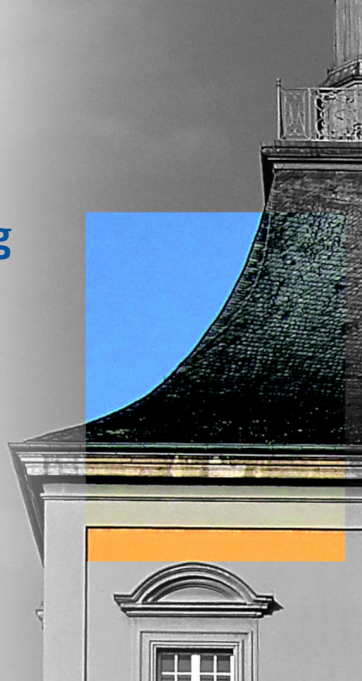
UNIVERSITÄT BONN

Algorithmen und Programmierung

Algorithmen II

Dr. Felix Jonathan Boes
boes@cs.uni-bonn.de
Institut für Informatik



KURZE WIEDERHOLUNG

Abstrakte Datentypen und Datenstrukturen

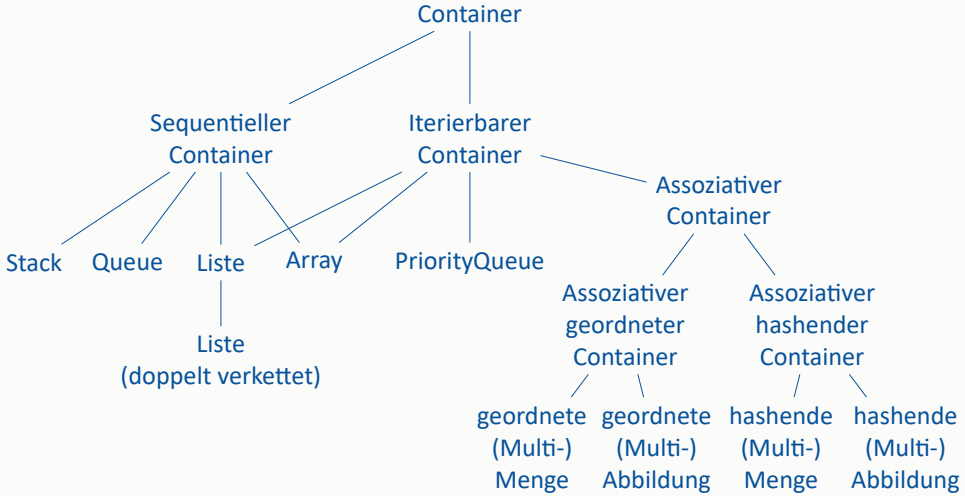
Offene Fragen

Wie sind die wichtigsten abstrakten Datentypen definiert?

Durch welche Datenstrukturen werden sie realisiert?

UNIVERSITÄT BONN

Die (wichtigsten) abstrakten Datentypen im Überblick



Wir beantworten hier:

Wie wird eine geordnete Menge realisiert?

UNIVERSITÄT BONN

Vorstufe: Binäre Suchbäume

Binäre Suchbäume speichern Knoten so, dass die Elemente beim Inorderdurchlauf aufsteigend sortiert sind.

Der Aufwand von Einfügen und Entfernen ist $\mathcal{O}(h)$, wobei h die Höhe des Baums bezeichnet.

Ziel: Suchbäume so modifizieren, dass $h = \Theta(\log(n))$ gilt. Dann kostet Einfügen und Entfernen $\mathcal{O}(\log(n))$. Das ist nötig um eine geordnete Menge zu realisieren.

AVL Bäume

Balancierte Binärbäume

UNIVERSITÄT BONN

Balancierte Binärbäume

(Balancekoeffizient)

Sei G ein Binärbaum und v ein Knoten in G mit linkem Teilbaum G_l^v und rechtem Teilbaum G_r^v. Der **Balancekoeffizient** $B(v)$ ist die Höhendifferenz

$$B(v) = h(G_r^v) - h(G_l^v)\,.$$

Der Knoten v heißt **balanciert** wenn $-1 \leq B(v) \leq 1$ erfüllt ist.

(Balancierter Binärbaum)

Ein Binärbaum ist ein **balancierter Binärbaum** wenn alle Knoten balanciert sind.

Wir haben gezeigt, dass sich bei balancierten Binärbäumen $h = \Theta(\log(n))$ gilt.

AVL Bäume

Knoten hinzufügen und entfernen

Nächstes Ziel

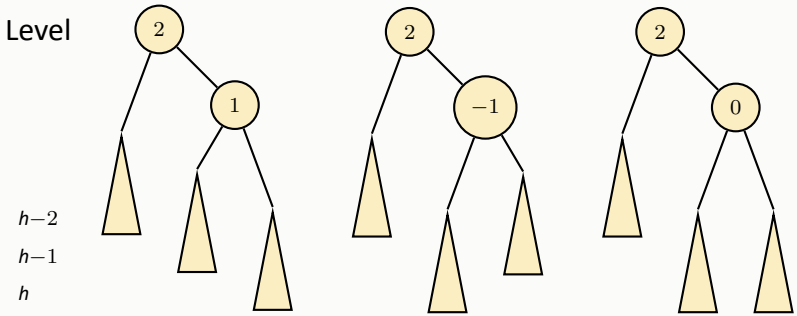
Wir wollen balancierte Binärbäume als Suchbäume nutzen

Um balancierten Binärbäumen Blätter hinzuzufügen und zu entfernen, passen wir die bekannten Operationen INSERT und DELETE an

UNIVERSITÄT BONN

Beobachtung

Beim Hinzufügen oder Entfernen betrachten wir den tiefsten Knoten v, der nicht mehr balanciert ist. Für $B(v) = 2$ sind drei Fälle möglich (für $B(v) = -2$ geht man analog vor). Die zugehörigen Teilbäume werden durch Dreiecke angedeutet.

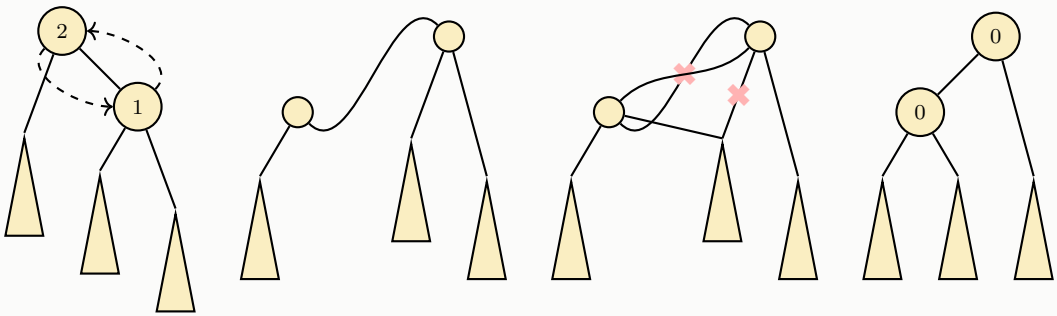


Fälle 1 und 2 treten beim Hinzufügen auf. Fälle 1, 2 und 3 treten beim Entfernen auf. Wir untersuchen zunächst Fall 1 und 2 beim Hinzufügen eines Blatts.

UNIVERSITÄT BONN

Korrektur im Fall 1

Um den Defekt im Fall 1 zu korrigieren, führen wir eine **Linksrotation** durch.



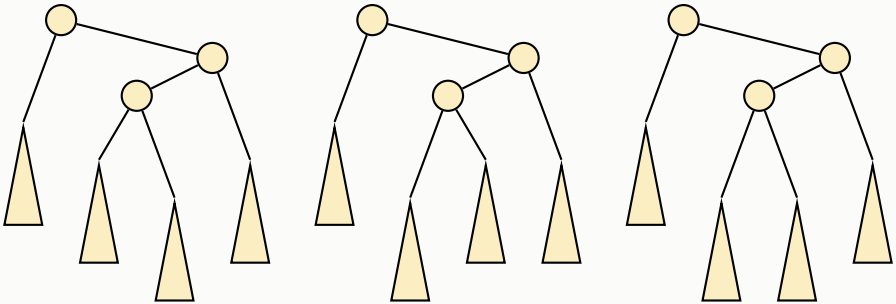
Das Hinzufügen des Blatts erhöht den Balancekoeffizient aller Vorfahren des Blatts um eins. Das Rotieren verringert den Balancekoeffizient aller Vorfahren des Knoten um den gereht wird um eins. Wir erhalten somit wieder einen balancierten Binärbaum.

Haben Sie Fragen?

UNIVERSITÄT BONN

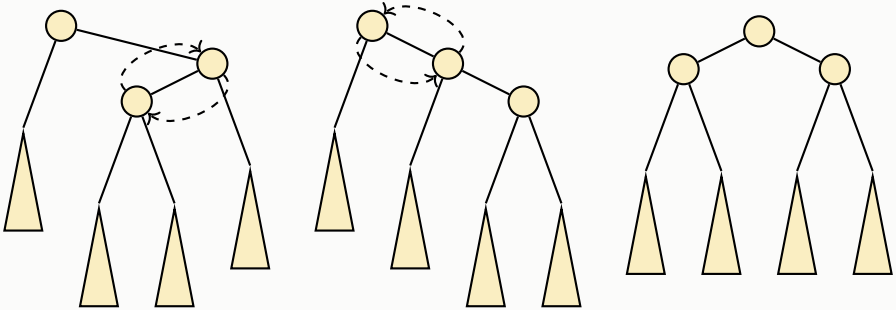
Korrektur im Fall 2

Um den Defekt im Fall 2 zu korrigieren, reicht eine Rotation nicht aus. Wir führen eine **Doppelrotationen** durch. Hier gibt es drei Unterfälle.



Korrektur im Fall 2

Wir führen eine **Doppelrotation** für den dritten Unterfall durch. Die Doppelrotation (und die Schlussfolgerung) ist für die anderen beiden Unterfälle identisch.



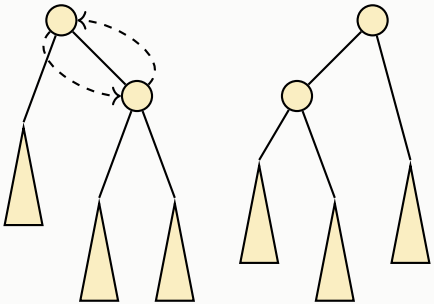
Das Hinzufügen des Blatts erhöht den Balancekoeffizient aller Vorfahren des Blatts um eins. Das Rotieren verringert den Balancekoeffizient aller Vorfahren des oberen Knoten um den gereht wird um eins.

Haben Sie Fragen?

UNIVERSITÄT BONN

Korrektur im Fall 3

Der dritte Fall tritt nur durch das Entfernen eines Knoten auf. Hier führen wir wieder eine Linksrotation durch.



Hier bleibt die Höhe erhalten.

Haben Sie Fragen?

UNIVERSITÄT BONN

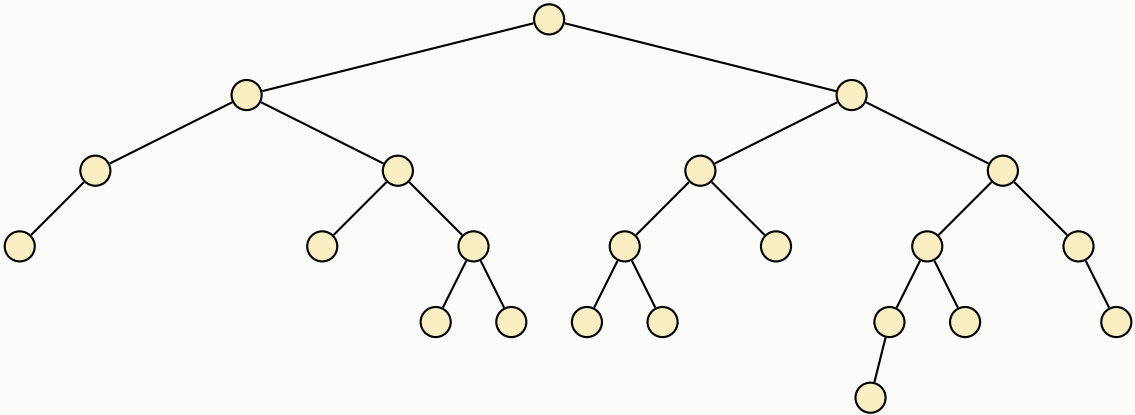
Zum Entfernen I

Beim Entfernen können die Fälle 1, 2 und 3 auf treten. In Fall 3 bleibt die Höhe des Teilbaums erhalten, in den Fällen 1 und 2 wird die Höhe verringert.

Beim Entfernen kann es deshalb (wegen der Fällen 1 und 2) nötig werden, weitere Rotationen durchzuführen.

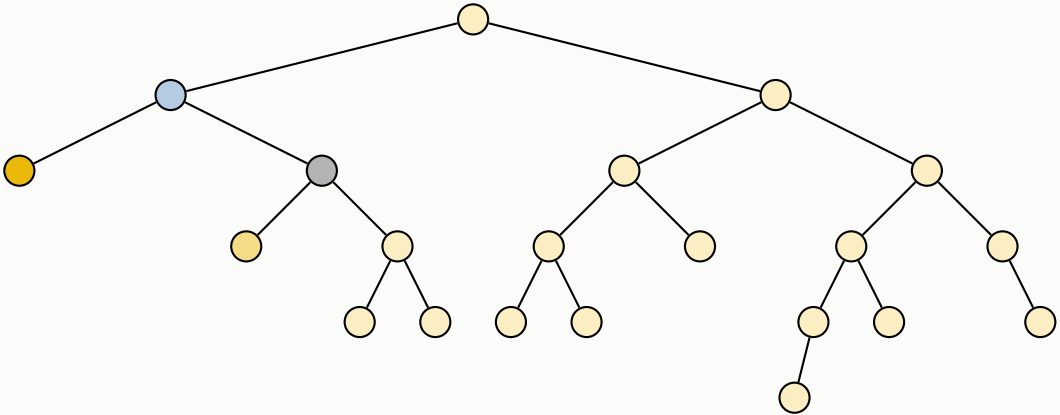
UNIVERSITÄT BONN

Beispiel Knoten Entfernen



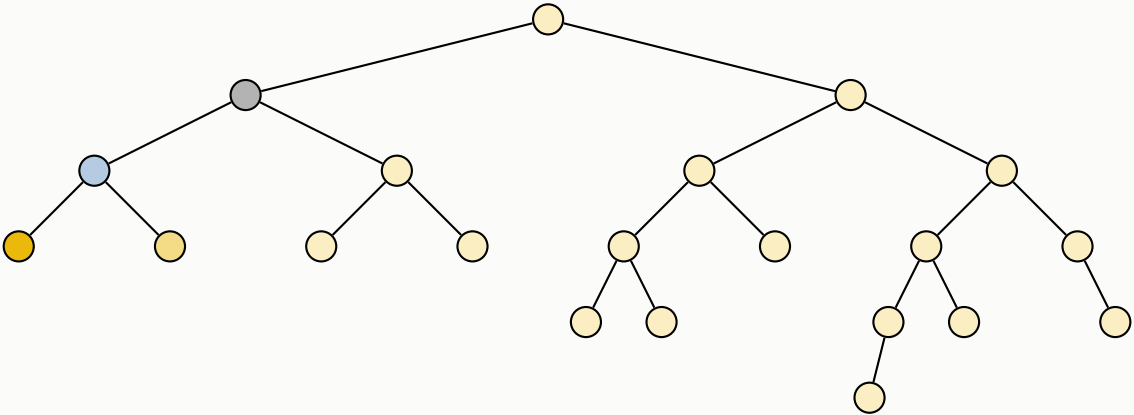
UNIVERSITÄT BONN

Beispiel Knoten Entfernen



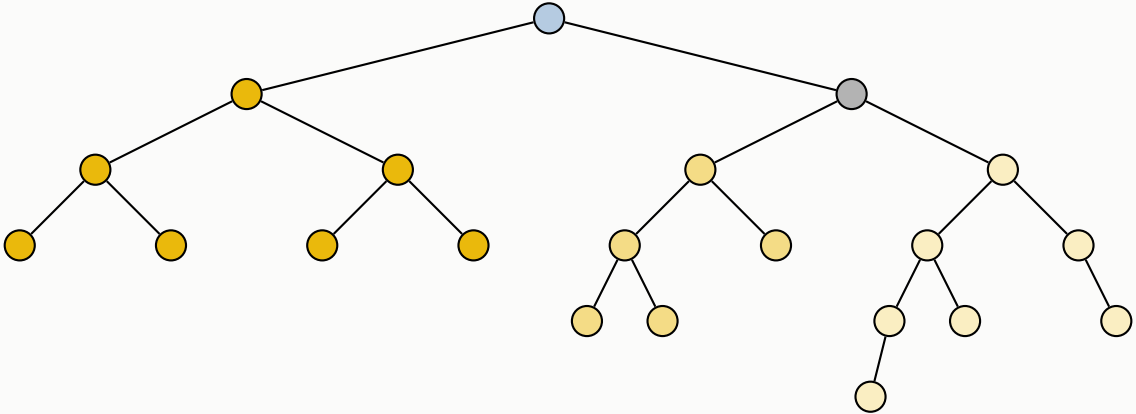
UNIVERSITÄT BONN

Beispiel Knoten Entfernen



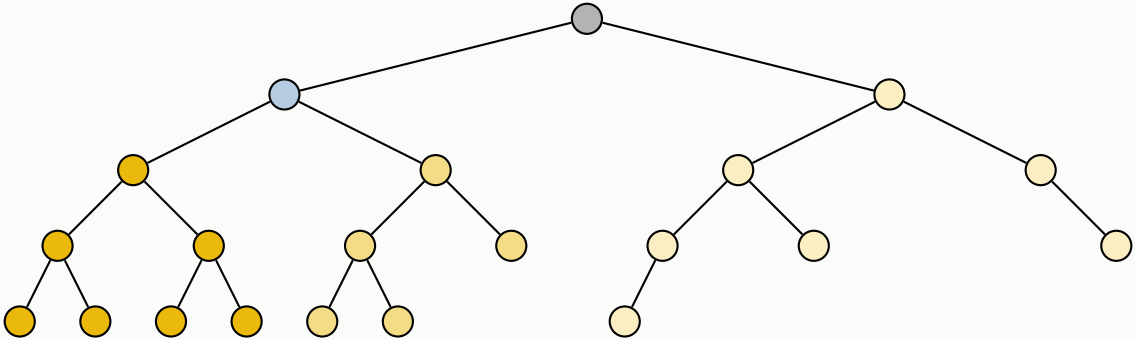
UNIVERSITÄT BONN

Beispiel Knoten Entfernen



UNIVERSITÄT BONN

Beispiel Knoten Entfernen



UNIVERSITÄT BONN

Zum Entfernen II

Wenn ein Knoten entfernt wird und dadurch die Balanceeigenschaft verletzt wird, gibt es (wie vorher auch) ein Blatt v sodass die Balancekoeffizienten auf dem Weg von v zur Wurzel einen Balancekoeffizient von $-2 \leq B \leq 2$ haben sowie alle anderen Knoten einen Balancekoeffizient von $-1 \leq B \leq 1$ haben.

Hier führt man auf dem Weg von v zur Wurzel an jedem Knoten der einen Balancekoeffizient ± 2 hat jeweils eine (Doppel)rotation durch. Dadurch erhält man wieder einen balancierten Binärbaum. In dieser Vorlesung führen wir die zugehörige Argumentation nicht aus. Allerdings ist die Argumentation nicht sehr schwer sofern man induktiv von v zur Wurzel vorgeht.

Da ein balancierter Binärbaum die Höhe $h = \Theta(\log(n))$ hat, sind nach dem Entfernen maximal $\mathcal{O}(h) = \mathcal{O}(\log(n))$ viele (Doppel)rotationen nötig.

Haben Sie Fragen?

Zwischenfazit

Um balancierten Binärbäume Elemente hinzuzufügen und zu entfernen haben wir INSERT und DELETE angepasst

Nach dem Einfügen eines Blatts ist maximal eine (Doppel)rotation nötig

Nach dem Entfernen eines Knoten sind maximal $\mathcal{O}(\log(n))$ viele (Doppel)rotationen nötig

Haben Sie Fragen?

AVL Bäume

Balancierte Binärbäume und Suchbäume

Wir zeigen

Die angepassten Operationen INSERT und DELETE sind verträglich mit Suchbäumen

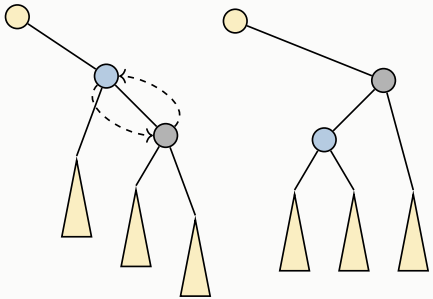
UNIVERSITÄT BONN

Problemstellung und Ziel

Unser schlussendliches Ziel ist es, balancierte Binärbäume als Suchbäume einzusetzen. Dazu werden wir noch prüfen, dass die Rotationsoperationen die Suchbaumeigenschaft respektiert.

Nachdem wir uns davon überzeugt haben, können wir Suchbäume mithilfe von balancierten Binärbäumen sehr effizient implementieren. Damit haben wir es endlich geschafft, geordnete Mengen zu realisieren.

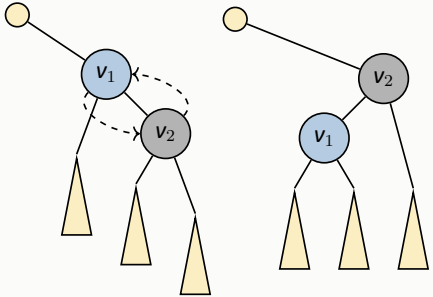
Suchbäume und Rotationen I



Bei jeder Rotation bleiben die Einträge der beteiligten Knoten und Teilbäume erhalten. Also ist die Suchbaumeigenschaft für alle Knoten oberhalb des Knoten, um den gedreht wird, weiterhin erhalten.

Die bei einer Rotation beteiligten Teilbäume ändern ggf. ihren Elternknoten, bleiben aber sonst erhalten. Also ist die Suchbaumeigenschaft der beteiligten Teilbäume weiterhin erhalten.

Suchbäume und Rotationen II



Der linke Teilbaum von v_1 bleibt erhalten.

Alle Knoten rechts von v_1 sind größer als v_1. Diese Eigenschaft ist, mithilfe eines direkten Vergleichs mit der Grafik, nach der Rotation ebenfalls erfüllt.

Der rechte Teilbaum von v_2 bleibt erhalten.

Es gilt $v_1 < v_2$ da v_1 die Suchbaumeigenschaft erfüllt. Nach der Rotation ist dann der gesamte linke Teilbaum von v_2 kleiner als v_2.

Die Rotationsoperationen sind also verträglich mit der Suchbaumeigenschaft.

Wir haben gezeigt

Die angepassten Operationen INSERT und DELETE sind verträglich mit Suchbäumen

Haben Sie Fragen?

AVL Bäume

Definition AVL-Bäume

AVL-Bäume

(AVL-Baum)

Ein binärer Suchbaum ist ein **AVL-Baum** wenn alle Knoten balanciert sind, und die, um (Doppel)rotationen ergänzten, Operationen verwendet werden.

Wir haben in dem vergangenen Abschnitt der Vorlesung gezeigt, dass die Operationen verträglich mit der Balanceeigenschaft und der Suchbaumeigenschaft sind.

Abstrakte Datentypen und Datenstrukturen

Assoziative Container und deren Realisierung - Geordnete Menge

Wir beantworten die offene Fragen

Durch welche Datenstruktur werden geordnete Mengen realisiert?

UNIVERSITÄT BONN

AVL-Bäume realisieren geordnete Mengen

Da die Keys untereinander vergleichbar sind, können Sie in einem AVL-Baum verwaltet werden.

Wir haben gezeigt, dass das Einfügen und Entfernen einen Aufwand von $\mathcal{O}(\log(n))$ hat, wobei n die Anzahl der enthaltenen Elemente bezeichnet. Deshalb realisieren AVL-Bäume geordnete Mengen[1].

[1] AVL-Bäume realisieren auch geordnete Multimengen, wenn man zulässt, dass die Elemente jedes linken Teilbaum kleiner oder gleich groß sind.

Zusammenfassung

Geordnete Mengen werden durch AVL-Bäume realisiert

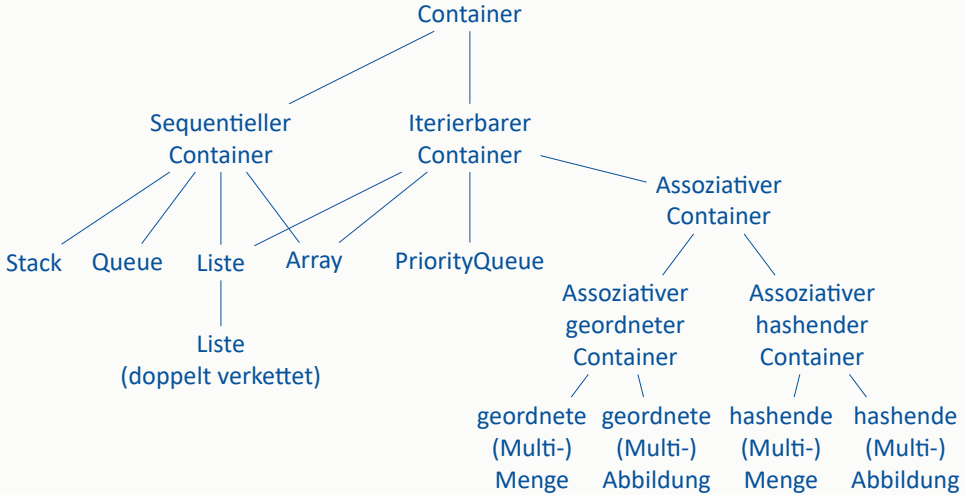
Haben Sie Fragen?

Abstrakte Datentypen und Datenstrukturen

Assoziative Container und deren Realisierung - Geordnete Abbildungen

UNIVERSITÄT BONN

Die (wichtigsten) abstrakten Datentypen im Überblick



Offene Fragen

Was ist eine geordnete Abbildungen?

Durch welche Datenstruktur wird sie realisiert?

UNIVERSITÄT BONN

Geordnete (Multi)abbildung (moralisch)

Erinnerung: Im Rahmen von assoziativen Containern bezeichnen wir konstante Objekte als **Keys** und nicht notwendigerweise konstante Objekte als **Values**.

Eine **geordnete Abbildung** / **geordnete Multiabbildung** ist ein assoziativer Container zum Speichern von Values, die über **untereinander vergleichbare** Keys indiziert werden. Dabei darf jeder Key höchstens einmal / mehrfach vorkommen.

UNIVERSITÄT BONN

Geordnete (Multi)abbildung (formaler)

Erinnerung: Eine Menge X heißt **total geordnet** bezüglich einer binären Relation $\preceq$, falls $\preceq$ reflexiv, transitiv, antisymmetrisch und total ist.

Eine **geordnete Abbildung** / **geordnete Multiabbildung** ist ein assoziativer Container zum Speichern von Key-Value-Paaren für total geordneten Keys. Dabei darf jeder Key höchstens einmal / mehrfach vorkommen.

UNIVERSITÄT BONN

Anwendung

Geordnete Abbildungen werden oft verwendet, um Funktionen zu modelieren, bei denen der Definitionsbereich aus untereinander vergleichbaren Keys besteht und der Bildbereich aus Values besteht.

In anderen Worten erlauben es geordnete Mengen Wörterbücher (Dictionaries) anzulegen, wobei der Zugriff auf die Values über untereinander geordnete Keys ermöglicht wird.

UNIVERSITÄT BONN

AVL-Bäume realisieren geordnete Abbildungen

Geordnete Abbildungen speichern Key-Value-Paare. Da die Keys untereinander vergleichbar sind, können Key-Value-Paare (über die Verwendung des Keys) in einem AVL-Baum verwaltet werden.

Wir haben gezeigt, dass das Einfügen und Entfernen einen Aufwand von $\mathcal{O}(\log(n))$ hat, wobei n die Anzahl der enthaltenen Elemente bezeichnet. Deshalb realisieren AVL-Bäume geordnete Abbildungen[2].

[2] Analog realisieren AVL-Bäume geordnete Multiabbildungen, wenn wir zulassen, dass die Elemente jedes linken Teilbaum kleiner oder gleich groß sind.

Zusammenfassung

Geordnete Abbildnugen werden durch AVL-Bäume realisiert

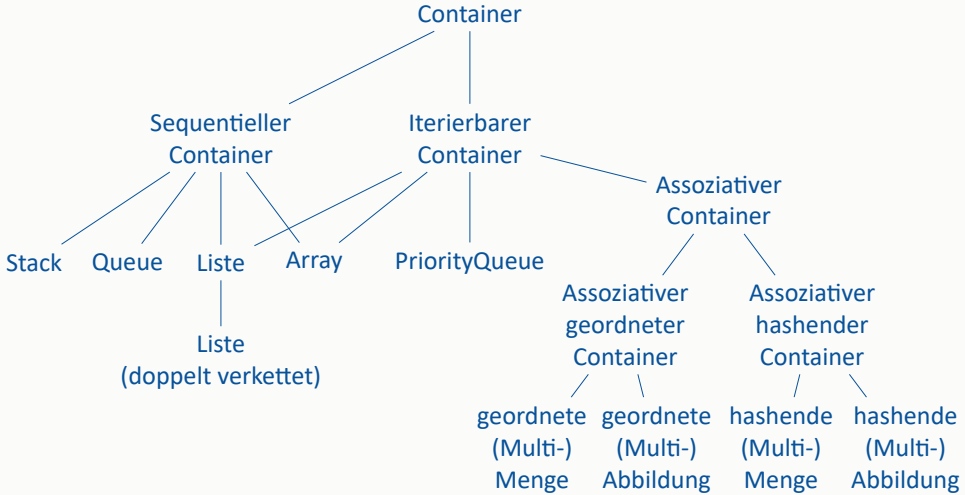
Haben Sie Fragen?

Abstrakte Datentypen und Datenstrukturen

Assoziative Container und deren Realisierung - Hashende Menge

UNIVERSITÄT BONN

Die (wichtigsten) abstrakten Datentypen im Überblick



Offene Fragen

Was ist eine hashende Menge?

Durch welche Datenstruktur wird sie realisiert?

UNIVERSITÄT BONN

Vorbereitung: Hashfunktionen

Erinnerung: Im Rahmen von assoziativen Containern bezeichnen wir konstante Objekte als **Keys** und nicht notwendigerweise konstante Objekte als **Values**.

Gegeben eine (möglicherweise unbeschränkte) Menge von Keys K und eine Menge S von Values (mit identischer, fester Bytegröße). Dann bezeichnet man jede Funktion $h\colon K \to S$ als **Hashfunktion**.

Die Keymenge nennt man **hashbar** bezüglich h. Außerdem bezeichnet man $h(x)$ als den **Hashwert**, **Hashcode** oder **Hash** von x.

UNIVERSITÄT BONN

Simple Beispiele

Wir betrachten drei simple Hashfunktionen.

- Mittlerer Buchstabe : *Strings* $\to$ *Buchstaben*.
- Anzahl 1-Bits : *uint64_t-Werte* $\to \{0, \ldots, 64\}$.
- Byteweise XOR : *Bytefolgen* $\to \{0x00, \ldots, 0xFF\}$.

UNIVERSITÄT BONN

Einige gewünschte Eigenschaften

Unter anderem sind die folgenden Eigenschaften wünschenswert.

- **Gleichförmigkeit**: Die Hashes sollen möglichst gleichverteilt sein
- **Lawineneffekt**: Bei Veränderung eines einzigen Bits des Inputs hat jedes Bit des Outputs eine Wahrscheinlichkeit von ca. 50 % sich zu ändern
- **Effizienz**: Der Hash soll möglichst schnell und ressourcenarm berechnet werden

Verbreitete Hashfunktionen

Die folgenden Hashfunktionen werden / wurden häufig verwendet

Datenspeicherung:

- Fowler-Noll-Vo (FNV), jenkins (lookup3), MurmurHash

Kryptographie:

- MD2, MD4, MD5 (alle unsicher)
- SHA-2, z.B. SHA-256 (Hash hat 256 bit)
- MD6, SHA-3 (Keccak)
- Bcrypt, Scrypt, LM-Hash, PBKDF2

UNIVERSITÄT BONN

Hashende (Multi)mengen (moralisch)

Erinnerung: Im Rahmen von assoziativen Containern bezeichnen wir konstante Objekte als **Keys** und nicht notwendigerweise konstante Objekte als **Values**.

Eine **hashende Menge** / **hashende Multimenge** ist ein assoziativer Container[3] zum Speichern von Keys die (bezüglich einer möglichst gleichverteilten Hashfunktion) **hashbar** sind. Dabei darf jeder Key höchstens einmal / mehrfach gespeichert werden.

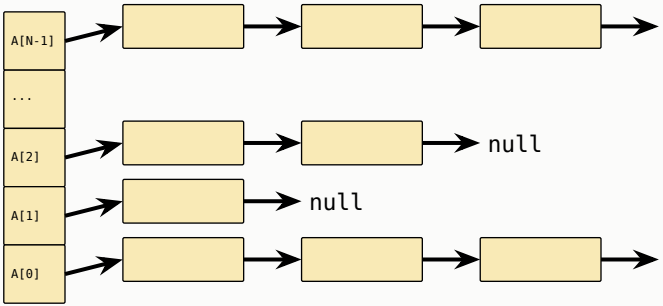
[3] Um Hashende Multimengen formal korrekt zu definieren, müssten alle vorkommenden, stochastischen Größen genau benannt werden sowie probabilistische Laufzeitbeschränkungen eingeführt und benannt werden. Aus Zeitgründen wollen wir in diesem Modul auf eine formal korrekte Diskussion dieses und darauf aufbauender Sachverhalten verzichten.

UNIVERSITÄT BONN

Hashende Multimenge als Datenstruktur I

Wir betrachten Hashfunktionen mit *uint64_t-Hashes*.

Eine **Hashtabelle** besteht aus einem dynamischen Array von verketteten Listen. Die verketteten Listen heißen auch **Buckets**. Die Größe des Array entspricht offenbar der Anzahl der Buckets und wird mit *N* bezeichnet.

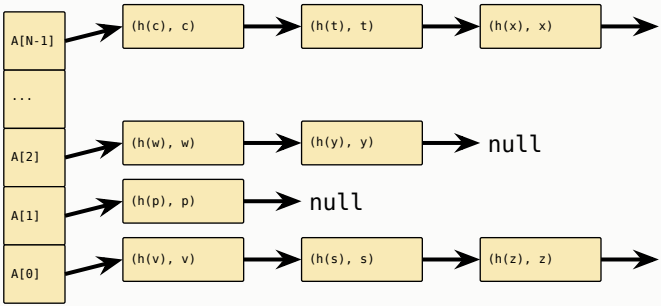


UNIVERSITÄT BONN

Hashende Multimenge als Datenstruktur II

Wir betrachten Hashfunktionen mit *uint64_t-Hashes*.

In den Buckets werden Hash-Key-Paare der Form $(h(x), x)$ gespeichert. Dabei liegt das Paar $(h(x), x)$ im Bucket $h(x)$ mod N.

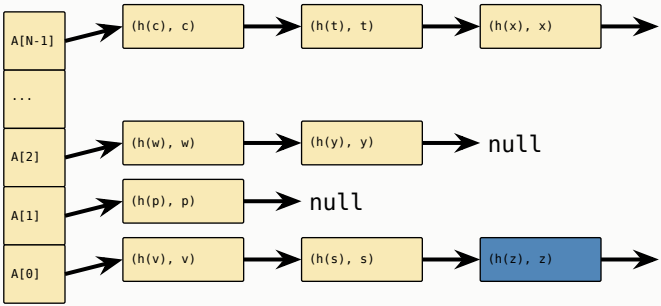


UNIVERSITÄT BONN

Hashende Multimenge als Datenstruktur III

Wir betrachten Hashfunktionen mit *uint64_t-Hashes*.

Um zu prüfen ob ein Element z in der Hashtabelle gespeichert ist, wird $i = h(z) \mod N$ berechnet und im Bucket $A[i]$ nach dem Paar $(h(z), z)$ gesucht.



UNIVERSITÄT BONN

Hashende Multimenge als Datenstruktur IV

Der **Belegungsfaktor** einer Hashtabelle mit n Elementen und N Buckets ist

$$\alpha = \frac{n}{N}$$

Der Belegungsfaktor spielt beim Einfügen (und Entfernen) eine wesentliche Rolle. Wenn der Belegungsfaktor kleiner als 0.75 ist, dann wird ein neues Element u wie erwartet eingefügt. Das heißt, es wird $i = h(u) \mod N$ berechnet und $(h(u), u)$ dem Bucket $A[i]$ hinzugefügt.

Wenn der Belegungsfaktor mindestens 0.75 ist, werden die Anzahl der Buckets verdoppelt und alle gespeicherten Elemente entsprechend der neuen Bucketanzahl verteilt, sodass $(h(t), t)$ in Bucket $h(t) \mod 2N$ landet. Anschließend wird das neue Element wie üblich eingefügt.

Beim Entfernen wird analog verfahren. Die untere Schranke für den Belegungsfaktor ist hier 0.25.

UNIVERSITÄT BONN

Hashende Mengen werden durch Hashtabellen realisiert (Skizze)

Hashtabellen realisieren iterierbare Container: Eine Hashtabelle ist ein Array von verketteten Listen und somit iterierbar.

Die zusätzlichen Operationen und Laufzeiten sind gegeben: Wird eine möglichst gleichverteilte, effizient berechenbare Hashfunktion verwendet, dann sind die Buckets (im Durchschnitt) sehr leer. Die Umsortierung der Elemente geschieht so selten, dass im Durchschnitt alle Operationen effizient genug sind. Also wird die hashende Menge durch die zugehörige Hashtabelle realisiert.

Geeignete Hashfunktionen

In der Anwendung besteht der Wunsch, immer eine geeignete Hashfunktion verwendet werden. Möglichst gleichverteilte, effizient berechenbare Hashfunktionen sind leider schwer zu bestimmen.

Für grundlegende Datentypen liefern die meisten Programmiersprachen solche Hashfunktionen mit. Für zusammengesetzte Datentypen ist es möglich, diese mitgelieferten Hashfunktionen zu kombinieren.

Haben Sie Fragen?

Zusammenfassung

Hashtabellen speichern Elemente in Buckets und verwenden dazu eine Hashfunktion

Möglichst gleichverteilte, effizient berechenbare Hashfunktion sind rar

Hashende Mengen werden durch Hashtabellen realisiert

Abstrakte Datentypen und Datenstrukturen

Assoziative Container und deren Realisierung - Hashende Abbildungen

Offene Fragen

Was ist eine hashende Abbildungen?

Durch welche Datenstruktur wird sie realisiert?

UNIVERSITÄT BONN

Hashende (Multi)abbildung (moralisch)

Erinnerung: Im Rahmen von assoziativen Containern bezeichnen wir konstante Objekte als **Keys** und nicht notwendigerweise konstante Objekte als **Values**.

Eine **hashende Multiabbildung** / **hashende Multiabbildung** ist ein assoziativer Container[4] zum Speichern von Key-Value-Paaren für Keys die (bezüglich einer möglichst gleichverteilten Hashfunktion) **hashbar** sind. Dabei darf jeder Key höchstens einmal / mehrfach gespeichert werden.

Wir argumentieren wie bei geordneten Multimengen und Multiabbildungen und sehen sofort ein, dass Hashtabellen geordnete Abbildungen realisieren.

[4] Um Hashende Multiabbildungen formal korrekt zu definieren, müssten alle vorkommenden, stochastischen Größen genau benannt werden sowie probabilistische Laufzeitbeschränkungen eingeführt und benannt werden. Aus Zeitgründen wollen wir in diesem Modul auf eine formal korrekte Diskussion dieses und darauf aufbauender Sachverhalten verzichten.

Haben Sie Fragen?

Zusammenfassung

Hashende Abbildungen werden durch Hashtabellen realisiert