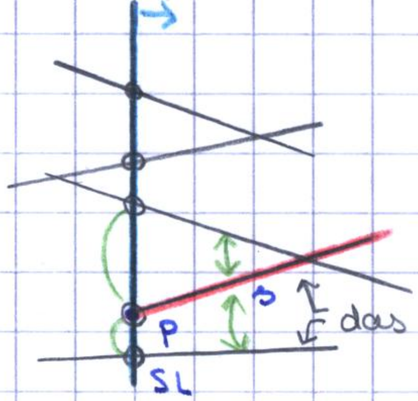
$S_1 \cup S_2$: alle Schnittpkte berechnet und ausgegeben. alle links von SL
$S_2$: alle Segmente, die möglicherweise noch zu Schnittpkte rechts von SL beitragen
$S_3$: unbekannt.

→ 3.2.4.1 Beobachtung:
Wenn die Segmente in $S_a = S \cap SL$ von unten nach oben gemäß ihrem Schnittpkt mit SL sortiert sind (erstmal sind keine vertikalen Segmente erlaubt), dann können wir Schnitt-Tests lokal auf benachbarte Segmente beschränken.

→ 3.2.4.2 Bem. + Def:
Allg. bei Plane Sweep-Algorithmen: Finde und verwalte die Position der Sweep Line SL, wo sich was verändert.
Diese Pkte heißen Transitionspkte oder Events.

→ 3.2.4.3 Events beim Segment-Schnitt:
"Wo ändert sich die sortierte Folge $S_2$?"

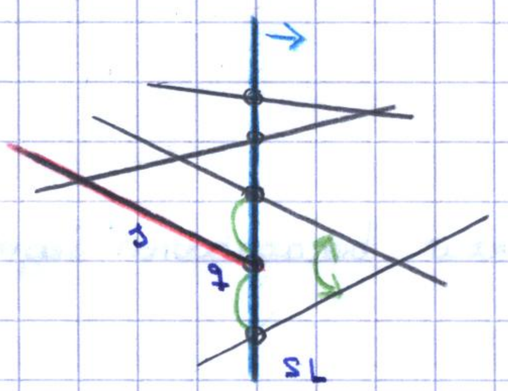
a).




das sind die Nachbarn, schauen ob schneiden

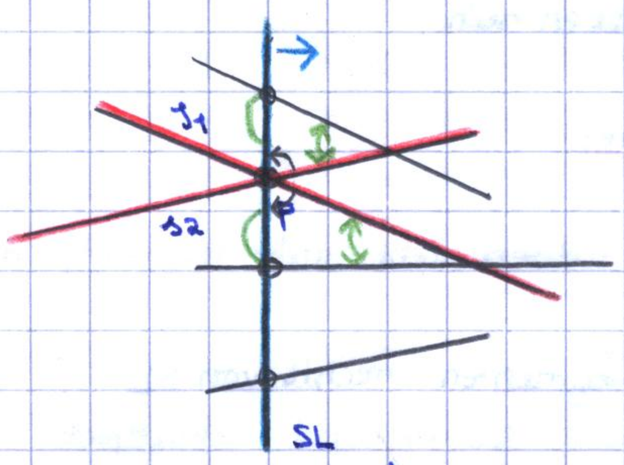
Linker Endpkt p eines Segments s:
→ Aktion: Füge s an die richtige Stelle in Folge $S_2$ ein und Schnittest mit Nachbarn.

b).




Rechter Endpkte q eines Segments s:
→ Aktion: Entferne s aus $S_2$ und Schnittest mit Nachbarn.

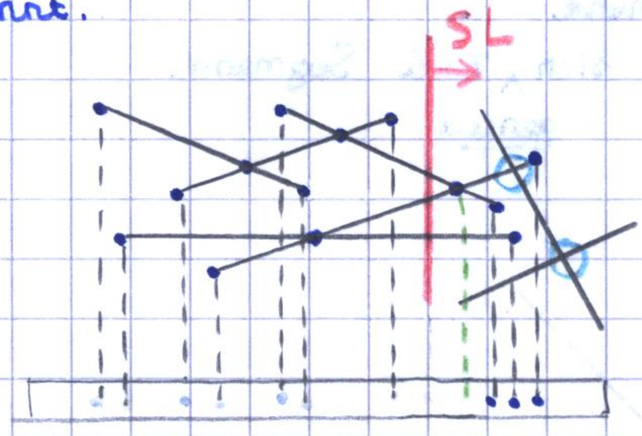
c).




Schnittpkt $p = s_1 \cap s_2$
→ Aktion: Vertausche $s_1$ und $s_2$ und Schnittest für neue Nachbarn.

## 3.2.5 Plane Sweep (allgemein)

→ 3.2.5.1 X-Struktur: Verarbeite Events von links nach rechts.
→ Datenstruktur für Events: X-Struktur. (event queue)
Zwei Fälle:
1). statische X-Struktur, dh. alle Events sind bekannt
→ Event Liste
Bsp: konvexe Hülle: X-Struktur ≙ sortierte Liste der Eingabepkte.
2). dynamische X-Struktur, dh. Events sind nicht alle bekannt, sondern werden zum Teil während des Sweeps berechnet.
Bsp: Segmentschnitt: Schnittpkte rechts von SL sind im Allg. noch nicht bekannt.




- bekannt!
- auch bekannt, weil als Schnittpkte benachbarter Segmente auf SL schon berechnet
- noch unbekannt!

X-Struktur

⇒ X-Str. = {alle End u. Anfangspkte der Segmente + die schon bekannten Schnittpkte}

→ 3.2.5.2 Y-Struktur: