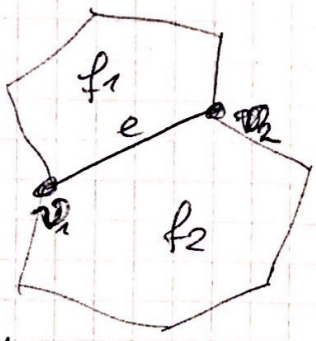
Terminologie / Grundbegriffe

Punkte durch Kleinbuchstaben

Inzident, adjazent:



Koinzident:

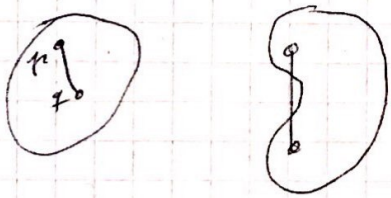
2 Pkte $p, q \in \mathbb{R}^d$ heißen koinzident $:\Leftrightarrow$

$\forall i: p_i = q_i$

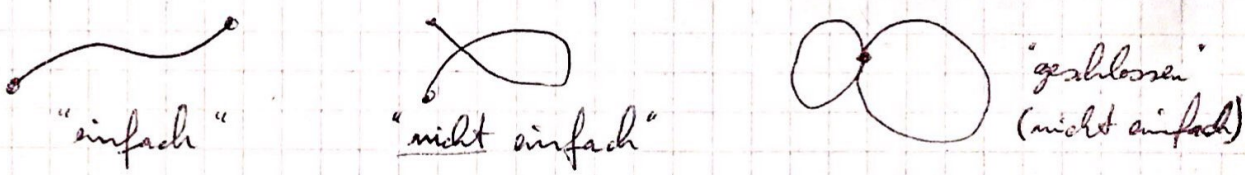
Konvex:

$K \subseteq \mathbb{R}^d$ "konvex" $:\Leftrightarrow$

$\forall p, q \in K: \overline{pq} \subseteq K$



Wege in der Ebene:



Jordan'scher Kurvensatz:

Sei w ein einfacher geschlossener Weg in $\mathbb{R}^2$.
Dann besteht $\mathbb{R}^2 \setminus w$ aus genau zwei, nicht zusammenhängenden Gebieten:
ein beschränktes G. (heißt "inneres Gebiet"),
und ein unbeschränktes G. ("äußeres").
Beide Gebiete haben als Rand w.

Quadtrees

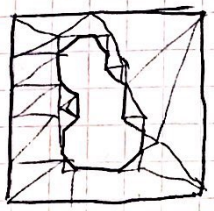
[Quelle: Buch von de Berg, Kreveld, Overmars, Schwarzkopf: "Comp. geom."]

Problem: "Meshing"

nicht Striping

gegeben: Polygonzug in 2D,

gesucht: Triangulierung der Ebene mit bst. "schönen" Eigenschaften.



analog in 3D

braucht man oft für FEM.

Was ist FEM?

Problem "Meshing" kann man gut mit Quadtrees lösen.

Quadtrees allg. praktische Datenstruktur.

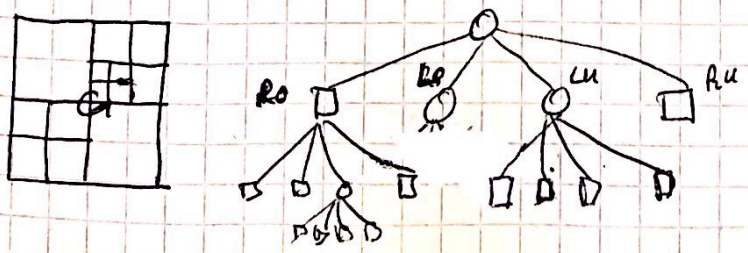
Vorerst Vereinfachung: Quadtrees für Punktmengen.

Quadtree := Tree; mit jeder innere Knoten hat 4 Kinder;
jeder Knoten korresp. mit Quadrat der Ebene;
Kinder ≙ Quadranten eines Knotens

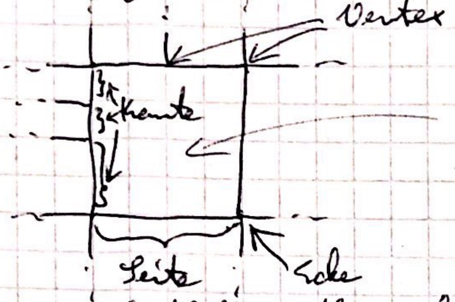
Subdivision: Blätter induzieren Unterteilung, Partitionierung der Ebene → "Quadtree ≙ Subdivision", oder
"Quadtree - Part." (engl. "quadtree subdivision")

Frage: Knoten eines (!) Levels induzieren ebenso eine Quadtree-Subdiv., ist regelmäßiges Gitter, falls Level voll besetzt.

Bsp.:



Terminologie:



(enthält evtl. mehr als 1 Kante)

oder Zelle aka Node
Quadrat kann mehr als 4 vertices haben!
(insbesondere, wenn innerer Knoten)
nur, falls innerer Knoten → Frage

Quadrate "benachbart" :⟺ gemeinsame Kante

Punkte im Quadtree speichern: Algo: construct Q.tree over Pts

NB: Q. kann noch viele andere Objs speichern. (Erinnerung: Warnock aus CG1)

Def. Quadrat zu Knoten v $q(v) := [x_v, x'_v] \times [y_v, y'_v]$.
often: v synonym to $q(v)$

Def. Quadtree Q:

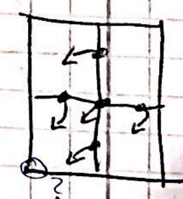
Geg. Menge Pkte $P \subseteq \mathbb{R}^2$

$Q(P) :=$ node $v \begin{cases} v, & v \text{ ist Blatt, falls } |P| \leq 1 \\ v \text{ mit Kindern } v_{RO}, \dots, v_{RU}, & |P| > 1 \end{cases}$

mit v_{RO} ist Qtree für $P(v_{RO}) := \{ p \in P \mid p_x > \frac{1}{2}(x_v + x'_v),\ p_y > \frac{1}{2}(y_v + y'_v) \}$

analog v_{LO} – –
v_{LU}
v_{RU}

Punkte auf den Kanten werden so zugeordnet:



Rekursive Def. für Quadtree → rek. Algo zum Aufbau eines Q.

[Daraus erkennt man auch schon, wie Pkte auf den äußeren Kanten des großen Quadrats zugeordnet werden:  über Rek.]

Frage: wie tief wird ein Quadtree?

Problem: Pktmenge wird nicht notw. gesplittet
Depth does not only depend on number of points, but also on "geometry"

Bsp.  P wird gar nicht aufgeteilt!

Frage: wovon hängt Tiefe des Quadtree ab?

Lemma: Depth

Sei P Menge von Pkte in Ebene,

s = Seitenlänge von $q(\text{root})$,

c = min. Pkt.abstand = $\min\{\|p_1 - p_2\| : p_1, p_2 \in P,\ p_1 \neq p_2\}$

$\Rightarrow$ Tiefe $d \leq \log\left(\frac{s}{c}\right) + \frac{3}{2}$.

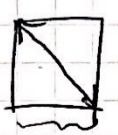
↑ log dualis

Bew.: oBdA $s = 1$

Seitenlänge eines Knoten v mit Tiefe $i = \frac{1}{2^i}$

max Dist innerhalb $v = \frac{\sqrt{2}}{2^i}$.

Betrachte innere Knoten auf Level i: $\frac{\sqrt{2}}{2^i} \geq c$ für interne Knoten,



← level i

weil interne Kn. ≥ 2 Pkte enthalten

$c \leq \min\{|p_1 - p_2| : p_1, p_2 \in \square\} \leq \frac{\sqrt{2}}{2^i}$

$\Rightarrow i \leq \log \frac{\sqrt{2}}{c} = \log \frac{1}{c} + \frac{1}{2}$

ist Bed. für Level innerer Knoten $\Rightarrow$ Bed. für Level von Blättern.

[Übung: wie tief wird Q.tree bei Punktmenge auf Gitter?]

Frage: wie lange dauert Konstruktion, wieviel Speicher?

Lemma: Complexity

Quadtree mit Tiefe d und n Punkten braucht

$O(n(d+1))$ Knoten und $O(n(d+1))$ Zeit.

entspricht $\log n$ Faktor bei binären Bäumen!

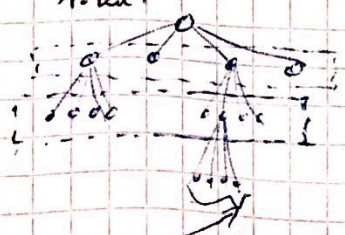
Binary full tree: $2d+1 \leq \#\text{nodes} \leq 2^{d+1} - 1$

Bew.:

$\#\text{Blätter} = (\#\text{innere Knoten}) \cdot 3 + 1$ (by induction)

$\Rightarrow$ genügt Schranke für innere Knoten.

1. Teil:



$\leftarrow$ $\#$ innere Knoten $\leq n$, denn pro Knoten mind. 2 Pkte, und auf jedem Schritt ist $\sum_v$ Pkte in innern Knv $\leq n$ Knoten überlappen nicht.

$\Rightarrow$ gilt für alle Tiefen $\Rightarrow$ Schranke

$n \cdot (d-1) + 2n \Rightarrow$ Schranke.

Pro Quadrupel von Blättern müssen ≥ 2 Blätter besetzt sein.

Bew. cont'd (2. Teil, Zeit):

Pro Knoten v braucht man $T(v) \in O(m)$ Zeit, m = # Elte in Knoten v.

Aber: # Elte pro Schicht $\leq n$

$\Rightarrow$ Σ-Zeit pro Schicht $\in O(n)$

$\sum_{\text{nodes } v \text{ on level}}$ Zeit für $v \in O(n)$

Häufige Operation: Nachbarzelle finden ("neighbor finding")

Gegeben: Knoten v

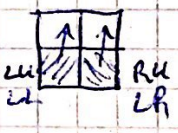
gesucht: v' oberer Nachbar von v
$\hat{=}$ oder links/rechts/unten

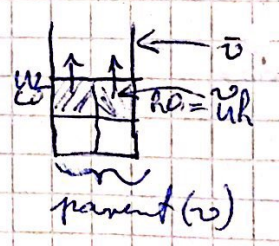
und so daß depth(v') $\leq$ depth(v)
$\hat{=}$ kann passieren, daß kein Knoten in der gewünschten Richtung mit gleicher Tiefe ex

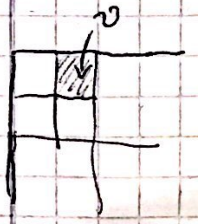
Also: North Neighbor (v)

Bem.: ab hier wissen wir, daß parent(v) existiert

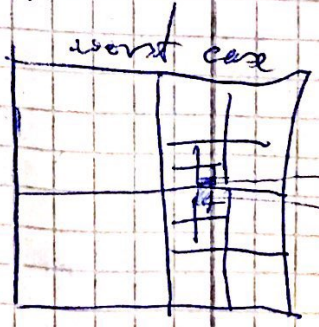
```
if v == root → return NULL
if v == UL-child von parent(v)
        → return LO-child von parent(v)
if v == RU-child → ...
v̄ := North Neighbor (parent(v))
if v̄ == NULL oder v̄ ist Blatt
        → return v̄
if v == LO-child
        → return UL-child von v̄
if v == LO-child ...
```







Also liefert nicht notw. Blatt! falls Blatt gesucht, muß man noch weiter nach unten laufen.



Laufzeit: $O(d)$ [$O(d+1)$?]

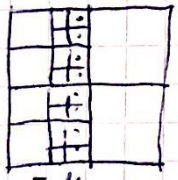
[Übung: Quadtree-Aufbau und Neighbor-Suche programmieren.]

Verallgemeinerte Nachbarsuche weiter hinten!

Balancierte Quadtrees:

unbalanciertes Bsp.:

oft unerwünscht, z.B. in der Anwendung Meshing für FEM.



Def. "balancierter Quadtree" :⇔

$|depth(v) - depth(v')| \leq 1$ für Nachbarn

Dann Q balanciert ⇔ Zwei benachbarter Quadrate unterscheidet sich höchstens um Faktor 2

Wie macht man balancierten Baum aus unbalancierten?

→ Füge Knoten ein, solange noch Nachbarn mit zu großer Tiefe anstoßen. Bsp.:

Algo: sketch verwalte Liste L aller Blätter

2 Schritte im wesentlichen:

1. Check, ob Blatt v gesplittet werden muß.
   Geht mit Neighbor Finding – Algo von vorhin
2. Wenn Blatt gesplittet wurde, checke ob seine Nachbarn auch gesplittet werden müssen.
   Auch mit Neighbor Finding.

Q: do we have to go deeper beneath the neighbor? → No

Balancierung kann man natürlich gleich während des Aufbaus eines Quadtrees machen, aber Darstellung und Analyse ist einfacher, wenn man es getrennt macht. Und i.A. kommt dasselbe raus.

Σ Übung: programmieren. Jeder einfache ausformulieren.]

Evtl. Frage: was passiert mit der Größe, wenn balanciert wird?

Dann: man könnte glauben, daß sich ein Split immer weiter ausbreitet ... — Glücklicherweise gilt 2

Lemma:

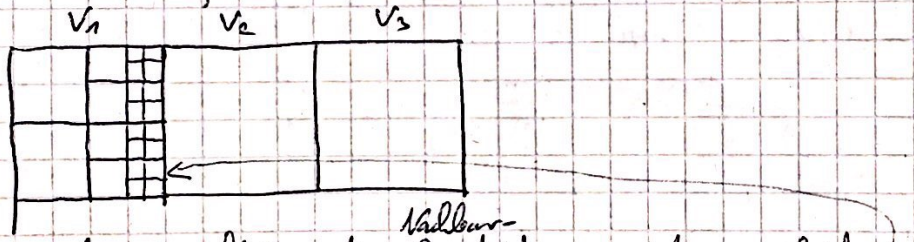
Sei Q Quadtree mit n Knoten, $\hat{Q}$ = balancierte Version davon.

Dann hat $\hat{Q}$ $O(n)$ Knoten,

und läßt sich in $O(n(d+1))$ konstruieren (Gesamtzeit).

Bew.:

1. $O(n)$ Knoten: wir beweisen, daß nur $O(n)$ splitting Operationen; daraus folgt Beh., da jede splitting Op. 4 neue Knoten anlegt.

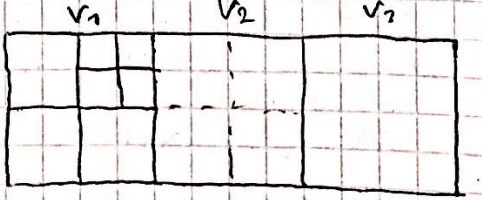


Beh.: egal wie klein die Quadrate an dieser Seite von v_2 sind, v_3 muß nie gesplittet werden. M.a.W.: Splits propagieren nicht beliebig weit.

Def: $D(v)$:= Tiefe des Teilbaumes unter v

Bew. tricky durch Induktion über Tiefe D des Teilbaumes unter v_1 von v_1:

Induktionsanfang: Base case

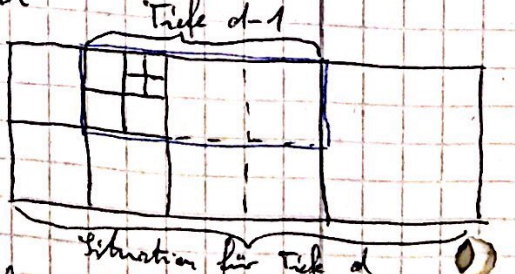


$D(v_2) = D(v_3) = 0$

$D(v_1) = 2$

$\Rightarrow$ v_2 wird 1x gesplittet, v_3 wird nicht gesplittet $\Rightarrow$ Beh.

Induktionsschritt: Inductive step Beh. gilt für Tiefen $D < d$



$D(v_1)$
Tiefe $(v_1) = d > 2$

$\Rightarrow$ v_2 wird (mind.) 1x gesplittet;

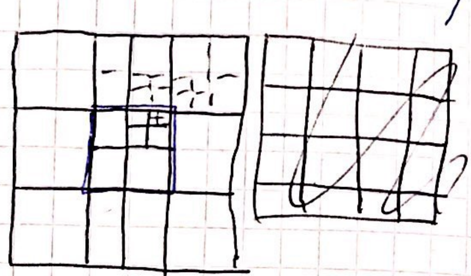
$D(\text{RO-child}(v_1)) = d-1$
Tiefe (RO-child (v_1)) $= d-1 \Rightarrow$

RO-child (v_2) wird nicht weiter gesplittet (vgl. Induktion)

$\Rightarrow$ v_3 wird nicht gesplittet.

(⇒ egal, wie oft ein Quadrat unterteilt wird (= welche Tiefe ein Quadrat / Quadtree-Zelle hat), es werden nur 8 Nachbar-Quadrate auf selben Level als Folge unterteilt.)

(Split kann „um die Ecke" propagieren.)



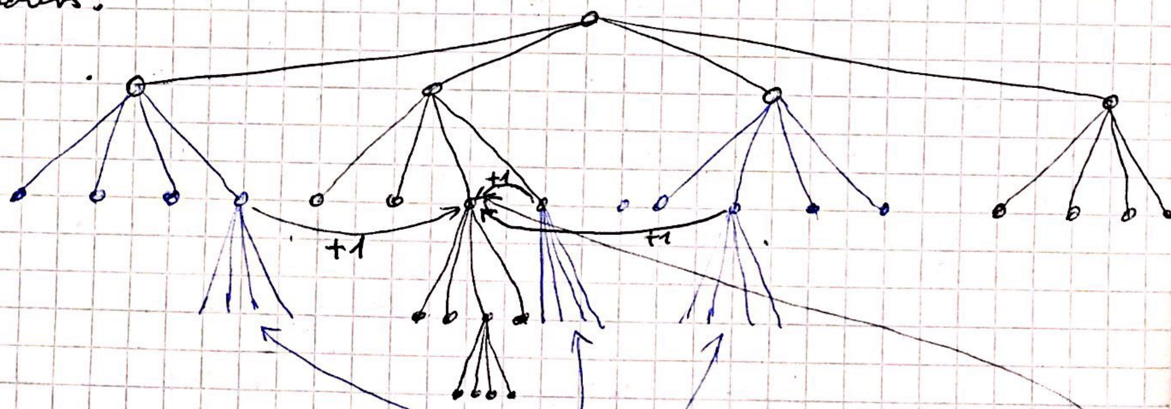
Mit dieser Beob. kann man jetzt die # Splits beschränken:

„alter Knoten" = Knoten im orig. Baum

„neuer Kn." = „ „ neuen, daraus hervorgehenden, balancierten Baum.

Split-Zähler (für alte Knoten) = wie oft hat ein alter Knoten einen Split verursacht. [neue Kn. brauchen keinen Split-Z.]

(Falls Knoten gesplittet wird, dann deswegen, weil einer der 8 Nachbarn zu tief ist → erhöhe den Split-Zähler dieses Verursachers.)



→ welche Knoten muß man unterteilen, weil dieser Knoten zu tief ist? z.B. diese – davon gibt es ≤ 8 Stück

⇒ der Split-Zähler jedes alten Kn. ≤ 8 am Ende, hat also ≤ 8·4 neue Knoten beigesteuert. ⇒ 1. Beh. des Lemmas.

2. Zeit:

Zeit pro Knoten ist $O(d+1)$ wegen konstanter Anzahl von Nachbarsuchen.; jeder Knoten kommt höchstens 1x dran ⇒ Beh.

Größe und Zeit läßt sich mit Schnitt-Knoten leichter zeigen.

Bemerkung: Die Operationen, die beim Balancieren gemacht werden, kann man natürlich auch gleich beim Aufbau des Quadtrees machen. d.h., wenn ein Blatt unterteilt wird, schaut man gleich nach, ob die Nachbarn entl. auch gesplittet werden müssen.