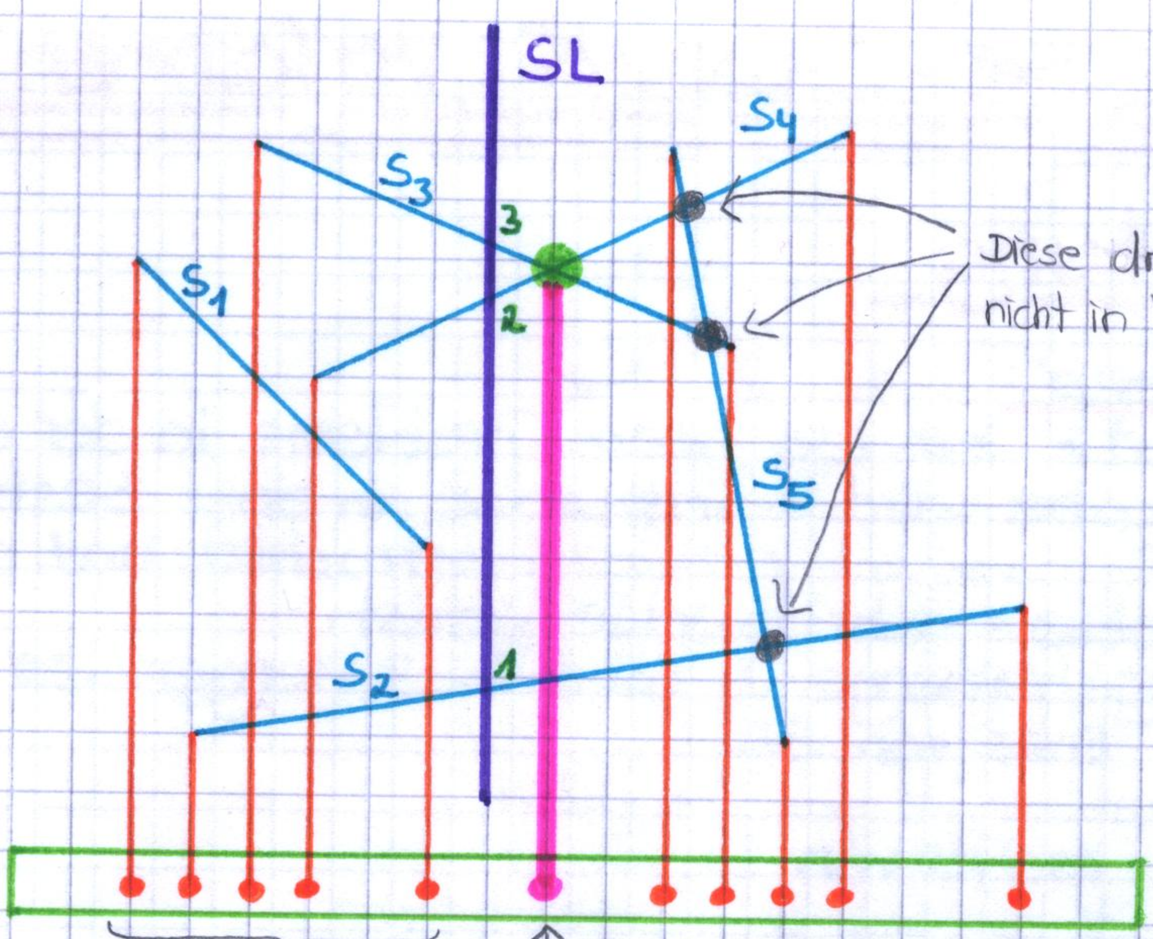
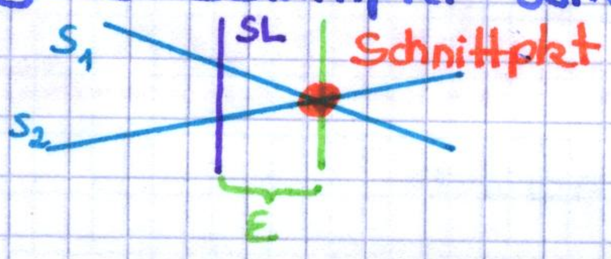


- **Y-Struktur:**
  Verwaltung der Menge ~~[illegible]~~, dh der ~~aktuellen~~ Schnittpkte ~~von Segmenten~~ mit SL haben
  (Segmente welche aktuell einen)
  Diese Segmente sollen von unten nach oben (dh gem ihrem Schnittpkt mit SL) entlang der SL sortiert werden.

- Dadurch können Schnitttests auf benachbarte Segmente beschrkt werden, nur diese müssen in X-Struktur enthalten sein.
  ⇒ Jeder Schnittpkt wird ε vor ihm erkannt



- **Operationen auf X- bzw Y-Struktur beim Segmentschnitt:**
  - **X-Struktur:**

X.insert(p)	// p = Event-Pkt	$O(\log n)$
X.delete(p)		$O(\log n)$
X.findmin()	// nächstes Event	$O(1)$

  - **Y-Struktur:**

Y.insert(s)		$O(\log n)$
Y.delete(s)		$O(\log n)$
Y.swap(s_1, s_2)		$O(1)$
Y.succ(s)	// Nachfolger	$O(1)$
Y.pred(s)	// Vorgänger	$O(1)$

- **Implementierung von X- und Y-Struktur:**
  - balancierter, blattorientierter binärer Baum
  - Für Y-Struktur: zusätzliche Verkettung der Blätter

- **Invariante:**
  X-Struktur enthält zu jedem Zeitpkt:
  - alle Endpkte rechts von SL
  - alle Schnittpkte von in Y benachbarten Segmenten rechts von SL

  ⇒ max n-1 Schnittpkte in X-Struktur ($\exists$ n Segmente)
  ⇒ **Platzbedarf: $O(n)$** wir speichern am Anfang alle linken & rechten Endpkte, das sind aber auch nur linear viele

  Ohne Invariante bis zu n^2 Schnittpkte ⇒ Platzbedarf quadratisch, wäre schlecht !!