



## Übung zur Vorlesung „Datenbanksysteme für Hörer anderer Fachrichtungen“

Richard Kuntschke (richard.kuntschke@in.tum.de)

### Lösungen zu Blatt 6

#### Aufgabe 1

Obere Schranke für die Höhe:

Tiefe	minimale Knotenzahl in einer Tiefe
0	1
1	2
2	$2 \cdot (k + 1)$
3	$2 \cdot (k + 1)^2$
$\vdots$	$\vdots$
$h$	$2 \cdot (k + 1)^{h-1}$

Gesamtsumme der Knoten in einem B-Baum:

$$\begin{aligned} 1 + \sum_{i=1}^h 2 \cdot (k + 1)^{i-1} &\stackrel{\text{geometrische Reihe}}{=} 1 + 2 \frac{(k + 1)^h - 1}{k + 1 - 1} \\ &= 1 + 2 \frac{(k + 1)^h - 1}{k} \end{aligned}$$

Minimale Anzahl der Elemente im Baum:

$$\begin{aligned} 1 + k \cdot 2 \cdot \frac{(k + 1)^h - 1}{k} &= 1 + 2(k + 1)^h - 2 \\ &= 2(k + 1)^h - 1 \end{aligned}$$

Es gilt also:

$$\begin{aligned} 2(k + 1)^h - 1 &\leq n \\ 2(k + 1)^h &\leq n + 1 \\ (k + 1)^h &\leq \frac{n + 1}{2} \\ h &\leq \log_{k+1} \left( \frac{n + 1}{2} \right) \end{aligned}$$

**Untere Schranke für die Höhe:**

Tiefe	maximale Knotenzahl in einer Tiefe
0	1
1	$2k + 1$
2	$(2k + 1)^2$
3	$(2k + 1)^3$
$\vdots$	$\vdots$
$h$	$(2k + 1)^h$

Maximale Anzahl der Knoten im Baum:

$$\begin{aligned} \sum_{i=0}^h (2k + 1)^i &\stackrel{\text{geometrische Reihe}}{=} \frac{(2k + 1)^{h+1} - 1}{2k + 1 - 1} \\ &= \frac{(2k + 1)^{h+1} - 1}{2k} \end{aligned}$$

Maximale Elementanzahl im Baum:

$$2k \cdot \frac{(2k + 1)^{h+1} - 1}{2k} = (2k + 1)^{h+1} - 1$$

Es gilt also:

$$\begin{aligned} n &\leq (2k + 1)^{h+1} - 1 \\ n + 1 &\leq (2k + 1)^{h+1} \\ \log_{2k+1}(n + 1) &\leq h + 1 \\ h &\geq \log_{2k+1}(n + 1) - 1 \end{aligned}$$

## Aufgabe 2

Obere Schranke für die Worst-Case Komplexität eines optimalen Verfahrens, das die Boxengröße minimiert:

Vollständiges Durchsuchen aller Möglichkeiten, die Elementmenge in zwei (nicht leere) Teilmengen aufzuspalten:

$$\text{Anzahl Möglichkeiten: } 2^n - 2$$

Komplexität dieses Verfahrens:  $O(2^n)$

Es ist klar, dass dieses Verfahren die optimale Lösung finden muss, da alle Möglichkeiten abgesehen werden.

**Frage:** Ist diese Komplexität dann auch die worst-case Komplexität?

Generell gilt, dass die Partitionierung eines gewichteten Graphen in  $k$  Teilmengen ( $k \geq 2$ ) NP-vollständig ist. Unser Problem entspricht genau dieser Partitionierung ( $k = 2$ ), so dass die

angegebene Komplexität (wohl) der worst-case Komplexität entspricht.

**Festlegungen:**

$m$  ist die Minimalanzahl der Einträge in einem Knoten.

$M$  ist die Maximalanzahl der Einträge in einem Knoten.

Splitalgorithmus:

- Für jede Dimension  $d$ 
  - Sortiere Einträge anhand der unteren Grenze des Eintrags bezüglich  $d$ :  
 $a_1, \dots, a_{M+1}$
  - Bilde zwei Gruppen mit den  $m$  größten bzw. kleinsten Werten:  
 $a_1, \dots, a_m$  und  $a_{M+1-(m-1)}, \dots, a_{M+1}$
  - Für jedes  $k \in \{1, \dots, M+1-2m\}$  bilde

$$\underbrace{a_1, \dots, a_{m+k}}_{lp} \quad \text{und} \quad \underbrace{a_{m+k+1}, \dots, a_{M+1}}_{hp}$$

\* Falls  $quality(lp, hp)$  den aktuell geringsten Wert aller bisher getesteten Partitionierungen hat, so merkt man sich  $(lp, hp)$ .

- Gib das zuletzt aktuelle  $(lp, hp)$  als Ergebnis aus.

Die Funktion  $quality$  bewertet die Partitionierung und es sind mehrere Implementierungen denkbar:

- $quality(lp, hp) = area(bb(lp)) + area(bb(hp))$   
hierbei bezeichnet  $bb$  die Funktion, die zu einer gegebenen Menge von Einträgen die bounding box bestimmt. Mit dieser Qualitätsfunktion wird also versucht die Überdeckung zu minimieren.
- $quality(lp, hp) = area(bb(lp) \cap bb(hp))$   
Mit dieser Qualitätsfunktion wird versucht die Überlappung zu minimieren.

Die hier vorgestellten Splitalgorithmen entstammen der Arbeit zu  $R^*$ -Bäumen („The  $R^*$ -tree: An Efficient and Robust Access Method for Points and Rectangles“, Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, Bernhard Seeger, in ACM SIGMOD 1990).