Algo 6.00 4.1

Buch: Algorithm Compendia, TIK
2. Auflage
+ wirken Abschnitte der Vorlesung

Sweep: 2.2.3

\[ \text{Sp: } \begin{bmatrix} - \frac{3}{2} & 2 & -1; \frac{5}{2} & 11 & -20 \end{bmatrix} \]
\[ \text{maxSofa } 10 \]

Aktivierung: 
\[ \begin{bmatrix} 3 \cdot -2 \cdot -1 \cdot 0 \cdot 1 \cdot 9 \cdot 3 \cdot -1 \cdot 3 \cdot -1 \cdot 8 \end{bmatrix} \]

\[ \text{gesucht: maximale Teilsumme konsekutiver Felder} \]
\[ \max \left( \max_{1 \leq k \leq n} \sum_{j=1}^{k} (A \cdot \cdot J, 0) \right) \]

erster Ansatz

\[ \text{MaxSofa: } 0 \]
\[ \text{foss i := } 1 \text{ to } n \]
\[ \text{foss k := } 1 \text{ to } n \]
\[ \text{sum := } 0 \]
\[ \text{foss } i := 1 \text{ to } k \]
\[ \text{sum := sum + AIj.J} \]
\[ \text{MaxSofa := max (MaxSofa, sum)} \]

\[ \text{land = } k \text{ vor } n \]
Algorithmus

**Anläufe: Summe fachseitig**

\[
\text{Max So Far} := 0 \\
\text{for } i := 1 \rightarrow n \\
\quad \text{sum} := 0 \\
\text{for } k := 1 \rightarrow m \\
\quad \text{sum} := \text{sum} + A(i,k) \\
\text{Max So Far} := \max (\text{Max So Far}, \text{sum})
\]

**Divide & Conquer:**

**logarithmic**

**nach baesser:**

**Sweep:** zu Beginn muss die Eigenschaft für Objekte durchlaufen werden, die von links nach rechts unterhalb sich befinden. Die Eigenschaft der linkssten betroffenen Objekte wird festgehalten, und die Auswirkungen dieser Eigenschaften werden entsprechend verfolgt.

Müssen nicht nur Max So Far, sondern auch Max Ending Here unterhalten.

Wie kann sich Max Ending Here verändern?

\[
\begin{pmatrix}
9 & -1 & 3 & 4 & 8 \\
-7 & 9
\end{pmatrix}
\]

und

\[
\begin{pmatrix}
9 & -1 & 3 & 4 & 8 \\
-7 & 9
\end{pmatrix}
\]

bleibt...
Algorithm 1.3
\[
\begin{align*}
\text{Max So Far} & := 0 \\
\text{Max Ending Here} & := 0 \\
\text{for } i := 1 \text{ to } n & \\
\text{Max Ending Here} & := \max(\text{Max Ending Here} + A[i], 0) \\
\text{Max So Far} & := \max(\text{Max So Far}, \text{Max Ending Here})
\end{align*}
\]

\[\Theta(n)\]

John Bentley: Programming Pearls

Sheep in 2D - Closest Pair Problem

Given: \(n\) Punkte in der Ebene

Geendet: kleinster Abstand zwischen zwei Punkte.

Algorithmus: Geordnet Koodinaten \((x_p, y_p) = \Phi\)

kann euklidische Abstände ausrechnen:

\[d(P, Q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}\]

hier: alle \(\binom{n}{2}\) Abstände berechnen \(\sim n^2\) Minimum bestimmen
Sweep ?

objekte: Punkte

Sweep line L

Beobachtung: Wenn einer Punkte r von Sweepline L getroffen wird (Ereignis!),
dann ist r mit Punktspunkte p in linken Halbraum um r
mit Punkt r M der alle Minimum M unterschieden

kreismitten sind schief:

Idee: Verschiebung:
statt Halbraue: Box zu

testen alle Punkte p in Box,
Ob sie zu r Abstand < M haben

2 Problem: (i) wie viele Punkte können in des Box liegen (Vermutung: 6)
(ii) wie findet man sie?
Lemma: In einer Isox können maximal 6 Punkte mit paarweiser Abstand ≥ M liegen.

Beweis: In jede Mini-box kann höchstens 1 Punkt liegen. Dann: 2 Punkte in einer Mini-box haben höchstens den Abstand

\[
\sqrt{\left(\frac{x-M}{2}\right)^2 + \left(\frac{y-M}{2}\right)^2} = \sqrt{\frac{16 + a}{36}} \\
\leq \frac{5}{6} \\
M < M.
\]

Lemma

zu (ii): Wie berechnen die Punkte in der Isox?

Wenn L auf neuen Punkt \( r = (x, y_r) \) stößt:
Benichte alle Punkte im Strifen mit y-Koordinate \( y_r - M \leq y_p \leq y_r + M \).
Algorithmus für Closest Pair Problem im $\mathbb{R}^2$

Sortiere Punkte nach $x$-Koord. $O(n \log n)$
bewege Sweepline L von links nach rechts

Wenn neuer Punkt $r$ getroffen:

- Suche alle Punkte $q$ mit $x_q < x_r - M$ aus Strafe

Finde alle Punkte $p$ im Sträflin mit $y_r - M \leq y_p \leq y_r + M$ $O(\log n + k)$

Teste, ob für ein solches $p$ $d(p,r) \leq c$

- $|p - r| < M$

- Falls ja: $M := |p - r|$

- Falls nein: in Sträflin enden: $O(\log n)$
Theorem 2.4: Der dichteste Abstand von n Punkten im \( \mathbb{R}^2 \) lässt sich in Zeit \( O(n \log n) \) bestimmen.

Frage: Was passiert hier:

1. Möglichkeit
   Algorithmus anpassen

2. Möglichkeit
   "Allgemeine Lage" verlassen