Probabilistische Algorithmen

- sind Algorithmen, deren Verhalten von einer Zufallskomponente beeinflusst wird
- typischerweise enthalten sie Anweisungen wie „eine Münze zu werfen“
- die Laufzeit Algorithmus ist eine Zufallsvariable, die durch die Zufallsbits bestimmt wird mit (hoffentlich) gutem Erwartungswert

Berechnungsmodell

- probabilistische TM, Variante einer NTM
- jeder nichtdeterministische Schritt ist ein Münzwürf-Schritt,
- die Wahrscheinlichkeit eines Zweiges ist:

$$Pr[b] = 2^{-k}$$

- Akzeptierungswahrscheinlichkeit

$$Pr[\text{M accepts w}] = \sum_{\text{accepting branches}} Pr[b]$$

- positive Fehlerwahrscheinlichkeit ε ist erlaubt:
  1. $w \in A$ impliziert $Pr[\text{M accepts w}] \geq 1 - \varepsilon$
  2. $w \notin A$ impliziert $Pr[\text{M rejects w}] \geq 1 - \varepsilon$

Fehlerwahrscheinlichkeit darf von Eingabe abhängen z.B. $\varepsilon = 2^{-n}$.

Die Klasse BPP

- wir suchen Algorithmen, die effizient sind
- verwenden den schlechtesten Berechnungszweig als Grundlage für die Komplexität

Definition (8.1)

BPP ist die Klasse der Sprachen, die von PPTM mit Fehlerwahrscheinlichkeit $\varepsilon = 1/3$ erkannt werden.

Jede andere Konstante zwischen 0 und 0.5 induziert dieselbe Komplexitätsklasse.

Verstärkungs-Lemma

Lemma 8.1: Sei $0 < \varepsilon < 1/2$ und $P(n)$ ein Polynom. Für jede PPTM M_1 mit Fehlerwahrscheinlichkeit ε existiert eine äquivalente PPTM M_2 mit Fehlerwahrscheinlichkeit $\varepsilon^{P(n)}$.

Beweisidee:

- M_2 simuliert M_1, indem sie M_1 $P(n)$-mal laufen läßt und nach der Mehrheit der Antworten entscheidet
- $\varepsilon = 1/3$ entspricht Urnenmodell mit roten und blauen Kugeln im Verhältnis 1:2
- Test auf rot durch (z.B. 100-maliges) Ziehen. Die dominante Farbe kommt mit höherer Wahrscheinlichkeit öfter vor
- die Kugeln entsprechen dem Ausgang der Berechnung von M_1

Verstärkungs-Lemma

Beweis:
Sei M_1 PPTM mit Fehler $\varepsilon < 1/2$, $P(n)$ Polynom.

$t = 2^{P(n)}$,
$a = 1/(4\varepsilon(1-\varepsilon))$,
$b = \max\{1, 1/\log a\}$,
$c = 2\log(bt)$
$k = \lceil bc \rceil$

$M_2 =$ „Auf Eingabe w

1. berechne k und wiederhole $2k$-mal das Folgende
2. simuliere M_1 auf w
3. falls M_1 häufiger akzeptiert, akzeptiere, sonst lehne ab“

Beweis zum Verstärkungs-Lemma

- z.z. M_2 ist äquivalent zu M_1 aber mit Fehler $\epsilon = 2^{P(n)}$
- oBdA sei $t \geq 9$, auf w irrt M_1 mit Wkt.$\delta \leq \varepsilon < 1/2$
- M_2 irrt auf w wenn mehr als k von $2k$ Läufen von M_1 irren, die Wkt dafür ist:

$$\sum_{k \leq i \leq 2k} Pr(M_1 \text{ irrt genau i-mal in 2k Versuchen})$$

$$= \sum_{k \leq i \leq 2k} \binom{2k}{i} \delta^i (1-\delta)^{2k-i}$$

Beweis zum Verstärkungs-Lemma

- z.z. M_2 ist äquivalent zu M_1 aber mit Fehler $\epsilon = 2^{P(n)}$
- oBdA sei $t \geq 9$, auf w irrt M_1 mit Wkt.$\delta \leq \varepsilon < 1/2$
- M_2 irrt auf w wenn mehr als k von $2k$ Läufen von M_1 irren, die Wkt dafür ist:

$$\sum_{k \leq i \leq 2k} Pr(M_1 \text{ irrt genau i-mal in 2k Versuchen})$$

$$= \sum_{k \leq i \leq 2k} \binom{2k}{i} \delta^i (1-\delta)^{2k-i}$$

$$\leq (k+1) \binom{2k}{k} \delta^k (1-\delta)^k$$

Beweis zum Verstärkungs-Lemma

- z.z. M_2 ist äquivalent zu M_1 aber mit Fehler $\epsilon = 2^{P(n)}$
- oBdA sei $t \geq 9$, auf w irrt M_1 mit Wkt.$\delta \leq \varepsilon < 1/2$
- M_2 irrt auf w wenn mehr als k von $2k$ Läufen von M_1 irren, die Wkt dafür ist:

$$\sum_{k \leq i \leq 2k} Pr(M_1 \text{ irrt genau i-mal in 2k Versuchen})$$

$$= \sum_{k \leq i \leq 2k} \binom{2k}{i} \delta^i (1-\delta)^{2k-i}$$

$$\leq (k+1) \binom{2k}{k} \delta^k (1-\delta)^k$$

$$\leq (k+1) 2^{2k} \delta^k (1-\delta)^k$$

Beweis zum Verstärkungs-Lemma

- z.z. M_2 ist äquivalent zu M_1 aber mit Fehler $\epsilon = 2^{P(n)}$
- oBdA sei $t \geq 9$, auf w irrt M_1 mit Wkt.$\delta \leq \varepsilon < 1/2$
- M_2 irrt auf w wenn mehr als k von $2k$ Läufen von M_1 irren, die Wkt dafür ist:

$$\sum_{k \leq i \leq 2k} Pr(M_1 \text{ irrt genau i-mal in 2k Versuchen})$$

$$= \sum_{k \leq i \leq 2k} \binom{2k}{i} \delta^i (1-\delta)^{2k-i}$$

$$\leq (k+1) \binom{2k}{k} \delta^k (1-\delta)^k$$

$$\leq (k+1) 2^{2k} \delta^k (1-\delta)^k$$

$$\leq (k+1)(4\delta(1-\delta))^k$$

Beweis zum Verstärkungs-Lemma II

Wkt., dass M_1 in mehr als k von 2k Läufen irrt ist kleiner als

$$(k+1)(4\delta(1-\delta))^k$$

- wegen $\delta \leq \varepsilon < 1/2$ gilt $\delta(1-\delta) \leq \varepsilon(1-\varepsilon)$ und damit:

$$(k+1)(4\delta(1-\delta))^k \leq (k+1)(4\varepsilon(1-\varepsilon))^k$$

Beweis zum Verstärkungs-Lemma II

Wkt., dass M_1 in mehr als k von 2k Läufen irrt ist kleiner als

$$(k+1)(4\delta(1-\delta))^k$$

- wegen $\delta \leq \varepsilon < 1/2$ gilt $\delta(1-\delta) \leq \varepsilon(1-\varepsilon)$ und damit:

$$(k+1)(4\delta(1-\delta))^k \leq (k+1)(4\varepsilon(1-\varepsilon))^k$$

- noch zu zeigen:

$$(k+1)(4\varepsilon(1-\varepsilon))^k \leq 2^{-P(n)}$$

Beweis zum Verstärkungs-Lemma II

Wkt., dass M_1 in mehr als k von 2k Läufen irrt ist kleiner als

$$(k+1)(4\delta(1-\delta))^k$$

- wegen $\delta \leq \varepsilon < 1/2$ gilt $\delta(1-\delta) \leq \varepsilon(1-\varepsilon)$ und damit:

$$(k+1)(4\delta(1-\delta))^k \leq (k+1)(4\varepsilon(1-\varepsilon))^k$$

- noch zu zeigen:

$$(k+1)(4\varepsilon(1-\varepsilon))^k \leq 2^{-P(n)}$$

- äquivalent zu

$$(k+1)(1/a)^k \leq 1/t \text{ genau dann wenn } a^k \geq (k+1)t$$

Beweis zum Verstärkungs-Lemma III

- zu zeigen ist noch
$$a^k \geq (k+1)t$$
- es gilt $a^k = a^{\lceil bc \rceil}$ und folgende Fallunterscheidung:
  1. falls $1/\log a \geq 1$ dann ist $b = 1/\log a$ und $a^{bc} = 2^c$
  2. falls $1/\log a < 1$ dann ist $b = 1$ und $a > 2$ und $a^{bc} = a^c > 2^c$

  und daher
$$a = a^{\lceil bc \rceil} \geq 2^c = 2^{2\log(bt)} = (bt)^2$$
$$(bt)^2 > bt(2 + 2\log(bt)) \geq t(2b + \lceil 2b \log(bt) \rceil)$$
$$t(2 + \lceil 2b \log(bt) \rceil) \geq t(1 + \lceil bc \rceil) = (k+1)t$$

□

Test auf Primzahl

- Primzahlen sind nur durch Eins und sich selbst teilbar
- naiver Primzahltest probiert alle möglichen Faktoren aus, wegen Dezimaldarstellung exponentiell in der Länge der Eingabe
- probabilistischer Test basiert nicht auf dem Finden von Faktoren

Zahlentheorie

- für jedes (ganzzahlige) $p > 1$ sei $\mathbb{Z}_p^+ = \{1, \ldots, p-1\}$ und $\mathbb{Z}_p = \mathbb{Z}_p{}^+ \cup \{0\}$ Menge der möglichen Reste bei Division durch p
- $x \mod p$ ist das kleinste nichtnegative y mit $x \equiv y \mod p$
- p und q sind relativ prim zueinander wenn $ggT(p, q) = 1$

Theorem ((8.1) Chinesischer Restsatz)

Wenn p und q relativ prim zueinander sind, dann gibt es eine Bijektion

$$f : \mathbb{Z}_{pq} \to \mathbb{Z}_p \times \mathbb{Z}_q$$

mit

$$f : r \mapsto (a, b) \text{ so, daß } \qquad r \equiv a \mod p \qquad r \equiv b \mod q.$$

Pseudoprimzahlen und Fermattest

Theorem ((8.2)Kleiner Satz von Fermat)

Sei p eine Primzahl und $a \in \mathbb{Z}_p^+$, dann gilt $a^{p-1} \equiv 1 \mod p$.

- $2^{7-1} = 2^6 = 64 \equiv 1 \mod 7$, da 7 prim ist, aber $2^{6-1} = 2^5 = 32 \equiv 2 \mod 6$
- Test ob $x^{p-1} \equiv 1 \mod p$ kann für Primzahltest verwendet werden ohne zu faktorisieren,
- Primzahlen bestehen alle möglichen Tests, Carmichael-Zahlen bestehen auch, sind aber relativ selten, Zahlen, die alle Fermattests bestehen heißen Pseudo-Primzahlen.

Probabilistischer Pseudoprimzahltest

PSEUDOPRIME=„Auf Eingabe p:

1. wähle $a_1, \ldots a_k$ zufällig aus $\mathbb{Z}_p^+$
2. berechne $a_i^{p-1} \mod p$ für jedes i
3. falls alle berechneten Werte 1 sind akzeptiere, sonst lehne ab."

- falls p keine Pseudoprimzahl ist, fällt der Test für jedes a_i mit Wkt 0.5 positiv aus, die Wahrscheinlichkeit daß p alle Tests besteht ist höchstens 2^{-k}
- PSEUDOPRIME arbeitet in Polynomzeit, da modulares Potenzieren in P liegt

Primzahltest

- Problem mit den Carmichaelzahlen soll vermieden werden
- Prinzip: 1 hat genau zwei Quadratwurzeln modulo einer Primzahl p, aber genau 4 oder mehr Quadratwurzeln modulo der meisten zusammengesetzten Zahlen
- falls Pseudoprime positiv, suche eine der Quadratwurzeln von 1 und stelle fest, ob sie 1 oder -1 ist, lehne ab, wenn nicht
- falls Fermattest für a positiv ist, also $a^{p-1} \equiv 1 \mod p$ dann ist $a^{((p-1)/2} \mod p$ eine Wurzel von 1
- falls der Wert wieder eine Eins ist, fahren wir mit der Halbierung des Exponenten fort bis eine von 1 und (-1) verschiedene Zahl entsteht.

Der Primzahltest PRIME mit Parameter k

PRIME=„Auf Eingabe p:

1. falls p gerade ist, akzeptiere falls $p = 2$, lehne ab sonst.
2. wähle $a_1, \ldots a_k$ zufällig aus $\mathbb{Z}_p^+$
3. für jedes i von 1 bis k
   4. berechne $a^{p-1} \mod p$ und lehne ab falls nicht 1
   5. zerlege $p - 1 = st$ für ungerades s und Zweierpotenz $t = 2^h$
   6. berechne die Folge $a_i^{s \cdot 2^0}, a_i^{s \cdot 2^1}, \ldots, a_i^{s \cdot 2^h} \mod p$
   7. falls eins der Folgenglieder nicht 1 ist, finde das letzte Element, daß nicht 1 ist und lehne ab wenn es nicht -1 ist.
4. falls alle Tests erfolgreich waren akzeptiere“

a_i heißt Zeuge für Zusammengesetztheit von p falls bei 4-7 abgelehnt wird.

Korrektheit von PRIME

Theorem

Für ungerade Zahlen p gilt:

1. *Falls p prim ist, dann Pr[Prime accepts p] = 1.*
2. *Falls p zusammengesetzt dann Pr[Prime accepts p] $\leq 2^{-k}$.*

Beweis

- falls p prim ist, existiert keine Zeuge a
- bricht PRIME in Nr. 4 ab, dann wäre $a^{p-1} \mod p \neq 1$, also p nicht prim
- bricht PRIME in Nr. 7 ab, dann existiert ein b mit $b \neq \pm 1 \mod p$ und $b^2 \equiv 1 \mod p$
- also $b^2 - 1 \equiv 0 \mod p$ und wegen $b^2 - 1 \equiv (b-1)(b+1) \mod p$ kann p nicht prim sein.

Korrektheitsbeweis für PRIME

Für p nicht prim: wenn a zufällig aus $\mathbb{Z}_p^+$ gewählt wird, dann gilt $Pr[a \text{ ist ein Zeuge}] \geq 1/2$

- z.z es existieren genauso viele Zeugen wie Nichtzeugen in $\mathbb{Z}_p^+$, (durch Konstruktion einer entsprechenden Zuordnung)
- für jeden Nichtzeugen besteht die in Nr.6 berechnete Folge nur aus 1 oder (-1) an einer Stelle gefolgt von Einsen
- man finde unter allen Nichtzeugen z denjenigen bei dem -1 an der höchsten Stelle j ($z^{s \cdot 2^j} \equiv -1 \mod p$) auftaucht
- nach Vor.: $p = qr$ mit q, r relativ prim, d.h. es existiert t mit

$$t \equiv z \mod q \text{ und } t \equiv 1 \mod r \text{ also}$$

- also ist t ein Zeuge, da Quadratwurzel von p und $t \neq \pm 1 \mod p$

Korrektheitsbeweis für PRIME

- n.z.z. dt ist eindeutig bestimmter Zeuge für jeden Nichtzeugen d
  1. wegen Wahl von j gilt $d^{s\cdot 2^j} \equiv \pm 1 \mod p$ und $d^{s\cdot 2^{j+1}} \equiv 1 \mod p$, also ist $dt \mod p$ ein Zeuge wegen $(dt)^{s\cdot 2^j} \neq \pm 1 \mod p$ und $(dt)^{s\cdot 2^{j+1}} \equiv 1 \mod p$
  2. Aus $t^{s\cdot 2^{j+1}} \mod p = t \cdot t^{s\cdot 2^{j+1}-1} = 1 \mod p$ folgt wenn $d_1 t \mod p = d_2 t \mod p$ dann

$$d_1 = t \cdot t^{s\cdot 2^{j+1}-1} d_1 \mod p = t \cdot t^{s\cdot 2^{j+1}-1} d_2 \mod p = d_2.$$

- also ist die Zahl der Zeugen mindestens genauso groß wie die der Nichtzeugen.

□

Algorithmen mit einseitigem Fehler

Theorem
PRIMES $\in$ *BBP*

- wenn PRIMES ablehnt, war p zusammengesetzt, wenn PRIMES akzeptiert, ist p höchstwahrscheinlich prim
- unkorrekte Antwort kann es nur für zusammengesetzte Zahlen geben, einseitiger Fehler

Definition (8.1)

Die Klasse der prob. Polynomzeitalgorithmen A für Sprachen L mit $Prob[A \text{ akzeptiert } w] > 1/2$ für $w \in L$ und $Prob[A \text{ lehnt } w \text{ ab}] = 1$ für $w \notin L$ heißt **RP**.

Theorem
COMPOSITES $\in$ *RP*

Branching Programme

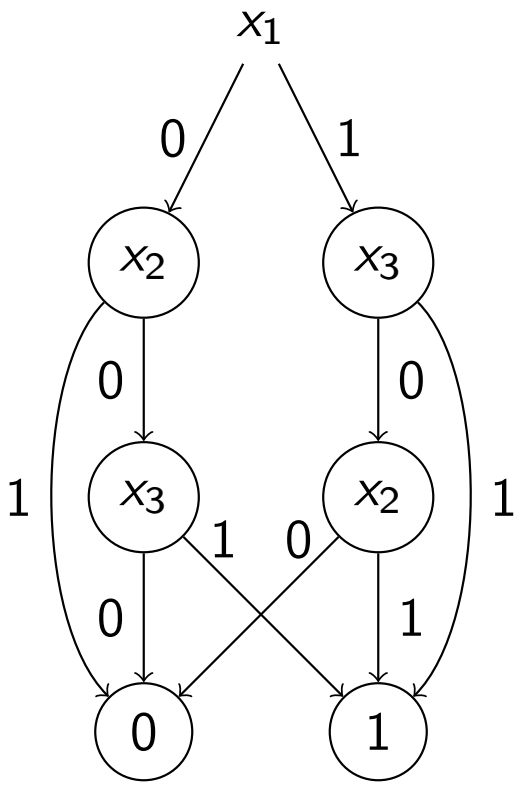
- Berechnungsmodell zur Darstellung von Entscheidungsprozessen auf Grundlage der Eingabewerte
- Darstellung durch Graphen, wobei die Knoten den Entscheidungen entsprechen
- wie testet man die Äquivalenz zweier Branching-Programme? i.A. co-NP-vollständig
- unter bestimmten Bedingungen gibt es probabilistische Polynomzeit-Algorithmen

Definition

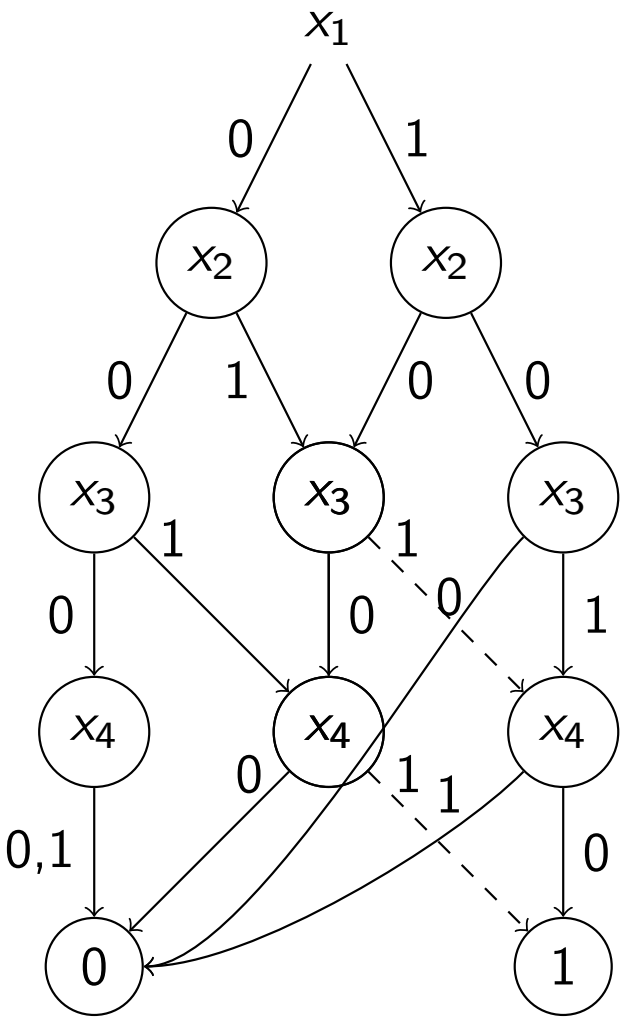
Definition: Ein Branching Programm ist ein gerichteter azyklischer Graph, wobei alle Knoten bis auf zwei Ausgabeknoten mit Variablen markiert sind. Markierte Knoten heißen Frageknoten.

- jeder Knoten hat zwei ausgehende Kanten, je eine mit 0 und 1 markiert
- Ausgabeknoten haben keine ausgehenden Kanten, ein Knoten ist Startknoten
- Branching Programme bestimmen Boole'sche Funktionen

Beispiele



Beispiele II



Eigenschaften

- die der Branching Programme entspricht der Klasse der Sprachen die mit logarithmisch viel Platz entscheidbar sind (SPACE=log(n)), jede Log-Space-Sprache über $\{0, 1\}$ kann mit einem Branching Programm mit polynomiell vielen Knoten entschieden werden
- Branching Programme sind äquivalent, wenn sie dieselbe Funktion berechnen,
- Testen ob zwei Branching Programme äquivalent sind liegt in co-NP
- für spezielle Fälle ist es einfacher

Read-Once-Branching-Programme

Jede Variable wird auf jedem gerichteten Pfad vom Start-zum Ausgabeknoten nur einmal „gefragt“

Theorem

$$EQ_{ROBP} = \{\langle B_1, B_2\rangle \mid B_1 \text{ und } B_2 \text{ sind äquivalente Branching-Programme}\}$$

$$EQ_{ROBP} \in BPP.$$

Beweisidee für $EQ_{ROBP} \in BPP$

1. Versuch
   - versuche Variablen $x_1, \ldots x_n$ in den Programmen zufällig zu belegen, und die Programme damit auszuwerten
   - wir akzeptieren wenn B_1 und B_2 auf dieser Belegung übereinstimmen, sonst lehnen wir ab
   - falls sich die Programme nur bei einer von 2^m möglichen Belegungen unterscheiden ist die Wkt, daß diese geprüft wird klein, also die Irrtumswahrscheinlichkeit groß
2. Versuch
   - wählen zufällig nicht Boole'sche Belegungen für die Variablen und bewerten B_1 und B_2 in einer passend definierten Art
   - man zeigt daß falls B_1 und B_2 nicht äquivalent sind, dann sind die Zufallsbewertungen wahrscheinlich ungleich

Beweis für $EQ_{ROBP} \in BPP$

- den Knoten und Kanten des Programms werden Polynome über $x_0 1 \ldots x_n$ zugeordnet:
  1. dem Startknoten das konstante Polynom 1,
  2. falls ein mit x markierter Knoten deas Polynom P zugeordnet bekommt, dann ordne der 1-Kante das polynom xP und der 0-Kante das polynom $(1 - x)P$ zu,
  3. jedem Knoten wird die Summe der Polynome der ankommenden Kanten zugeordnet,
  4. das Polynom des 1-Knotens wird dem ganzen Programm zugeordnet

Beweis für $EQ_{ROBP} \in BPP$ II

Sei K ein Körper mit wenigstens $3m$ Elementen $D =$„Auf Eingabe von $\langle B_1, B_2 \rangle$

1. wähle zufällig Elemente $a_1, \ldots a_m$ aus K
2. werte die Polynome p_1 und p_2 an den Stellen $a_1, \ldots a_m$ aus
3. falls $p_1(a_1, \ldots a_m) = p_2(a_1, \ldots a_m)$ akzeptiere sonst lehne ab.“

Der Algorithmus entscheidet EQ_{ROBP} mit Fehlerwahrscheinlichkeit von höchstens $1/3$.

Offene Fragen

1. Ist $RP = coRP$?
2. Gilt $RP \subseteq NP \cap coNP$?
   (Da $co - RP \subseteq co - NP$, würde $RP = co - RP$ implizieren daß $RP \subseteq NP \cap co - NP$.)
3. Gilt $BPP \subseteq NP$.