Algorithmik

Sebastian Iwanowski
FH Wedel

4. Vorlesungswoche

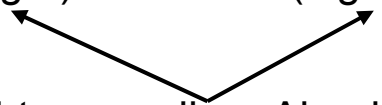
Algorithmik 4

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

Sortiertes Feld als Dictionary:

member (key) Laufzeit Θ(log n) w.c. und Θ(log log n) a.c. erreichbar



nicht vom selben Algorithmus

insert (key, newdata) Laufzeit Θ(n) w.c. und a.c.

insert (key, newdata) Laufzeit Θ(n) w.c. und a.c.

Referenzen zum Nacharbeiten und Vertiefen:

Skript Alt, S. 30 – 35 (für member, Wiederholung vom letzten Mal)

Algorithmik 4

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

Hashing

Datentyp: Indiziertes Array mit m Speicherplätzen

Arbeitsweise:

- Es gibt hash-Funktion h: Keys → {0,…,m-1}
- Jedes Element wird auf Position h(k) abgespeichert,
  wenn diese Position noch frei ist (wobei k der Schlüssel des Elements ist)
- Wenn die Position h(k) besetzt ist,
  muss eine Kollisionsvermeidung gemacht werden (verschiedene Strategien)

Alle 3 Dictionary-Funktionen — Laufzeit $\Theta(n)$ w.c. und $\Theta(n/m)$ a.c.
$\Rightarrow$ für $n \in O(m)$: Laufzeit $\Theta(1)$ a.c.

Referenzen zum Nacharbeiten und Vertiefen:

Foliensatz Hashing zu dieser Vorlesung
Cormen, Kap. 11

Algorithmik 4

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

Hashing: Kollisionsvermeidung

Datentyp: Indiziertes Array mit m Speicherplätzen

Hashlisten

- An Position h(k) wird anstelle des Datums ein Pointer auf eine lineare Liste vorgehalten.
  Alle Daten, die auf h(k) abgebildet werden, werden sequentiell in die Liste eingefügt.

Offenes Hashing

- Wenn die Position h(k) besetzt ist,
  wird mit spezieller Sondierungsregel eine andere Position bestimmt
- Für die Sondierungsregel gibt es viele Strategien

Referenzen zum Nacharbeiten und Vertiefen:

Foliensatz Hashing zu dieser Vorlesung
Cormen, Kap. 11

Algorithmik 4

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

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 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 4

Realisierung der Kernfunktionen eines 2-3-Baumes

Hausaufgabe (zum 15.05.):

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.

Algorithmik 4

Realisierung der Kernfunktionen eines 2-3-Baumes

Insert23 (gibt Wurzel eines 2-3-Baumes zurück, der newdata enthält).

1. Bestimme den Kindknoten, in den eingefügt werden muss.
2. Rufe insert23rec für diesen Kindknoten auf.
3. Falls nur eine Wurzel zurückkommt, lass alles beim 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.

Insert23rec (gibt 1 oder 2 Wurzeln eines 2-3-Baumes zurück, der newdata enthält).

1. Bestimme den Kindknoten, in den eingefügt werden muss.
2. Rufe insert23rec für diesen Kindknoten rekursiv auf.
3. Falls nur eine Wurzel zurückkommt, lass alles beim 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 auf, verteile die Kinder zu je zwei und gib die beiden neuen Wurzeln zurück.

Algorithmik 4

Realisierung der Kernfunktionen eines 2-3-Baumes

Delete23 (gibt Wurzel eines 2-3-Baumes zurück, aus dem newdata gestrichen wurde).

1. Bestimme den Kindknoten, in dem gestrichen werden muss.
2. Rufe delete23rec für diesen Kindknoten auf.
3. Falls die Summe der von diesem zurückgegebenen Kinder und der anderen Enkel nur 3 ist, adoptiere alle neuen Enkel als Kinder und gib die eigene Wurzel zurück.
4. Falls die Summe der von diesem zurückgegebenen Kinder und der anderen Enkel mehr als 3 ist, generiere neue Kinder (2 oder 3), auf welche die neuen Enkel gleichmäßig verteilt werden, und gib die eigene Wurzel zurück.

Delete23rec (gibt 1 bis 3 Kinder zurück, aus denen newdata gestrichen wurde).

1. Bestimme den Kindknoten, in den eingefügt werden muss.
2. Rufe delete23rec für diesen Kindknoten rekursiv auf.
3. 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.
4. 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.
5. 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.