

Informatik 2

Baumstrukturen

Reiner Hüchting

7. Mai 2022

Themenüberblick – Bäume

Bäume

Binäre Suchbäume

Balancierte Bäume

Heaps

Speicherung von Bäumen

Themenüberblick – Bäume

Bäume

Binäre Suchbäume

Balancierte Bäume

Heaps

Speicherung von Bäumen

Bäume

Wiederholung: Binäre Suche

- + Halbierung des Suchraums in jedem Schritt
- Liste muss sortiert sein
- Nachträgliches Sortieren ist keine Option (**zu langsam**)

Idee: Elemente direkt an der richtigen Stelle einfügen.

- ▶ Bei Arrays zwei Möglichkeiten:
 1. Richtige Stelle suchen und dann Elemente verschieben.
 2. Vertauschungen wie z.B. bei Insertionsort.
- ▶ Bei verketteten Listen kann direkt eingefügt werden.

Bäume

Gesucht: Datenstruktur für effizientes Einfügen von Elementen

- ▶ Kein Verschieben von Elementen
- ▶ Suchraum sollte mit jedem Schritt halbiert werden.

Idee:

- ▶ Pointerstruktur wie bei verketteten Listen.
- ▶ Jedes Element hat zwei Nachfolger:
 1. kleinere Elemente
 2. größere Elemente
- ▶ i.W. immer noch eine verkettete Liste
 - ▶ Struktur reflektiert das Verhalten der Suche.

Bäume

Definition (Graph)

Ein **Graph** ist ein Tupel (V, E) mit folgenden Eigenschaften:

- ▶ V ist eine Menge von **Knoten**.
- ▶ $E \subseteq V \times V$ ist eine Menge von **Kanten**.

Intuition:

- ▶ Knoten sind zu ordnende **Objekte** (Datensätze).
- ▶ Kanten sind **Verweise** zwischen den Knoten (meist **Pointer**).
- ▶ Unterscheidung: **gerichtete** und **ungerichtete** Graphen
 - ▶ Bei ungerichteten Graphen haben Kanten keine Richtung.
 - ▶ Zu jeder Kante gibt es eine Kante in die Rückrichtung.

Bäume

Definition (Baum)

Ein **Baum** ist ein gerichteter Graph mit folgenden Eigenschaften:

- ▶ Jedes Element hat höchstens einen Vorgänger.
- ▶ Es gibt genau ein Element ohne Vorgänger (die **Wurzel**).

Anders ausgedrückt:

„Ein Baum ist ein **zusammenhängender gerichteter azyklischer Graph**, bei dem jedes Element höchstens einen Vorgänger hat.“

Definition (Binärbaum)

Ein **Binärbaum** ist ein Baum, bei dem jedes Element höchstens zwei Nachfolger hat.

Bäume

Sprechweise

- ▶ Nachfolger eines Knotens heißen **Kinder**.
- ▶ Ein Knoten ohne Kinder ist ein **Blatt**.
- ▶ Kinder werden meist in **linke** und **rechte** Kinder unterteilt.
- ▶ Ein Kind eines Knotes ist die Wurzel eines **Teilbaums**.

Themenüberblick – Bäume

Bäume

Binäre Suchbäume

Balancierte Bäume

Heaps

Speicherung von Bäumen

Binäre Suchbäume

Definition (Binärer Suchbaum)

Ein **binärer Suchbaum** ist ein Binärbaum, für den gilt:

- ▶ Jedem Knoten ist ein **Schlüssel** zugeordnet.
 - ▶ Auf den Schlüsseln ist eine **totale Ordnung** definiert.
 - ▶ D.h. man kann sie vergleichen.
- ▶ Für jeden Knoten gilt die **In-Order-Eigenschaft**:
 - ▶ Die Elemente des linken Teilbaums sind kleiner.
 - ▶ Die Elemente des rechten Teilbaums sind größer.

Motivation

- ▶ Oft als **Wörterbücher** verwendet.
 - ▶ Suche nach einem **Schlüssel** liefert dazugehörigen **Wert**.
- ▶ Stichwörter in Programmiersprachen:
 - ▶ Map, assoziatives Array, Dictionary, Key-Value-Paare

Binäre Suchbäume

Suchen von Elementen

Ansatz: Wie bei der binären Suche.

- ▶ Steige bei der Suche in den Baum hinab.
- ▶ Gehe jeweils nach links oder rechts, wenn der gesuchte Wert kleiner oder größer als der aktuelle Knoten ist.
- ▶ Ergebnis ist der Wert zum gesuchten Schlüssel.

Algorithmus:

1. Starte bei Wurzel.
2. Falls aktueller Knoten leer: **NICHT GEFUNDEN**.
3. Vergleiche gesuchten Wert mit Wert des aktuellen Knotens:
 - ▶ Falls gleich: **GEFUNDEN**.
 - ▶ Falls kleiner: Fahre bei linkem Teilbaum fort.
 - ▶ Falls größer: Fahre bei rechtem Teilbaum fort.

Binäre Suchbäume

Anzeige der Elemente im Baum

- ▶ Ein Baum ist der Vorstellung nach immer noch eine Liste.
- ▶ In welcher Reihenfolge werden die Elemente angezeigt?
- ▶ Anzeige in natürlicher Sortierung: **In-Order-Durchlauf**.
 - ▶ Rekursiver Abstieg in den Baum.
 - ▶ Die Wurzel wird **zwischen** den Knoten des linken und des rechten Teilbaumes angezeigt.
- ▶ Alternativen: **Pre-** oder **Post-Order-Durchlauf**.
 - ▶ Die Wurzel wird **vor** bzw. **nach** den Knoten des linken und des rechten Teilbaumes angezeigt.

Binäre Suchbäume

Einfügen von Elementen

Ansatz: Fast wie bei der Suche.

- ▶ Steige in den Baum ab, bis der Knoten als linkes oder rechtes Kind angehängt werden kann.

Algorithmus:

1. Starte bei Wurzel.
2. Falls **aktueller Knoten leer**, füge neuen Datensatz hier ein.
3. Vergleiche neuen Wert mit Wert des aktuellen Knotens:
 - ▶ Falls kleiner: Fahre bei linkem Teilbaum fort.
 - ▶ Falls größer: Fahre bei rechtem Teilbaum fort.

Binäre Suchbäume

Löschen eines Elements

Ansatz: Suche den zu löschenden Wert und „überbrücke“ Pointer darauf ähnlich wie bei einer verketteten Liste.

- ▶ Problem: Der gelöschte Knoten könnte **zwei** Kinder haben.
- ▶ Einfache Lösung: Lösche den gesamten Teilbaum und füge die Kinder nacheinander wieder ein.
- ▶ Besser: Suche **In-Order-Nachfolger** des gelöschten Knotens und setze diesen stattdessen ein.

Themenüberblick – Bäume

Bäume

Binäre Suchbäume

Balancierte Bäume

Heaps

Speicherung von Bäumen

Balancierte Bäume

Zusammenfassung: Eigenschaften von Suchbäumen

- ▶ Datenstruktur zum effizienten Speichern von Listen.
- ▶ geordnete Speicherung von Werten:
 - ▶ Linker Teilbaum enthält kleinere Werte als die Wurzel.
 - ▶ Rechter Teilbaum enthält größere Werte als die Wurzel.
- ▶ Neue Werte werden direkt an der richtigen Stelle eingefügt.
- ▶ Dadurch schnelles Suchen, Einfügen und Löschen von Werten.

Problem: Bäume können aus der Balance geraten.

- ▶ Neue Elemente werden ggf. nur auf einer Seite angehängt.
- ▶ Der Baum wird zu einer einfach verketteten Liste.
- ▶ Man spricht von einem **entarteten Baum**.

Balancierte Bäume

Definition (Tiefe eines Knotens in einem Baum)

Die **Tiefe** eines Knotens ist die Länge des Pfades bis zur Wurzel.

- ▶ Die Wurzel hat Tiefe 0.
- ▶ Die Kinder der Wurzel haben Tiefe 1.
- ▶ ...

Definition (Höhe eines Baumes)

Die **Höhe** eines Baumes ist die maximale Länge eines Pfades von der Wurzel bis zu einem Blatt.

- ▶ alternativ: Die Höhe ist die maximale Tiefe eines Knotens.

Balancierte Bäume

Definition (Balancierter Baum)

Ein Baum ist **balanciert**, wenn für jeden Knoten gilt, dass sich die Höhe des linken und rechten Teilbaumes höchstens um ein bestimmtes Verhältnis unterscheiden.

- ▶ Dafür muss der Baum ggf. nach Einfügen oder Löschen eines Elements **reorganisiert** werden.
- ▶ Hilfreiches Maß: **Balancefaktor** bf eines Knotens k .
 - ▶ $bf(k)$ ist die Differenz zwischen der Höhe des rechten Teilbaumes und der Höhe des linken Teilbaumes.
 - ▶ $bf(k) = h(\text{rechtes Kind}) - h(\text{linkes Kind})$

Aufgabe: Entwerfen Sie Algorithmen, die ...

1. ...die Tiefe eines Knotens in einem Baum bestimmen.
2. ...die Höhe eines Baumes bestimmen.
3. ...den Balancefaktor jedes Knotens ausgeben.

Balancierte Bäume

Definition (AVL-Baum)

Ein **AVL-Baum** ist ein binärer Suchbaum, bei dem der Balancefaktor jedes Knotens im Bereich $\{-1, 0, 1\}$ liegt.

Erhaltung der AVL-Eigenschaft

- ▶ Beim Einfügen oder Löschen kann die Eigenschaft verloren gehen.
- ▶ Der Baum (oder ein Teilbaum) muss **rotiert** werden.
- ▶ Intuitiv: **nach rechts Rotieren** bedeutet, die Wurzel in den rechten Teilbaum zu verschieben und eine neue Wurzel aus dem linken Teilbaum zu holen.

Balancierte Bäume

Rotationsarten (Einfachrotationen)

- ▶ Links-Rotation:
 - ▶ Wurzel wird in den linken Teilbaum abgesenkt.
 - ▶ Rechtes Kind wird die neue Wurzel.
 - ▶ Linkes Kind der neuen Wurzel wird zum rechten Kind der alten.
- ▶ Rechts-Rotation:
 - ▶ Wurzel wird in den rechten Teilbaum abgesenkt.
 - ▶ Linkes Kind wird die neue Wurzel.
 - ▶ Rechtes Kind der neuen Wurzel wird zum linken Kind der alten.

Diese beiden Rotationen stellen die Balance wieder her, wenn das Ungleichgewicht **ganz außen** im Baum ist.

Balancierte Bäume

Ungleichgewichtssituationen

Wir unterscheiden, auf welcher Seite des Baumes das Ungleichgewicht besteht:

Links-Links

- ▶ Balancefaktoren der Wurzel und des linken Kindes negativ.
- ▶ Balance wird durch Rechtsrotation wieder hergestellt.

Rechts-Rechts

- ▶ Balancefaktoren der Wurzel und des rechten Kindes positiv.
- ▶ Balance wird durch Linkssrotation wieder hergestellt.

Balancierte Bäume

Ungleichgewichtssituationen

Entsprechend gibt es noch die Situationen *Links-Rechts* und *Rechts-Links*:

Links-Rechts

- ▶ Balancefaktor der Wurzel negativ.
- ▶ Balancefaktor des linken Kindes positiv.
- ▶ Balance wird durch **Links-Rechts-Rotation** wieder hergestellt:
 1. Linksrotation durch das linke Kind.
 2. Rechtsrotation durch die Wurzel.

Balancierte Bäume

Ungleichgewichtssituationen

Entsprechend gibt es noch die Situationen *Links-Rechts* und *Rechts-Links*:

Rechts-Links

- ▶ Balancefaktor der Wurzel positiv.
- ▶ Balancefaktor des rechten Kindes negativ.
- ▶ Balance wird durch **Rechts-Links-Rotation** wieder hergestellt:
 1. Rechtsrotation durch das rechte Kind.
 2. Linksrotation durch die Wurzel.

Balancierte Bäume

Implementierung von AVL-Bäumen

- ▶ Einfügen und Löschen wie bisher.
- ▶ Dabei zusätzlich Balancefaktoren berechnen.
- ▶ Sobald ein Knoten mit Balancefaktor -2 gefunden wird, linkes Kind prüfen:
 - ▶ Kind hat Balancefaktor -1 : Rechtsrotation
 - ▶ Kind hat Balancefaktor $+1$: Links-Rechts-Rotation
- ▶ Sobald ein Knoten mit Balancefaktor $+2$ gefunden wird, rechtes Kind prüfen:
 - ▶ Kind hat Balancefaktor $+1$: Linksrotation
 - ▶ Kind hat Balancefaktor -1 : Rechts-Links-Rotation

Balancierte Bäume

Weitere Idee: Abschwächung des AVL-Prinzips

- ▶ Keine perfekte, sondern näherungsweise Balancierung.
- ▶ Idee: Knoten in rot oder schwarz einfärben.
 - ▶ Baum ist balanciert, wenn man nur schwarze Knoten betrachtet.
 - ▶ Anzahl der roten Knoten ist begrenzt.
- ▶ Vorteil: Es muss nicht jedes Mal neu balanciert werden.

Definition (Rot-Schwarz-Bäume)

Ein **Rot-Schwarz-Baum** ist ein Binärbaum, bei dem jeder Knoten eine Farbe (Rot oder Schwarz) hat.

- ▶ Jedes Blatt ist schwarz.
- ▶ Ein roter Knoten hat nur schwarze Kinder.
- ▶ Jeder Pfad von einem Knoten zu seinen Blättern hat die gleiche Anzahl schwarzer Knoten.

Balancierte Bäume

Optimierung des Suchbaumprinzips: B-Bäume

Bei einem **B-Baum** kann ein Knoten mehr als zwei Kinder haben und mehr als einen Schlüssel tragen.

- ▶ Trägt der Knoten n Schlüssel, so hat er $n + 1$ Kinder.
- ▶ Kind 0 enthält Werte, die kleiner sind als der erste Schlüssel.
- ▶ Kind 1 enthält Werte, die zwischen erstem und zweitem Schlüssel liegen usw.

Eigenschaften

- ▶ Die Anzahl der Schlüssel pro Knoten ist variabel
 - ▶ Meist zwischen n und $2n$ für vorgegebene Zahl n .
- ▶ Alle Blätter haben die gleiche Tiefe.
 - ▶ ggf. Zusatzschlüssel in inneren Knoten benutzen.

B-Bäume sind eine typische Datenstruktur in Datenbanken und Dateisystemen.

Themenüberblick – Bäume

Bäume

Binäre Suchbäume

Balancierte Bäume

Heaps

Speicherung von Bäumen

Heaps

Bisheriger Ansatz: Bäume als Listen

- + Suchbaumeigenschaft garantiert korrekte Sortierung.
- + Balancierungsoperationen für schnellen Zugriff.
- Problem: Selbst AVL-Bäume können noch unnötig hoch werden.

Alternatives Gütekriterium: Vollständigkeit

- ▶ Versuche, den Baum möglichst perfekt zu balancieren.
- ▶ Verzichte dafür auf korrekte Sortierung.
 - ▶ Baum sollte immer noch partiell sortiert sein.

Heaps

Definition (vollständiger Binärbaum)

Ein vollständiger Binärbaum ist ein Binärbaum, bei dem alle Ebenen voll besetzt sind.

- ▶ Ausnahme: Die unterste Ebene muss nicht vollständig sein. In diesem Fall sind die Knoten von links durchgehend besetzt.

Intuition

- ▶ Jeder Knoten (außer den Blättern) hat zwei Kinder.
- ▶ Vollständige Bäume sind der Idealfall:
Perfekt balanciert und minimale Suchtiefe.

Heaps

Definition (Heap)

Ein **Heap** ist ein vollständiger Binärbaum, bei dem der Wert jedes Knotens kleiner ist als der seiner Kinder.

Beobachtungen

- ▶ Die Wurzel ist das kleinste Element.
- ▶ Der Baum ist **partiell sortiert**:
Beim Absteigen werden die Elemente größer.

Alternative Definitionen:

- ▶ Ein Heap wie oben definiert heißt **min-Heap**.
- ▶ **Max-Heap**: Die Wurzel ist **größer** als ihre Kinder.

Heaps

Verwendung von Listen:

- ▶ Oft muss nicht auf alle Elemente einer Liste schnell zugegriffen werden.
 - ▶ Aufgabenlisten mit Deadlines
 - ▶ Priorisierung von Datenverkehr oder Aufgaben
 - ▶ Suchalgorithmen (z.B. in Navigationssystemen)
- ▶ Nur das Element mit dem geringsten oder höchsten Wert wird sofort gebraucht.
- ▶ Solche Datenstrukturen können effizient mit Heaps implementiert werden.

Heaps

Definition (Priority Queue)

Eine **Priority Queue** ist ein abstrakter Listen-Datentyp, bei dem jedem Element eine Priorität zugeordnet wird. Operationen:

- ▶ insert fügt ein neues Element ein.
- ▶ pop liefert das Element mit der niedrigsten/höchsten Priorität und entfernt es aus der Liste.
- ▶ Optional: peek bzw. top liefert das höchste Element, ohne es zu entfernen.

Heaps

Heaps als Priority Queues

- ▶ Höchstes bzw. niedrigstes Element steht in Wurzel.
- ▶ Kann ohne jeden Aufwand gefunden werden.
- + Sehr gut als Priority Queue geeignet.

Wie funktionieren `insert` und `pop`?

- ▶ Idee: Elemente können – ähnlich wie bei Bubblesort – durch Vertauschungen auf- und absteigen.
- ▶ `insert`:
 1. Füge neues Element am Ende ein.
 2. Lasse Knoten aufsteigen, bis er richtig einsortiert ist.
- ▶ `pop`:
 1. Ersetze Wurzel durch letztes Element.
 2. Lasse neue Wurzel absinken, bis sie richtig einsortiert ist.

Heaps

Aufsteigen von Knoten: Die Operation **Heapify-Up**

$k :=$ Neuer Knoten

while $k < \text{wurzel}(k)$ **do**

 Vertausche k mit $\text{wurzel}(k)$

- Wird auch **Bubble-Up** genannt.

Einsinken von Knoten: Die Operation **Heapify-Down**

$k :=$ Wurzel

while $k >$ eines der Kinder **do**

 Vertausche k mit kleinerem Kind

- Wird auch **Bubble-Down** genannt.

Heaps

Zusammenfassung: Heaps als Priority Queues

- ▶ Niedrigstes/höchstes Element wird sofort gefunden.
- ▶ insert und pop in logarithmischer Zeit möglich.
- ▶ Ähnlich gute Eigenschaften wie Suchbäume.
- + Heap ist immer vollständig.
 - ▶ Einfügen und Löschen sogar etwas schneller.
- Kein geordneter Durchlauf durch den Baum möglich.
 - ▶ Ungeeignet, wenn Elemente sortiert angezeigt werden sollen.

Heaps

Heaps als Sortierhilfe

- ▶ Erinnerung: Selectionsort
 1. Suche und entferne kleinstes Element aus alter Liste.
 2. Füge Element am Ende in neue Liste ein.
- ▶ Optimierung: Speichere alte Liste als Heap.
 - + Schnelleres Auffinden des kleinsten Elements.
- ▶ Resultat: Sortiervorgang wird ähnlich schnell wie Quicksort oder Heapsort.

Heaps

Heapsort

Baue Heap aus Elementen der Liste.

while Heap nicht leer **do**

 Hänge kleinstes Element aus Heap an Liste an.

 Entferne kleinstes Element aus Heap.

Themenüberblick – Bäume

Bäume

Binäre Suchbäume

Balancierte Bäume

Heaps

Speicherung von Bäumen

Speicherung von Bäumen

Vollständige Bäume können sehr effizient gespeichert werden.

- ▶ Knoten durchnummerieren:
 - ▶ Wurzel hat die Nummer 0.
 - ▶ Hat ein Knoten die Nummer n ,
so haben seine Kinder $2n + 1$ und $2n + 2$.
- ▶ Ein vollständiger Binärbaum mit n Knoten kann in einem Array der Länge n gespeichert werden.
- ▶ Dadurch kann z.B. Heapsort **in-place** sortieren.
 - ▶ **Achtung:** Um den Heap nicht im Array verschieben zu müssen, ist es besser, einen Max-Heap zu verwenden und die Liste vom Ende her aufzubauen.