Algorithmik

Sebastian Iwanowski
FH Wedel

3. Lösungen für das Wörterbuchproblem
3.2 2-3-Bäume

Algorithmik 3

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

3.2 2-3-Baum

Datentyp: Suchbaum mit folgenden Eigenschaften:

i. Alle Daten sind in den Blättern, welche die gleiche Tiefe haben
ii. Alle inneren Knoten haben 2 oder 3 Kinder
iii. In den inneren Knoten stehen zu jedem Kind die jeweils größten Schlüssel aller Daten, die in den 2 bzw. 3 Kinderbäumen enthalten sind.

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

Referenzen zum Nacharbeiten und Vertiefen:

Skript Alt S. 44 – 51 (Kap. 3.1.5)
Levitin, Kap. 6.3

Algorithmik 3

Realisierung der Kernfunktionen eines 2-3-Baumes

member23 (gibt Referenz auf key-Datum zurück).

1. Falls Wurzel ein Blatt ist:
   Falls Datum in Wurzel key enthält, gib Referenz darauf zurück, anderenfalls null.
2. Anderenfalls: Bestimme Kind, in dem weiter gesucht werden muss:
   Es handelt sich um das kleinste Kind, dessen Schlüssel größer gleich key ist.
3. Falls solch ein Kind existiert: Rufe member23 rekursiv für dieses Kind auf.
4. Anderenfalls: Gib null zurück (Schlüssel ist offenbar nicht vorhanden).

Algorithmik 3

Realisierung der Kernfunktionen eines 2-3-Baumes

insert23 (gibt Wurzel eines 2-3-Baumes zurück, der newdata enthält).

1. Falls Wurzel ein innerer Knoten ist (anderenfalls Spezialbehandlung!):
   Bestimme den Kindknoten, in den eingefügt werden muss
   Es handelt sich um das kleinste Kind, dessen Schlüssel größer gleich key ist, wenn ein solches exisitiert,
   bzw. um das größte Kind sonst
2. Rufe insert23rec für diesen Kindknoten auf.
3. Falls nur eine Wurzel zurückkommt, adoptiere diese als neues Kind (statt des alten) und gib eigene Wurzel zurück.
4. Falls zwei Wurzeln zurückkommen und nur ein weiteres Kind existiert, adoptiere beide Wurzeln als neue Kinder
   und gib eigene Wurzel zurück.
5. Falls zwei Wurzeln zurückkommen und schon zwei weitere Kinder existieren,
   spalte die eigene Wurzel zu zwei neuen Kindern auf, generiere eine neue Wurzel und hänge die neuen Kinder
   daran. Gib dann die neue Wurzel zurück. (*Anm.: Jetzt hat sich die Tiefe um 1 erhöht*)

insert23rec (gibt 1 oder 2 Wurzeln eines 2-3-Baumes zurück, der newdata enthält).

1. Falls Wurzel ein Blatt ist: Generiere ein weiteres Blatt mit newdata und gib dieses und die alte Wurzel zurück (also 2 Wurzeln).
2. Anderenfalls: Bestimme den Kindknoten, in den eingefügt werden muss.
3. Rufe insert23rec für diesen Kindknoten rekursiv auf.
4. Falls nur eine Wurzel zurückkommt, adoptiere diese als neues Kind (statt des alten) und gib eigene Wurzel zurück.
5. Falls zwei Wurzeln zurückkommen und nur ein weiteres Kind existiert, adoptiere beide Wurzeln als neue Kinder und gib
   eigene Wurzel zurück.
6. Falls zwei Wurzeln zurückkommen und schon zwei weitere Kinder existieren,
   spalte die eigene Wurzel auf, verteile die Kinder zu je zwei und gib die beiden neuen Wurzeln zurück.

Algorithmik 3

Realisierung der Kernfunktionen eines 2-3-Baumes

delete23 (gibt Wurzel eines 2-3-Baumes zurück, aus dem key-Datum gestrichen wurde).

1. Rufe delete23rec auf.
2. Falls nur ein Kind zurückkommt, gib dieses als Wurzel der gesamten Datenbank zurück.
   (***Anm.: Jetzt hat sich die Tiefe um 1 verringert***.)
3. Anderenfalls generiere eine neue Wurzel mit den zurückgegebenen Kindern und gib diese als Wurzel der gesamten Datenbank zurück.

delete23rec (gibt 1 bis 3 Kinder der aufgerufenen Wurzel zurück, aus denen key-Datum gestrichen wurde).

1. Falls die Kinder Blätter sind, gib die Kinder zurück, die key nicht enthalten.
2. Anderenfalls bestimme den Kindknoten, aus dem entfernt werden muss:
   Es handelt sich um das kleinste Kind, dessen Schlüssel größer gleich key ist.
3. Falls solch ein Kindknoten nicht existiert, brich die Rekursion ab und gib die bisherigen Kinder zurück.
4. Anderenfalls rufe delete23rec für diesen Kindknoten rekursiv auf.
5. Falls die Summe der von diesem zurückgegebenen Kinder und der anderen **Enkel** nur 3 ist, generiere ein Kind mit diesen Enkeln als Kindern und gib das neue Kind zurück.
6. Falls die Summe der von diesem zurückgegebenen Kinder und der anderen **Enkel** zwischen 4 und 6 ist, generiere zwei Kinder mit diesen Enkeln als Kindern und gib die neuen Kinder zurück.
7. Falls die Summe der von diesem zurückgegebenen Kinder und der anderen **Enkel** zwischen 7 und 9 ist, generiere drei Kinder mit diesen Enkeln als Kindern und gib die neuen Kinder zurück.

Algorithmik 3

Realisierung der Kernfunktionen eines 2-3-Baumes

Hausaufgabe:

1. Entwerfen Sie eine Datenstruktur für einen 2-3-Baum, der Integerzahlen verarbeiten kann
   (hierbei ist der Schlüssel gleich dem Datum)
2. Implementieren Sie die Funktionen member23, insert23 und delete23
   in einer Sprache Ihrer Wahl
3. Testen Sie die Funktionen in einer geeigneten Umgebung

Referenzen zum Nacharbeiten und Vertiefen:

Lisp-Modul tree23.txt auf dem Handout-Server,
Funktionen insert23core, insert23rec, delete23core, delete23rec und member23

Die in tree23.txt implementierte Datenstruktur setzt den Schlüssel gleich dem Datum und verarbeitet beliebige Vergleichs- und Identitätsfunktionen.
Außerdem werden verschiedene Eindeutigkeitsvarianten sowie Spezialfälle zur Verfügung gestellt.