Algorithmik

Sebastian Iwanowski
FH Wedel

5. Vorlesungswoche

Algorithmik 5

Implementierung von Dictionaries

Ein Dictionary ist eine Datenstruktur für mit einem Schlüssel vergleichbare Elemente,
welche die Funktionen member (key), insert (key, newdata) und delete (key) zur Verfügung stellt

(a,b)-Baum: Verallgemeinerung von 2-3-Bäumen

Datentyp: Suchbaum mit folgenden Eigenschaften:

i. Alle Daten sind in den Blättern, welche die gleiche Tiefe haben.
ii. Alle inneren Knoten außer der Wurzel haben mindestens a und höchstens b Kinder.
Die Wurzel hat mindestens 2 Kinder.
iii. In den inneren Knoten stehen die jeweils größten Schlüssel aller Daten,
die in den Kinderbäumen enthalten sind.

Alle 3 Dictionary-Funktionen	Laufzeit $\Theta(\log n)$ w.c. und a.c.
	Beweis einfach

Referenzen zum Nacharbeiten und Vertiefen:

Skript Alt S. 44 – 51 (Kap. 3.1.5)

Algorithmik 5

Implementierung von Dictionaries

Ein Dictionary ist eine Datenstruktur für mit einem Schlüssel vergleichbare Elemente,
welche die Funktionen member (key), insert (key, newdata) und delete (key) zur Verfügung stellt

AVL-Baum

Datentyp: Binärer Suchbaum mit folgender Eigenschaft:

Für jeden Knoten unterscheidet sich die Höhe der beiden Kindbäume um maximal 1.

Der Erhalt der geforderten Invariante wird durch Rotationen / Doppelrotationen
von Knoten gewährleistet,
welche beim insert bzw. Delete in konstanter Zeit pro Knoten ausgeführt werden.

Alle 3 Dictionary-Funktionen	Laufzeit $\Theta(\log n)$ w.c. und a.c.
	Beweis kompliziert (über Fibonacci-Abschätzung)

Referenzen zum Nacharbeiten und Vertiefen:

Skript Alt S. 41 – 44 (Kap. 3.1.4)	Genaue Funktionsweise und Laufzeitabschätzung darin enthalten, aber nicht prüfungsrelevant

Algorithmik 5

Implementierung von Dictionaries

Ein Dictionary ist eine Datenstruktur für mit einem Schlüssel vergleichbare Elemente,
welche die Funktionen member (key), insert (key, newdata) und delete (key) zur Verfügung stellt

Rot-Schwarz-Baum

Datentyp: Binärer Suchbaum, in dem jeder Knoten rot oder schwarz ist,
mit folgenden Eigenschaften:

i. Jedes Blatt ist schwarz
ii. Rote Knoten haben schwarze Kinder.
iii. Jeder Weg von der Wurzel zum Blatt hat dieselbe Anzahl von schwarzen Knoten.

Der Erhalt der geforderten Invarianten wird durch Rotationen / Doppelrotationen
von Knoten gewährleistet,
welche beim insert bzw. delete in konstanter Zeit pro Knoten ausgeführt werden.

Alle 3 Dictionary-Funktionen Laufzeit $\Theta(\log n)$ w.c. und a.c.

Beweis übersichtlich nachvollziehbar

Referenzen zum Nacharbeiten und Vertiefen:

Skript Alt S. 54 (Kap. 3.1.7)
Cormen (inkl. Laufzeitabschätzung)

Algorithmik 5

Implementierung von Dictionaries

Ein Dictionary ist eine Datenstruktur für mit einem Schlüssel vergleichbare Elemente,
welche die Funktionen member (key), insert (key, newdata) und delete (key) zur Verfügung stellt

Rot-Schwarz-Baum

Satz: Rot-Schwarz-Bäume sind leicht zu (2,4)-Bäumen transformierbar und umgekehrt.

➔: Jeder schwarze Knoten wird mit seinen roten Kindern verschmolzen.
Die Kinder des oder der bisherigen roten Kindes und das eventuell vorhandene schwarze Kind des alten Knotens werden die Kinder des neuen Knotens.

➔:
i. Ein Knoten mit 2 Kindern wird ein schwarzer Knoten mit zwei schwarzen Kindern.
ii. Ein Knoten mit 3 Kindern a_1, a_2, a_3 wird ein schwarzer Knoten mit schwarzen Kind a_1 und einem neuen roten Kind, das a_2, a_3 als schwarze Kinder hat.
iii. Ein Knoten mit 4 Kindern wird ein schwarzer Knoten mit zwei neuen roten Kindern, die jeweils zwei der bisherigen Kinder als schwarze Kinder haben.

Hausaufgabe zum 22.05.: Beweisen Sie, dass die in beiden Richtungen erzeugten Bäume die jeweils behauptete Eigenschaft haben!

Referenzen zum Nacharbeiten und Vertiefen:

Skript Alt S. 54 (Kap. 3.1.7)

Algorithmik 5

Implementierung von Dictionaries

Ein Dictionary ist eine Datenstruktur für mit einem Schlüssel vergleichbare Elemente,
welche die Funktionen member (key), insert (key, newdata) und delete (key) zur Verfügung stellt

Trie-Baum (von Re**trie**val) für Strings über einem k-elementigen Alphabet

Datentyp: k-ärer Suchbaum mit folgenden Eigenschaften:

i. Die Wurzel ist ein leerer Knoten.
ii. Jeder Knoten enthält einen Buchstaben, der in einem String vorkommt.
Für jedes Wort, das diesen Knoten benutzt, gibt es einen Kindknoten
mit dem jeweils nächsten Buchstaben des Strings.
iii. Jeder Knoten hat eine Marke, die anzeigt, ob hier ein Wort endet oder nicht.

Alle 3 Dictionary-Funktionen Laufzeit: Θ(Länge des Strings) w.c. und a.c. (klar)
Laufzeit für Suche in Baum mit n Strings: logk(n) a.c. (Knuth 3, S. 507)
Speicherplatz für n Strings: n/log k a.c. (Knuth 3, S. 507)

Referenzen zum Nacharbeiten und Vertiefen:

Skript Alt S. 54 – 56 (Kap. 3.1.8) mit verschiedenen Verbesserungsmöglichkeiten,
Knuth 3 ab S. 493 alle nicht prüfungsrelevant

Algorithmik 5

Optimale binäre Suchbäume

Problemstellung:

i. Gegeben eine Menge $S = \{a_1, a_2, \ldots, a_n\}$
mit bekannten Wahrscheinlichkeiten p_i für die Suche nach ai
und q_i für die Suche nach einem Element a mit $a_i < a < a_{i+1}$.

ii. Konstruiere einen Suchbaum, der die erwartete Anfragezeit (# Vergleiche mit Elementen a_i) minimiert.

Lösung durch Algorithmus von Bellman (1957)

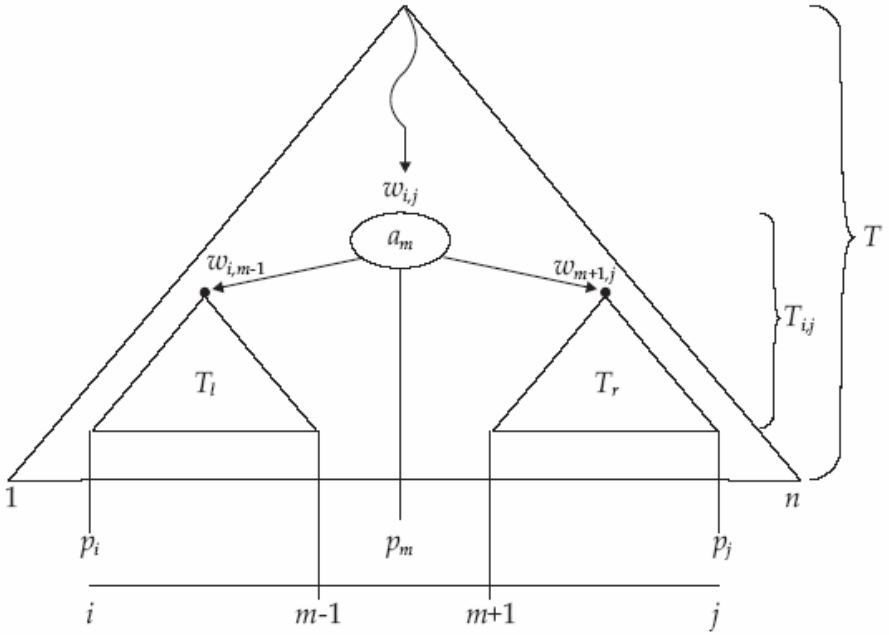
Konstruktionszzeit: $O(n^3)$ (leicht beweisbar)

Verbesserung: $O(n^2)$ (in Knuth 3 detailliert beschrieben)

Referenzen zum Erarbeiten:

Skript Alt S. 65 – 70 (Kap. 3.3) Andere Referenzen eher unübersichtlicher

Algorithmus von Bellman



Interner Suchbaum mit seinen Teilbäumen

Seien folgende Variablen für den Algorithmus vereinbart:

$r_{i,j}$:= Index der Wurzel für $T_{i,j}$
$c_{i,j}$:= Kosten von $T_{i,j} = w_{i,j} \cdot P_{T_{i,j}}$
$w_{i,j}$:= Wahrscheinlichkeit, dass $a \in [a_i, a_j]$ (wie bisher).

Behauptung 3.3.6. Es gilt

$$\begin{aligned} c_{i,j} &= w_{i,j} \cdot P_{T_{i,j}} = w_{i,j} + c_{i,m-1} + c_{m+1,j} \\ w_{i,j} &= w_{i,m-1} + p_m + w_{m+1,j}. \end{aligned}$$

Algorithm 3: [Bellman, 1957] Iterative Suche nach dem optimalen Suchbaum T.

1: **for** $i = 0, \ldots, n$ **do**
2: $\quad w_{i+1,i} = q_i$
3: $\quad c_{i+1,i} = 0$
4: **end for**
5: **for** $k = 0, \ldots, n-1$ **do**
6: $\quad$ **for** $i = 1, \ldots, n-k$ **do**
7: $\quad\quad j = i + k$
8: $\quad\quad$ Bestimme m mit $i \leq m \leq j$, so dass $c_{i,m-1} + c_{m+1,j}$ minimal ist.
9: $\quad\quad r_{i,j} = m$
10: $\quad\quad w_{i,j} = w_{i,m-1} + w_{m+1,j} + p_m$
11: $\quad\quad c_{i,j} = c_{i,m-1} + c_{m+1,j} + w_{i,j}$
12: $\quad$ **end for**
13: **end for**

aus: Skript Alt

Berechnete Wurzelindizes und Kosten

Konstruktion des Suchbaums

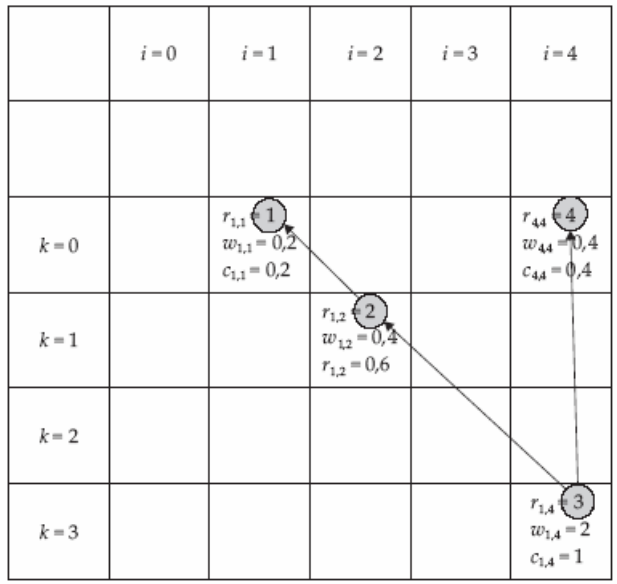
Beispiel:

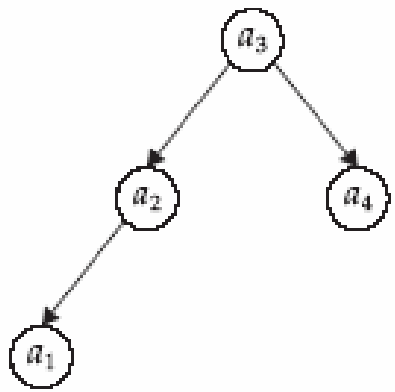
p_1=0 p_2=0,1 p_3=0,2 p_4=0,2

q_0=0,1 q_1=0,1 q_2=0,1 q_3=0,1 q_4=0,1

i	0	1	2	3	4
$Init$	$w_{1,0} = 0$ $c_{1,0} = 0$	$w_{2,1} = 0,1$ $c_{2,1} = 0$	$w_{3,2} = 0,1$ $c_{3,2} = 0$	$w_{4,3} = 0,1$ $c_{4,3} = 0$	$w_{5,4} = 0,1$ $c_{5,4} = 0$
$k = 0$		$r_{1,1} = 1$ $w_{1,1} = 0,2$ $c_{1,1} = 0,2$	$r_{2,2} = 2$ $w_{2,2} = 0,3$ $c_{2,2} = 0,3$	$r_{3,3} = 3$ $w_{3,3} = 0,4$ $c_{3,3} = 0,4$	$r_{4,4} = 4$ $w_{4,4} = 0,4$ $c_{4,4} = 0,4$
$k = 1$			$r_{1,2} = 2$ $w_{1,2} = 0,4$ $c_{1,2} = 0,6$	$r_{2,3} = 3$ $w_{2,3} = 0,6$ $c_{2,3} = 0,9$	$r_{3,4} = 3$ $w_{3,4} = 0,7$ $c_{3,4} = 1,1$
$k = 2$				$r_{1,3} = 2$ $w_{1,3} = 0,7$ $c_{1,3} = 1,3$	$r_{2,4} = 3$ $w_{2,4} = 0,9$ $c_{2,4} = 1,6$
$k = 3$					$r_{1,4} = 3$ $w_{1,4} = 2$ $c_{1,4} = 1$

Tabelle 3.1: Tabelle zur Speicherung der Berechnungen des Algorithmus





aus: Skript Alt