Randomisierte Algorithmen

Köberl Doris

Priewasser Johannes
Dhillon Tajinder

Was sind randomisierte Algorithmen?

- Mögliche Sichtweisen:
  - zufälliger Wert zur Verfügung gestellt
  - nicht eindeutig festgelegt nächster Schritt
  - aus Vielzahl deterministischen Algorithmen ausgewählt

Eigenschaften:

- Fehler
- Verschiedene Berechnungen
- Erwartungswert
- Analyse

Unterscheidung

- Las-Vegas-Algorithmen
  - korrekt
  - Macao-Algorithmen
- Monte-Carlo-Algorithmen
  - p-korrekt
  - Keinen Näherungswert

Fehler

- Abhängig Eingabe Zufallszahl
- Mehrheitliche Ergebnis
  - Allgemeine Idee
    - $p > ½$

Erlaubte Fehler

- Einseitiger Fehler
  - Ein Fehler
  - Bsp.: Miller-Rabin-Test
- Zweiseitiger Fehler
  - Beide erlaubten Fehler < 1/2

Einige Standardmethoden:

- Fingerabdrücke
- Zufälliges Auswählen und Umordnen
- Lastbalancierung
- Isolierung und Symmetrieberechnung

Beispiele zu Monte-Carlo-Verfahren

Randomisierter Identitätstest,

Vergleich von Wörtern

Randomisierter Identitätstest

Vergleich von Objekten:

- „Fingerabdruck“ der Objekte berechnen
- Vergleich des Fingerabdruckes
- Fehlerwahrscheinlichkeit beachten

z.B.: A, B, C sind n x n Matrizen
Es soll geprüft werden, ob A*B = C ist.

Multiplikationsalgorithums von Coppersmith und Winograd löst dieses Problem mit der Komplexität $O(n^{2.376})$

Randomisierter Identitätstest

- Algorithmus von Freivalds: $O(n^2)$

```
r = n unabhängige Zufallsbits;
x = B * r;
y = A * x;
z = C * r;

if (y ≠ z)
  return false;
else
  return true;
```

Randomisierter Identitätstest

Auswertung:

- $A*B = C \rightarrow$ Algorithmus immer richtig
- $A*B \neq C \rightarrow$ Fehlerwahrscheinlichkeit $\leq 50\%$

Verbesserung:

- Zahlenbereich von r auf 2^k erweitern
- k Wiederholungen durchführen
- Fehlerwahrscheinlichkeit beider Methoden 2^{-k}

Vergleich von Wörtern

- 2 Bitfolgen: $a_1,\ldots,a_n$ und $b_1,\ldots,b_n$
- Dargestellt als: $a = \sum_{i=1}^{n} a_i * 2^{i-1}$; $b = \sum_{i=1}^{n} b_i * 2^{i-1}$
- Überprüfen ob a = b

Vergleich von Wörtern

- Algorithmus:

p = Primzahl zufällig gewählt ≤ $n^2 * \ln(n^2)$;
a = Zahlenwert der Bitfolge $a_1,\ldots,a_n$
b = Zahlenwert der Bitfolge $b_1,\ldots,b_n$

```
if (a mod p = b mod p)
        return true;
else
        return false;
```

Beispiele zu Las-Vegas-Verfahren

Und-Oder-Bäume,

randomisierter Quicksort

Und-Oder-Bäume

- Definition:
  Für $k \geq 1$ sei T_k der wie folgt rekursiv definierte Binärbaum der Höhe 2k, dessen innere Knoten abwechselnd mit „Λ“ und „V“ markiert sind.
  - Die Wurzel von T_1 ist ein Λ Knoten und hat 2 V Knoten als Nachfolger, wobei diese 2 Blätter als Nachfolger haben.
  - Für $k \geq 2$ ergibt sich T_k aus T_1, indem man dessen Blätter durch Kopien von T_{k-1} ersetzt.

- Anzahl der Blätter = 4^k

Und-Oder-Bäume

```
proc AndNodeEval (T)
if IsLeaf (T)
    return value(T);
T‘ = zufälliger Unterbaum von T;
OrNodeEval (T‘);
if (r = 0)
    return 0;
else
    T‘‘ = der andere Unterbaum;
return OrNodeEval (T‘‘);
```

```
proc OrNodeEval (T)
T‘ = zufälliger Unterbaum von T;
r = AndNodeEval(T‘);
if (r = 1)
    return 1;
else
    T‘‘ = der andere Unterbaum;
    return AndNodeEval (T‘‘);
```

Randomisierter Quicksort

Komplexität

- QS: O(n)
- Random QS: O(n*log(n))

Quellen

- Unterlagen der Vorlesung randomisierte Algorithmen der Universität Karlsruhe
- Unterlagen der Vorlesung randomisierte Algorithmen der TU Berlin