Zusammenfassung Konvexe Hülle

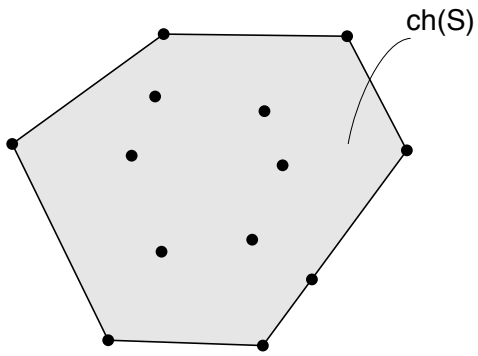
Elmar Langetepe
University of Bonn

Konstruktive Definition!

Lemma 4.1 Sei S eine Menge von n Punkten in der Ebene, dann ist $ch(S)$ ein konvexes Polygon mit Eckpunkten aus S.

Induktiver Beweis!

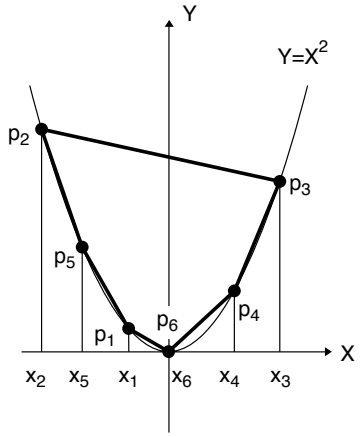
$$ch(S) = \bigcap_{\substack{K \supseteq S \\ K \text{ konvex}}} K$$



Untere Schranke!

- Diskrete Aufgabenstellung: Konvexes Polygon
- Bestimme Rand mit Knoten sortiert in CW (CCW) Ordnung! Datenstruktur!
- Sortieren $\leq_n$ Konvexe Hülle

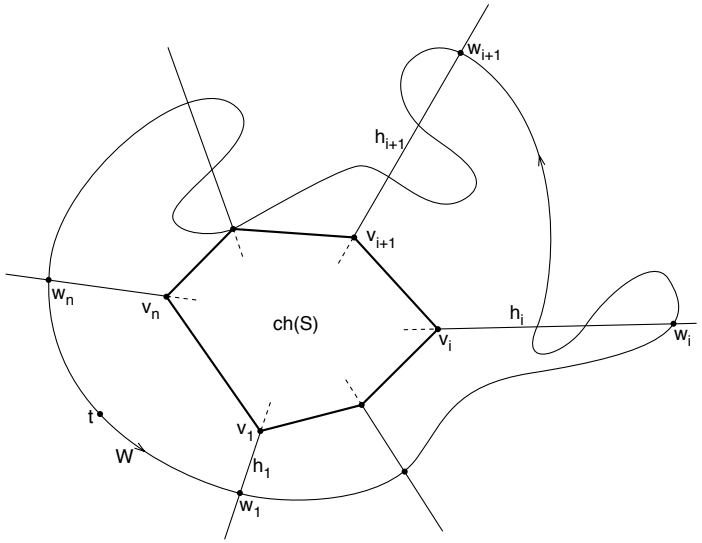
Lemma 4.2: Die Konstruktion der konvexen Hülle von n Punkten in der Ebene hat Zeitkompleximtät $\Omega(n \log n)$.



Strukturelle Eigenschaft!

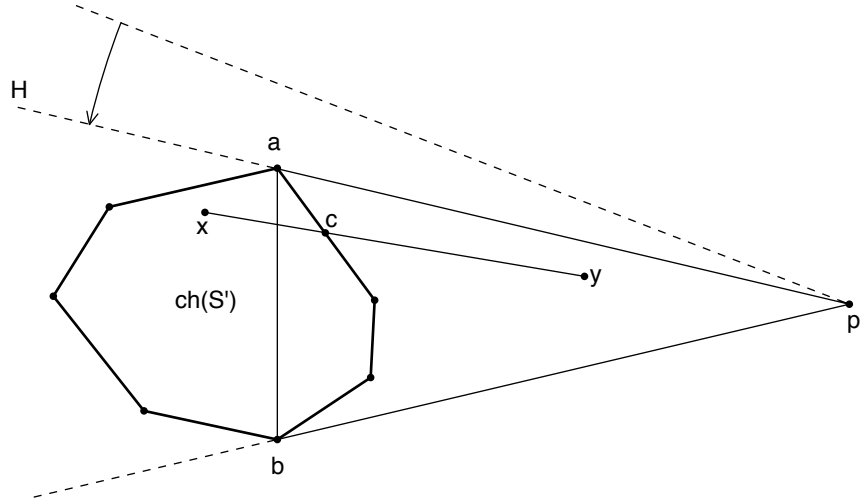
Lemma 4.3: Punktmenge S gegeben. Sei W ein einfacher, geschlossener Weg, der den Rand von $ch(S)$ umschließt. Dann ist der Rand von $ch(S)$ höchstens so lang wir W.

$|v_i v_{i+1}| \leq |w_i w_{i+1}| \leq |W_{i,i+1}|$ wegen Winkel!



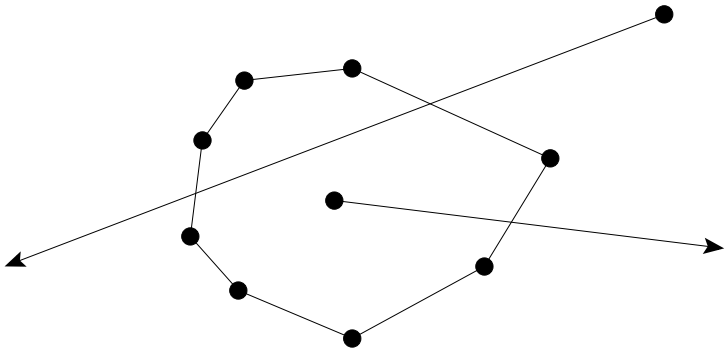
Inkrementelle Konstruktion!

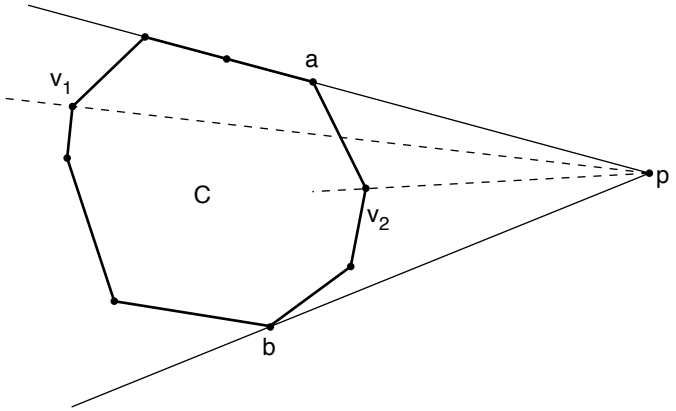
- Beweis Lemma 4.1, Einfügen eines neuen Punktes
- Problem I: Innerhalb/außerhalb
- Problem II: Tangentenbestimmung, Vereinigung



Problem I und II lösen

- Punkt p, bilde Strahl in eine Richtung, teste alle Segmente auf Schnitt: $O(n)$
- Von p aus sukzessive auf Tangente testen (Orientationtest): $O(n)$





Laufzeitverbesserung durch balancierte Bäume

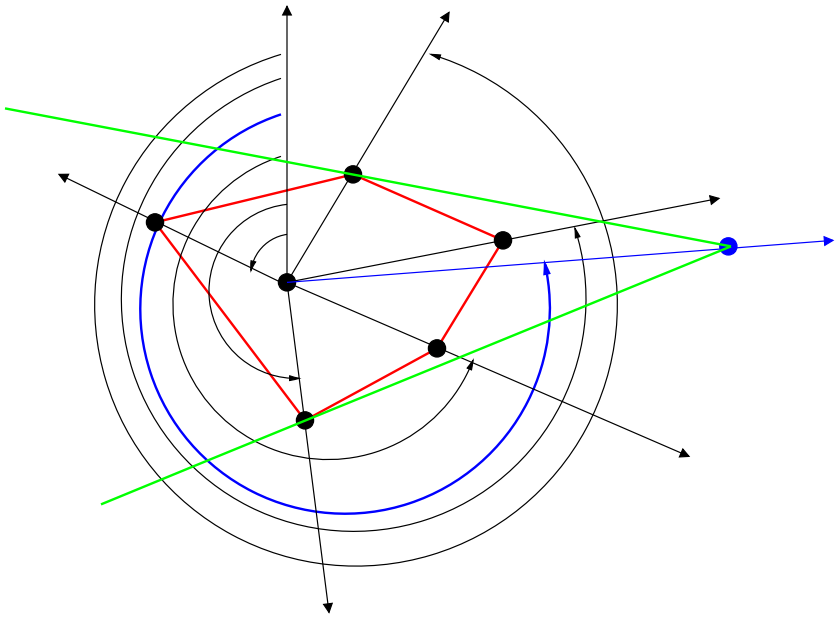
Theorem 4.5 Die konvexe Hülle einer Menge von n Punkten in der Ebene zu konstruieren hat Zeitkomplexität $\Theta(n \log n)$ mit linearem Speicher.

Beweis: Verbesserung Problem I/II in jeweils $O(\log n)$ statt $O(n)$

$O(n \log n)$ statt $O(n^2)$!

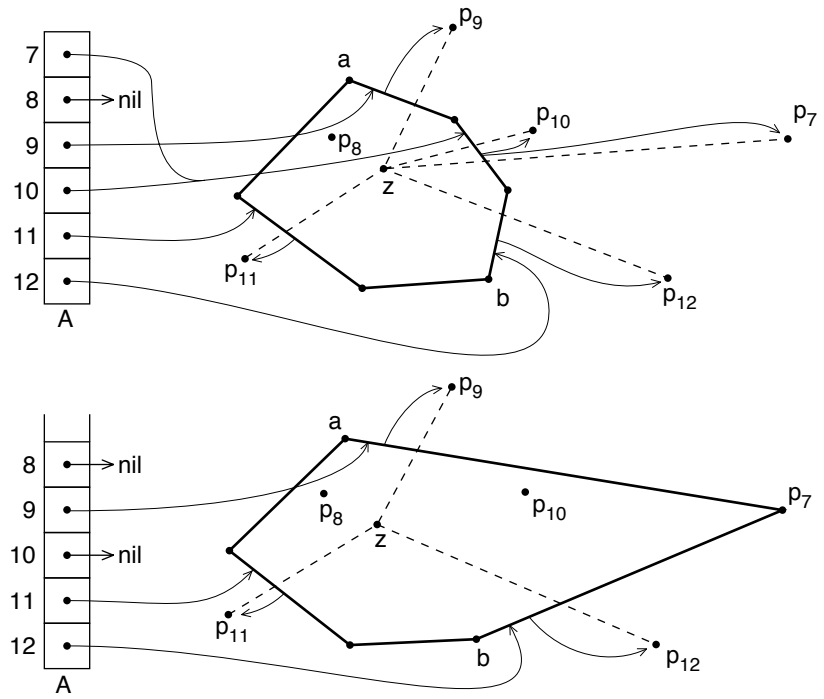
Verbesserung Problem I+II: Balancierte Bäume

- Sektoren in einem Baum abspeichern, nach Winkeln. Query Sektor/innerhalb: $O(\log n)$
- Tangenten:
  (Hülle) Punkte entfernen/Punkt einfügen $O(n)$ (amortisiert)
  (DS) Entfernen der Winkel, Einfügen Winkel, je $O(\log n)$



Ohne balancierte Bäume auskommen?

- Einfaches Verfahren, das *randomisiert* sehr gut funktioniert
- Randomisierte inkrementelle Konstruktion, alle Punkt-Reihenfolgen gleich wahrscheinlich
- Idee: Speichern, wo neue Punkte liegen (Zeiger-Array, Kante)



Aktualisierungsaufwand!

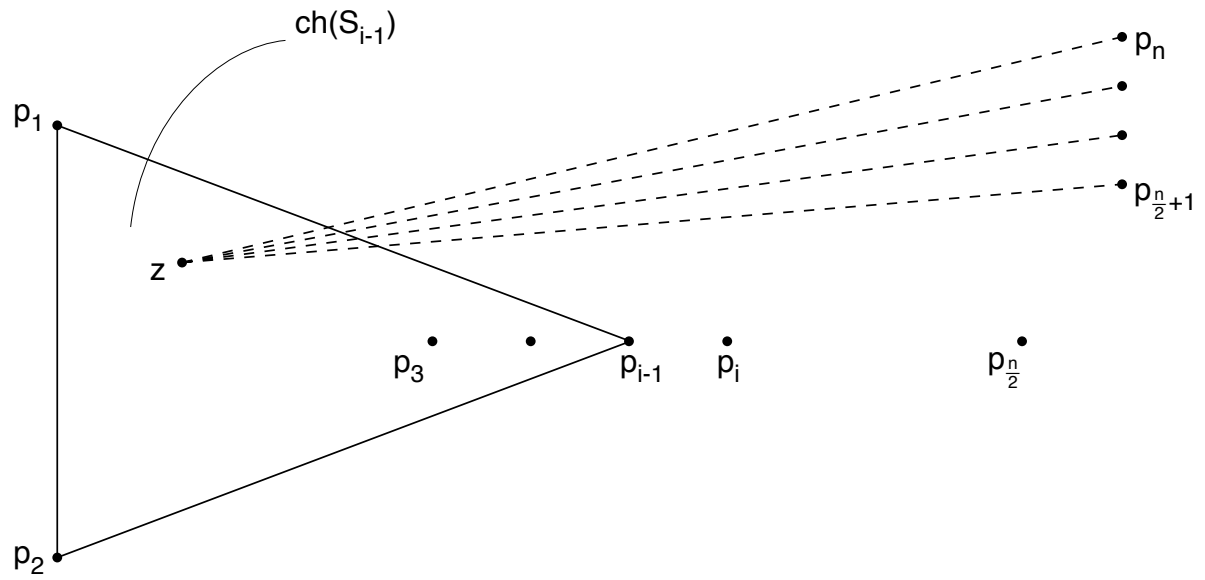
- Neuen Punkt einfügen. Kante erkennen, Tangentenpunkte a, b ermitteln, Kanten löschen, amortisiert insgesamt $O(n)$
- Array aktualisieren: Einige Punkte bekommen neue Kante oder liegen nun innerhalb!
- Konfliktpaare: p_i einfügen, p_j noch nicht bearbeitet:

  (p_i, p_j) in Konflikt $\iff$ zp_j schneidet eine Kante von $ch(S_{i-1})$, die beim Einfügen von p_i entfernt wird

- Für alle diese neuer Zeiger (NIL oder Kante mit p_i)
- Laufzeit: Anzahl der Konfliktpaare, zählen

Worst-Case Anzahl Konflikte

Schlechte Reihenfolge!



Gute Eingabereihenfolge? Zufällige Eingabereihenfolge!

Ergebnis

Theorem 4.6: Wird jede der $n!$ vielen Eingabereihenfolgen einer festen Punktmenge $S = \{p_1, p_2, \ldots, p_n\}$ mit gleicher Wahrscheinlichkeit gewählt, dann ergibt sich für die inkrementelle Konstruktion eine erwartete Anzahl von $O(n \log n)$ Operationen ($O(n \log n)$ Konfliktpaare).

Buch Kapitel

Kapitel 4 Seite 154 oben – S. 163 mitte