Algorithmische Geometrie
Vorlesung vom 10.06.05

Wie in der letzten Vorlesung bereits festgestellt wurde, ist $O(\sqrt{n}+m)$ Laufzeit für eine Suchanfrage in einem *k*-d-Baum, mit k=2, nötig.

Analyse: Anzahl der besuchten Knoten

Angenommen, die Suchregion sei *U*.
Dann gibt es folgende Möglichkeiten, welche Arten von Knoten besucht werden:

a.) Knoten *v* mit $Reg(v) \subset U$
$\Rightarrow$ Es sind $\leq m$ dieser Knoten möglich.

Sei *L* eine Seite des Rechtecks *U*.

b.) Primärknoten *v* mit $Reg(v) \cap L \neq \varnothing$ [und $Reg(v) \not\subset U$]

$\Rightarrow$ Bei einer Tiefe *s* des Baumes gibt es p_s Stück von diesen, wobei $p_s \leq 2^{\left\lceil \frac{s}{2} \right\rceil}$.

c.) Andere Knoten *v* mit $Reg(v) \cap L \neq \varnothing$, also Nachkommen eines =-Zeiger eines Primärknotens und der Tiefe *s*.
Bei diesen Knoten gibt es zwei Möglichkeiten, wie die Suchregion *U* bezüglich den Knoten im Unterbaum verlaufen kann:

i. Die Suche ist ein Weg nach unten in dem Unterbaum zu der in *Abb. 2* dargestellten Situation. Dabei wird Binärsuche angewendet, um *L* zu finden.

$\Rightarrow$ Insgesamt sind das $\log n - s$ Knoten, die in diesem Unterbaum besucht werden können.

ii. Ist die Suche ein Weg in dem Unterbaum zur Situation in *Abb. 1*, dann müssen Knoten betrachtet werden, deren Region nicht ganz in *L* liegt. Es müssen beide Endpunkte des Intervalls gesucht werden.

$\Rightarrow$ In diesem Fall können insgesamt höchstens $2(\log n - s)$ Schritte zurückgelegt werden bei der Suche nach den Endpunkten. Man betrachtet ausgehend von dem linken und rechten Endpunkt deren Unterbäume weiter.

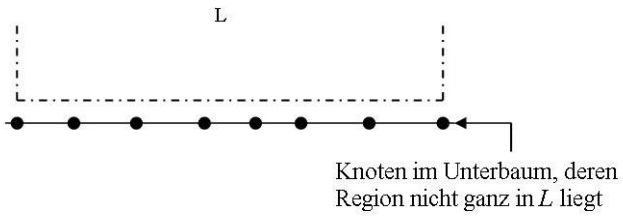

Abbildung 1

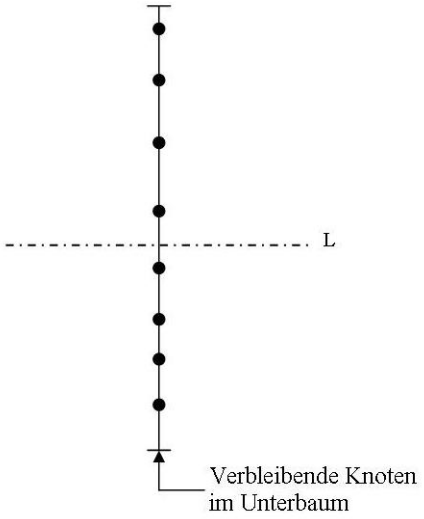

Abbildung 2

Für die Gesamtanzahl der besuchten Knoten aus den Punkten b.)-c.) ergibt sich nach den obigen Betrachtungen:

$$
\begin{aligned}
&\leq \sum_{s=0}^{\lceil \log n \rceil} \underbrace{2^{\lceil \frac{s}{2} \rceil}}_{\substack{\#\ \text{Anzahl der Primärknoten} \\ \text{der Tiefe } s}} \text{der} \left(\underbrace{1}_{\substack{\text{Knoten} \\ \text{selbst}}} + \underbrace{2(\log n - s)}_{\text{Nachkommen im =-Unterbaum}} \right) \\
&= 2 \sum_{s=0}^{\lceil \log n \rceil} 2^{\frac{s}{2}} \left(1 + 2(\log n - s)\right) \\
&\leq 2\sqrt{n} \sum_{s=0}^{\log n + 1} \left(2^{\frac{1}{2}} \right)^{\log n - s} \cdot \left(1 + 2(\log n - s)\right) \\
&= 2\sqrt{n} \sum_{s=-1}^{\log n} \left(2^{\frac{1}{2}} \right)^{-r} \cdot (1 + 2 \cdot r), \qquad \textit{wobei} \quad r = \log n - s \\
&\leq 2\sqrt{n} \sum_{r=-1}^{\infty} \frac{1 + 2r}{\sqrt{2}^{r}}, \\
&= O(\sqrt{n})
\end{aligned}
$$

Der Faktor $\sum_{r=-1}^{\infty} \frac{1+2r}{\sqrt{2}^{r}}$ in der Herleitung ist konstant, da es sich hierbei um eine konvergente Reihe handelt, so dass sich dadurch $O(\sqrt{n})$ ergibt.

Nach diesen Überlegungen wird für den Besuch aller Knoten des 2-d-Baum aus a.)-c.) $O(\sqrt{n} + m)$ Zeit benötigt.

Für den *k*-d-Baum allgemein ergibt sich $O\left(n^{1-\frac{1}{k}} + m \right)$. (Beweis: Übung*)*

Range-Bäume („Bereichsbäume“, Range Trees)

Im Folgenden soll mit den Bereichsbäumen eine weitere geometrische Datenstruktur zur Speicherung einer Menge von Datenpunkten betrachtet werden. Durch diese können orthogonale Suchanfragen schneller beantwortet werden, allerdings haben Bereichsbäume dafür auch einen höheren Platzbedarf.

Im eindimensionalen Raum ist ein Bereichsbaum für eine Bereichsanfrage auf n Zahlen ein binärer Suchbaum (vgl. Abb. 3).

Bei einer Bereichsfrage werden alle Punkte zwischen Anfrage und Ende des Abfragebereichs gesucht. Die Antwortzeit für eine solche Anfrage beläuft sich auf $O(\log n + k)$ mit $k = | U \cap S | =$ *Anzahl der gesuchten Elemente*.

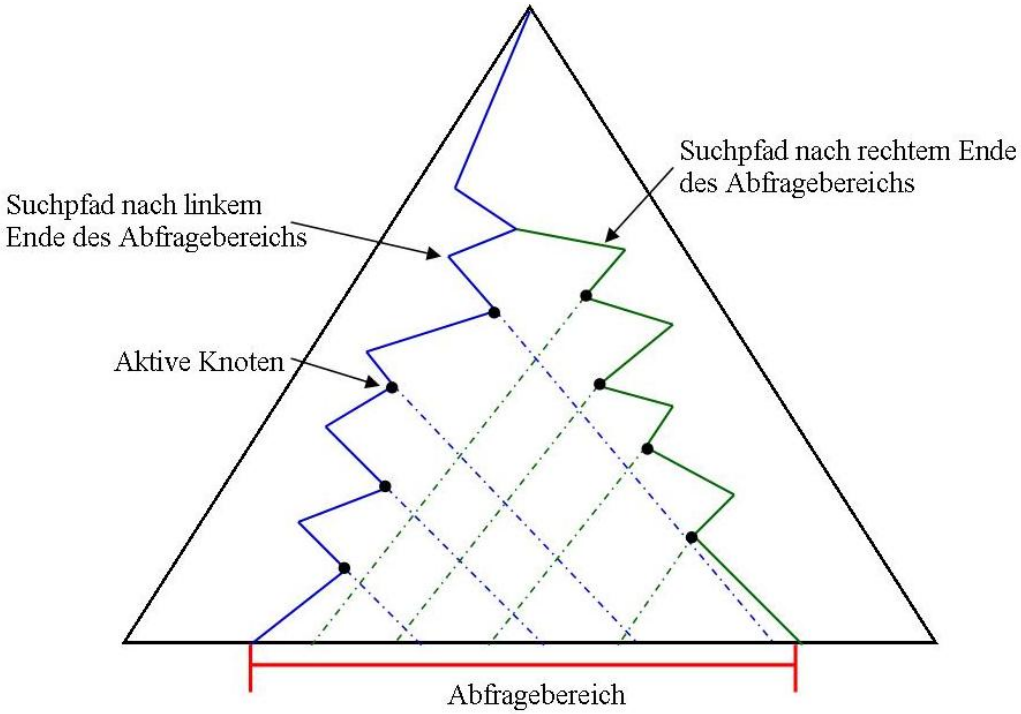

Abbildung 3: Balancierter Suchbaum für eine 1-dimensionale orthogonale Bereichsabfrage

Anhand der Abbildung erkennt man, dass auf dem Suchpfad zu dem linken Ende des Abfragebereichs auch alle rechten Unterbäume Elemente der Antwort sind. Analog dazu sind auf dem Suchpfad zum rechten Ende alle Elemente der linken Unterbäume in der Antwort enthalten. Diese Unterbäume werden „aktive Unterbäume“ genannt und hängen an „aktiven Knoten“ (vgl. Abb. 3). Aus allen Knoten der aktiven Unterbäume ergibt sich das oben genannte k in der Laufzeit für eine Anfrage.

Man kann eine Laufzeit von $O(\log n)$ erreichen, wenn man eine „Zahlversion“ der Bäume verwendet. Dabei wird in jedem Knoten zusätzlich die Größe der Unterbäume gespeichert.

Die Verallgemeinerung der im Beispiel gezeigten Idee auf höhere Dimensionen sind die sog. „**Range-Trees**“.

Für zwei Dimensionen kann man sich einen Bereichsbaum wie folgt vorstellen:

Sei $S \subset R^2$, $|S| = n$
S_1 = x-Koordinaten von Punkten in S und S_2 = y-Koordinaten von Punkten in S

1. Zunächst baut man einen balancierten binären Suchbaum für S_1 auf, damit eine effiziente Suche in den x-Koordinaten möglich ist. Dieser Suchbaum wird die sog. „Primärstruktur“ genannt (vgl. Suchbaum rechts oben in Abb. 4).

2. An jedem Knoten v der Primärstruktur hat man einen Zeiger auf einen balancierten binären Suchbaum bezüglich der y-Koordinaten. Dieser Suchbaum enthält alle Punkte des Unterbaums von v und wird als „Sekundärstruktur“ bezeichnet (vgl. Suchbaum für den Knoten 11 rechts unten in Abb. 4).

Bemerkung: Im d-dimensionalem Raum bezieht sich die Primärstruktur auf die erste Koordinate. An jedem Knoten gibt es einen Zeiger auf eine (d-1)-Suchstruktur bezüglich der anderen Koordinaten, also auf die Sekundärstruktur.

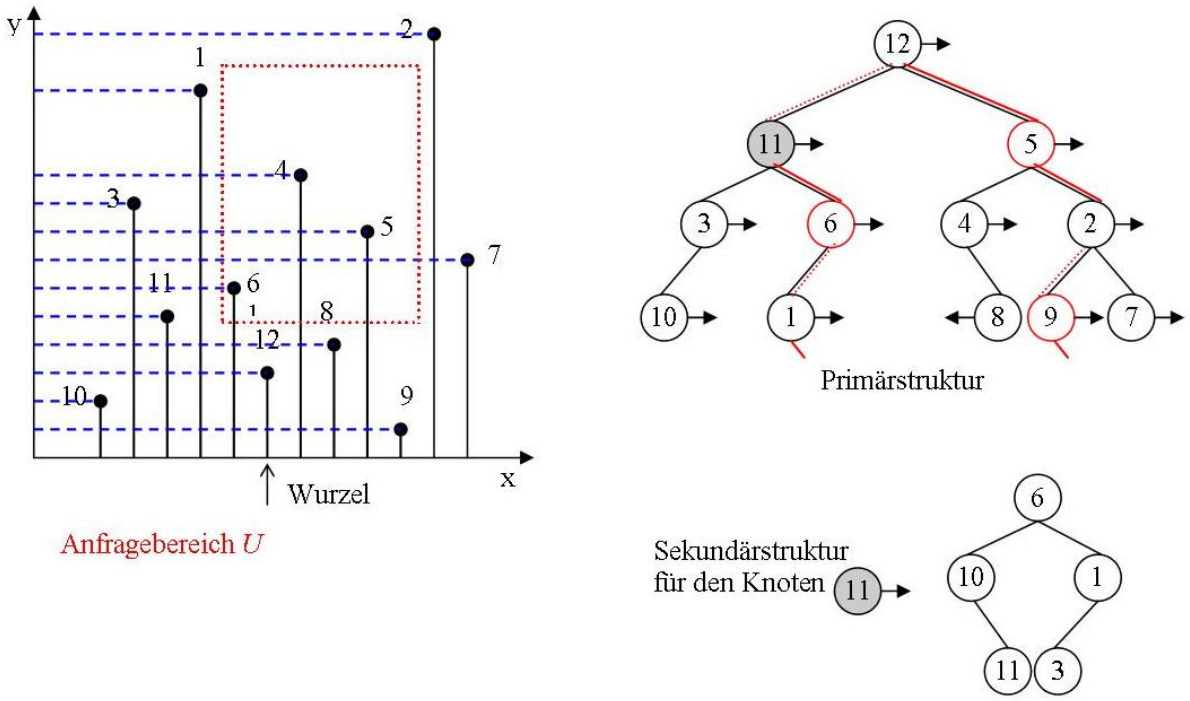

Abbildung 4: Beispiel einer Suchstruktur für eine Menge von Punkten

Platzbedarf im 2-dimensionalen Fall

Jeder Punkt wird ein Mal in der Primärstruktur abgespeichert. Außerdem wird jeder Punkt so oft abgespeichert, wie er Vorfahren in der Primärstruktur hat, wofür $O(\log n)$ Zeit benötigt wird.
Für die Vorverarbeitungszeit insgesamt ergibt sich daraus $O(n \log n)$.

Ablauf einer orthogonalen Bereichsabfrage

Sei *S* die Anfrage nach dem Rechteck $U = [x_1, x_2] \times [y_1, y_2]$

1. Finde die „aktiven Knoten“ aus dem Intervall $[x_1, x_2]$ in der Primärstruktur.
2. Von den aktiven Knoten aus durchlaufe die zugehörigen Suchbäume in der Sekundärstruktur mit $[y_1, y_2]$ und gib alle Knoten in den zugehörigen aktiven Unterbäumen aus.
   Beim Durchlaufen der Unterbäume beginnt man an den Knoten unterhalb der aktiven Knoten (vgl. Abb. 5). Die aktive Knoten selbst werden separat auf Zugehörigkeit zum Suchbereich überprüft.

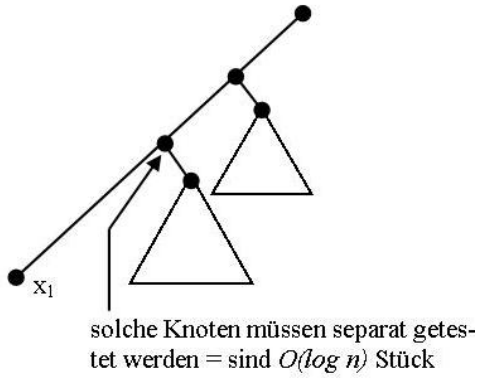

Abbildung 5: Durchlaufen der aktiven Unterbäume

In Abb. 4 ist links der Suchbereich rot hervorgehoben. Auf der rechten Seite in der Primärstruktur sind der Suchpfad sowie die aktiven Knoten ebenfalls in rot gekennzeichnet. Aus dem Baum geht hervor, dass alle Punkte rechts von Punkt 1 und die Punkte, die rechts von 9, aber links von 7 liegen in den Suchbereich fallen.

Antwort auf ein orthogonale Bereichsabfrage

Man überlegt sich zunächst, dass es $O(\log n)$ aktive Knoten in der Primärstruktur geben kann.
Für jeden dieser aktiven Knoten gibt es $O(\log n)$ aktive Knoten in der Sekundärstruktur.
Für die Antwortzeit ergibt sich damit insgesamt $O(\log^2 n + k)$, wobei k die Anzahl der Punkte ist, die im Suchbereich *U* liegen.