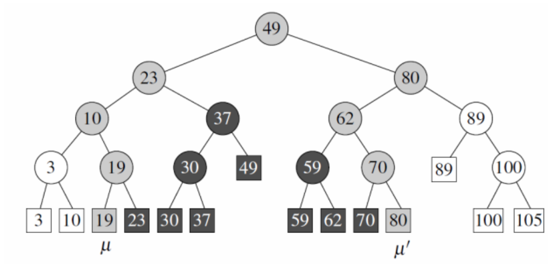
Range trees (Bereichsbäume)

1 Einleitung

Range trees erlauben orthogonale Bereichsanfragen in einer *d*-dimensionaler Punktemenge in *O*($log^d n+k$) Zeit bei einem Platzverbrauch von *O*($n\ log^{d-1} n$). Verwendet werden diese Bäume beispielsweise im Rechteckschnittproblem.

1.1 1-dimensionaler range tree

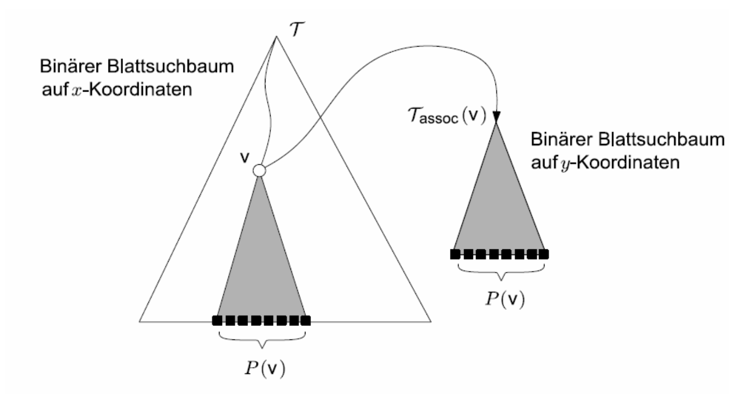


Die 1-dimensionale Variante erlaubt auf einer Zahlenmenge eine Bereichsanfrage, die in logarithmischer Zeit bewältig werden kann. Dazu verwendet man einen Blattsuchbaum, bei dem die Blätter in einer verketteten Liste gespeichert werden. Dies hat zur Folge, dass für eine Bereichsanfrage nach dem einen Ende gesucht werden kann (das geht in logarithmischer Zeit wenn ein balancierter Baum eingesetzt wird), und dann wird einfach so lange der Liste gefolgt, bis das erste Element ausserhalb des Bereiches gefunden wird. Will man nur die Anzahl Punkte in einem Intervall wissen, so muss bei jedem inneren Knoten zusätzlich gespeichert werden, wie viele Punkte in diesem Teilbaum gespeichert sind. Dann werden einfach die Kinder aller *„ausgewählter Teilbäume“* (siehe Abschnitt 2 und Graphik) gezählt, was in logarithmischer Zeit machbar ist.

2 d-dimensionaler range tree

Die d-dimensionale Erweiterung speichert für jeden Teilbaum T_x alle Knoten in T_x in einem *(d-1)*-dimensionalen Baum. Genauer:

- Die Basis besteht aus einem balancierten binären Blattsuchbaum *T*, sortiert nach der ersten Koordinate der Datensätze.
- Jeder innere Knoten *v* besitzt einen Zeiger auf einen weiteren Blattsuchbaum $T_{assoc}(v)$. In diesem sind Kopien aller Blätter gespeichert, jedoch sortiert nach der nächsten Koordinate. Es gibt also für jede Dimension eine Menge von Bäumen und jeder Schlüssel kommt in asymptotisch *O*($log^{d-1} n$) vielen Bäumen vor.



Beispiel eines 2-dimensionalen Bereichbaumes

Eine Suchanfrage funktioniert nun folgendermassen: Zuerst wird nach der 1. Koordinate im Basisbaum gesucht: Dazu wird nach den beiden Enden des Anfrageintervalls *[a,b]* von der Wurzel aus gesucht, wobei sich die beiden Suchpfade irgendwann in einem Knote v_{split} trennen. Von dort folgen wir einem Pfad nach links zu *a*, wobei wir jedesmal, wenn wir links abbiegen, eine (d-1)-dimensionale Bereichsanfrage im rechten Kind durchführen. Der Basisfall ist dann die 1-dimensionale Bereichsanfrage wie oben beschrieben. Symmetrisch gehen wir auch für den Pfad zu *b* vor.