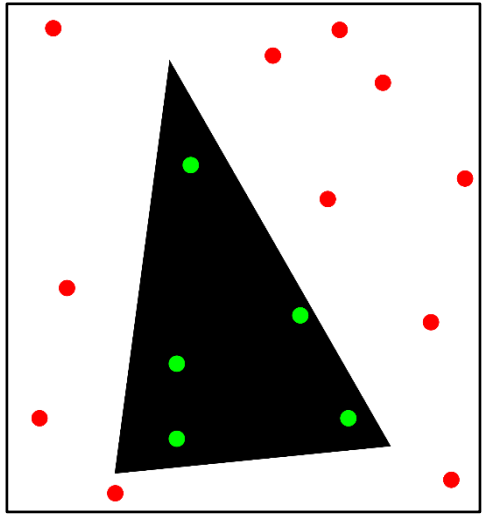
Technische Universität Braunschweig

Einführung in algorithmische Geomtrie – Quadtrees

Arne Schmidt

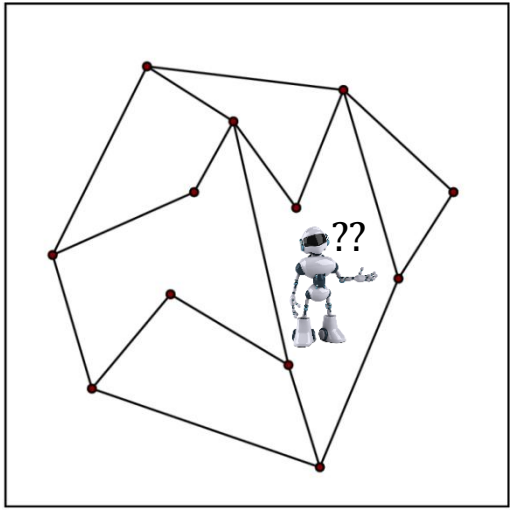
Kapitel 2 - Lokalisierungsprobleme

Probleme



Welche Punkte liegen in dem Bereich?

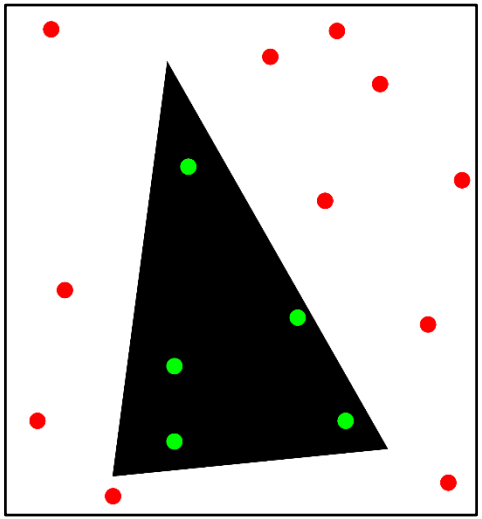
→ Range Query



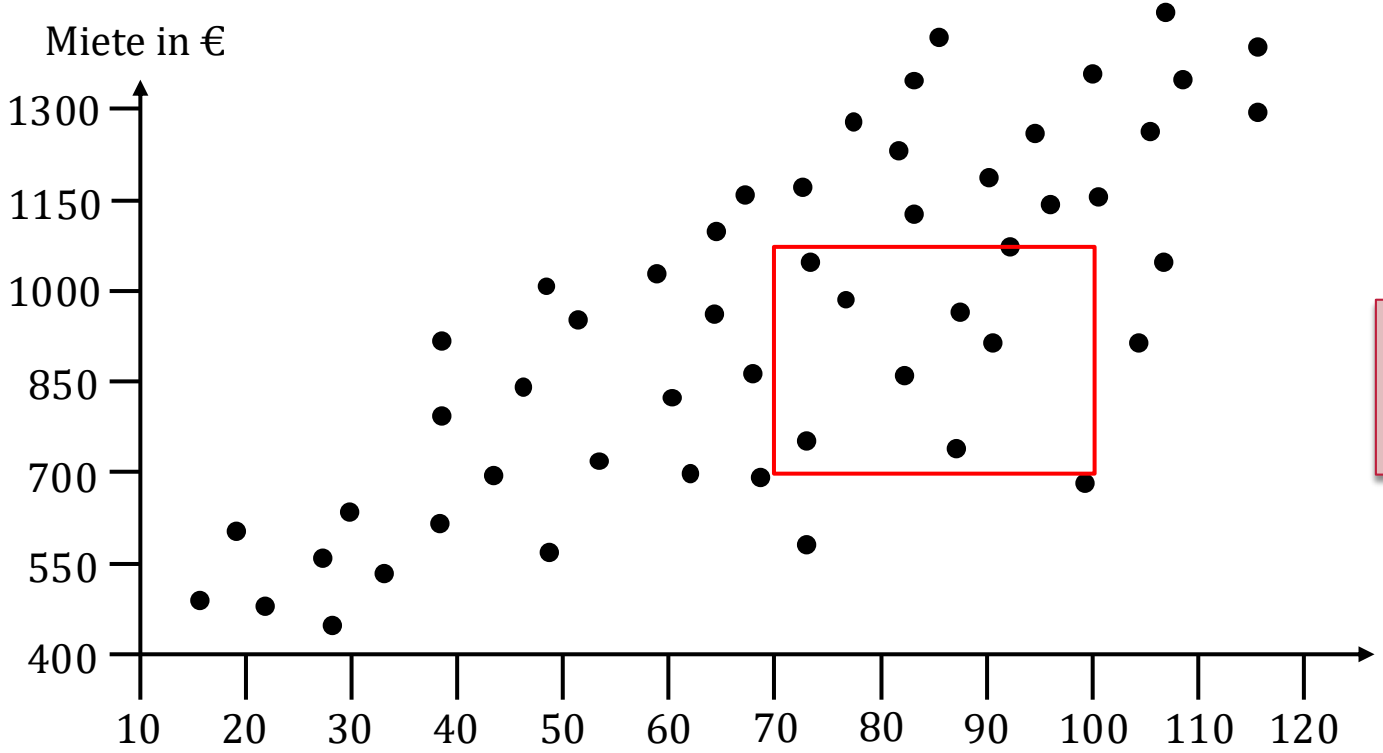
In welchem Bereich liegt ein Punkt?

→ Punktlokalisierung

Kapitel 2.1 – Range Queries



Motivation – Die Wohnungssuche

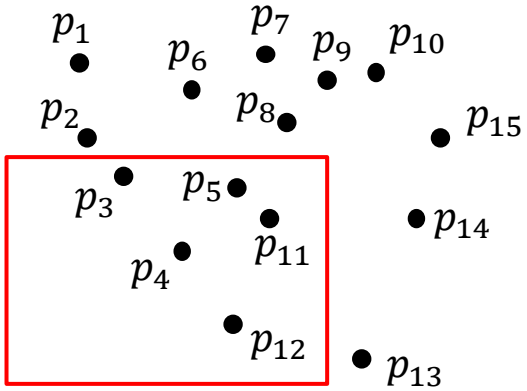


Suche Wohnung:
Maximal 1050€
Mindestens 700€
Maximal 100m²
Mindestens 70m²

Aufgabe: Gib alle Wohnungen im roten Quadrat aus!

Range Query

Problem 2.1 (Range Query, 2D)
Gegeben: Eine Punktmenge $P := \{p_1, \dots, p_n\} \in \mathbb{R}^2$ und Rechteck $R \subset \mathbb{R}^2$.
Gesucht: Menge $P' \subseteq P$, sodass für alle $p \in P$ gilt $p \in P' \Leftrightarrow p \in R$.



$$P' = \{p_3, p_4, p_5, p_{11}, p_{12}\}$$

Überlegungen

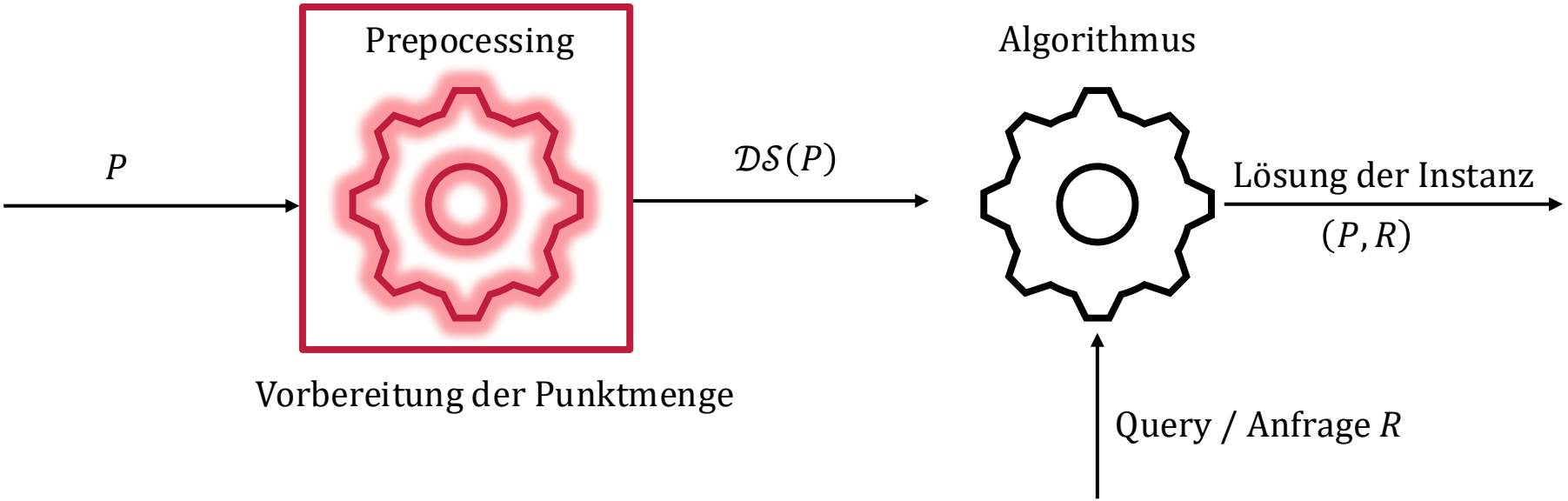
Einfach zu lösen!

Teste jeden Knoten, ob er in R liegt.

Das liefert einen Algorithmus mit Laufzeit $O(n)$

Was passiert, wenn mehrere Anfragen für die gleiche Punktmenge kommen?

Preprocessing und Queries



Seien $\mathcal{T}_P$ und $\mathcal{T}_A$ die benötigten Zeiten für das Preprocessing, bzw. des Algorithmus. Für N viele Anfragen ergibt sich dann eine Gesamtlaufzeit von $\mathcal{T}_P + N \cdot \mathcal{T}_A$.
Ziele: Halte beides, $\mathcal{T}_P$ und $\mathcal{T}_A$, möglichst klein!

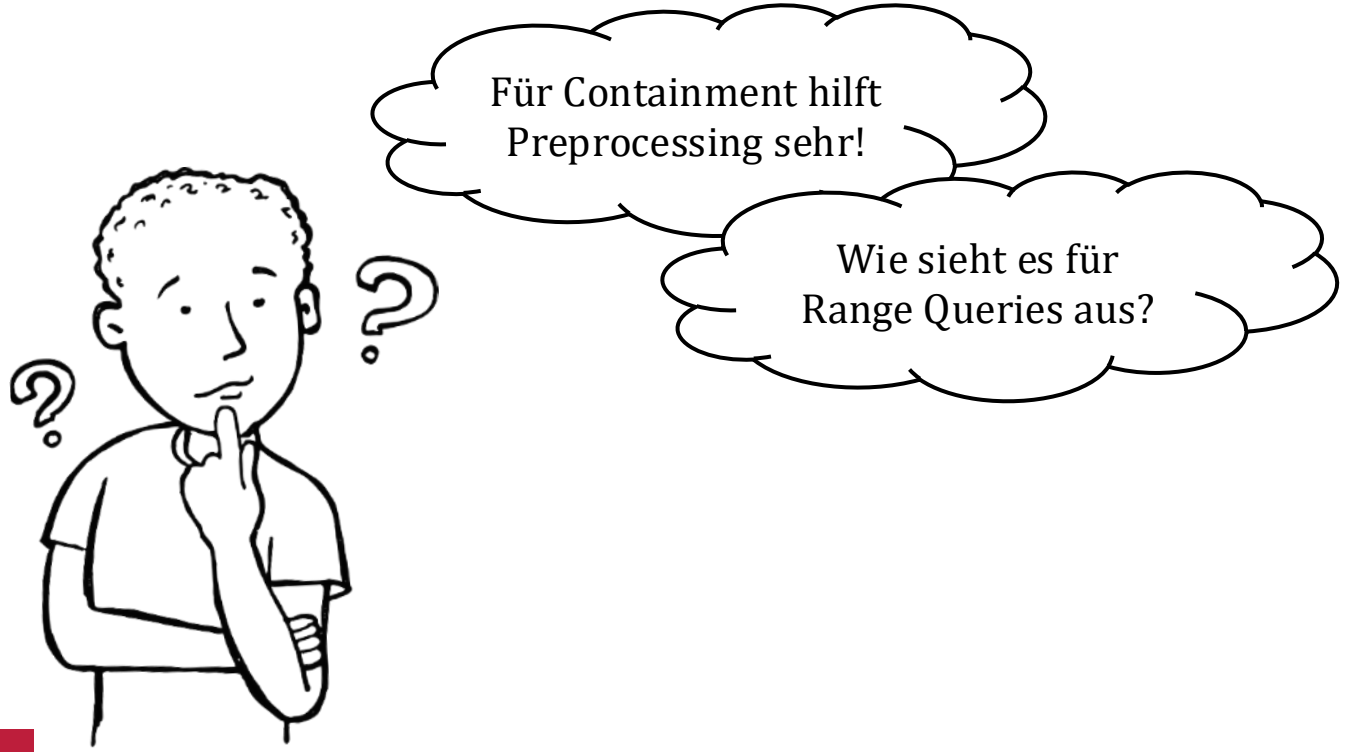
Detour: Containment

Problem
Gegeben: Menge $P \coloneqq \{p_1, \dots, p_n\} \subset \mathbb{N}$ und Zahl $z \in \mathbb{N}$.
Frage: Ist $z \in P$?

Fragen:

- Wie schnell kann man das Problem ohne Preprocessing lösen?
- Welche Datenstruktur eignet sich, damit man viele Anfragen für die gleiche Menge P schneller lösen kann?
- Wie schnell kann die Datenstruktur berechnet werden?
- Wie schnell kann eine Anfrage beantwortet werden?

Zurück zur Range Query



Laufzeitschranken

Unabhängig von der Datenstruktur:

- Liegen k Punkte im Rechteck, benötigen wir $\Omega(k)$ Schritte!

Gibt es eine Datenstruktur, welche:

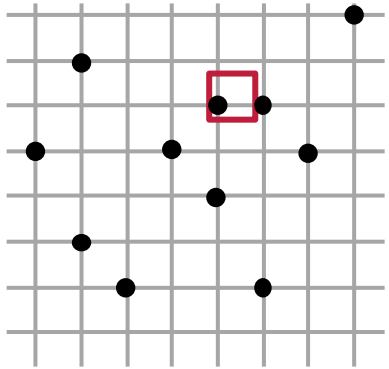
- Schnell erstellt werden kann
- Wenig Platz verbraucht
- Range-Queries schneller als $O(n)$ für kleine k löst?

Wünschenswert wäre eine Laufzeit von $O(\log(n) + k)$.

Definition 2.2 (Output-Sensitive Algorithmen)
Hängt die Laufzeit eines Algorithmus A sowohl von der Größe des Inputs als auch von der Größe des Outputs ab, dann ist A ein **output-sensititve Algorithmus**.

$k \in O(1)$

Betrachten wir zunächst einen ganz speziellen Fall.
$P \subset \mathbb{Z}^2$ und R ist ein Quadrat mit Seitenlänge 1.
→ R kann maximal 4 Punkte abdecken.

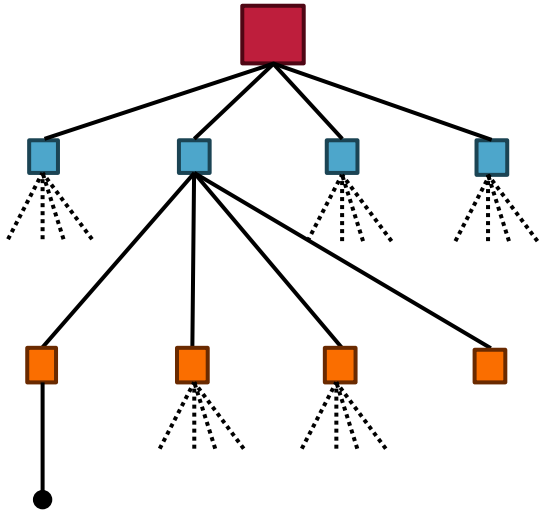


Ziel: Finde Datenstruktur, sodass ein Großteil der Punkte schnell ausgeschlossen werden kann.

Teile den zu durchsuchenden Raum in zwei Hälften

Schaue darüber, in welchem Teil R liegt.

Kapitel 2.1.1 – Quadtrees



Ideensammlung und Probleme

A. Wie kann man den Raum in zwei Hälften teilen?
B. Angenommen, wir haben den Raum mit einer Linie in zwei Hälften geteilt. Wie geht es dann weiter? Beachte folgende Situationen
    i. Linie schneidet R nicht → R ist vollständig in einer der Hälften. → Rekursion auf dieser Hälfte!
    ii. Linie schneidet R → Rekursion auf beide Hälften!

Für B. können wir also probieren, den Raum rekursiv in Hälften zu teilen und bei einer Anfrage versuchen jeweilige Hälften direkt auszuschließen!

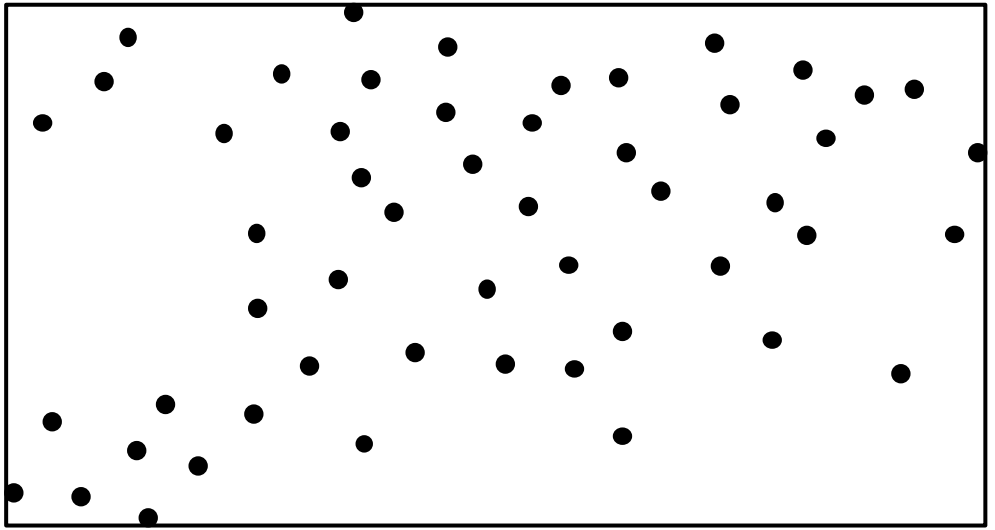
Wie können wir den **unendlichen** Raum rekursiv zerteilen?

Raum einschränken

Idee: Suche eine Box, welche
- Alle Punkte einschließt
- So klein wie möglich ist

Zur Vereinfachung:
Die Seiten sollen parallel zur x- bzw. y-Koordinate verlaufen.

→ Achsen-paralleles Rechteck



Problem 2.3 (Axis-Aligned Bounding Box, 2D)
Gegeben: Eine Punktmenge $P := \{p_1, \dots, p_n\} \in \mathbb{R}^2$.
Gesucht: Achsen-paralleles Rechteck $R \subset \mathbb{R}^2$, sodass für alle $p \in P$ gilt $p \in R$ und Fläche von R minimal.

Algorithmus AABB

Satz 2.4
Problem 2.3 (Axis-Aligned Bounding Box) kann in Zeit $O(n)$ gelöst werden.

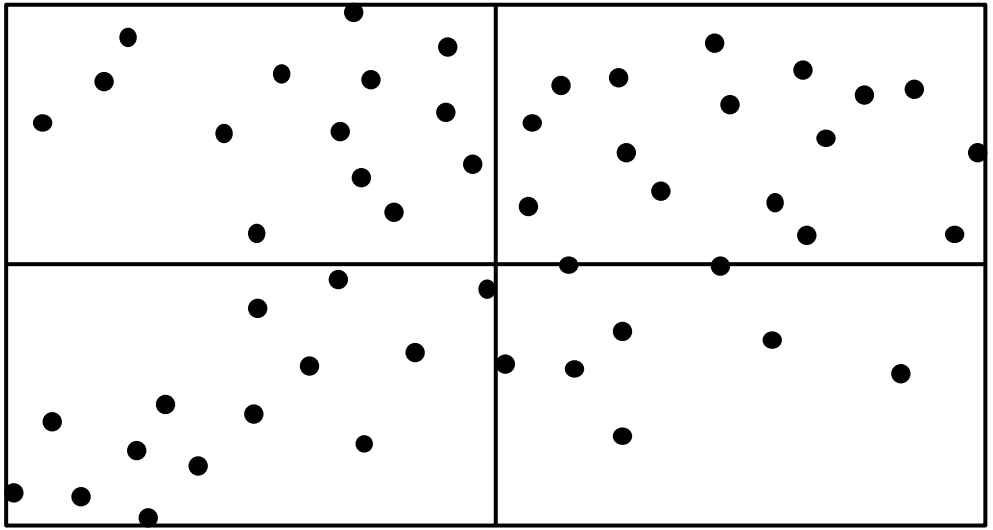
Suche:

$$x^*_{\min} := \min_{p \in P} x_p$$
$$y^*_{\min} := \min_{p \in P} y_p$$
$$x^*_{\max} := \max_{p \in P} x_p$$
$$y^*_{\max} := \max_{p \in P} y_p$$

Gib Rechteck mit Eckpunkten $(x^*_{\min}, y^*_{\min})$ und $(x^*_{\max}, y^*_{\max})$ zurück.

Rekursives Teilen!

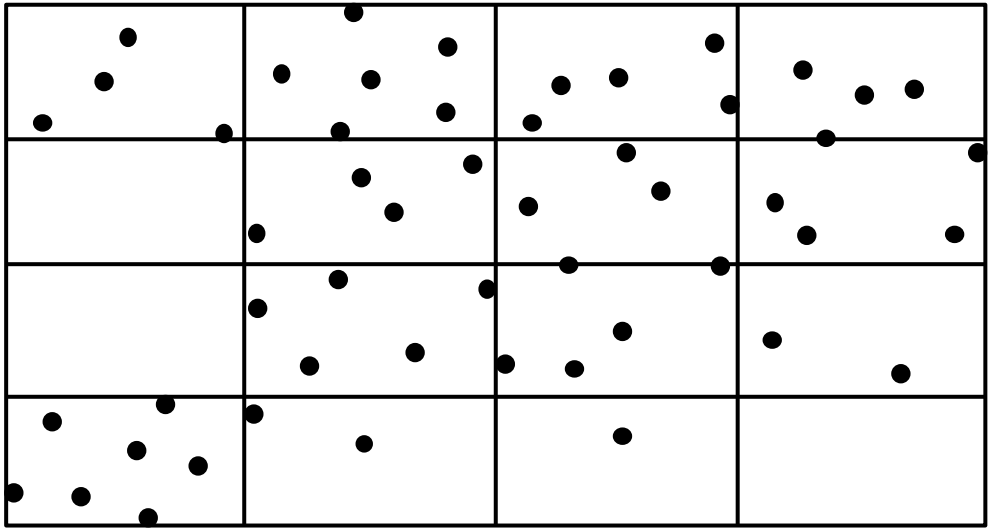
Da wir in zwei Dimensionen sind, können wir gleichzeitig die Breite und Höhe in zwei Teile scheiden!



Rekursives Teilen!

Da wir in zwei Dimensionen sind, können wir gleichzeitig die Breite und Höhe in zwei Teile scheiden!

→ Wir erhalten vier Boxen $B_1, \dots, B_4$ mit Punktmengen $P_1, \dots, P_4$.
→ Rekursion auf B_i und P_i für alle $i = 1, \dots, 4$

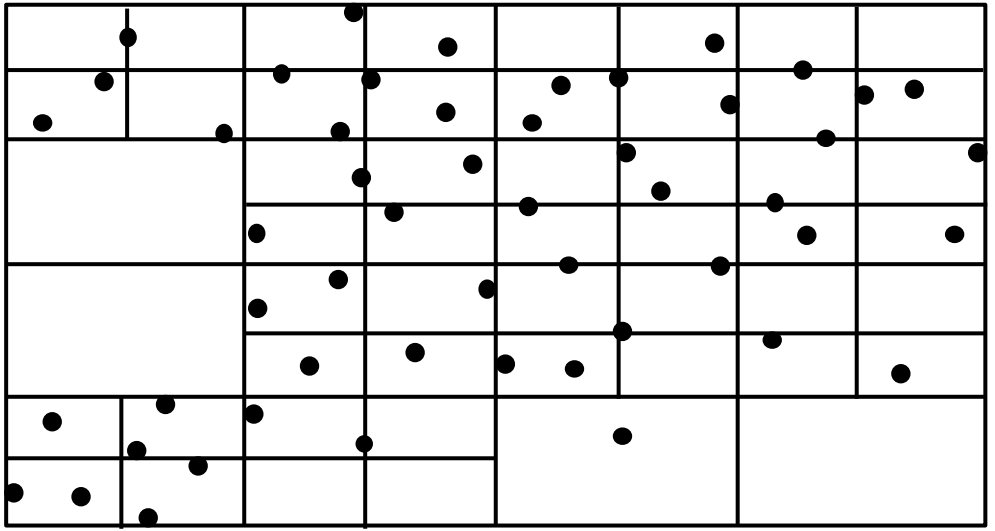


Rekursives Teilen!

Da wir in zwei Dimensionen sind, können wir gleichzeitig die Breite und Höhe in zwei Teile scheiden!

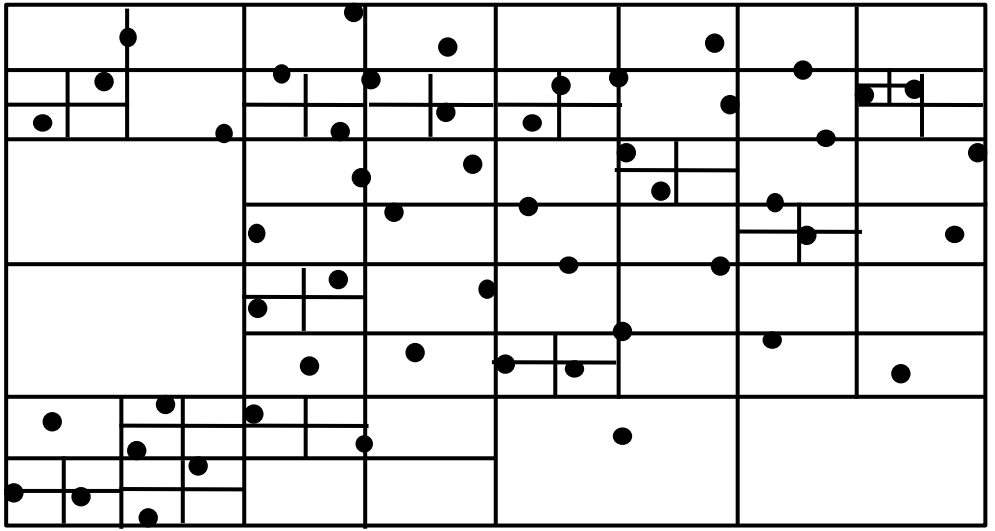
→ Wir erhalten vier Boxen $B_1, \dots, B_4$ mit Punktmengen $P_1, \dots, P_4$.
→ Rekursion auf B_i und P_i für alle $i = 1, \dots, 4$

→ Was machen wir, wenn $P_i = \emptyset$?
→ Was machen wir, wenn P_i nur noch einen Punkt besitzt?
→ Stoppe Rekursion!



Rekursives Teilen!

Da wir in zwei Dimensionen sind, können wir gleichzeitig die Breite und Höhe in zwei Teile scheiden!

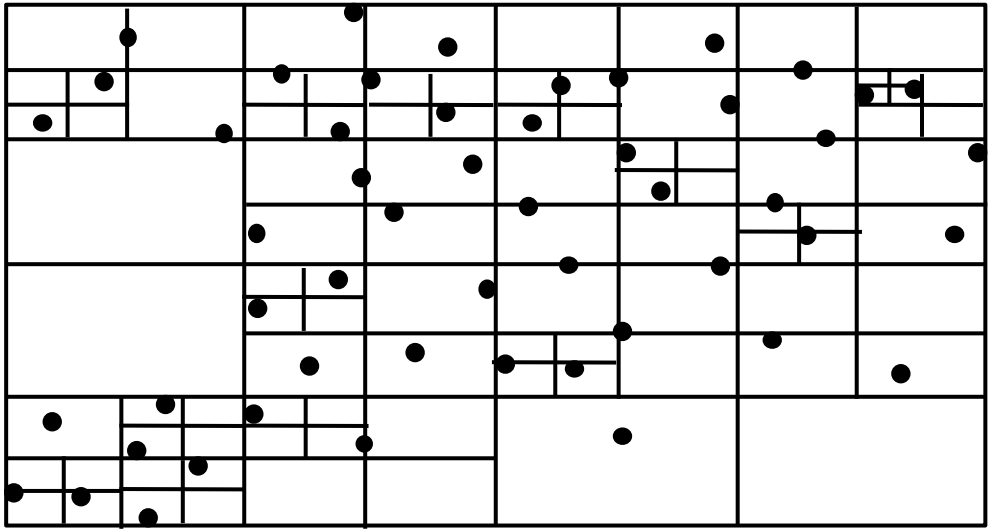


→ Wir erhalten vier Boxen $B_1, \dots, B_4$ mit Punktmengen $P_1, \dots, P_4$.
→ Rekursion auf B_i und P_i für alle $i = 1, \dots, 4$

→ Was machen wir, wenn $P_i = \emptyset$?
→ Was machen wir, wenn P_i nur noch einen Punkt besitzt?
→ Stoppe Rekursion!

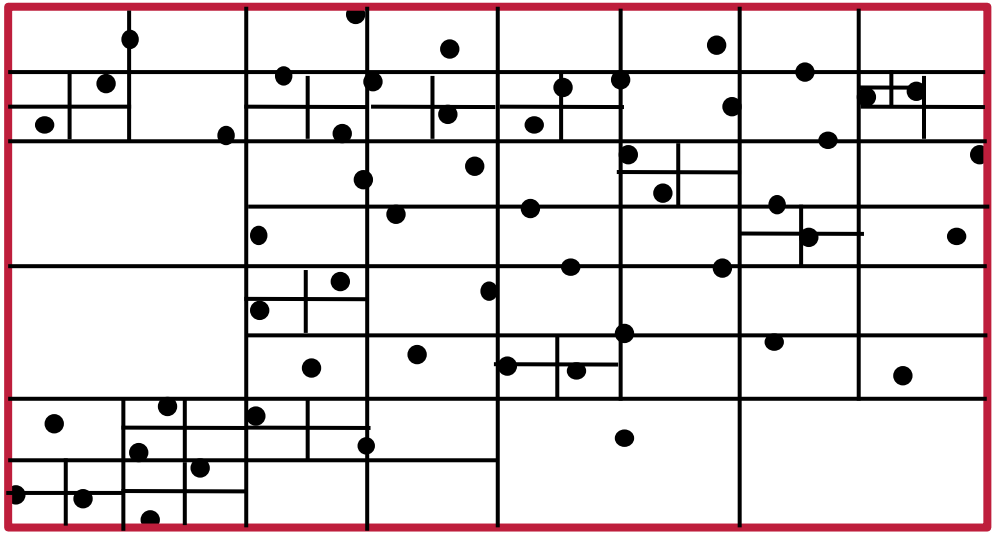
Hierarchie von Räumen

Stelle die unterteilten Räume als Knoten dar!



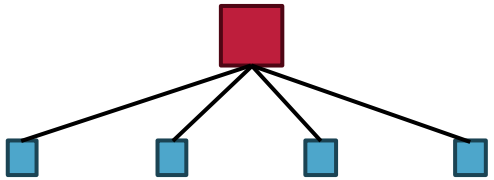
Hierarchie von Räumen

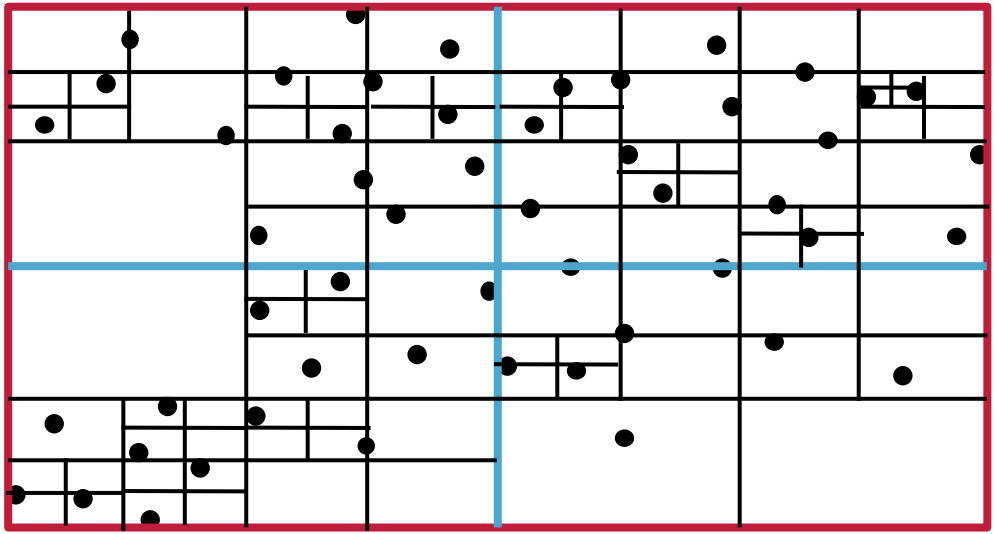
Stelle die unterteilten Räume als Knoten dar!



Hierarchie von Räumen

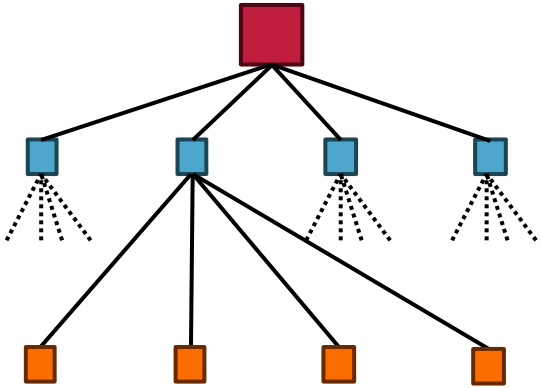
Stelle die unterteilten Räume als Knoten dar!

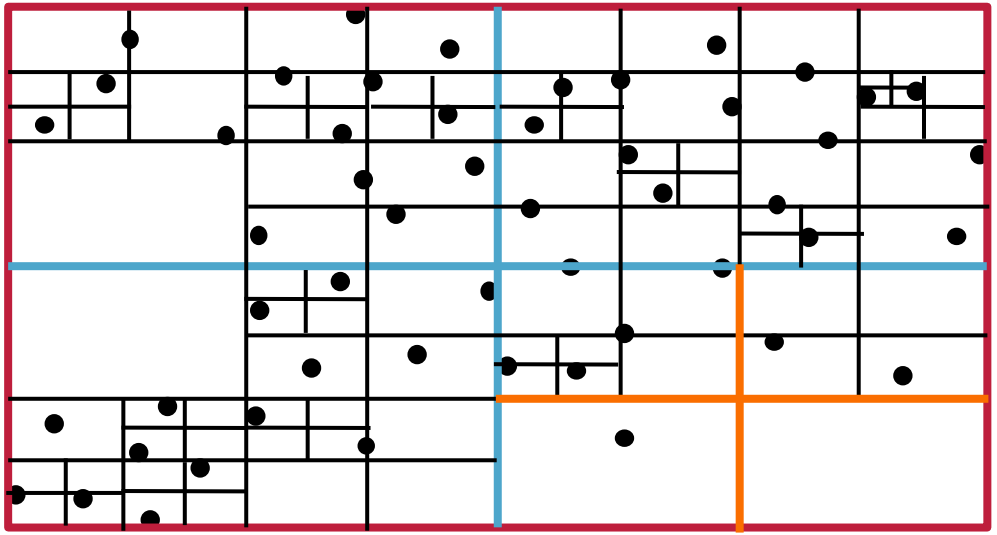




Hierarchie von Räumen

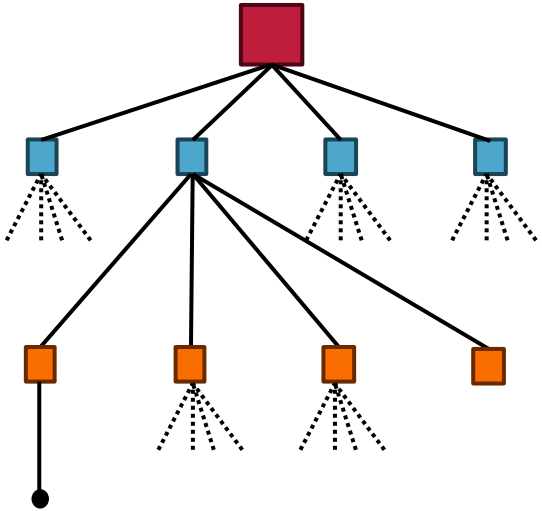
Stelle die unterteilten Räume als Knoten dar!

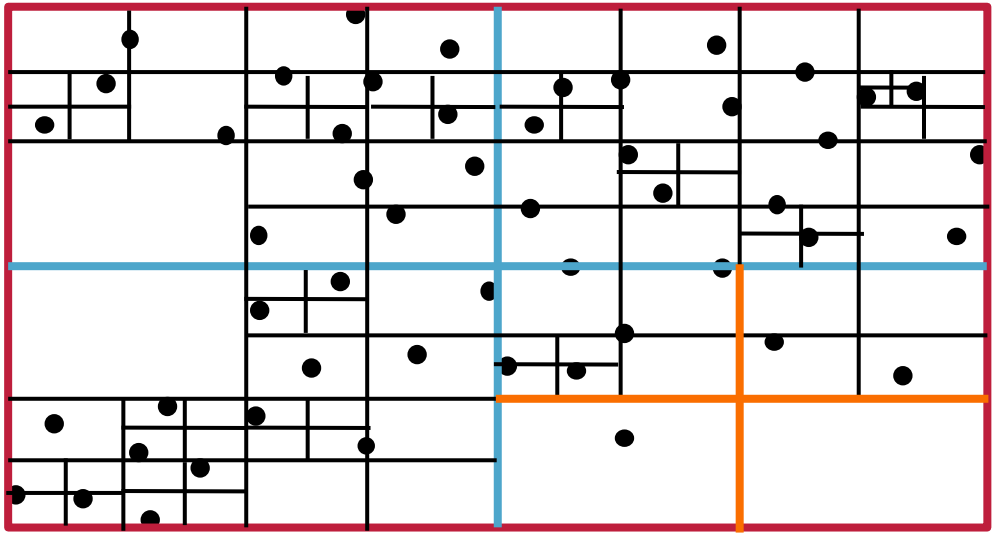




Hierarchy von Räumen

Stelle die unterteilten Räume als Knoten dar!





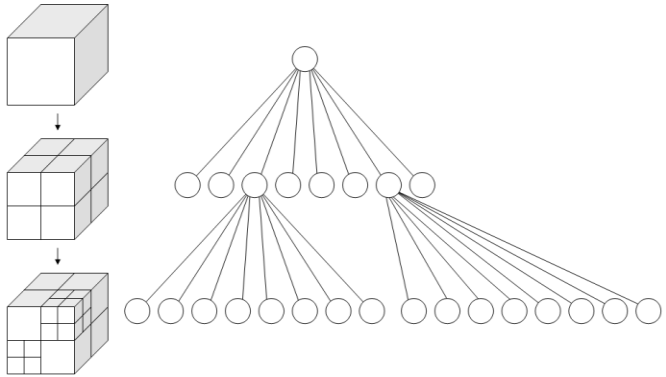
Jeder Punkt ist ein Blatt. Ist ein Raum leer, besitzt er keine Kinder (und ist auch ein Blatt).

Quadtree

Definition 2.5 (Quadtree)
Ein **Quadtree Q** ist eine Baumstruktur, in welcher jeder innere Knoten vier Kinder besitzt.
- Jeder innere Knoten entspricht dabei einem Rechteck.
- Jedes Blatt entspricht einem Punkt oder einem leeren Rechteck.
- **root(Q)** ist der Wurzelknoten von Q.

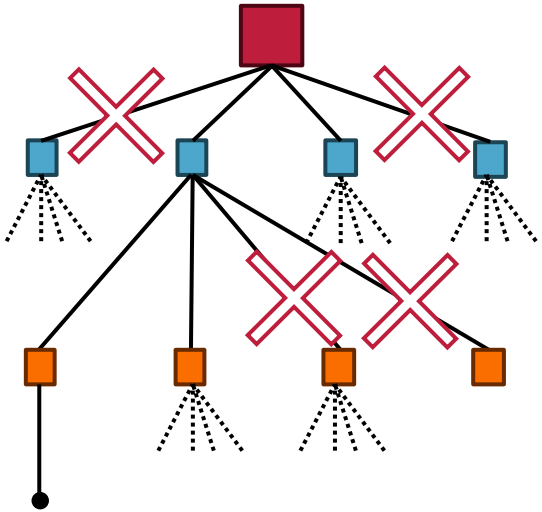
Das lässt sich auch für 3D definieren.
→ Teile in 3 Dimensionen.
→ 8 Teilquader!
→ Octree

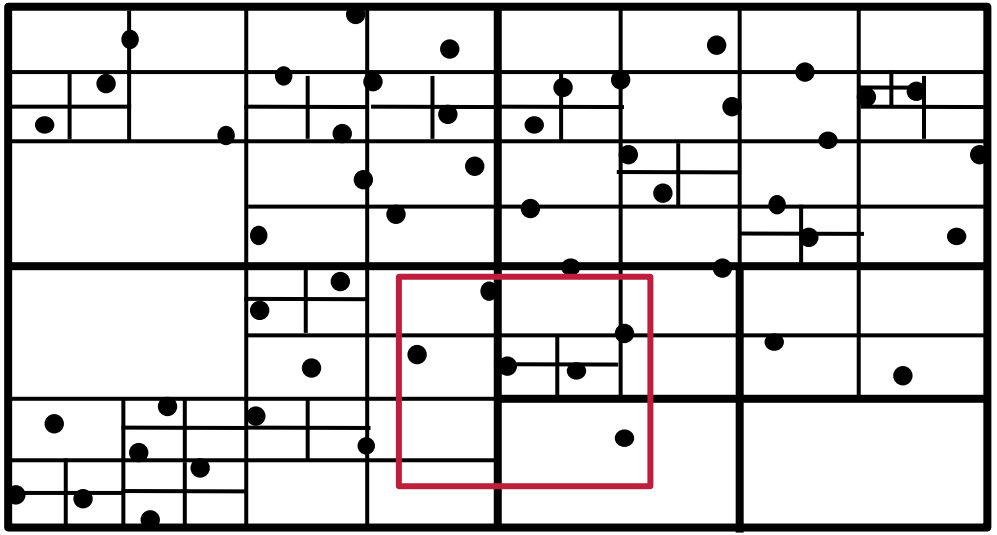
Wie lösen wir hierauf nun die Range Query?



https://de.wikipedia.org/wiki/Octree#/media/Datei:Octree2.png

Range Query mit Quadtrees





Welche Kinder überlappen sich mit dem Range-Query-Rechteck?
Suche auf diese rekursiv weiter.

Range Query mit Quadtrees

Algorithmus 2.6 (`Report(Q, R)`)
Input: Quadtree Q für Punktmenge P, Rechteck R
Output: Punktmenge $P' := P \cap R$

1. If root(Q) hat keine Kinder und root(Q) ist ein Punkt then
   1. Gib P aus.
2. If root(Q) besitzt keine Kinder then return.
3. Für jedes Kind Q' von Q:
   1. If $Q' \cap R \neq \emptyset$ then `Report(Q', R)`

Man kann noch weitere Schritte hinzufügen. Bspw:
Wenn root(Q) $\subseteq R$, gib P aus.

Laufzeit

Korrektheit ist relativ klar.

Welche Laufzeit besitzt der Algorithmus?

Wie sieht ein Worst-Case-Beispiel aus?

Wie viele Punkte können in einer Aufteilung enthalten sein?

Höhe von Quadtrees

Satz 2.7
Algorithmus 1.6 löst das Range-Query-Problem in Zeit $O(n + h)$, wobei n die Größe der Punktmenge ist und h die Höhe des Quadtrees.

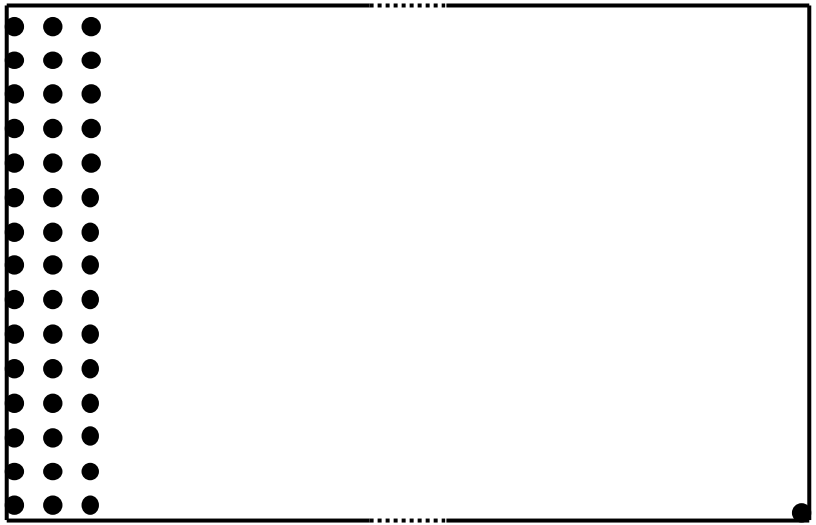
Können wir h irgendwie abschätzen?

Lemma 2.8
Sei s die Länge der Diagonalen der Bounding Box einer Punktmenge P und c der kleinste Abstand zwischen zwei Punkten in P. Dann ist die Höhe des Quadtrees maximal $O\left(\log\left(\frac{s}{c}\right)\right)$.

Beweisidee: Skaliere Instanz mit $\frac{1}{c}$. Wie oft muss man die Boundingbox (mit Diagonale s/c) unterteilen, damit die Knoten des Quadtrees ein Rechteck repräsentieren, dessen Diagonale kürzer als 1 ist? → Etwa $\log\left(\frac{s}{c}\right)$ mal.

Worst-Case-Instanz

Es gibt Punktmengen, für die der Quadtree eine Höhe von $\Omega\left(\log\left(\frac{s}{c}\right)\right)$ besitzt.



Für zufällig verteilte Punkte ist die Höhe des Quadtrees eher $\log(n)$ und die Laufzeit für die Range-Query damit $O(n)$.

Überlegungen

Laufzeit lohnt sich (theoretisch) nicht

Problem: Wir teilen den Raum und nicht die Knoten auf.

Wie teilen wir die Punkte auf?

AVL-Bäume teilen 1-dimensionale Punkte auf…

Was tun in 2D?

Nächste Woche: KD-Trees