5.3. Priority - Search - Tree:

5.3.1. Def: Sei $S \subset \mathbb{R}^2$

Ein Priority - Search-Tree (PST) T ist knotenorientierter Suchbaum nach den x-Koord. der Pkte aus S.

5.3.2. Speichern der Pkte:

Ann: paarw. verschiedene x-Koordinaten.

pes wird in einem Knoten auf Suchpfad nach p_x gespeichert und zwar so, dass T bzgl. der y-Koord. einen Maximumsheap bildet.

⇒ Auf jedem Pfad werden y-Koord. kleiner

⇒ Wurzel enthält Knoten mit max. y-Koord.

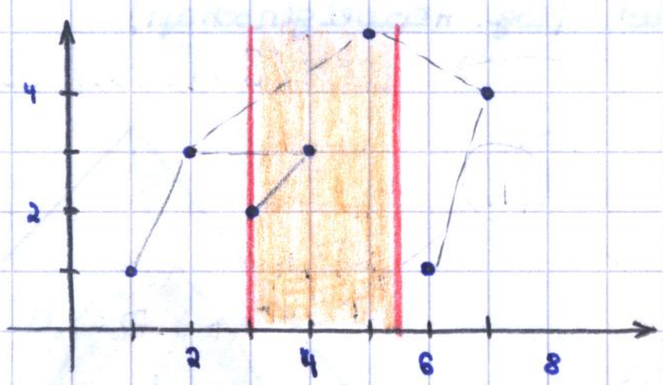
PST hat zwei Eigenschaften:
- Suchbaum nach x-Koord. der Pkte
- Heap nach y-Koord. der Pkte.

Aufbau (rekursiv):

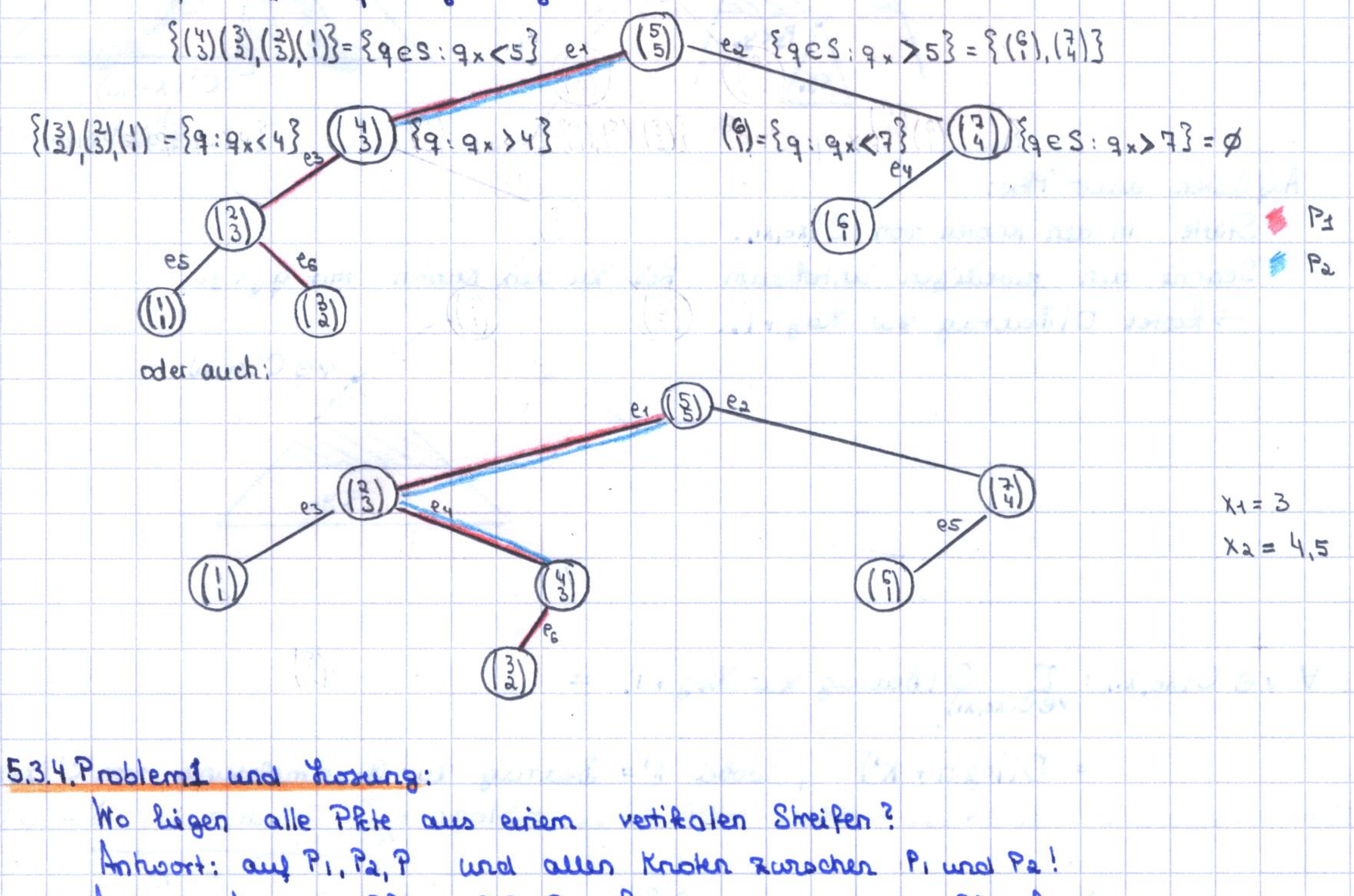
Wurzel w speichert $p \in S$ mit max. y-Koord.
- linker Unterbaum T_ℓ von w ist PST für $\{q \in S : q_x < p_x\}$
- rechter Unterbaum T_r von w ist PST für $\{q \in S : q_x > p_x\}$.

5.3.3. Beispiel:

$S = \{(1,1), (2,3), (3,2), (4,3), (6,1), (5,5), (7,4)\}$.



Im Streifen: $(3,2), (4,3), (5,5)$



oder auch:

$x_1 = 3$

$x_2 = 4,5$

5.3.4. Problem und Lösung:

Wo liegen alle Pkte aus einem vertikalen Streifen?

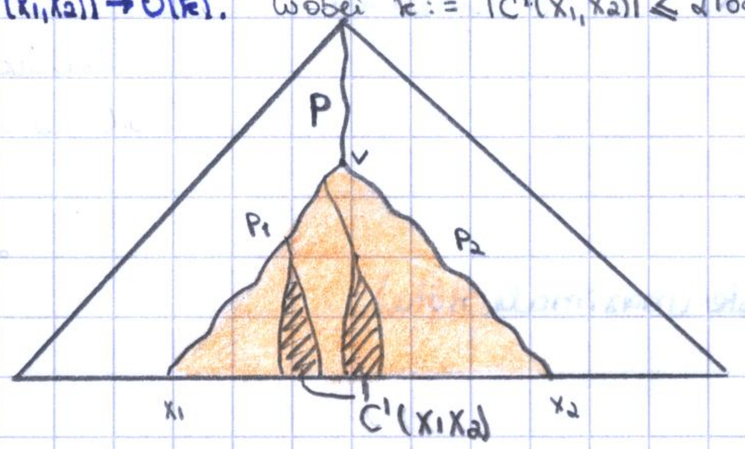
Antwort: auf P_1, P_2, P und allen Knoten zwischen P_1 und P_2!

Achtung: Auf den Pfaden P, P_1, P_2 können auch andere Pkte liegen.

Berechnung der Pkte:
- Filtere die Pkte auf Pfaden nach $[x_1, x_2]$ in Zeit $2 \cdot \log n + k$.
- Gib alle Pkte, die in Knoten zwischen P_1, P_2 gespeichert sind, aus $|C'(x_1, x_2)| \to O(\tilde{k})$. wobei $\tilde{k} := |C'(x_1, x_2)| \leq 2 \log n$

$2(\text{Höhe}(v) + 1 + O(k_1) + 1 + O(k_2) + \dots) =$
$= 2 \cdot (\text{Höhe}(v) + O(k_1) + O(k_2) + \sum 1) =$
$= 2 \cdot (\log n + \tilde{k})$

$\tilde{k}$:= # Knoten in orange.



Im Bsp: 1 Bild ⇒ $\{(5,5),(4,3),(2,3),(3,2)\}$ ⊃ gesuchte Menge

2 Bild ⇒ $\{(5,5),(2,3),(4,3),(3,2)\}$ ⊃ gesuchte Menge

Platz: $O(n)$!

Gesamtlaufzeit: $2 \log n + k + O(\log n + \tilde{k}) =$
$= O(\log n + K)$

K = # Ausgaben. $(= k + \tilde{k})$.