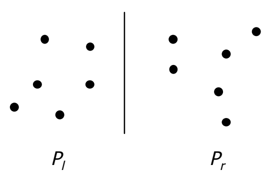
Das Closest Points Problem

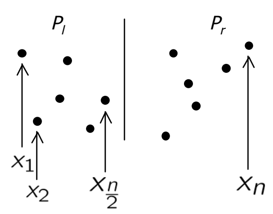
- Gegeben: Menge von n 2D-Punkten
- Gesucht: das Paar, das am dichtesten beieinander liegt
- Offensichtlich gibt es $\frac{n(n-1)}{2}$ Paare, die Komplexität eines naïven Algo ist also $O(n^2)$
- Bemerkung:
  - 1D-Version: wird gelöst durch Sortieren
  - Komplexität $O(n \log n)$
- Mit Divide-and-Conquer lässt sich auch für 2D-Fall $O(n \log n)$ erreichen

Divide-and-Conquer-Algorithmus

- Idee:
  - sortiere Punkte nach ihrer x-Koordinate und teile zur Hälfte,
  - das dichteste Paar ist entweder in einer der Hälften oder hat in jeder Hälfte ein Mitglied
- Phase 1: sortiere die Punkte nach ihrer x-Koordinate

$$p_1, p_2, \ldots, p_{\frac{n}{2}}, p_{\frac{n}{2}+1}, \ldots, p_n$$

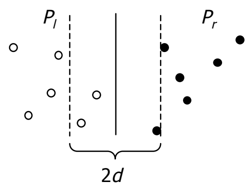


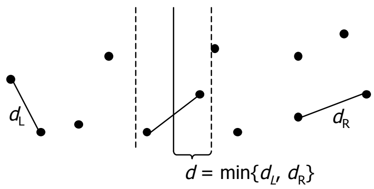


- Phase 2:
  - berechne rekursiv das dichteste Paar und die minimalen Abstände d_l und d_r in

$$P_l = \{p_1, p_2, \ldots, P_{\frac{n}{2}}\} \text{ und}$$
$$P_r = \{P_{\frac{n}{2}+1}, \ldots, p_n\}$$

- Phase 3: finde das dichteste Paar (◦,•) im "Band" um die Mitte mit der Breite 2*d*, wobei bekannt ist, daß kein (◦,◦)- oder (•,•)-Paar dichter zusammen ist als *d*

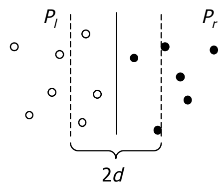





```
def close_pts( A ):
  sortiere A nach der x-Koordinate
  n = len( A )
  dL = close_pts( A[1:n/2] )          # T(n/2)
  dR = close_pts( A[n/2+1:n] )        # T(n/2)
  d = min( dL, dR )
  dM = suche Lösung in der Mitte      # ?
  return min( dL, dR, dM )
```

Phase 3

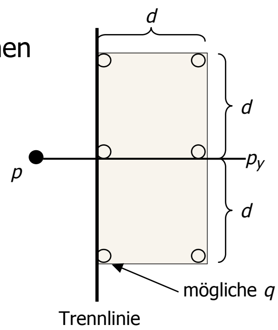
- Sortiere Punkte innerhalb des "Bandes" nach *y*-Koordinaten
- Gehe Punkte im Band der Reihe nach durch
- Für jeden solchen Punkt *p* betrachte alle Punkte *q* aus dem Band, die
  1. auf der "anderen" Seite der Trennlinie liegen
  2. deren *y*-Koordinate im Intervall $[p_y-d, p_y+d]$ liegen
- Pointer für *p* geht linear, Pointer für *q* "oszilliert"
- Bestimme alle Abstände und wähle den kleinsten
- Behauptung: zu jedem *p* aus dem Band kommen nur maximal 6 Punkte *q* aus dem Band in Frage → Aufwand für das gesamte Band ist $O(n) + O(n \log n)$



Beweis

- Ann.: p und q sind potenzielle dichteste Punkte aus dem Band

1. Es gilt: *p* muß (oBdA) links und *q* rechts von der Trennlinie liegen
2. *q* kann nicht rechts außerhalb des Bandes liegen
3. nur Punkte mit *y*-Koordinate im Intervall $[p_y-d, p_y+d]$ können Partner von *p* sein
4. Keine 2 Punkte im grauen Rechteck können dichter als *d* aneinander liegen!
5. Max 6 Punkte mit dieser Eigenschaft können in das Rechteck gepackt werden



Zeit-Komplexität

- Einzelschritte:
  1. Divide = Partitionieren (Sortieren nach x-Koordinate) → $O(n \log n)$
  2. Rekursion= $2T\left(\frac{n}{2}\right)$)
  3. Merge = Sortieren nach y-Koordinaten + konstanter Aufwand pro Punkt im Band → $O(n \log n) + O(n)$
- Annahme: $n = 2^k$

$$
\begin{aligned}
T(1) &= 1 \\
T(n) &= 2T\left(\tfrac{n}{2}\right) + 2n\log n \\
&= 4T\left(\tfrac{n}{4}\right) + 4\left(\tfrac{n}{2}\right)\log\left(\tfrac{n}{2}\right) + 2n\log n \\
&= 4T\left(\tfrac{n}{4}\right) + 2n(\log n - 1) + 2n\log n \\
&\quad \ldots \\
&= 2^k T\left(\tfrac{n}{2^k}\right) + 2n\big(\log n + (\log n - 1) + \cdots + (\log n - k + 1)\big) \\
&= n + 2n(1 + 2 + \cdots + \log n) = n + 2n\tfrac{(\log n + 1)\log n}{2} \\
&\in O\big(n\log^2 n\big)
\end{aligned}
$$

Bemerkungen

- Verbesserung:
  - Sortierung nach x-Koord muß man nur 1x am Anfang machen
  - Preprocessing-Schritt: sortiere alle Punkte nach ihrer y-Koordinate
  - Teile die sortierte Liste vor den rekursiven Aufrufen in zwei Unterlisten für die linke und die rechte Hälfte, wobei die Sortierung nach y-Koordinaten erhalten bleibt

  → Komplexität: $O(n \log n)$
- Es gilt sogar:
  Das Closest-Pair-Problem für k-dim. Punkte läßt sich in Zeit $O(n \log n)$ lösen!
- (Bemerkung: Häufig ist das nicht der Fall, d.h., Algos, die in 2D effizient sind, sind im k-dim. nicht mehr effizient [*curse of dimensionality*])