

# Algorithmen und Datenstrukturen I

## Treaps

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik  
Hochschule München

Letzte Änderung: 18.03.2018 18:16

### Inhaltsverzeichnis

Idee . . . . .	1
Treaps . . . . .	1
Bedingungen . . . . .	2
Beispiel . . . . .	3
Suchen und Einfügen . . . . .	3
Entfernen . . . . .	4
Treaps mit zufälligen Prioritäten . . . . .	4
Literaturhinweis . . . . .	4

### Idee

- fügt man  $N$  Schlüssel der Reihe in einen zunächst leeren Suchbaum ein, kann ein degenerierter Suchbaum entstehen
- unter den  $N!$  möglichen Anordnungen von  $N$  Schlüsseln tritt das nicht allzu häufig auf
- daher sind die Erwartungswerte für einen *zufällig* erzeugten Binärbaum mit  $N$  Schlüsseln nur von der Größenordnung  $\mathcal{O}(\log N)$

### Treaps

- geg. Menge  $S$  von Objekten mit der Eigenschaft, jedes  $x \in S$  hat zwei Komponenten

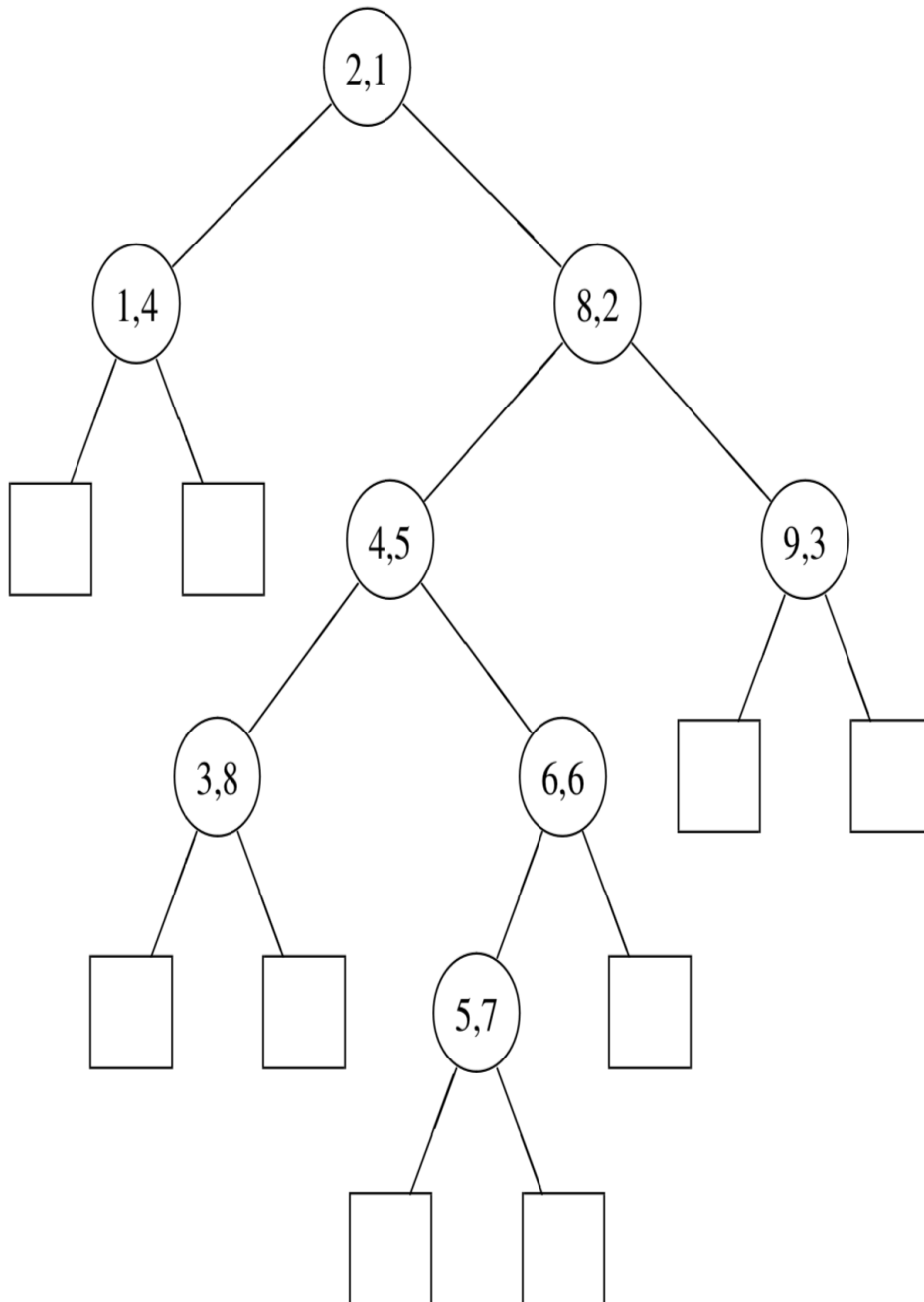
- Schlüsselkomponente  $x.key$  und
- Prioritätskomponente  $x.priority$
- ein **Treap** ist
  - ein binärer Suchbaum für die Schlüsselkomponente
  - ein Min-Heap für die Prioritätskomponente

## Bedingungen

für jeden Knoten  $p$ , der das Element  $x$  speichert, gelten die beiden folgenden Bedingungen:

**Suchbaumbedingung:** für jedes Element  $y$  im linken Teilbaum von  $p$  gilt  $y.key \leq x.key$  und für jedes Element  $y$  im rechten Teilbaum von  $p$  gilt  $x.key \leq y.key$

**Heapbedingung:** für jedes in einem Nachfolger von  $p$  gespeicherte Element  $z$  gilt  $x.priority \leq z.priority$

**Beispiel****Suchen und Einfügen**

- Feststellung: für jede Menge
- Suche wie im Suchbaum

- Einfügen zunächst wie im Suchbaum
- passt die Heapbedingung nicht für das neu eingefügte Element
  - durch Rotation nach links oder rechts solange nach oben bewegen, bis die Heapbedingung wieder gilt

## Entfernen

- durch Rotationen das zu entfernende Element  $x$  solange abwärts bewegen,
- bis beide Nachfolger Blätter sind
- Entscheidung wie zu rotieren ist, hängt von den beiden Nachfolgern ab
  - das Element mit der kleineren Priorität muss nach oben wandern

## Treaps mit zufälligen Prioritäten

- ein **randomisierter Suchbaum** für eine Menge  $S$  von Schlüsseln ist ein Treap
  - mit Schlüsseln aus  $S$  und
  - unabhängig und gleich verteilt zufällig gewählten Prioritäten
- keine zwei Schlüssel sollen gleiche Priorität erhalten
- geordnet nach wachsenden Prioritäten soll jede Permutation der Schlüssel gleich wahrscheinlich sein
- Prioritäten, z.B. erzeugt durch Zufallszahlengenerator, sollten vor dem Benutzer verborgen bleiben
  - sonst könnte er durch “einseitige” Wahl von Schlüsseln (und Prioritäten) doch degenerierte Bäume erzeugen

## Literaturhinweis

Die Graphen in diesem Kapitel sind aus dem Buch *Algorithmen und Datenstrukturen* von Ottmann & Widmayer.