Kleiner Exkurs: Closest Points in 2D

Aufgabe:

Input
Gegeben: Menge Punkte $P = \{p_1, p_2, \ldots, p_n\} \in \mathbb{R}^2$

Output
Gesucht: Pktepaar $p_a, p_b \in P$,
das den kleinsten Abstand hat
p_a, p_b heißt "closest pair".

Naiver Algo hat $O(n^2)$ Aufwand

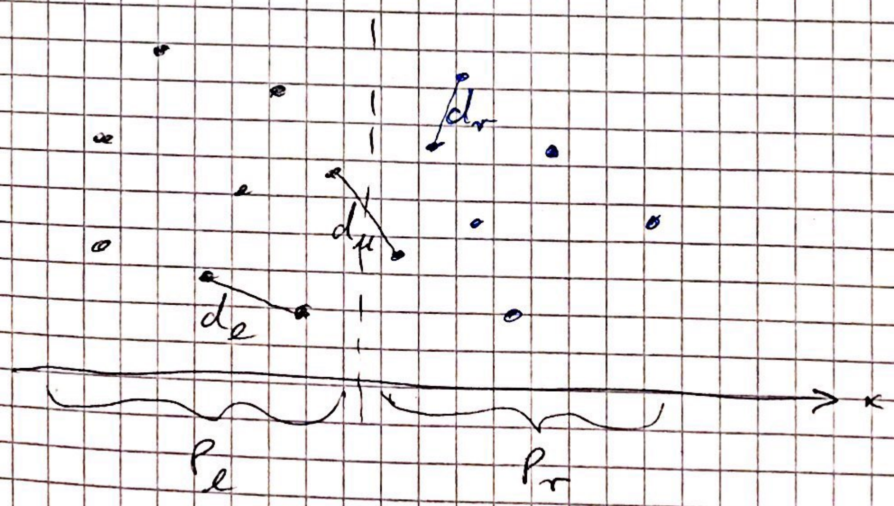
Bemerkung: das analoge Problem in 1D ist einfach
Lsg ist Sortieren $\rightarrow O(n \log n)$

Geht es auch in 2D in $O(n \log n)$?

Ansatz: Divide-and-Conquer

Idee:

- Sortiere P entlang x-Achse
- Closest Pair liegt entweder
  - in linker Hälfte von P, oder
  - rechter Hälfte, oder
  - p_a in linker, p_b in rechter Hälfte



Closest-pair (P)

Phase 1: sortiere P bzgl. x

$\rightarrow \underbrace{p_1, \dots, p_{n/2}}_{P_\ell}, \underbrace{p_{n/2+1}, \dots, p_n}_{P_r}$

Phase 2: Rekursion

Closest-pair(P_ℓ) $\Rightarrow d_\ell$

bestimme minimale Abstände in P_ℓ, P_r $\rightarrow d_\ell, d_r$

Phase 3:

$d := \min(d_\ell, d_r)$

Betrachte nur Pkte $\bar{P} = \{ p \mid p_x \in [m-d, m+d] \} \subseteq P$, wobei m der Median der x-Koord ist (= x-Koord von $p_{n/2}$ oder $p_{n/2+1}$)

Bestimme Closest Pair und d_u in $\bar{P}$

Lösung ist $\min(d_\ell, d_r, d_u)$ und zugehöriges Pkte-Paar

Details zu Phase 3:

Sortiere $\bar{P}$ bzgl. y-Koord

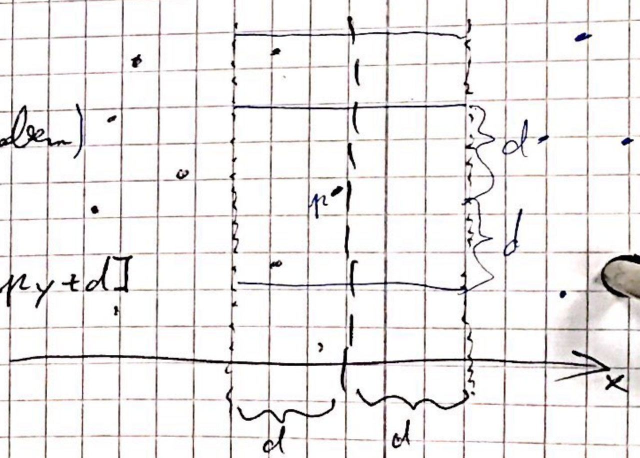
Betrachte alle $p \in \bar{P}$ (von unten nach oben)

Für jedes solche p: betrachte

(*) alle $q \in \bar{P}$ mit $q_y \in [p_y - d, p_y + d]$

Berechne alle Abstände dieser Paare $(p, q) \Rightarrow d_u = \min$

(Anm.: Pointer für q kann von p aus nach oben/unten "wandern")



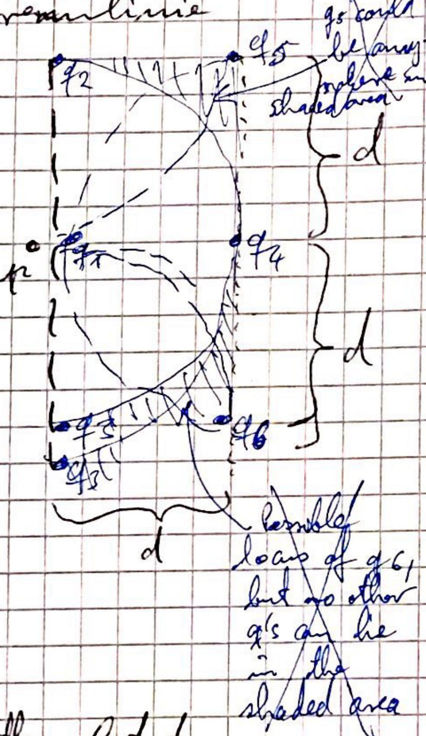
Behauptung:

In der inneren Schleife (*) werden max 6 q's betrachtet

$\Rightarrow$ Aufwand für $\underset{\text{Phase 3}}{P} \in \underset{\text{Sortieren}}{O(n \log n)} + \underset{\text{Schleife über } \overline{P}\ p\text{'s}}{O(n)}$

Beweis der Behauptung:

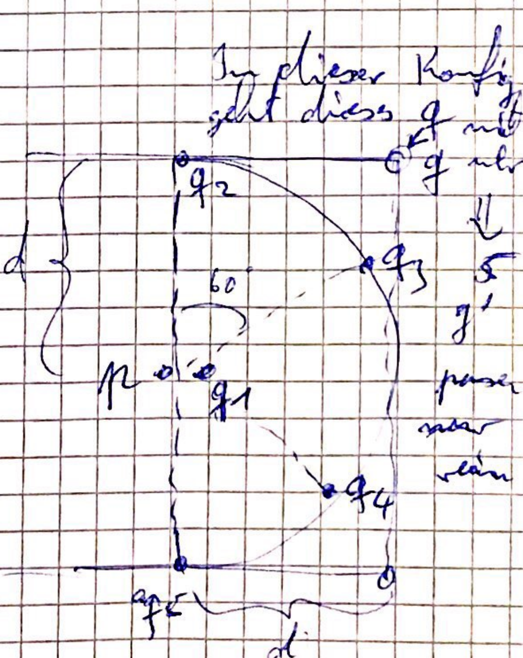
OBdA p links, q rechts der Trennlinie



1. in worst case könnte ein q_1 ganz dicht auf der "andren" Seite liegen
2. weitere q's müssen mindestens Abstand d von q_1 haben $\Rightarrow q_2, q_3, q_4$
3. weitere q's müssen mind. Abstand d von $q_2 - q_4$ haben $\Rightarrow q_5, q_6$
4. weitere q's brauchen nicht betrachtet werden, da diese sonst schon Abstand $> d$ von p hätten

Alternative argument:

points within $d \times 2d$-box ≤ 6, if all pairs of points within this box must be at least d apart



Zeitaufwand:

$$T(n) = 2T\left(\frac{n}{2}\right) + O(n \log n) + O(n \log n)$$

$\uparrow$ Phase 2 $\quad \uparrow$ Phase 1 $\quad \uparrow$ Phase 3

$\Rightarrow T(n) \in O(n \log^2 n)$

Erweiterung auf $\mathbb{R}^d$: $T(n) \in O(n \log^{d-1} n)$

Verbesserung:

- Sortierung bzgl. x muß man nur 1x machen (Preprocessing)
- Weiteres Preprocessing: Sortiere P bzgl. y
  $\rightarrow$ speichere in Liste L
- Zur Laufzeit:
  in Phase 2, vor Rekursion, teile L auf in L_l und L_r, so daß y-Sortierung erhalten bleibt, und $L_l = P_l$ und $L_r = P_r$
  (geht mit einem Pass durch L)

$\Rightarrow T(n) = 2T\left(\frac{n}{2}\right) + O(n) = O(n \log n)$

Bemerkung: 1. sogar in $\mathbb{R}^d$ geht Closest Pair in Zeit $O(n \log n)$
(Ist eher selten, daß man nicht die "curse of dimensionality" hat)

2. In 2D kann man die Anzahl der Distanz-Berechnungen auf 2 pro Pkt drücken

["An Optimized ..." Pereira & Lobos, 2015]