Segmentbäume

Sei $U = \{x_1, ..., x_N\}$ mit $x_1 < ... < x_N$
und S sei eine Menge von Intervallen mit Endpunkten $\in U$, wobei $|S| = n$.

Definition Segmentbaum

Ein Segmentbaum ist ein balancierter binärer Suchbaum für Intervalle
$(-\infty, x_1), [x_1, x_1], ..., (x_N, \infty)$.

Beispiel

Sei $U = \{1, 2, 3, 4, 5\}$. Dann ergibt sich folgendes Bild:

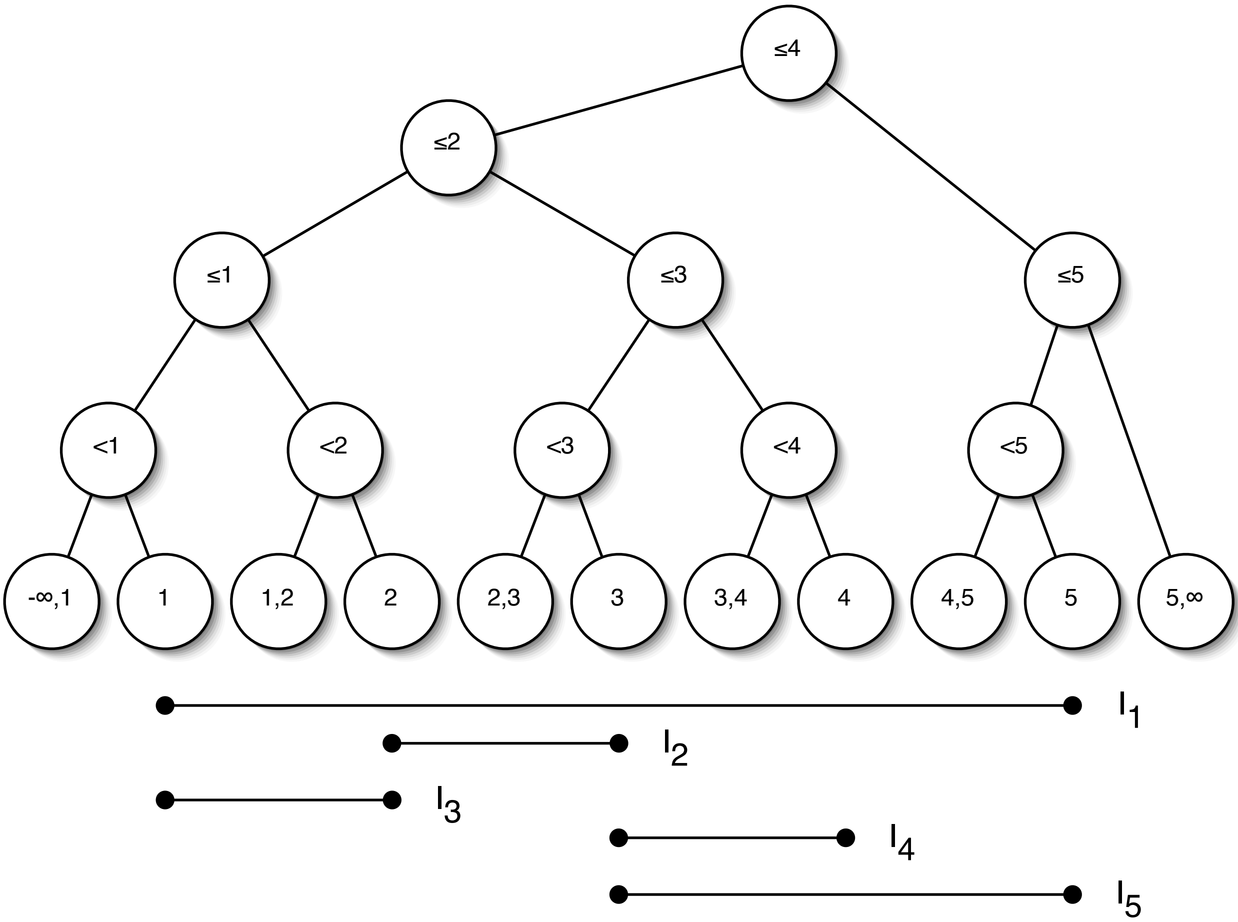

Abbildung 1: Beispiel für einen Segmentbaum

Lemma

Sei T Segmentbaum der Höhe h für S, $I \in S$. Dann gilt:
a) I gehört zu höchstens $2h$ Knotenlisten
b) $I = \dot{\bigcup}_{v\ mit\ I \in KL(v)} \quad x\_Bereich(v)$

Beweis

a) Betrachte alle Knoten einer festen Tiefe:

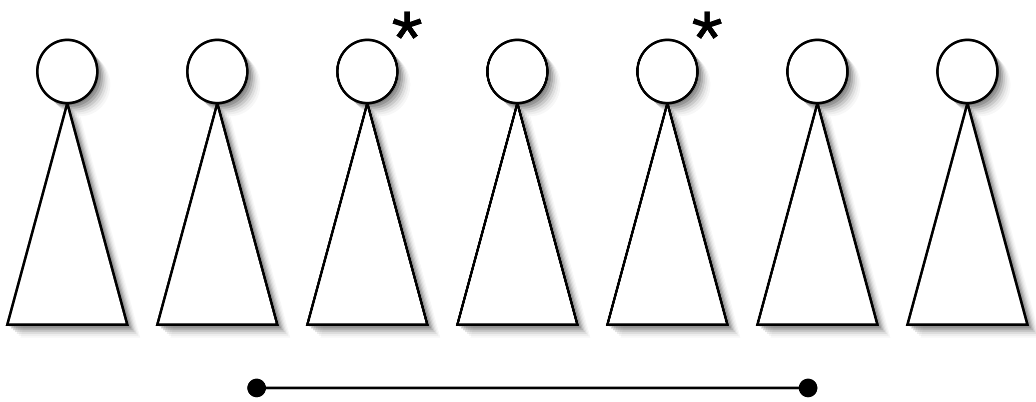

Abbildung 2: Knoten einer festen Tiefe mit Unterbäumen und Intervall I

Dann können nur Knoten mit Stern markiert sein. Damit kann I in höchstens $2h$ Knotenlisten enthalten sein.
b)
$\supseteq$: Ist klar nach Definition
$\subseteq$: Falls $x \in I$ ist, existiert ein Weg $v_0, v_1, ...$ bis zur Wurzel im Baum mit $x \in x\_Bereich(v_i)$, sei v oberster dieser Knoten, dessen $x-$Bereich noch ganz in I liegt $\Rightarrow I \in KL(v)$.

Konstruktion, Platzbedarf

Satz

Sei $U \in R$, $|U| = N$ und S eine Menge von n Intervallen mit Endpunkten aus U. Dann gilt:
a) Ein Segmentbaum für S kann in Zeit $O(N + n \log N)$ konstruiert werden und hat Platzbedarf $O(N + n \log N)$.
b) Gegeben $I \in S$, die Knoten v des Baumes mit $I \in KL(v)$ können in Zeit $O(\log n)$ gefunden werden.
c) Einfügen/Streichen von Intervallen in/aus S ist möglich in Zeit $O(g(n) \log N)$, wobei $g(n)$ die Zeit ist, die man benötigt, um Intervalle in einer Knotenliste einzufügen oder zu streichen.

Beweis

a) und b)
Baumstruktur kann in $O(N)$ Zeit aufgebaut werden, da U sortiert vorliegt. Die Knotenlisten erhalten die Knoten, die ein Intervall I enthalten, durch Suche nach den Endpunkten x_1, x_2 (genauer: $x_1 - \epsilon, x_2 + \epsilon$).

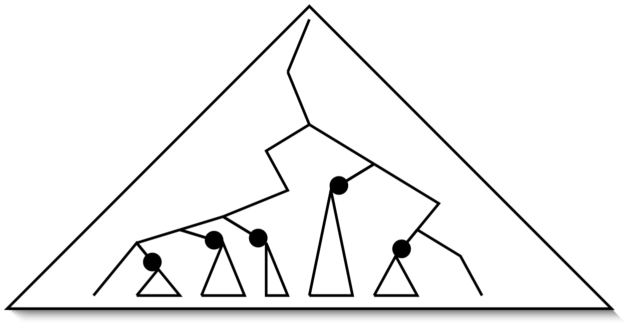

Abbildung 3: Wurzeln der aktiven Unterbäume

Konten, die Wurzeln der nach rechts vom linken Suchweg und nach links vom rechten Suchweg abgehenden Teilbäume sind, werden mit I markiert.
c) Analog zu b), um Knotenliste zu finden dort einfügen/streichen.

Anwendungsbeispiel

Gegeben sind m achsenparallele Rechtecke in der Ebene.

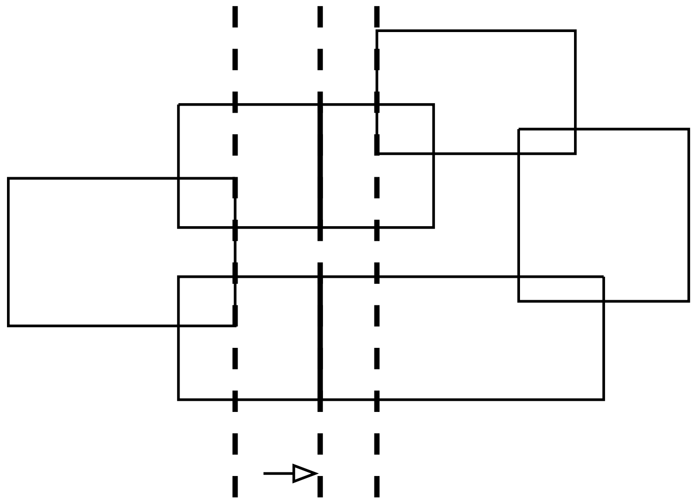

Abbildung 4: Berechnung der Vereinigung von Rechtecken mit Sweepline

Berechne hierzu die Fläche der Vereinigungen. Das geht in $\Theta(m^2)$ mit bruteforce

Sweeplinealgorithmus

Sei U die Menge der $y-$Koordinaten der Ecken der Rechtecke $\Rightarrow |U| \leq 2m$
Sweepline an einer bestimmten Stelle:
SLS: besteht aus Segmentbaum, der die Projektionen der geschnittenen Rechtecke in x-Richtung enthält.
Knotenliste: nur die Länge der Knotenliste ist abgespeichert. Zusätzlich merkt sich jeder Knoten v die Gesamtlänge $TL(v)$ aller Segmente, die von Intervallen überdeckt werden, die zu der Knotenliste von v oder einem seiner Nachfolger gehören.
Es gilt:

$$\mathrm{TL}(v) = \begin{cases} \text{Länge } x\_Bereich(v) & KL(v) \neq \emptyset \\ 0 & v \text{ Blatt und } KL(v) = \emptyset \\ TL(LKind(v)) + TL(RKind(v)) & \text{sonst} \end{cases}$$

$TL(Wurzel)$ = Länge der Vereinigung der Intervalle in S (für die aktuelle Sweepline)
Ereignispunkte: $x-$Koordinate der Ecken der Rechtecke
EPS: Heap, der sie enthält
Zwischen zwei Ereignispunkten:

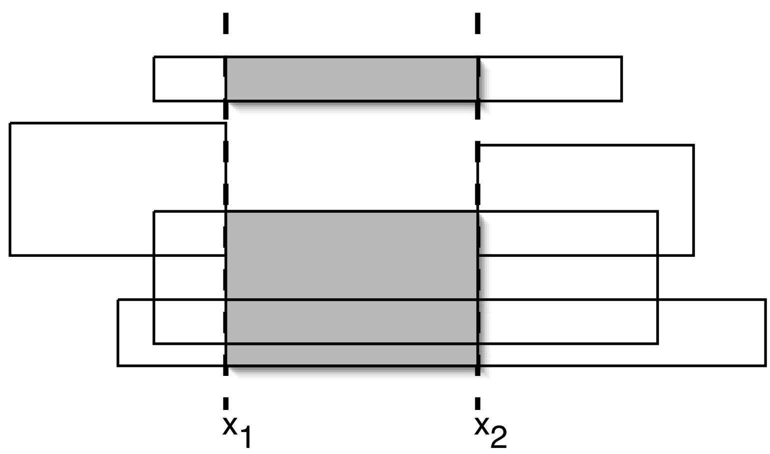

Abbildung 5: Fläche zwischen 2 Ereignispunkten

$TL(Wurzel) \cdot (x_2 - x_1)$ (Wurzel zur Zeit, wo Sweepline zwischen x_1 und x_2) zur bisherigen Fläche addieren. Dies läßt sich in $O(\log n)$ realisieren.
Einfügen eines Rechtecks R:
Finde in $O(\log m)$ Zeit alle Knoten v, die (Proj. von) R in ihren Knotenlisten haben (siehe Satz) und erhöhe ihre Zähler um 1. Falls der Zähler vorher 0 war, setze $TL(v)$ = Länge von $x\_Bereich(v)$. Aktualisiere TL für alle Vorfahren von solchen v's. d Das passiert auf zwei Wegen bis zur Wurzel also in $O(\log m)$.

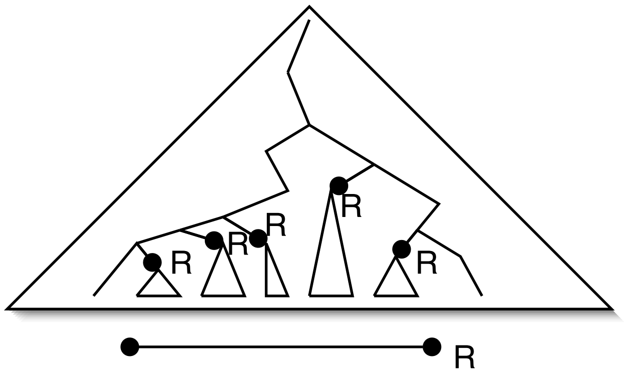

Abbildung 6: Knoten in Segmentbaum, in deren Knotenlisten R eingefügt wird

Das Entfernen von Rechtecken bzw. deren Projektion funktioniert analog.
Die Laufzeit des gesamten Algorithmus beträgt also $O(m \log m)$.