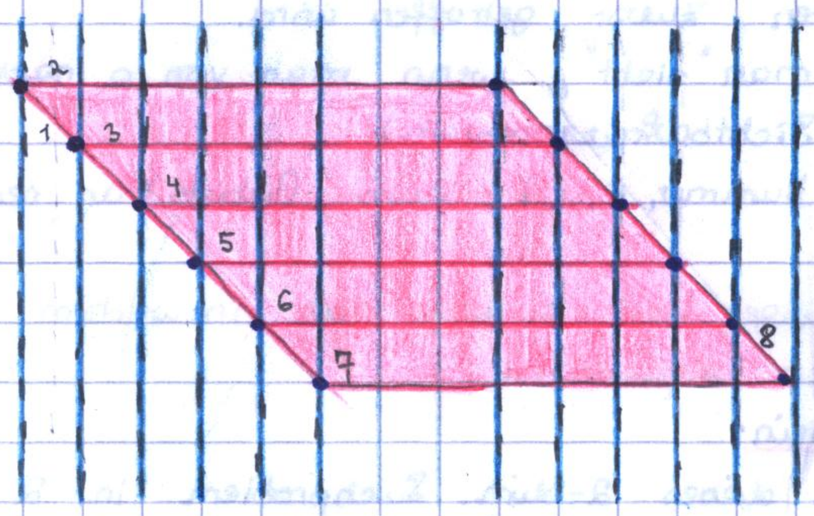
⟶ Streifenmethode: ergebnis:

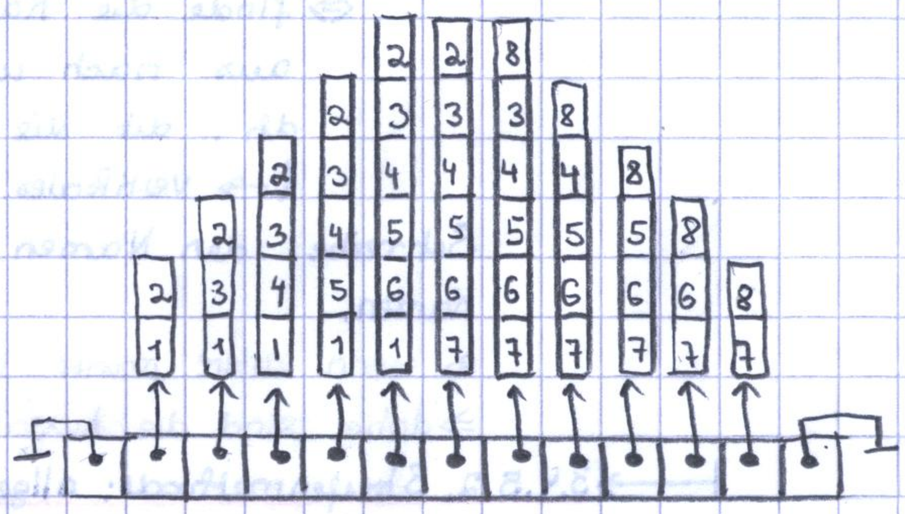
Laufzeit: $O(\log n)$ (optimal)

Platzbedarf: $O(n^2)$ (unpraktisch für große n) $O(n)+O(n^2)=O(n^2)$

Ziel: $O(\log n)$ Suchzeit und $O(n)$ Platz.

Beispiel für quadratischen Platz:





Gesamt: n Knoten (hier $n=12$)

Jede der $n/2$ Kanten ist in $n/2$ Streifen gespeichert.

⟶ 3.4.5.4. Triangulierungsmethode: allgemein:

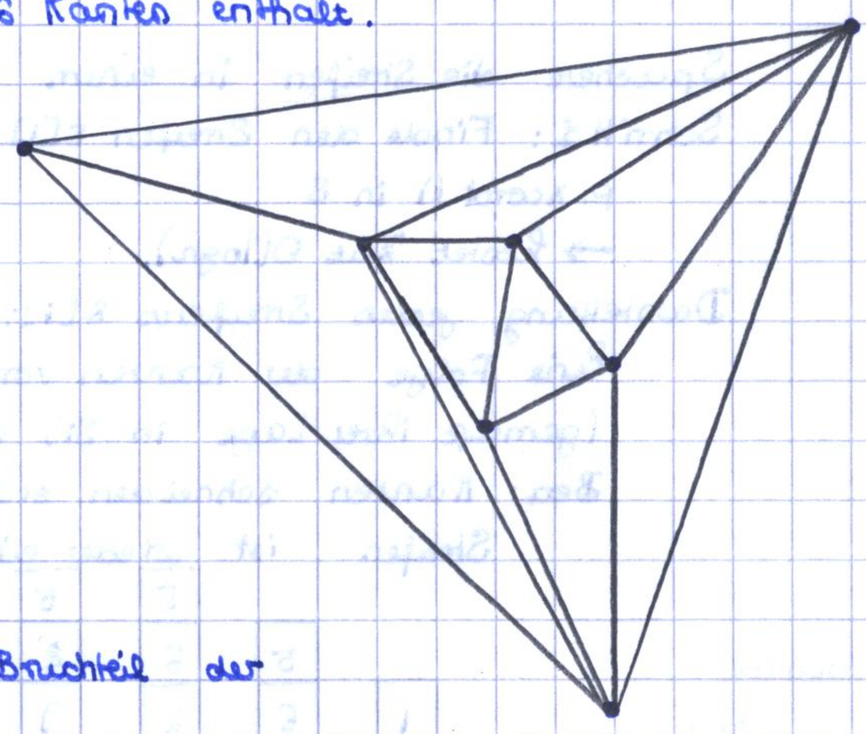
- Ber. eine optimale hierarch. Ds. für das Point Location Problem.
- Methode geht zurück auf Kirkpatrick (1983)

Sei G ein planarer Graph (d.h. zwei versch. Kanten überschneiden sich nicht u. ungerichtet) auf n Knoten und weiter sei jedes Face von G ein Dreieck (auch das äußere Face)

⇒ G ist eine Triangulierung und die konvexe Hülle ist auch Dreieck. $|CH|=3$.

Wir lösen das Point Location-Problem für G optimal. Später leiten wir für jede planare Unterteilung eine optimale hierarch. Ds. her.

Die konv. Hülle besteht aus 3 Knoten, Und wir wissen weiter, dass jede Triangulierung dieser n Knoten $3n-6$ Kanten enthält. ($3n-k-3$, $k=|CH|$).



⟶ 3.4.5.5 Triangulierungsmethode: Idee für Algorithmus:

Konstruiere eine Folge $S_1 .. S_R$ von Triangulierungen, so dass gilt:

(1) $S_1 = G$

(2) S_R ist das äußere Dreieck von G

(3) $R = O(\log n)$

(4) S_{i+1} besteht aus einem konstanten Bruchteil der Knoten von S_i

(5) Für jeden Query Point q, den wir S_{i+1} lokalisiert haben, können wir in Zeit $O(1)$ in S_i lokalisieren.

Geg: $S_1 .. S_R$

dann lösen wir das Point Location Problem wie folgt:

- in Zeit $O(1)$ lokalisieren wir q in S_R
- dann unter Benutzung von (5) lokalisieren wir nacheinander q in $S_{R-1}, S_{R-2}, .., S_1 = G$.

Da $R = O(\log n)$ haben wir eine Suchzeit von $O(\log n)$

Mit (4) folgt, dass für jede Folge der Triangulierung nur Platz $O(n)$ benötigt wird.