(ii) Mische L_1 und L_2 zu einer sortierten Gesamtliste L_P zusammen in Zeit $O(m)$ (→ siehe Mergesort)
Analog: Sortiere Folge L_Q der Ecken von Q in Zeit $O(\ell)$.
(iii) Mische L_P und L_Q in Zeit $O(m+\ell) = O(n)$ (wobei $n := m+\ell$) zu einer sortierten Gesamtfolge zusammen.
→ dann Graham's Scan.

1.4.2. Allgemein: Sei $S \subset \mathbb{R}^2$, $|S| = n$.

```
CONVEX_HULL(S).
if |S| = 1 then
    output S
else
    teile S in zwei möglichst gleich große Teile S1 und S2   } DEVIDE
    (zB. |S1| = ⌈|S|/2⌉ und |S2| = ⌊|S|/2⌋
    P ← CONVEX_HULL(S1)                                       } CONQUER
    Q ← CONVEX_HULL(S2)
    berechne CH(P ∪ Q) wie in letzter Vorlesung gezeigt.      } MISCHSCHRITT
fi.
```

hier wird die eigentliche Rechnung durchgeführt.

1.4.3. Laufzeit:
Teilen: $O(n)$
Mischen: $O(n)$
Merge auf Listen
Graham's Scan (ohne Sortieren).

$$\Rightarrow T(n) = \begin{cases} C_0, & n = 1 \\ C_1 \cdot n + 2 \cdot T(n/2), & n > 1 \end{cases}$$

Das ist also dieselbe Rekursion wie für Mergesort (aber mit anderen Konstanten natürlich).
⇒ Gesamtlaufzeit des Verfahrens ist $O(n \log n)$ (siehe Analyse von Mergesort).

1.5. Eine Anwendung von CONVEX_HULL.

1.5.1 Problem: Geg. sind n Halbebenen $H_1 \dots H_n$ von $\mathbb{R}^2$
Berechne $P = \bigcap_{i=1}^{n} H_i$

1.5.2. Def: Eine abgeschlossene Halbebene = { alle Pkte auf gleicher Seite einer Geraden ℓ } ∪ ℓ.

1.5.3 Anm: Schnitt P ist konvexes (möglicherweise unbeschränktes) Polygon, da der Schnitt konvexer Mengen wieder konvex ist.

1.5.4. Ziel und Lösungsansatz:
Ziel: Berechne die Folge der Ecken von P gg. den Uhrzeigersinn.
Lösung: Zurückführung auf konvexe Hülle einer geeigneten Pktmenge.
Dazu transformieren wir die definierenden Geraden in "duale" Pkte.

1.5.5. Definition: (Geometr. Transformation)
a). Sei $\ell = \{y : \underbrace{y = ax+b}_{\text{Geradengleichung von } \ell}, x \text{ bel.}, a,b \text{ fest}\}$ eine nicht vertikale Gerade.
Der Punkt $D(\ell) := (a,b)$ heißt der duale Punkt zu ℓ.
b). Sei $p = (a,b) \in \mathbb{R}^2$ ein Pkt. Die Gerade $D(P) = \{y : y = -ax + b, x \text{ Beliebig}\}$ heißt die duale Gerade zu p.
Abbildung D erlaubt die relative Lage von Objekten.

1.5.5 Lemma: Pkt p liegt auf (oberhalb / unterhalb) einer Geraden ℓ
⇔ Gerade $D(P)$ liegt auf (oberhalb / unterhalb) Pkt $D(\ell)$.

Beweis: Sei $p = (p_x, p_y)$, $\ell = \{y \in \mathbb{R} : y = ax + b, x \in \mathbb{R}\}$ (a,b fest).
$\Rightarrow D(\ell) = (a,b)$ und $D(P) = \{y : y = -p_x x + p_y, x \in \mathbb{R}\}$

- sei p gelegen auf ℓ
$\Leftrightarrow p_y = a \cdot p_x + b$
$\Leftrightarrow -b = a p_x - p_y$
$\Leftrightarrow b = -p_x a + p_y$
$\Leftrightarrow D(P)$ liegt auf $D(\ell)$.

- sei p gelegen oberhalb ℓ (unterhalb analog).
$\Leftrightarrow ax + b < p_y \ \forall x \in \mathbb{R}$
Bzw. sogar $a p_x + b < p_y$.
$\Leftrightarrow -a p_x - b > -p_y$
$\Leftrightarrow -a p_x + p_y > b$
$\Leftrightarrow D(P)$ liegt oberhalb $D(\ell)$.

1.5.7. Folgerung: Wenn $p = \ell_1 \cap \ell_2 \Rightarrow D(\ell_1), D(\ell_2) \in D(p)$.
Bew. siehe Kursordner. eigentlich " $\Leftrightarrow$ "

1.5.8. Betrachte folgendes Problem:
Seien $\ell_1 .. \ell_n$ n nicht vertikale Geraden im $\mathbb{R}^2$.
$\ell_i^+ :=$ Halbraum oberhalb von ℓ_i $(i \in \{1..n\})$
$\ell_i^- :=$ Halbraum unterhalb von ℓ_i $(i \in \{1..n\})$.
Sei weiter $m \leq n$.
Nun berechne: $S := \ell_1^+ \cap \ldots \cap \ell_m^+ \cap \ell_{m+1}^- \cap \ldots \cap \ell_n^-$.

$\mapsto$ **1.5.8.1. Beobachtung:** Unser ursprüngliches Problem kann in dieser Weise formuliert werden.

Sei $S^+ := \bigcap_{i=1}^{m} \ell_i^+$ und $S^- := \bigcap_{j=m+1}^{n} \ell_j^-$

$\Rightarrow S = S^+ \cap S^-$
Wir berechnen S^+ und S^- getrennt.
Zunächst S^+ (S^- analog):
S^+ ist ein nach oben unbeschränktes konvexes Polygon.

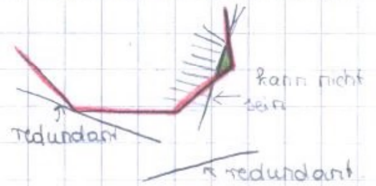
$\mapsto$ **1.5.8.2. Def:** Eine Gerade ℓ_i, $1 \leq i \leq m$ heißt redundant, falls sie nicht zum Rand von S^+ beiträgt, d.h. es gibt keine Kante von S^+ auf ℓ_i.

Beobachtung: Redundante Geraden können ignoriert werden.
Frage: Wie findet man sie? $\rightarrow$ Dualität.

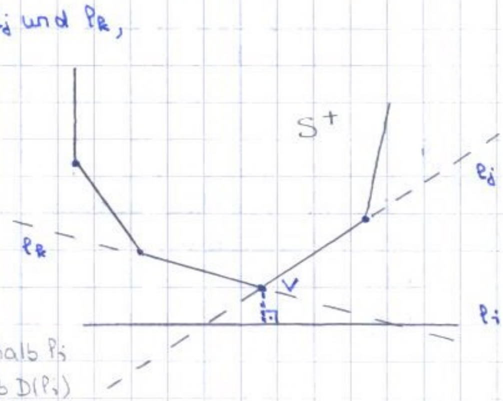
$\mapsto$ **1.5.8.3. Lemma:** ℓ_i ist redundant $(i \in \{1..m\})$
$\Leftrightarrow$ Pkt $D(\ell_i)$ ist keine Ecke der oberen konvexen Hülle von $\{D(\ell_1) \ldots D(\ell_m)\}$.

Beweis:
Vorbemerkung:
Sei ℓ_i redundant
$\Rightarrow S^+ \cap \ell_i = \emptyset$ oder Ecke von S^+
Sei v die Ecke von S^+, die ℓ_i am nächsten liegt.

Bea: Zwei Ecken kann es im Schnitt nicht geben, denn:



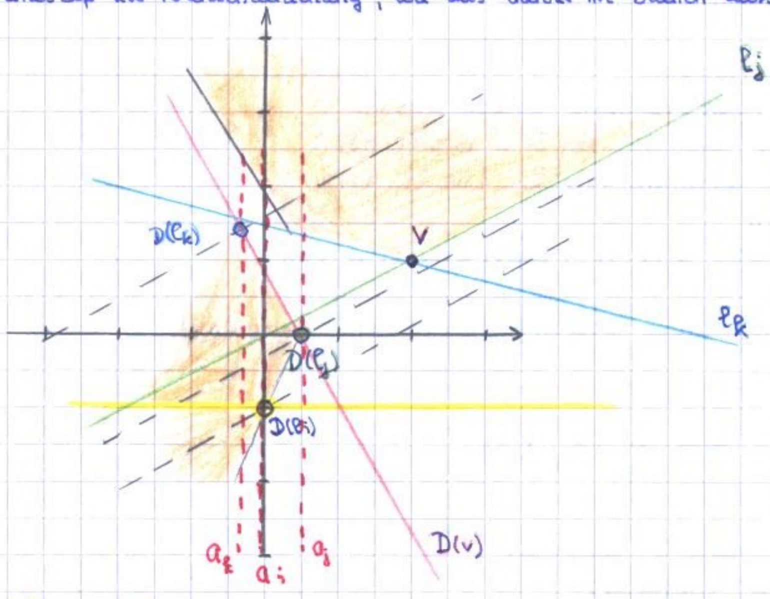
Beobachtung:
1). $\exists$ zwei nicht redundante Geraden ℓ_j und ℓ_k mit $v = \ell_j \cap \ell_k$.
Skizze:
2). Steigung von ℓ_i liegt zw. Steigungen von ℓ_j und ℓ_k, da sonst entweder ℓ_i nicht redundant oder eine andere Ecke als v näher zu ℓ_i liegt.



3). Pkt $D(\ell_i)$ liegt auf oder unterhalb der Geraden $D(v)$
Bea: v ist auf oder oberhalb ℓ_i
$\Leftrightarrow D(v)$ auf oder oberhalb $D(\ell_i)$
$\Leftrightarrow D(\ell_i)$ auf oder unterhalb $D(\ell_i)$
4). $D(\ell_j)$ und $D(\ell_k)$ liegen auf der Geraden $D(v)$
Bea: v liegt auf ℓ_j und ℓ_k
$\Leftrightarrow D(v)$ liegt auf $D(\ell_j)$ und $D(\ell_k)$

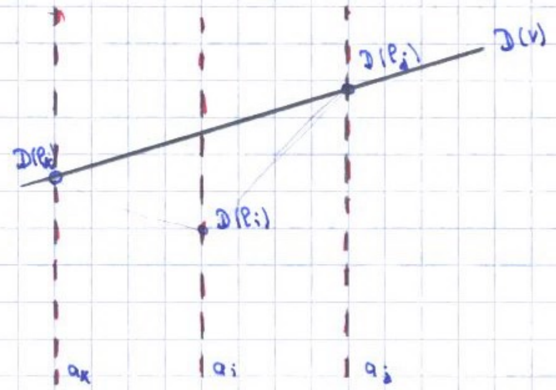
$\rightarrow$

Kleines Bsp zur Veranschaulichung, wie das Ganze im Dualen aussieht:



$\ell_j = \{y : y = \frac{1}{2}x\}$
$\Rightarrow D(\ell_j) = (\frac{1}{2}, 0)$
$\ell_k = \{y : y = -\frac{1}{4}x + \frac{3}{2}\}$
$\Rightarrow D(\ell_k) = (-\frac{1}{4}, \frac{3}{2})$.
$v = (2,1)$
$\Rightarrow D(v) = \{y : y = -2x + 1\}$
$\ell_i = \{y : y = -1\}$
$\Rightarrow D(\ell_i) = (0,-1)$

D.h. also: Situation im Dualen:



Der eigentliche Beweis:

„$\Rightarrow$" sei ℓ_i redundant

z.z. $D(\ell_i)$ ist keine Ecke der oberen $CH(\{D(\ell_1) .. D(\ell_m)\})$

Setze: $D(\ell_i) := (a_i, b_i)$
$D(\ell_j) := (a_j, b_j)$
$D(\ell_k) := (a_k, b_k)$

d.h. a_i, a_j und a_k sind Steigungen von ℓ_i, ℓ_j und ℓ_k

Beob. 2) $\Rightarrow a_k \leq a_i \leq a_j$ $\Rightarrow$ $D(\ell_j)$ ist weiter rechts als $D(\ell_i)$ $\Rightarrow$ $D(\ell_i)$ kann nicht mehr die rechteste Ecke der ober. Hülle sein!

Beob. 3) $\Rightarrow$ $D(\ell_i)$ liegt auf oder unterhalb $D(v)$

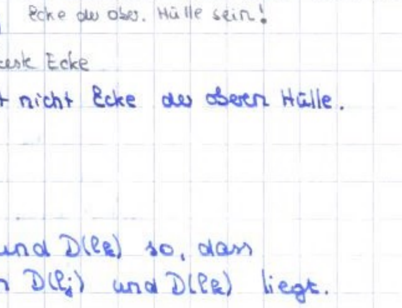
$D(\ell_k)$ ist weiter links $\Rightarrow$ $D(\ell_i)$ nicht die linkeste Ecke
$\rightarrow$ $D(\ell_i)$ auf oder unterhalb $D(v)$ $\Rightarrow$ $D(\ell_i)$ ist nicht Ecke der oberen Hülle.

„$\Leftarrow$" (rückwärts lesen von „$\Rightarrow$")

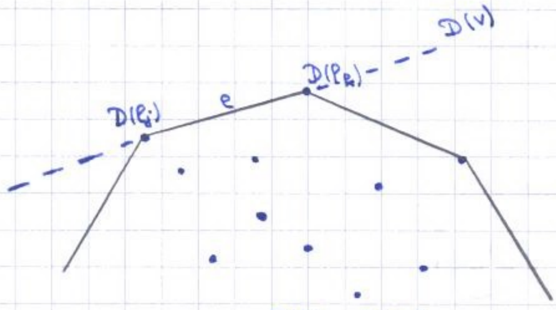
Sei $D(\ell_i)$ nicht Ecke der oberen Hülle.

z.z. ℓ_i redundant

$D(\ell_i)$ nicht Ecke der oberen Hülle $\Rightarrow$ $\exists$ $D(\ell_j)$ und $D(\ell_k)$ so, dass $D(\ell_i)$ auf oder unterhalb der Geraden durch $D(\ell_j)$ und $D(\ell_k)$ liegt.



1.5.8.4. Lemma: $v \in \mathbb{R}^2$ ist Ecke von S^+
$\Leftrightarrow$ Die Gerade $D(v)$ enthält eine Kante der oberen Hülle von $\{D(\ell_1) \ldots D(\ell_m)\}$.



Beweis:

„$\Leftarrow$": $D(v)$ enthält eine Kante e der oberen Hülle von $\{D(\ell_1) .. D(\ell_m)\}$.
z.z. v ist Ecke von S^+
Sei $e = (D(\ell_j), D(\ell_k))$
Dualität: $v \in \ell_j \cap \ell_k$
obere Hülle $\Rightarrow$ alle Pkte $D(\ell_i)$, $i \in \{1..m\}$ liegen unterhalb oder auf $D(v)$
Dualität $\Rightarrow$ Gerade ℓ_i $(1 \leq i \leq m)$ liegt unterhalb oder auf v
d.h. v liegt über oder auf ℓ_i.
$\Rightarrow v \in S^+$
v = Schnittpkt zweier Geraden $\Rightarrow$ v ist Ecke von S^+
(bea: ℓ_j und ℓ_k sind nicht redundant
denn wenn sie es wären $\Rightarrow$ $D(\ell_j)$ und $D(\ell_k)$ wären nicht Ecken von $CH(D(\ell_1) .. D(\ell_m))$ nach L.1.5.8.3 $\Rightarrow$ ↯)

„$\Rightarrow$" v ist Ecke von S^+
$\Rightarrow$ $v = \ell_j \cap \ell_k$
$1.5.7 \Rightarrow D(\ell_j), D(\ell_k) \in D(v)$
$\Rightarrow D(v)$ enthält also die Kante $e := (D(\ell_j), D(\ell_k))$ } (*)
bleibt z.z: diese Kante ist die Kante der konvexen Hülle.
Es gilt: $v \in S^+$
$\Rightarrow$ v liegt über oder auf ℓ_i $\forall i \in \{1..m\}$
$\Rightarrow$ $D(\ell_i)$ liegt unter oder auf $D(v)$ $\forall i \in \{1..m\}$
$\Rightarrow$ $D(v)$ enthält eine Kante der oberen Hülle oder $D(v)$ ist redundant
Wegen (*) kann $D(v)$ nicht redundant sein
$\Rightarrow$ $D(v)$ enthält eine Kante der oberen Hülle.

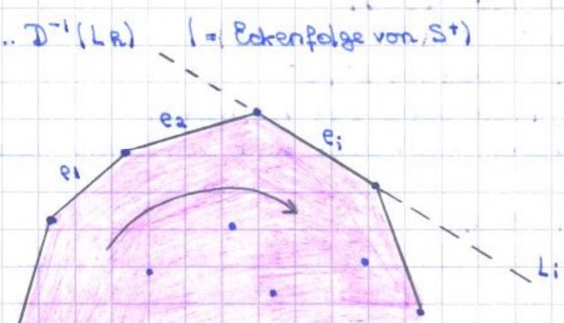
Dieses Lemma liefert einen Algorithmus zur Berechnung von $S^+ = \bigcap_{i=1}^{m} \ell_i^+$

1.5.8.5. Algorithmus:

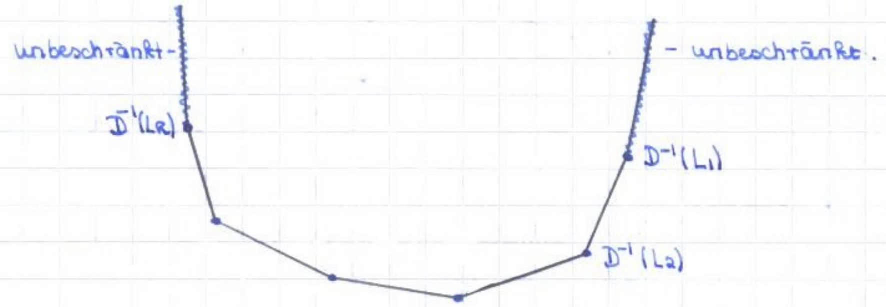
1. Berechne die dualen Pkte $p_i = D(\ell_i)$, $i = 1..m$ $\quad O(m)$
2. Berechne die obere konvexe Hülle H von $\{p_1 .. p_m\}$ $\quad O(m \log m)$
   Seien $e_1 .. e_k$ die Kanten von H von links nach rechts
3. Berechne die Folge der Geraden $L_1 .. L_k$ so, dass $e_i \subset L_i$ $\quad O(m)$
4. Ausgabe:
   $D^{-1}(L_1) \ldots D^{-1}(L_k)$ (= Eckenfolge von S^+) $\quad O(m)$

Achtung: $D^{-1} \neq D$.

$L_i: y = ax + b$
$\Rightarrow D^{-1}(L_i) = (-a, b)$



. $e_1 \dots e_k$ Kanten von H von links nach rechts.
⇒ $L_1 \dots L_k$ nach Steigungen absteigend sortiert. (siehe Skizze ⇒ klar!)
⇒ Ecken von S^+ sind nach x-Koordinaten absteigend sortiert, d.h. von rechts nach links.



Frage: Wie finden wir die beiden unbeschränkten Kanten?
Antwort: Linke Gerade mit minimaler Steigung.
Rechte Gerade mit maximaler Steigung.

1.5.8.6. Zwischenresultat:

$S^+ = \bigcap_{i=1}^{m} \ell_i^+$ kann in Zeit $O(m \log m)$ berechnet werden.

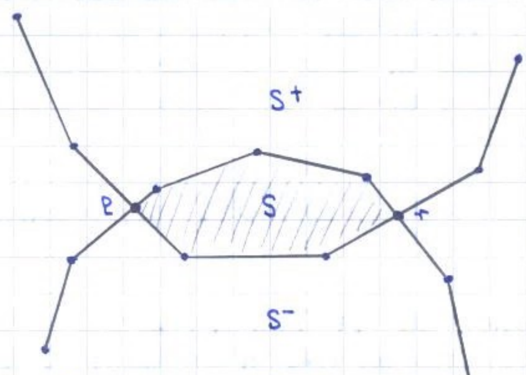
$\underbrace{O(m)}_{D, D^{-1} \dots} + \underbrace{O(m \log m)}_{\text{Graham's Scan}}$

Falls $\ell_1 \dots \ell_m$ nach Steigung sortiert gegeben sind, dann Laufzeit $O(m)$ (⇝ Graham's Scan)

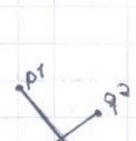
$S^- = \bigcap_{j=m+1}^{n} \ell_j^-$ kann genauso (symmetrisch) berechnet werden.

Es bleibt noch $S = S^+ \cap S^-$ auszurechnen.

1.5.8.7. Berechne S.



Finde Schnittpkte ℓ und r der Ränder von S^+ und S^-



Achtung: Sonderfälle:
- ex. nicht ($S = \emptyset$)
- $\ell = r$ ($S = \{\ell\} = \{r\}$)
- ℓ, r Ecken von S^+, S^-

Allgemein: Innere Schnittpkte von Kanten
Voraussetzung: Die Ränder sind von links nach rechts sortiert.

⇒ Laufzeit: $O(n)$.

S^+ [ ] S^- [ ] ↓ Mischen $O(n)$ S [ ]

und gleichzeitig jeweils das El., welches eingefügt wird, in int S^+ und S^- schreiben und vergleichen.
Sobald Änderung ⇒ Kante gefunden. ⇒ Schnittpkt. berechnen
⇒ Gesamtlaufzeit: $O(n)$.

1.5.9. Satz (Zusammenfassung)

1). Der Schnitt von n Halbebenen kann in Zeit $O(n \log n)$ berechnet werden.
2). Falls die definierenden Geraden nach Steigung sortiert gegeben sind, dann ist die Laufzeit $O(n)$.

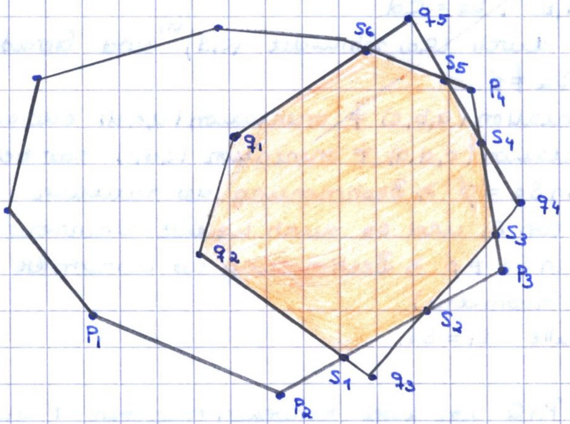
1.5.10 Bemerkungen:

1) Die geometrische Transformation der Dualität wird auch noch für andere Probleme dieser Vorlesung wichtig sein.
2) Der Schnitt von n Halbebenen kann auch direkt berechnet werden.
   ⇝ Divide & Conquer.

1.5.11. Schnitt von Halbebenen durch Divide and Conquer

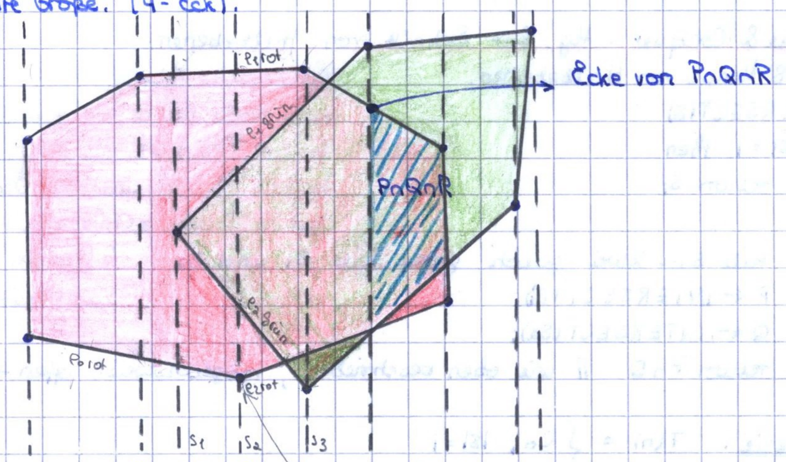
Schnitt von zwei konvexen Polygonen.
hier abgeschlossen (offen → Übung, voriger Abschnitt)
Geg: Seien $P = (p_1 .. p_m)$, $Q = (q_1 .. q_l)$ konvexe abgeschlossene Polygone, geg. durch Eckenfolge gg. den Uhrzeigersinn.
Ausgabe: Eckenfolge von $P \cap Q$ gegen Uhrzeigersinn.

⟼ **1.5.11.1. Beispiel.**
Mögliche Ausgabe:
$q_1 q_2 s_1 s_2 s_3 s_4 s_5 s_6$



⟼ **1.5.11.2. Idee:**
Zerlege Ebene in Regionen so, dass für jede Region R $(P \cap Q) \cap R$ einfach zu berechnen ist.

⟼ **1.5.11.3 Regionen:** vertikale Streifen.
Zeichne durch jede Ecke von P und Q eine senkrechte Gerade.
Jeweils zwei benachbarte Senkrechten definieren eine Region, nämlich den vertikalen Streifen dazwischen.
Dann ist für jeden Streifen R $R \cap P$ und $R \cap Q$ ein Trapezoid, d.h. haben konstante Größe. (4-Eck).



Es gilt $P \cap Q \cap R = (P \cap R) \cap (Q \cap R)$
wegen konst. Größe braucht Zeit $O(1)$

1.5.11.4. Algorithmus:

1). Berechne Streifen von links nach rechts sortiert in Zeit $O(n)$, wobei $n = m + \ell$. Gesamtzahl der Ecken, durch Mischen der Eckenfolgen von P und Q. (siehe D&C für CH.)

2). Berechne für jeden Streifen R Trapezoide $P \cap R$ und $Q \cap R$. Das geht auch in $O(n)$, da wir Eckenfolgen (siehe 1) von links nach rechts sortiert haben.

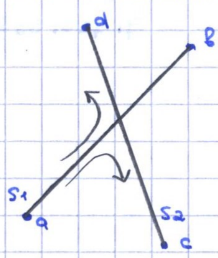
3) Berechne für jeden Streifen R $\quad P \cap Q \cap R := (P \cap R) \cap (Q \cap R)$
Zeit $O(1)$ pro Streifen.

4). Zusammen kleben der Teile aus 3)
Insbesondere Eliminierung von Ecken, die nicht zur Ausgabe gehören.
Laufzeit: $O(n)$ (Durchlaufen der Senkrechten von links nach rechts).
→ Eckenfolge von $P \cap Q$ gg. Uhrzeigersinn.

1.5.11.5. Implementierungsdetails:

Teilprobleme:

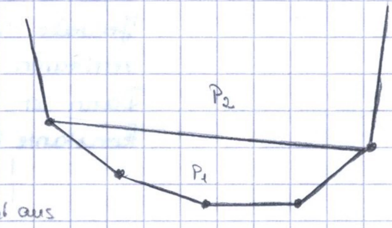
- Test ob Segment s_1 ein anderes schneidet.
  $s_1 = \overline{a,b}$, $s_2 = \overline{c,d}$
  Gerade durch (a,b) schneidet $(c,d)^{(1)}$ und Gerade durch (c,d) schneidet $(a,b)^{(2)}$
  $\Leftrightarrow s_1 \cap s_2 \neq \emptyset$
  (1) orientation$(a,b,c) \neq$ orientation(a,b,d) oder beide 0
  (2) orientation$(c,d,a) \neq$ orientation(c,d,b) oder beide 0.
- falls $s_1 \cap s_2 = \emptyset \Rightarrow$ Bestimmung der relativen Lage von s_1 und s_2 durch weitere Orientierungstests.
- falls $s_1 \cap s_2 \neq \emptyset \Rightarrow$ Bestimmung der Schnittpkte (anal. Geometrie)
- Sonderfälle: $s_1 = s_2$.



1.5.12. Satz: Der Schnitt $P \cap Q$ von zwei konvexen Polygonen P und Q kann in Zeit $O(n)$ berechnet werden, wobei n = Summe der Ecken von P und Q.
Mischschritt

1.5.13. Fragen:

1) Geht das schneller, wenn alle Polygone im Voraus geg. sind, für die man evtl. Schnittoperationen ausführt.
2). Existiert Algorithmus, der output-sensitiv ist?
Dh. Laufzeit hängt von Größe der Ausgabe ab.
→ siehe nächster Abschnitt über konvexe Polygone.



1.5.14. Divide & Conquer - Alg. für Schnitt von Halbebenen.

Sei S Menge von Halbebenen.

```
INTERSECT(S)
if |S| = 1 then
     return S;
else
     teile S in zwei gleich große Teile S1 und S2
     P ← INTERSECT(S1);
     Q ← INTERSECT(S2);
     return P ∩ Q   // wie oben beschrieben, möglicherweise offen → Übung.
```

Ein unbeschr. Polygon besteht aus
- einem beschr. Pol. P_1 und unbeschr. Pol. $P_2 \Rightarrow P = P_1 \cup P_2$

Berechne $P_1 \cap Q_1$ mit D&C und Mischritt von oben $\Rightarrow O(n \log n)$
Berechne $P_2 \cap Q_1$ und $P_1 \cap Q_2$ als Schnitt von Polygon mit drei Geraden in Zeit $3 \cdot O(n \log n) = O(n \log n)$. Anschließend berechne $P_2 \cap Q_2$ als Schnitt von 6 Geraden in $O(9) = O(1)$
$\Rightarrow$ Gesamtlaufzeit: $O(n \log n)$.

1.5.15 Laufzeit:
$$T(n) = \begin{cases} c_0, & |S| = 1 \\ c_1 n + 2 \cdot T(n/2), & |S| > 1 \end{cases}$$

$\Rightarrow T(n) = O(n \log n)$
Alternative zu Dualitätsalgorithmus

1.5.16. Frage: Welcher Algorithmus ist in welchen Fällen schneller? Wahrscheinlich Divide & Conquer