# Zusammenfassung: Prioritätssuchbaum

Elmar Langetepe
University of Bonn

# k-dimensionaler Range Tree

Theorem 3.11: Ein $k$-dimensionaler Bereichsbaum für $n$ Punkte im $\mathbb{R}^k$ kann in Zeit $O(n(\log n)^{k-1})$ mit Platz $O(n(\log n)^{k-1})$ aufgebaut werden. Eine Bereichsanfrage mit Hyperrechteck $q \subset \mathbb{R}^k$ kann in Zeit $O(a + (\log n)^k)$ beantwortet werden. Dabei ist $a$ die Größe der Antwort.

Speicherplatz Punkt $p = (x_1, x_2, \ldots, x_k)$

- Baum $T^1$:
  1) Baum $T^1$ einmal im Blatt
  2) In $\log n$ vielen Intervallen $I(v_i)$ von $T^1$
- Induktiv, Aufsummieren!

# k-dimensionaler Range Tree

Theorem 3.11: Ein $k$-dimensionaler Bereichsbaum für $n$ Punkte im $\mathbb{R}^k$ kann in Zeit $O(n(\log n)^{k-1})$ mit Platz $O(n(\log n)^{k-1})$ aufgebaut werden. Eine Bereichsanfrage mit Hyperrechteck $q \subset \mathbb{R}^k$ kann in Zeit $O(a + (\log n)^k)$ beantwortet werden. Dabei ist $a$ die Größe der Antwort.

Query! $q = (I_1, I_2, \ldots, I_k)$

- $T^1$: Intervalle $I(v_i)$ die $I_1$ ausschöpfen max. $2 \log n$ viele
- Induktiv: $O((\log n_i)^{k-1} + a_i)$ für $T^{k-1}_{v_i}$
- Aufsummieren!

# k-dimensionaler Range Tree

Theorem 3.11: Ein $k$-dimensionaler Bereichsbaum für $n$ Punkte im $\mathbb{R}^k$ kann in Zeit $O(n(\log n)^{k-1})$ mit Platz $O(n(\log n)^{k-1})$ aufgebaut werden. Eine Bereichsanfrage mit Hyperrechteck $q \subset \mathbb{R}^k$ kann in Zeit $O(a + (\log n)^k)$ beantwortet werden. Dabei ist $a$ die Größe der Antwort.

Beweis: Aufbau!

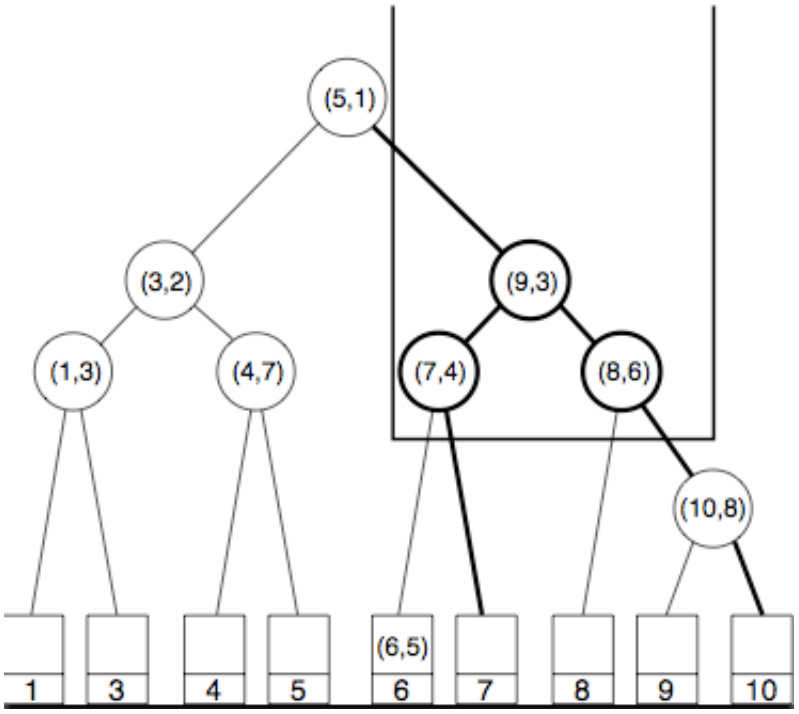
1. Sortieren: $k$ Listen $L_1, L_2, \ldots, L_k$, $O(kn \log n)$
2. Aufteilungsschritt $n_i$ für Knoten $v_i$
3. Aufbau, rekursiv!

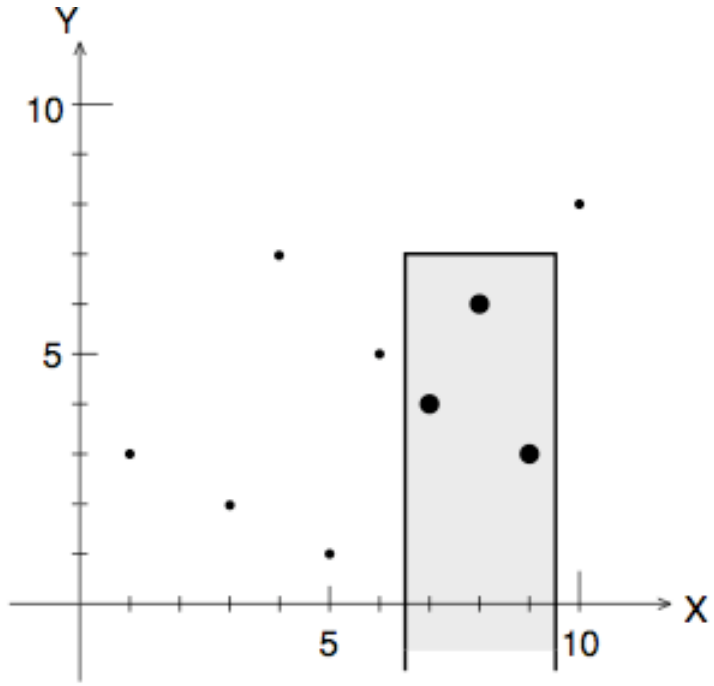
Induktiv: $O(n_i \log n_i)^{k-2})$ für $T_{v_i}^{k-1}$
Aufsummieren: $\sum_{i=1}^{2n-1} n_i \in O(n \log n)$

# Prioritätssuchbaum

- Einfache Struktur für Punkte in der Ebene
- Halbstreifenanfrage $H = [x_1, x_2] \times (-\infty, y]$
- Finde Punkte aus $D$ in $H$
- Platzoptimal und effizient!
- Eindimensionaler Bereichsbaum für $X$ Koordinaten
- Heap für $Y$-Koordinaten

# Prioritätssuchbaum Beispiel




- 1. Jeder Punkt auf dem Weg zu seiner $X$-Koordinate
- 2. Punkte entlang des Pfades nach $Y$-Koordinaten
- 3. So nah wie möglich an der Wurzel

# Ergebnis

Theorem 3.14 Ein Prioritätssuchbaum für $n$ Punkte in der Ebene kann in Zeit $O(n \log n)$ aufgebaut werden. Er benötigt $O(n)$ viel Platz. Eine Halbstreifenanfrage kann in Zeit $O(a + \log n)$ beantwortet werden. Dabei ist $a$ die Größe der Antwort.

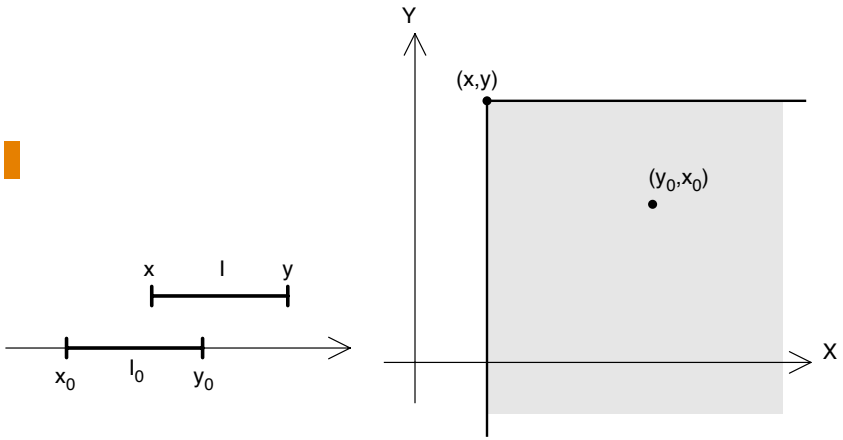
Beweis: Aufbau! Geht immer!

- $X$-sortiertes Skelett, nach aufsteigenden $Y$-Koordinaten einfügen
- Induktiv: Wurzel, Teilbäume $v_1$, $v_2$

Query

- $X$-Grenzen in $\log n$,
- Grenzen entlanggehen: $X$-Koord. im Innern ok!
- Nach $Y$-Koord. bei Tiefe $Y$ aufhören!

# Anwendung Schnittanfrage mit Intervallen




Lemma 3.16 Seien $I_0 = [x_0, y_0]$ und $I = [x, y]$ zwei Intervalle, dann gilt: $I_0$ überlappt mit $I \iff x \leq y_0$ und $x_0 \leq y$

# Ergebnis Schnittanfrage

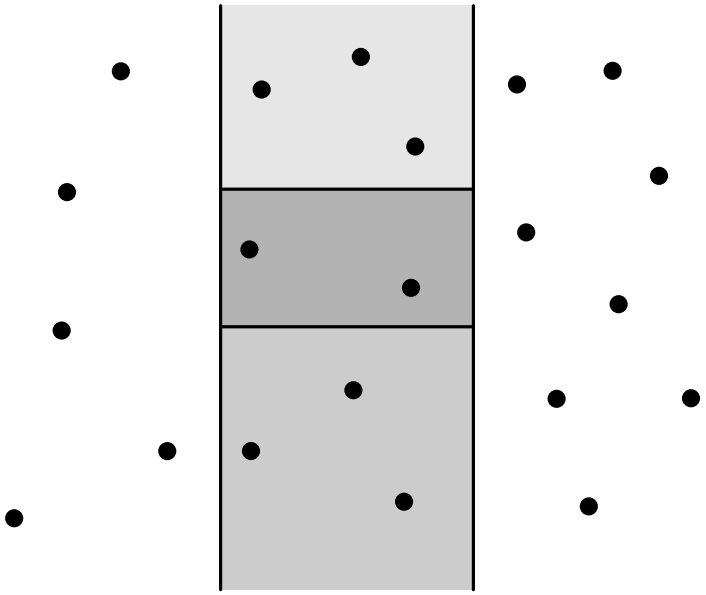
Theorem 3.17 Man kann $n$ Intervalle mit Platz $O(n)$ so abspeichern, dass sich eine Überlappungsanfrage eines Intervalls $I_0$ in Zeit $O(a + \log n)$ beantworten läßt. Dabei ist $a$ die Größe der Antwort.

Intervalle in Punkte übertragen.
Anfrage mit Viertelebene $[x_0, \infty) \times (-\infty, y_0]$.

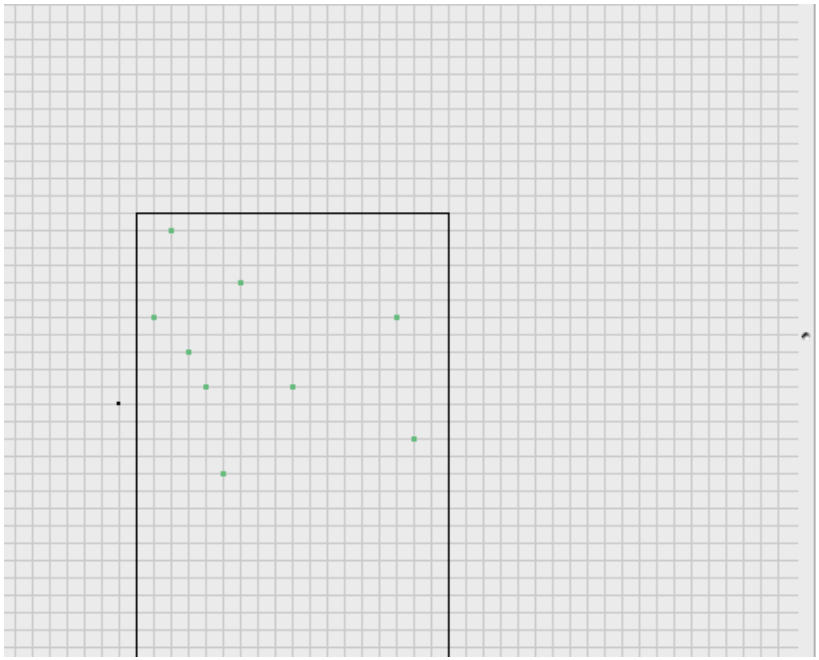
# Anwendung Rechteckanfrage

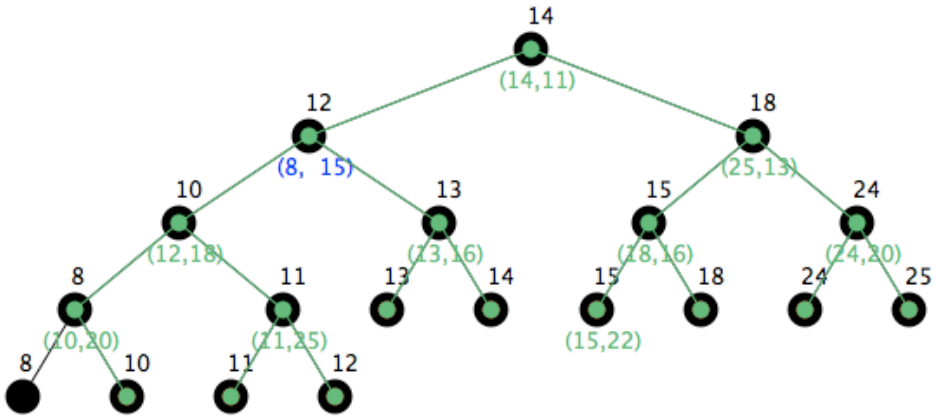
Zwei Halbstreifenanfragen ergeben eine Rechteckanfrage!

Problem: Nicht output-sensitiv!



# Beispiel: Prioritätssuchbaum Query Rechteck?



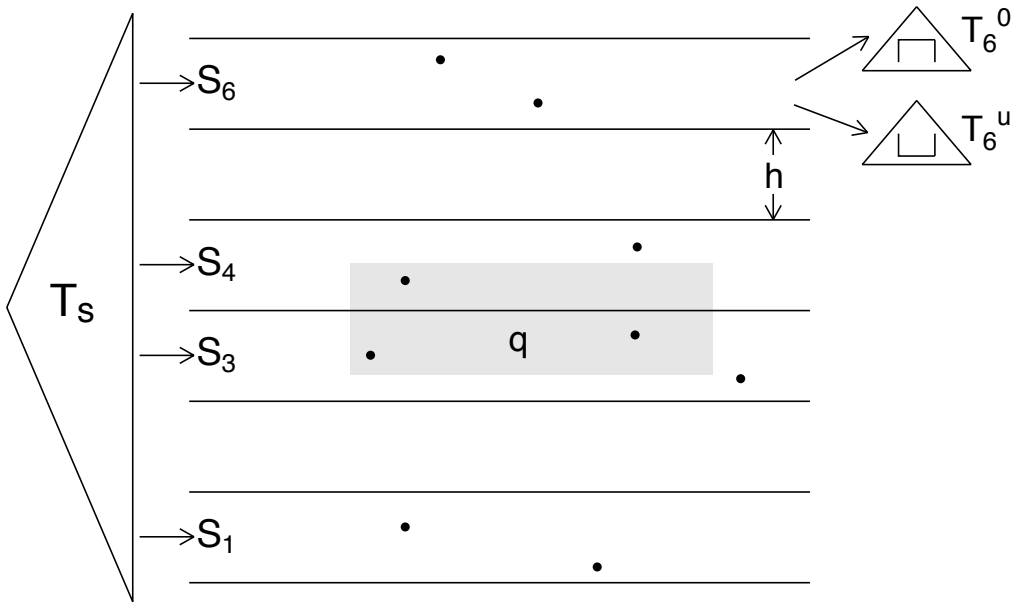



Anfrage: $[9, 27] \times (-\infty, 26]$, alle grünen Punkte

Rechteckanfrage: $[9, 27] \times [24, 26]$, nur Knoten (11,25) aber alle absuchen

# Anwendung Rechteckanfrage mit fester Höhe

Theorem 3.18 Sei $h > 0$ fest. $n$ Punkte in der Ebene lassen sich so mit Platz $O(n)$ abspeichern, dass jede Rechteckanfrage mit Höhe $h$ in Zeit $O(\log n + a)$ ($a$ Größe der Antwort) beantwortet werden können.




Beweis:

# Buch Kapitel

Kapitel 3.3.3 Seite 140 oben – S. 147 mitte