

# Randomisierte Algorithmen

## 5. Zwei spieltheoretische Aspekte

Thomas Worsch

Fakultät für Informatik  
Karlsruher Institut für Technologie

Wintersemester 2019/2020

# Überblick

Und-Oder-Bäume und ihre deterministische Auswertung

Analyse eines randomisierten Algorithmus für die Auswertung von UOB

Zwei-Personen-Nullsummen-Spiele

Untere Schranken für randomisierte Algorithmen

# Überblick

Und-Oder-Bäume und ihre deterministische Auswertung

Analyse eines randomisierten Algorithmus für die Auswertung von UOB

Zwei-Personen-Nullsummen-Spiele

Untere Schranken für randomisierte Algorithmen

## 5.1 Definition (Und-Oder-Baum)

$T_k$  vollständiger binärer Baum der Höhe  $2k$

- ▶ innere Knoten abwechselnd mit “ $\wedge$ ” und “ $\vee$ ” markiert
- ▶ Wurzel von  $T_1$  ist  $\wedge$ -Knoten mit zwei  $\vee$ -Knoten als Nachfolger
- ▶ Jeder dieser Knoten hat zwei Blätter als Nachfolger.
- ▶  $T_k$  entsteht aus  $T_1$ , indem die Blätter durch Kopien von  $T_{k-1}$  ersetzt werden
  
- ▶  $T_k$  hat  $n = 4^k$  Blätter
- ▶ im folgenden  $x_1, \dots, x_{4^k}$  genannt

## Und-Oder-Bäume

- ▶ Festlegung boolescher Werte an allen Blättern eines UOB impliziert für alle inneren Knoten einschließlich Wurzel einen Wert
- ▶ Berechnung des Wurzelwertes „bottom up“ durch
  - ▶ Besuch aller  $n = 4^k$  Blätter und
  - ▶ Berechnung der Werte aller inneren Knoten möglich
- ▶ **Frage:** Geht es besser?

## 5.4 Satz

Für jedes  $k \geq 1$  und jeden deterministischen Algorithmus  $A$  gilt:

- ▶ Es gibt eine Folge  $x_1, \dots, x_{4^k}$  von Bits so, dass  $A$  bei der Auswertung von  $T_k$  mit den  $x_i$  als Blattwerten *alle*  $n = 4^k$  Blätter besucht.

## 5.4 Satz

Für jedes  $k \geq 1$  und jeden deterministischen Algorithmus  $A$  gilt:

- ▶ Es gibt eine Folge  $x_1, \dots, x_{4^k}$  von Bits so, dass  $A$  bei der Auswertung von  $T_k$  mit den  $x_i$  als Blattwerten *alle*  $n = 4^k$  Blätter besucht.
- ▶ Wert der Wurzel = Wert des zuletzt besuchten Blattes
- ▶ sowohl Wurzelwert = 0 als auch Wurzelwert = 1 kann erzwungen werden

## 5.5 Beweis

### Induktion über $k$

▶ **Induktionsanfang  $k = 1$ :**

- ▶  $A$  muss mindestens ein Blatt besuchen, o. B. d. A.  $x_1$ .
- ▶ Setze  $x_1 = 0$ . Damit ist kein innerer Wert festgelegt.
- ▶  $A$  muss ein weiteres Blatt besuchen.
- ▶ O. B. d. A. zwei Möglichkeiten:
  1. Zweites besuchtes Blatt ist  $x_2$ . Setze  $x_2 = 1$ .  
Damit nur Wert des  $\vee$ -Knotens klar, Wurzelwert nicht.  
 $A$  muss weiteres Blatt besuchen, o. B. d. A.  $x_3$ .  
Setze  $x_3 = 0$ .  $A$  muss  $x_4$  besuchen  $\leadsto$  Wurzelwert.
  2. Zweites besuchtes Blatt ist  $x_3$ . Setze  $x_3 = 0$ .  
Kein innerer Wert festgelegt  
 $A$  muss weiteres Blatt besuchen, o. B. d. A.  $x_2$ .  
Setze  $x_2 = 1$ .  $A$  muss  $x_4$  besuchen  $\leadsto$  Wurzelwert



## 5.5 Beweis (2)

- ▶ **Induktionsschritt  $k - 1 \rightsquigarrow k$ :**
  - ▶ Fasse  $T_k$  als  $T_1$ -Baum auf, dessen Blätter durch  $T_{k-1}$ -Bäume ersetzt sind.
  - ▶ Bezeichne die „Blätter“ von  $T_1$  mit  $y_1, \dots, y_4$ .
  - ▶ Analog Induktionsanfang kann durch geeignete Wahl der  $y_i$  erzwungen werden, dass  $A$  alle 4 Werte ermitteln muss.
  - ▶ Induktionsvoraussetzung: für jeden  $T_{k-1}$ -Baum gibt es Blattwerte, die gewünschtes  $y_i$  liefern und erzwingen, dass  $A$  alle darunter liegenden Blätter besuchen muss.
  - ▶ Also muss  $A$  in diesem Fall alle Blätter überhaupt besuchen.

# Überblick

Und-Oder-Bäume und ihre deterministische Auswertung

Analyse eines randomisierten Algorithmus für die Auswertung von UOB

Zwei-Personen-Nullsummen-Spiele

Untere Schranken für randomisierte Algorithmen

## 5.6 Algorithmus: randomisierte UOB-Auswertung

## 5.6 Algorithmus: randomisierte UOB-Auswertung

```

proc AndNodeEval(T)
if IsLeaf(T) then
    return value(T) fi
⟨andernfalls:⟩
T' ← ⟨zufälliger T-U.baum⟩
r ← OrNodeEval(T')
if r = 0 then
    return 0
else
    T'' ← ⟨and. T-U.baum⟩
    return OrNodeEval(T'')
fi

```

```

proc OrNodeEval(T)
T' ← ⟨zufälliger T-U.baum⟩
r ← AndNodeEval(T')
if r = 1 then
    return 1
else
    T'' ← ⟨and. T-U.baum⟩
    return AndNodeEval(T'')
fi

```

*AndNodeEval*(*root*)

## 5.7 Satz

Der Erwartungswert für die Anzahl der von Algorithmus 5.6 besuchten Blätter ist für jede Folge  $x_1, \dots, x_{4^k}$  höchstens  $3^k = n^{\log_4 3} \approx n^{0.792\dots}$ .

## 5.8 Beweis

### Induktion

- ▶  $E$ : ein Erwartungswert für die Anzahl besuchter Blätter
- ▶ **Induktionsanfang  $k = 1$** : Überprüfen aller 16 möglichen Kombinationen  $x_1, \dots, x_4$ . Z. B. 0100 (Rest analog):
  1. Falls erst linker Teilbaum:  
gleich wahrscheinlich wird erst und nur die 1 oder erst die 0 und dann die 1 besucht,  
anschließend im rechten Teilbaum beide Blätter.  
 $E = 1/2 \cdot 1 + 1/2 \cdot 2 + 2 = 7/2$ .
  2. Falls erst rechter Teilbaum; nach Besuch beider Blätter klar: der  $T_1$ -Baum den liefert Wert 0.  
 $E = 2$ .

Beide Fälle gleich wahrscheinlich, also insgesamt

$$E = 1/2 \cdot 7/2 + 1/2 \cdot 2 = 11/4 < 3 (= 3^1 = 3^k).$$

## 5.8 Beweis (2)

### Induktionsschritt $k - 1 \rightsquigarrow k$ :

- ▶ Betrachte zunächst  $\vee$ -Knoten mit zwei  $T_{k-1}$ -Bäumen darunter.  
Zwei Fälle:

#### O1. $\vee$ -Knoten wird 1 liefern:

- ▶ Dann muss mindestens ein  $T_{k-1}$ -Baum dies auch tun.
- ▶ Mit Wahrscheinlichkeit  $p \geq 1/2$  wird ein Unterbaum untersucht, der 1 liefert.
- ▶ Mit Wahrscheinlichkeit  $1 - p \leq 1/2$  werden beide Unterbäume untersucht.
- ▶  $E \leq p \cdot 3^{k-1} + (1 - p) \cdot 2 \cdot 3^{k-1} = (2 - p) \cdot 3^{k-1} \leq 3/2 \cdot 3^{k-1}$ .

#### O2. Der $\vee$ -Knoten wird 0 liefern:

- ▶ Dann müssen beide  $T_{k-1}$ -Bäume dies auch tun.
- ▶ Nach Induktionsvoraussetzung  $E \leq 2 \cdot 3^{k-1}$  ist.

## 5.8 Beweis (3)

### Induktionsschritt $k - 1 \rightsquigarrow k$ : Teil 2a

- ▶ Betrachte Wurzel des  $T_k$ -Baumes mit zwei eben untersuchten Bäumen darunter. Zwei Fälle:
  - U1. Der  $\wedge$ -Knoten wird 0 liefern:
    - ▶ Dann muss mindestens ein U.baum dies auch tun.
    - ▶ Mit Wahrscheinlichkeit  $p \geq 1/2$  wird ein Unterbaum untersucht, der 0 liefert.
    - ▶ Mit Wahrscheinlichkeit  $1 - p \leq 1/2$  werden beide Unterbäume untersucht.
    - ▶ Gemäß O1 und O2 ist folglich

$$\begin{aligned} E &\leq p \cdot 2 \cdot 3^{k-1} + (1 - p) \cdot (3/2 \cdot 3^{k-1} + 2 \cdot 3^{k-1}) \\ &= 7/2 \cdot 3^{k-1} - p \cdot 3/2 \cdot 3^{k-1} \\ &\leq 11/4 \cdot 3^{k-1} \\ &\leq 3^k \end{aligned}$$



## 5.8 Beweis (4)

### Induktionsschritt $k - 1 \rightsquigarrow k$ : Teil 2b

- ▶ Betrachte Wurzel des  $T_k$ -Baumes mit zwei eben untersuchten Bäumen darunter. Fall:
  - U2. Der  $\wedge$ -Knoten wird 1 liefern:
    - ▶ Dann müssen beide Unterbäume dies auch tun.
    - ▶ Gemäß O1 ist daher  $E \leq 2 \cdot 3/2 \cdot 3^{k-1} \leq 3^k$ .

# Überblick

Und-Oder-Bäume und ihre deterministische Auswertung

Analyse eines randomisierten Algorithmus für die Auswertung von UOB

Zwei-Personen-Nullsummen-Spiele

Untere Schranken für randomisierte Algorithmen

## 5.9 Spiele

- ▶  $n \geq 2$  *Spieler*
- ▶ Spieler  $i$  hat (endliche) Menge  $S_i$  *reiner Strategien*  $s_j^i$  zur Auswahl
- ▶  $u_i : S_1 \times \cdots \times S_n \rightarrow \mathbb{R}$  gibt für jede Kombination von Strategien den *Nutzen* oder *Gewinn* von Spieler  $i$  an.

## 5.10 Zwei-Personen-Nullsummen-Spiele

- ▶  $n = 2$  Spieler
- ▶ für die Nutzenfunktionen gilt:  $u_1 = -u_2$ .
- ▶ Es genügt Matrix  $\mathbf{M}$  mit
  - ▶  $|S_1|$  Zeilen
  - ▶  $|S_2|$  Spalten
  - ▶  $M_{ij} = u_1(s_i^1, s_j^2)$
- ▶ *Zeilenspieler* und *Spaltenspieler*
- ▶ Einheitsvektor  $\mathbf{e}_i$  (passender Länge) entspricht reiner Strategie  $i$
- ▶  $u_1(s_i^1, s_j^2) = \mathbf{e}_i^T \mathbf{M} \mathbf{e}_j$

## 5.11 Gemischte Strategien

- ▶ *gemischte Strategie*: Wahrscheinlichkeitsverteilung  $\mathbf{p}$  auf der Menge der reinen Strategien eines Spielers.
- ▶ Sind  $\mathbf{p}$  und  $\mathbf{q}$  gemischte Strategien für Zeilen- und Spaltenspieler, dann ist

$$\mathbf{p}^T \mathbf{M} \mathbf{q}$$

der zu erwartende Gewinn für den Zeilenspieler.

## 5.12 Satz (von Neumann, 1928)

Für Zwei-Personen-Nullsummen-Spiele mit Matrix  $M$  gilt:

$$\max_p \min_q \mathbf{p}^T M \mathbf{q} = \min_q \max_p \mathbf{p}^T M \mathbf{q}$$

und es gibt Verteilungen  $\mathbf{p}^*$  und  $\mathbf{q}^*$ , für die  $\mathbf{p}^{*T} M \mathbf{q}^*$  dieser Wert ist.

## 5.13 Korollar (Loomis, 1946)

Für Zwei-Personen-Nullsummen-Spiele mit Matrix  $\mathbf{M}$  gilt:

$$\max_{\mathbf{p}} \min_j \mathbf{p}^T \mathbf{M} \mathbf{e}_j = \min_{\mathbf{q}} \max_i \mathbf{e}_i^T \mathbf{M} \mathbf{q}$$

## 5.14 Beweis

- ▶ genügt zu zeigen:

- ▶ für jedes  $\mathbf{p}$  gilt:

$$\min_{\mathbf{q}} \mathbf{p}^T \mathbf{M} \mathbf{q} = \min_j \mathbf{p}^T \mathbf{M} \mathbf{e}_j$$

zeige: „ $\geq$ “ („ $\leq$ “ ist trivial)

- ▶ rechte Seiten der Gleichungen aus Satz 5.12 und Korollar 5.13 analog

- ▶ Für beliebiges  $\mathbf{p}$  ist  $\mathbf{p}^T \mathbf{M}$  ein Zeilenvektor  $\mathbf{v}^T$ .

Es sei  $v_{\min} = \min\{v_i\}$ .

für ein  $j$  also  $v_{\min} = \mathbf{v}^T \mathbf{e}_j$

- ▶ dann für jedes  $\mathbf{q}$ :  $\mathbf{v}^T \mathbf{q} = \sum v_i q_i \geq \sum v_{\min} q_i = v_{\min} = \mathbf{v}^T \mathbf{e}_j$



## 5.15 Korollar

Loomis:

$$\max_{\mathbf{p}} \min_j \mathbf{p}^T \mathbf{M} \mathbf{e}_j = \min_{\mathbf{q}} \max_i \mathbf{e}_i^T \mathbf{M} \mathbf{q}$$

## 5.15 Korollar

Loomis:

$$\max_{\mathbf{p}} \min_j \mathbf{p}^T \mathbf{M} \mathbf{e}_j = \min_{\mathbf{q}} \max_i \mathbf{e}_i^T \mathbf{M} \mathbf{q}$$

Für alle Verteilungen  $\mathbf{p}$  und  $\mathbf{q}$  gilt:

$$\min_j \mathbf{p}^T \mathbf{M} \mathbf{e}_j \leq \max_i \mathbf{e}_i^T \mathbf{M} \mathbf{q}$$

# Überblick

Und-Oder-Bäume und ihre deterministische Auswertung

Analyse eines randomisierten Algorithmus für die Auswertung von UOB

Zwei-Personen-Nullsummen-Spiele

Untere Schranken für randomisierte Algorithmen

## 5.16 Algorithmenentwurf als „Spiel“ (1)

- ▶ 2 Spieler
  - ▶ Zeilenspieler: ein „böser“ Widersacher
  - ▶ Spaltenspieler: ein Algorithmenentwerfer
- ▶ reine Strategien
  - ▶ Widersacher: „gemeine“ Eingaben
  - ▶ Algorithmenentwerfer: deterministische Algorithmen
- ▶ Einträge von  $M$ : Gewinn  $C(I, A)$  des Widersachers
  - ▶ z.B.  $C(I, A)$  = Laufzeit von Algorithmus  $A$  für Eingabe  $I$
  - ▶ oder der Verbrauch irgendeiner anderen Ressource
- ▶ Ziele
  - ▶ Widersacher:  $C(I, A)$  maximieren
  - ▶ Algorithmenentwerfer:  $C(I, A)$  minimieren

## 5.16 Algorithmenentwurf als „Spiel“ (2)

- ▶ reine Strategien
  - ▶ Widersacher: „gemeine“ Eingaben
  - ▶ Algorithmenentwerfer: deterministische Algorithmen
- ▶ gemischte Strategien

## 5.16 Algorithmenentwurf als „Spiel“ (2)

- ▶ reine Strategien
  - ▶ Widersacher: „gemeine“ Eingaben
  - ▶ Algorithmenentwerfer: deterministische Algorithmen
- ▶ gemischte Strategien
  - ▶ Widersacher: Wahrscheinlichkeitsverteilungen auf einer Menge der Eingaben
  - ▶ Algorithmenentwerfer: Wahrscheinlichkeitsverteilungen auf einer Menge deterministischer Algorithmen

## 5.16 Algorithmenentwurf als „Spiel“ (2)

- ▶ reine Strategien
  - ▶ Widersacher: „gemeine“ Eingaben
  - ▶ Algorithmenentwerfer: deterministische Algorithmen
- ▶ gemischte Strategien
  - ▶ Widersacher: Wahrscheinlichkeitsverteilungen auf einer Menge der Eingaben
  - ▶ Algorithmenentwerfer: **randomisierte Algorithmen**

## 5.17 Satz (Minimax-Methode von Yao)

- ▶ endliche Menge  $\mathcal{I}$  von Eingaben gleicher Größe  $n$  und endliche Menge  $\mathcal{A}$  von Algorithmen für ein Problem
- ▶  $C(I, A)$  Laufzeit von  $A \in \mathcal{A}$  für  $I \in \mathcal{I}$ .
- ▶  $\mathbf{p}$  und  $\mathbf{q}$ : Wahrscheinlichkeitsverteilungen auf  $\mathcal{I}$  bzw.  $\mathcal{A}$ .
- ▶ Mit  $I_p$  bzw.  $A_q$  werde ein gemäß der Verteilung  $\mathbf{p}$  bzw.  $\mathbf{q}$  aus  $\mathcal{I}$  bzw.  $\mathcal{A}$  gewählte Eingabe bzw. Algorithmus bezeichnet.
- ▶ Dann gilt für alle  $\mathbf{p}$  und  $\mathbf{q}$ :

$$\min_{A \in \mathcal{A}} \mathbf{E} [C(I_p, A)] \leq \max_{I \in \mathcal{I}} \mathbf{E} [C(I, A_q)]$$



## 5.18 Erläuterungen

$$\min_{A \in \mathcal{A}} \mathbf{E} [C(I_p, A)] \leq \max_{I \in \mathcal{I}} \mathbf{E} [C(I, A_q)]$$

- ▶ Erwartungswert  $\mathbf{E} [C(I_p, A)]$ 
  - ▶ für gegebenes  $A$  die „erwartete Laufzeit“ bei zufälliger Wahl von  $I_p$  gemäß Verteilung  $\mathbf{p}$ .
  - ▶ Minimum der Erwartungswerte, also für den „besten“ Algorithmus, von Bedeutung
- ▶ Erwartungswert  $\mathbf{E} [C(I, A_q)]$ 
  - ▶ für gegebenes  $I$  die „erwartete Laufzeit“ eines randomisierten Algorithmus.
  - ▶ Maximum der Erwartungswerte, also für schlimmste Eingabe, von Bedeutung
- ▶  $\min_{A \in \mathcal{A}} \mathbf{E} [C(I_p, A)]$  ist eine untere Schranke für Laufzeit

## 5.19 Bemerkung

Einen Satz analog zu 5.17 kann man auch für Monte-Carlo-Algorithmen beweisen. Hierauf gehen wir nicht weiter ein.

## 5.20 – 5.21

▶  $\bar{\vee}$ -Funktion:

$x$	$y$	$x\bar{\vee}y$
0	0	1
0	1	0
1	0	0
1	1	0

- ▶ Und-Oder-Baum kann auch als Baum mit lauter inneren  $\bar{\vee}$ -Knoten aufgefaßt werden:

$$\begin{aligned} \overline{\overline{(x_1 \vee x_2) \wedge (x_3 \vee x_4)}}} &= \overline{\overline{(x_1 \vee x_2) \vee (x_3 \vee x_4)}}} \\ &= (x_1 \bar{\vee} x_2) \bar{\vee} (x_3 \bar{\vee} x_4) \end{aligned}$$

- ▶ Für  $p = \frac{3-\sqrt{5}}{2} \approx 0.381966\dots$  gilt:  $(1-p)^2 = p$ .
- ▶ Wenn an jedem Eingang eines  $\bar{\vee}$ -Gatters unabhängig mit Wahrscheinlichkeit  $p$  eine 1 vorliegt, ist daher mit gleicher Wahrscheinlichkeit  $p$  auch die Ausgabe eine 1.

## 5.22 Satz

- ▶  $T$ : vollständiger balancierter Baum aus  $\bar{v}$ -Knoten, Blätter haben unabhängig voneinander mit Wahrscheinlichkeit  $p = (3 - \sqrt{5})/2$  den Wert 1
- ▶  $W(T)$  sei das Minimum (über alle deterministischen Algorithmen) der erwarteten Anzahl von Schritten zur Auswertung von  $T$ .
- ▶ Dann gibt es auch einen Algorithmus  $A$ , der eine erwartete Anzahl von nur  $W(T)$  Schritten macht und außerdem die folgende Eigenschaft hat:
  - ▶ Besucht  $A$  ein Blatt  $v'$ , das zu einem Teilbaum  $T'$  gehört und später ein Blatt  $u$ , das *nicht* zu  $T'$  gehört,
  - ▶ dann gilt für alle Blätter  $a''$  von  $T'$ , die  $A$  überhaupt besucht:  $A$  besucht  $a''$  vor  $u$ .

## 5.23 Satz

Die erwartete Anzahl der Blätter, die ein randomisierter Algorithmus zur Auswertung von UOB mit  $n$  Blättern besucht, ist mindestens

$$n^{\log_2((1+\sqrt{5})/2)} = n^{0.694\dots}$$

## 5.24 Beweis

- ▶ Betrachte einen Algorithmus wie in Satz 5.22
- ▶ Auswertung von  $\bar{V}$ -Bäumen, deren Blätter unabhängig voneinander mit Wahrscheinlichkeit  $p = \frac{3-\sqrt{5}}{2}$  auf 1 gesetzt sind.
- ▶ In Abhängigkeit von der Höhe  $h$  sei  $W(h)$  die erwartete Anzahl besuchter Blätter.
- ▶ Es ist

$$\begin{aligned} W(h) &= W(h-1) + (1-p)W(h-1) \\ &= (2-p)W(h-1) \end{aligned}$$

$$\text{also } W(h) = (2-p)^{h-1}W(1) = (2-p)^h$$

- ▶ Einsetzen von  $h = \log_2 n$  und  $p$  ergibt

$$W(T) = (2-p)^{\log_2 n} = 2^{(\log_2(2-p))(\log_2 n)} = n^{\log_2(2-p)} = n^{0.694\dots}$$

## 5.25

- ▶ Genauere (und schwierigere) Analyse liefert sogar untere Schranke von  $n^{\log_4 3} \approx n^{0.792\dots}$ .
- ▶ Die durch unseren randomisierten Algorithmus gegebene obere Schranke ist also optimal.

## 5.26

- ▶ Da Erwartungswert für Anzahl besuchter Blätter  $n^{0.792\dots}$
- ▶ gibt es eine Berechnung, die max. so viele Blätter besucht.



## 5.26

- ▶ Da Erwartungswert für Anzahl besuchter Blätter  $n^{0.792\dots}$
- ▶ gibt es eine Berechnung, die max. so viele Blätter besucht.
  
- ▶ also: mit Wahrscheinlichkeit  $> 0$  findet randomisierter Algorithmus Teilmenge von  $\leq n^{0.792\dots}$  Blättern, aus denen schon der Wurzelwert folgt.
- ▶ Also *existiert* für jede Verteilung von Bits auf alle Blätter solch „kleine“ Teilmenge von Blättern, deren Kenntnis für die Bestimmung des Wurzelwertes ausreicht.
- ▶ Hausaufgabe: es reichen sogar immer  $n^{0.5}$

## 5.26

- ▶ Da Erwartungswert für Anzahl besuchter Blätter  $n^{0.792\dots}$
- ▶ gibt es eine Berechnung, die max. so viele Blätter besucht.
  
- ▶ also: mit Wahrscheinlichkeit echt größer 0 findet randomisierter Algorithmus Teilmenge von  $\leq n^{0.792\dots}$  Blättern, aus denen schon der Wurzelwert folgt.
- ▶ Also *existiert* für jede Verteilung von Bits auf alle Blätter solch „kleine“ Teilmenge von Blättern, deren Kenntnis für die Bestimmung des Wurzelwertes ausreicht.
- ▶ Hausaufgabe: es reichen sogar immer  $n^{0.5}$
  
- ▶ Aber: Jeder deterministische Algorithmus muss für manche Eingaben alle Blätter besuchen.
- ▶ Es ist also manchmal „sehr schwierig“, deterministisch eine solche kleine Teilmenge von Blättern zu finden.

## Zusammenfassung

- ▶ Bei der Auswertung von Und-Oder-Bäumen kann man randomisiert weniger Blattbesuche erwarten, als jeder deterministische Algorithmus für manche Bäume durchführen *muss*.
- ▶ Die Minimax-Methode von Yao liefert eine Möglichkeit, untere Schranken für die erwartete Laufzeit randomisierter Algorithmen herzuleiten.