Σ ein endliches Alphabet,
$X = (x_1, x_2, \ldots, x_n) \in \Sigma^n$ und
$Y = (y_1, y_2, \ldots, y_m) \in \Sigma^m$,
$n > m$

Frage: Existiert $j \in \{1, \ldots, n-m+1\}$, so dass
$\underbrace{x_j \ldots x_{j+m-1}}_{=:X(j)} = y_1 \ldots y_m$ gilt?

Offensichtlich: naiver Algorithmus mit Laufzeit
$$\mathcal{O}\left(m(n-m+1)\right) = \mathcal{O}\left(mn\right)$$

Besserer deterministischer Algorithmus:

$\left.\begin{array}{l}\text{Knuth, Morris, Pratt (1977)}\\ \text{Boyer, Moore (1977)}\end{array}\right\}$ Laufzeit $\mathcal{O}\left(n+m\right)$

$\longrightarrow$ VL Effiziente Algorithmen

Hier: randomisierter Algorithmus von Karp, Rabin (1987) mit erwarteter Laufzeit $\mathcal{O}(n+m)$

Las Vegas Algorithmus

Vorteil: im allgemeinen kleinere konstante Faktoren, geringerer Speicherbedarf

Idee: Ersetze im naiven Algorithmus Vergleich der jeweiligen Zeichenketten durch Vergleich ihrer Fingerabdrücke

können effizient berechnet werden

Vereinfachung: $\Sigma = \{0, 1\}$

Definition:

Fingerabdrucksfunktion $f_p := \mathbb{Z} \rightarrow \mathbb{Z}_p$ ist definiert als

$f_p(x) := x \bmod p$.

Für Binärzahlen $z \in \{0,1\}^m$ definieren wir

$f_p(z) := \mid z \mid_2 \bmod p$ mit $\mid z \mid_2 := \sum_{i=0}^{m-1} z_i \cdot 2^i$

Hier: p als Binärzahl codiert

Wichtige Beobachtung:

Berechnung von $f_p(X(j+1))$ aus $f_p(X(j))$ einfach:

$|X(j+1)|_2 = 2 \cdot (|X(j)|_2 - x_j 2^{m-1}) + x_{j+m} 2^0$

$\longrightarrow f_p(X(j+1)) = 2(f_p(X(j)) - x_j 2^{m-1}) + x_{j+m} 2^0 \bmod p$

$\longrightarrow \mathcal{O}(1)$ Operationen über $\mathbb{Z}_p$

Algorithmus: Binäres Pattern Matching

Eingabe: $x = (x_1, \ldots, x_n) \in \{0,1\}^n$,
$y = (y_1, \ldots, y_m) \in \{0,1\}^m$; $m \leq n$

1. Wähle zufällig gleichverteilt p aus der Menge der Primzahlen aus $[1, t \cdot m \cdot \ln(tm)]$.
2. Berechne $F_y := f_p(y)$ und $wert := 2^{m-1} \bmod p$
3. Berechne $F_x := f_p(X(1))$. $(*)$
4. while $F_x \neq F_y$ and $j \leq n - m$ do
   begin
   if $x_j = 1$ then $F_x := F_x - wert$
   $F_x := (2F_x + x_{j+m}) \bmod p$ $(*)$
   $j := j + 1$
   end
5. if $F_x = F_y$ then Output „(Vermutlich) ja“. STOPP
   else Output „Nein“. STOPP

Beobachtung:

- Algorithmus Binäres Pattern Matching ist Monte Carlo Algorithmus mit einseitigem Fehler
- j und F_x werden so aktualisiert, dass an Stellen $(*)$ gilt $F_x = f_p(X(j))$
- Parameter t bestimmt Fehlerwahrscheinlichkeit

Frage: Wie groß ist die Fehlerwahrscheinlichkeit, dass „(Vermutlich) ja“ ausgegeben wird, die Antwort jedoch „Nein“ lauten muss?

Indikatorvariable $Q_j := \begin{cases} 1, & \text{wenn } f_p(X(j)) = F_y, \text{ jedoch } X(j) \neq y \\ 0, & \text{sonst} \end{cases}$

$Pr(Q_j = 1)$?

Analyse der Fehlerwahrscheinlichkeit:

Primzahlsatz $\longrightarrow$ $\pi(tm \cdot \ln(tm)) = \Theta(tm)$
Wann gilt $u \equiv v \bmod p$, für $u, v \in \mathbb{Z}$?

Genau dann, wenn $u - v \equiv 0 \bmod p$,
also $\mid u - v \mid$ Vielfaches von p

Für wie viele Primzahlen kann $u - v \equiv 0 \bmod p$ gelten?
Maximal $\lfloor \log(\mid u - v \mid) \rfloor$. Warum?

Hier: $0 \leq \mid y \mid_2 < 2^m$ und $0 \leq |X(j)|_2 < 2^m$,
also $0 \leq \mid \mid y \mid_2 - |X(j)|_2 \mid < 2^m$, $1 \leq j \leq n - m + 1$

$\longrightarrow$ # Primzahlen, für die $F_y = f_p(X(j))$ gelten kann, höchstens m

Fortsetzung Analyse Fehlerwahrscheinlichkeit:

Größe der Primzahlen nicht entscheidend, sondern Kardinalität der Menge der Primzahlkandidaten

p zufällig gewählt

$$\longrightarrow Pr(Q_j = 1) = \mathcal{O}\left(\frac{m}{\pi(tm\ln(tm))}\right) = \mathcal{O}(\tfrac{1}{t})$$

$$\longrightarrow Pr(Q_1 = 1 \vee Q_2 = 1 \vee \ldots \vee Q_{n-m+1} = 1) \leq (n - m + 1)\mathcal{O}\left(\tfrac{1}{t}\right)$$

Für $t = n^2$

$\longrightarrow$ Fehlerwahrscheinlichkeit begrenzt durch $\mathcal{O}\left(\frac{1}{n}\right)$

Jetzt: Modifikation zum Las Vegas Algorithmus

Ersetze 5. durch if $F_x = F_y$
then if $X(j) = y$
then Output „Ja“
else if $j \leq n - m$ then „setze while-Schleife fort“
else Output „Nein“
else Output „Nein “

Obere Schranke für die erwartete Laufzeit:

$$Q := Q_1 + Q_2 + \ldots + Q_{n-m+1}$$

$$E(Q) = (n - m + 1) \cdot \mathcal{O}\left(\frac{1}{t}\right)$$

$\uparrow$

Linearität des Erwartungswertes

$\longrightarrow$ erwartete Laufzeit höchstens

$$\mathcal{O}(n+m) + m \cdot \mathcal{O}\left(\frac{n}{t}\right) = \mathcal{O}(n+m)$$

$\nearrow$

Annahme, dass Operationen in $\mathbb{Z}_p$ Kosten 1

Anmerkungen:

Algorithmus von Karp, Rabin

- leicht erweiterbar auf zweidimensionales Pattern Matching
- auch einsetzbar, wenn Zeichenkette X nur inkrementell gegeben ist, z.B. in Situationen, wo X aufgrund seiner Größe nicht in den Hauptspeicher passt

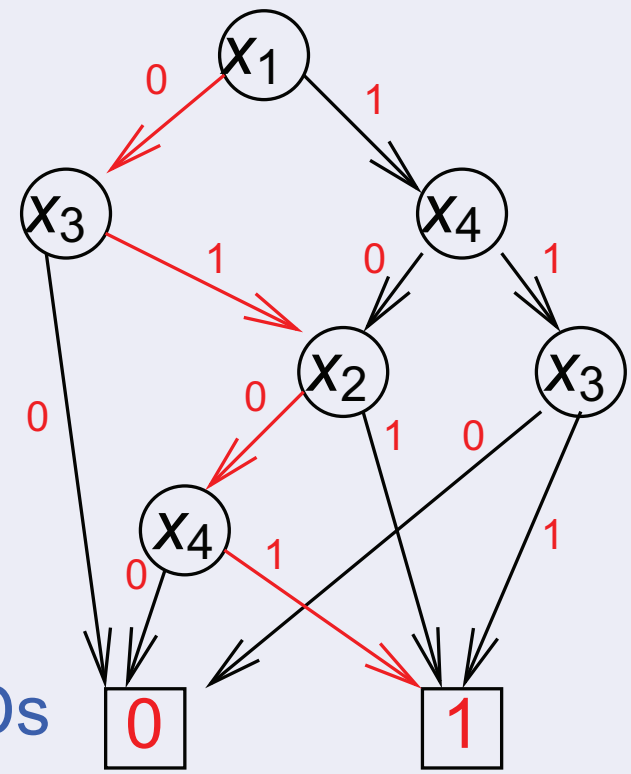
Äquivalenztest für FBDDs (Signaturmethode)

Eingabe:	FBDDs G_f und G_g, die die Booleschen Funktionen f und g aus B_n darstellen
Frage:	Gilt $f \equiv g$, also $f(a) = g(a)$ für alle $a \in \{0,1\}^n$?
Naive Idee:	Wähle zufällig gleichverteilt $a \in \{0,1\}^n$ und werte f und g auf a aus.
Problem:	$\#a$s mit $f(a) \neq g(a)$ kann sehr klein sein.
Ausweg:	„Raten“ der Variablenbelegung, Variablen jedoch aus größerem Definitionsbereich

Definition: Binary Decision Diagramms

- gerichteter azyklischer Graph
- Entscheidungsknoten:
  - markiert mit Variablen
  - ausgehende 0- und 1-Kante
- eine Quelle und 2 Senken: 0 und 1
- Einschränkungen:
  - read-once Eigenschaft FBDDs/BP1s
  - oblivious

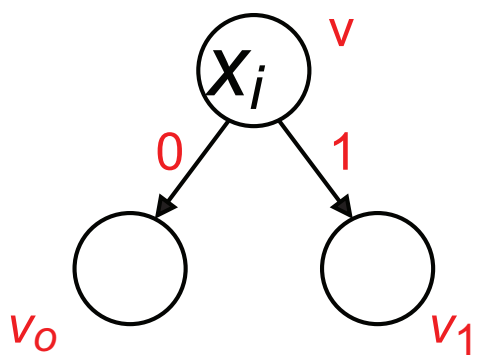
} OBDDs



Ziel: Konstruktionsvorschrift $G \curvearrowright p_G$,
wobei Funktionswert von G auf a gleich $p_G(a)$
für alle $a \in \{0,1\}^n$

Jedem Knoten v in G kann bottom-up Polynom p_v
zugeordnet werden:

- $p_{0-Senke} = 0$, $p_{1-Senke} = 1$
-



$$p_v := x_i p_{v_1} + (1 - x_i) p_{v_0}$$

- an Quelle von G wird p_G dargestellt

- Wähle Körper $\mathbb{Z}_p$ mit $p > 2n$
- Betrachte p_G über $\mathbb{Z}_p$:
  - wohldefiniert
  - $p_G(a)$ gleich Funktionswert von G auf a für alle $a \in \{0,1\}^n$

    auf jedem Weg jede Variable höchstens einmal
  - multilineares Polynom / multilinearer Ausdruck über $x_1, \ldots, x_n$
- Definiere $f^* : \mathbb{Z}_p^n \to \mathbb{Z}_p$ (Erweiterung der durch G dargestellten Funktion)

  als $f^*(r_1, \ldots r_n) := p_G(r_1, \ldots r_n)$ mit $r_i \in \{0, \ldots, p-1\}$, $1 \leq i \leq n$

Definition:

Sei $I \subseteq \{1, \ldots, n\}$ und $m_I := \prod_{i \in I} x_i$. $\leftarrow$ aus $\mathbb{F}$

Ein multilineares Polynom / multilinearer Ausdruck über Körper $\mathbb{F}$ ist ein Polynom der Form

$$p(x_1, \ldots, x_n) = \sum_{I \subseteq \{1,\ldots,n\}} c_I \cdot m_I,$$

wobei c_I Koeffizienten aus $\mathbb{F}$ sind.

Lemma:

Sei p ein multilineares Polynom über n Variablen. Wenn es $r = (r_1, \ldots, r_n) \in \mathbb{F}^n$ mit $p(r) \neq 0$ gibt, dann auch $a \in \{0, 1\}^n$ mit $p(a) \neq 0$.

Beweis:

- Sei $p(x_1, \ldots, x_n) = \sum_{I \subseteq \{1,\ldots,n\}} c_I \cdot m_I$.

mindestens ein I mit $c_I \neq 0$

- Wähle I' mit $c_{I'} \neq 0$ mit minimaler Kardinalität.

Beobachtung:

$m_I(a) = 1$ für $a \in \{0,1\}^n \Leftrightarrow \forall i \in I\colon a_i = 1$

- Wähle $a_i := \begin{cases} 1, & \text{wenn } i \in I' \\ 0, & \text{sonst} \end{cases}$

$\longrightarrow m_{I'}(a) = 1 \longrightarrow c_{I'} m_{I'}(a) \neq 0$

Beweis (Fortsetzung):

- $c_l m_l(a) = 0$ für $l \neq l'$
  - $\mid l \mid < \mid l' \mid$: $c_l = 0$ nach Definition von l'
  - $\mid l \mid \geq \mid l' \mid$: zu wenig oder „falsche“ Variablen auf 1 gesetzt.

$\longrightarrow$ $p(a) = c_{l'} m_{l'}(a) \neq 0$

Algorithmus: FBDD Äquivalenz

Eingabe: FBDD G_f und FBDD G_g

1. Berechne aus G_f und G_g f^* und g^*.
2. Wähle $r = (r_1, \ldots, r_n) \in \mathbb{Z}_p^n$ zufällig gleichverteilt.
3. Berechne $f^*(r)$ und $g^*(r)$.
4. if $f^*(r) = g^*(r)$ then Output „Vermutlich äquivalent“
   else Output „Nicht äquivalent“

Satz:

Algorithmus FBDD Äquivalenz ist co-RP-Algorithmus für $L_{FBDD} = \{(G_f, G_g) \mid G_f \text{ und } G_g \text{ FBDDs und } f = g)\}$.

Beweis:

- Laufzeit polynomiell ✓
- $f = g$, also $f(a) = g(a)$ für alle $a \in \{0,1\}^n$

  $\longrightarrow$ $f^*(r) = g^*(r)$ für alle $r \in \mathbb{Z}_p^n$

  $\longrightarrow$ Output „Vermutlich äquivalent“
- $f \neq g$, also existiert $a \in \{0,1\}^n$ mit $f(a) \neq g(a)$

  $\longrightarrow$ $f^*(a) \neq g^*(a)$

  $\longrightarrow$ $d := f^* - g^*$ von 0 verschieden, Grad durch n beschränkt

Beweis (Fortsetzung):

Grad von d durch n beschränkt, denn

- nur über $x_1, \ldots, x_n$ definiert
- multilinear

$\xrightarrow[\text{Schwartz-Zippel}]{\text{Satz von}}$ für zufällig gleichverteilt gewählte Belegung $r \in \mathbb{Z}_p^n$ gilt

$$\begin{aligned} \Pr_r(d(r) \neq 0) &\geq \frac{(p-1)^n}{p^n} \\ &= (1 - \tfrac{1}{p})^n \\ &\geq 1 - \frac{n}{p} \\ &\overset{p>2n}{\geq} \frac{1}{2} \end{aligned}$$

Satz: Schwartz-Zippel

Sei p Primzahl und A multilinearer Ausdruck über $x_1, \ldots, x_n$ mit $A \bmod p \neq 0$.

$\rightarrow$ # Belegung $r = (r_1, \ldots, r_n) \in \{0, \ldots, p-1\}^n$ mit $A(r) \bmod p \neq 0$ mindestens $(p-1)^n$.

$\rightarrow$ Output „Nicht äquivalent “mindestens mit Wahrscheinlichkeit $\frac{1}{2}$

■

Bemerkung:

- Algorithmus FBDD Äqivalenz geht zurück auf Fortune, Hopcroft, Schmidt (1978)
- Kein deterministischer Polynomialzeit-Algorithmus bekannt
- $L_{BDD} = \{(G_f, G_g) \mid G_f$ und G_g BDDs und $f = g\}$ co-NP-vollständig