Grundlagen der Algorithmischen Geometrie

Elmar Langetepe
University of Bonn

Organisatorisches

- Bachelor BA-INF 114 Wahlpflicht, 4-6 Semester

  – Vorlesung 4 SWS, 5.5 LP, Übung 2 SWS, 3.5 LP
  – Prüfungsform: Mündlich
  – Studienleistungen: Erfolgreiche Übungsteilnahme

- Grundlage: Algorithmische Geometrie (Rolf Klein), Folien, Tafel, Wiederholung
- Mailingliste

  `https://mailbox.informatik.uni-bonn.de/mailman/listinfo.cgi/vl-alggeom`

Organisatorisches

- Übungen: (TVS)
  Montag 10-12, Raum A6b
  Montag 12-14, Raum A6b
  Dienstag 14-16, Raum A7b
  Freitag 14-16, Raum A6b
- Wöchentlich, Beginn: 20.04, Übungsblattausgabe 13.04
- Aufgaben, Lösungen schriftlich Einzelabgabe, Korrektur, Besprechung, 50 Prozent-Regel, Lösung präsentieren
- Bachelorarbeiten (Themenhinweise)
- Projektgruppe Computational Geometry
  (Java,`www.geometrylab.de`)

Ziele der Vorlesung

Ziele der Vorlesung

- Fachlich: Erlernen und Einüben grundlegender und typischer Techniken der algorithmischen Geometrie und ihre Anwendung auf praxisrelevante Probleme

- Integrative Schlüsselkompetenzen: Präsentation eigener Lösungsansätze und zielorientierte Diskussion im Rahmen der Übungen

- Beispiel: Voronoi Diagramm, Lokalisationsplanung!

Kapitel 2 Sweep Technik

- Ausfegen der Ebene mit Sweepline
- Komplexitätsreduktion: n-Dim nach (n-1)-Dim
- Sortieren und gem. Sort. fegen
- SSS (Sweep-Status-Struktur): Invariante in der Nähe der Sweepline
- ES (Ereignisstruktur): Haltepunkte der Sweepline
- Ereignisverarbeitung: Aktualisierung SSS
- Laufzeitanalyse: (# Ereignisse) x Kosten(Verab)
- Korrektheit Ergebnis: Invariante erfüllt
- Einfache Beispiele, komplexe Beispiele

Algorithmus Maximum

```
MaxSoFar := q[1];
for j := 2  to n do
      if MaxSoFar < q[j]
      then MaxSoFar := q[j];
write("Das Maximum ist ", MaxSoFar)
```

Algorithmus Closest Pair

$MinDistSoFar := x'[2] - x'[1];$
$ClosPos := 2;$
for $j := 3$ **to** n **do**
    if $MinDistSoFar > x'[j] - x'[j-1]$
    then
        $MinDistSoFar := x'[j] - x'[j-1];$
        $ClosPos := j;$
write("`Dichtestes Paar:`", $x'[ClosPos - 1], x'[ClosPos]$)

Laufzeit: Closest Pair

Laufzeit: Closest Pair

Korollar 2.1 Die Berechnung des Closest Pairs von n reellen Zahlen hat Zeitkomplexitäzt $O(n \log n)$.

Laufzeit: Closest Pair

Korollar 2.1 Die Berechnung des Closest Pairs von n reellen Zahlen hat Zeitkomplexitäzt $O(n \log n)$.

- Real RAM Modell
- Reelle Zahlen
- Standard Arithmetik: $+, -, *, /$; (Erweiterte Arithmetik)
- Exakte Vergleiche: $<, \leq, =, \neq, \geq, >$
- 1 Speicherplatz je Zahl
- Kosten einer Berechnung: konstant
- Analyse der Laufzeit: O/Ω-Notation

Beispiel Variation

Beispiel Variation

-1	2	4	-3	9	-7	-6	5	-8

Beispiel Variation

-1	2	4	-3	9	-7	-6	5	-8

- Naives Verfahren? Laufzeit?
- Verbesserung
- Divide and Conquer
- Optimal?

MaxSoFar, MaxEndingHere

MaxSoFar, MaxEndingHere

-1	2	-3	5	-2	-1	6	-2	0	-1	-9	3	-1	3	?

MaxSoFar, MaxEndingHere

-1	2	-3	5	-2	-1	6	-2	0	-1	-9	3	-1	3	?

- $MaxSoFar = 8$
- $MaxEndingHere = 5$
- Was passiert beim nächsten Element?

Algorithmus: Maximum Subvektor!

Algorithmus: Maximum Subvektor!

MaxSoFar $:= 0$;
MaxEndingHere $:= 0$;
for $j := 1$ **to** n **do**
    MaxEndingHere $:= \max(0, MaxEndingHere + Variation[j])$;
    MaxSoFar $:= \max(MaxSoFar, MaxEndingHere)$;
write("`Maximale Teilsumme:`", *MaxSoFar*)

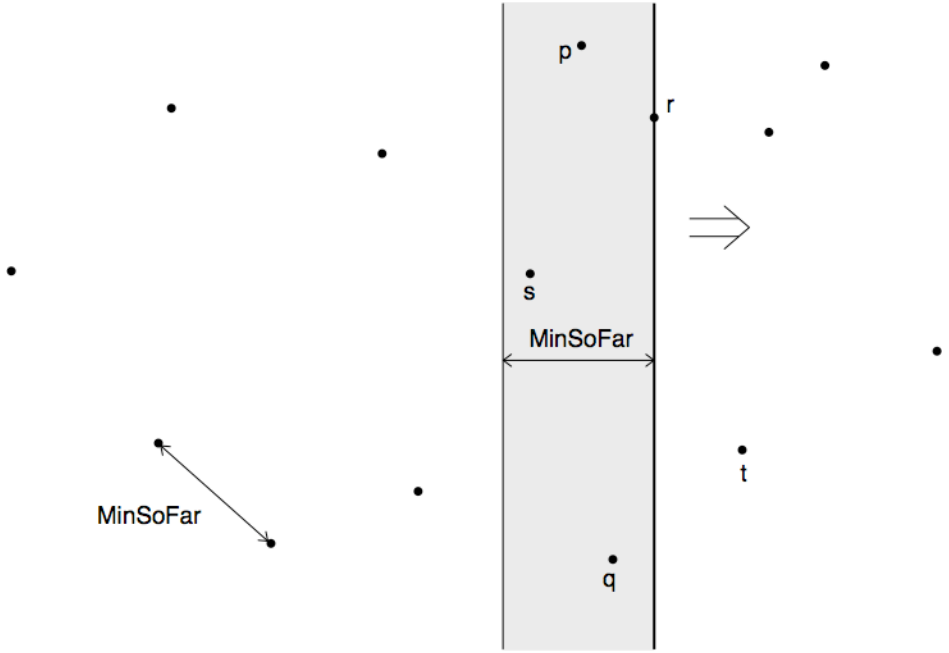
Algorithmus: Maximum Subvektor!

MaxSoFar := 0;
MaxEndingHere := 0;
for $j := 1$ **to** n **do**
    MaxEndingHere := $\max(0, \textit{MaxEndingHere} + \textit{Variation}[j])$;
    MaxSoFar := $\max(\textit{MaxSoFar}, \textit{MaxEndingHere})$;
write("`Maximale Teilsumme:`", *MaxSoFar*)

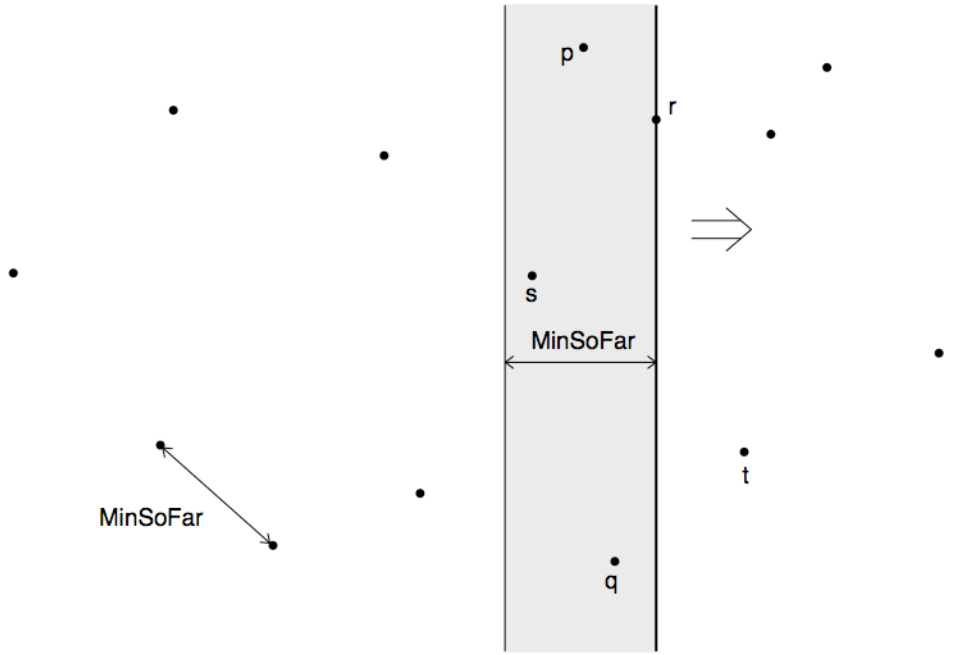
Theorem 2.2 Die Berechnung des Maximum Subvektors n konsekutiver reeller Zahlen hat Zeitkomplexitäzt $\Theta(n)$.

Closest Pair: SSS!

Closest Pair: SSS!



Closest Pair: SSS!



1. Neuer Punkt!
2. Verlassen Streifen
3. Anpassen Streifen

Closest Pair: Initialisierung!

(* Initialisierung *)
sortiere die n Punkte nach aufsteigenden X-Koordinaten
und füge sie ins Array P ein;

(* alle Punkte sind paarweise verschieden *)
füge $P[1], P[2]$ in *SSS* ein;
MinSoFar $:= |P[1]P[2]|$;
links $:= 1$;
rechts $:= 3$;

Closest Pair: Sweep!

(* Sweep *)
while (* I_1 gilt *) *rechts* $\leq n$ **do**
    if $P[$*links*$].x + MinSoFar \leq P[$*rechts*$].x$
    then (* alter Punkt verläßt Streifen *)
        entferne $P[$*links*$]$ aus *SSS*;
        links := *links* $+ 1$
    else (* neuer Punkt betritt Streifen; I_2 gilt *)
        MinSoFar := *MinDist*(*SSS*, $P[$*rechts*$]$, *MinSoFar*);
        füge $P[$*rechts*$]$ in *SSS* ein;
        rechts := *rechts* $+ 1$;
(* Ausgabe *)
write("`Der kleinste Abstand ist`", *MinSoFar*)

Laufzeit $MinDist(SSS, P[rechts], MinSoFar)$

Laufzeit $MinDist(SSS, P[rechts], MinSoFar)$

Lemma 2.3: Sei $M > 0$ und P eine Menge von Punkten im $\mathbb{R}^2$ von denen je zwei den Abstand $\geq M$ haben. Ein Rechteck mit Kantenlängen M und $2M$ enthält höchstens 10 Punkte.

Laufzeit $MinDist(SSS, P[rechts], MinSoFar)$

Lemma 2.3: Sei $M > 0$ und P eine Menge von Punkten im $\mathbb{R}^2$ von denen je zwei den Abstand $\geq M$ haben. Ein Rechteck mit Kantenlängen M und $2M$ enthält höchstens 10 Punkte.

- DS: AVL-Bäume
- Einfügen, Entfernen $O(\log n)$
- Neues Minimum $O(\log n)$

Laufzeit $MinDist(SSS, P[rechts], MinSoFar)$

Lemma 2.3: Sei $M > 0$ und P eine Menge von Punkten im $\mathbb{R}^2$ von denen je zwei den Abstand $\geq M$ haben. Ein Rechteck mit Kantenlängen M und $2M$ enthält höchstens 10 Punkte.

- DS: AVL-Bäume
- Einfügen, Entfernen $O(\log n)$
- Neues Minimum $O(\log n)$

Theorem 2.4: Der minimale Abstand aller Paare einer n-elementigen Punktmenge in der Ebene läßt sich in Zeit $O(n \log n)$ bestimmen.