Assignment 1
Advanced Algorithms & Data Structures PS

Christian Müller 1123410
Daniel Kocher, 0926293

March 9, 2016

Input : Menge S bestehend aus vertikalen Segmenten und Endpunkten von horizontalen Segmenten, welche bezüglich ihrer x-und y Koordinaten paarweise verschieden sind.
S sei nach horizontalen Koordinaten sortiert (Algorithmus ist leicht zu adaptieren falls nicht).

Output : Alle Schnittpunkte von vertikalen Segmenten mit horizontalen Segmenten, von denen mindestens ein Endpunkt in S ist

Variables: $L(S)$ enthält die y-Koordinaten aller linken Eckpunkte in S, deren rechter Partner nicht in S.
$R(S)$ enthält die y-Koordinaten aller rechten Eckpunkte in S, deren linker Partner nicht in S.
$V(S)$ enthält die y-Intervalle der vertikalen Segmente in S.
$L(S)$ und $R(S)$ sind nach steigenden y-Koordinaten sortierte, verkettete Listen.
$V(S)$ sind nach steigenden unteren Endpunkten sortierte, verkettete Listen.

1 **Function** *ReportCuts(S)*
2 **if** $|S| \leq 1$ **then**
`// Initialisierung von` $L(S)$, $R(S)$ `und` $V(S)$
3 Sei $s = (x, y) \in S$;
4 **if** *s ist linker Endpunkt* **then** $L(S) \leftarrow \{y\}$, $R(S) \leftarrow \emptyset$, $V(S) \leftarrow \emptyset$;
5 **else if** *s ist rechter Endpunkt* **then** $L(S) \leftarrow \emptyset$, $R(S) \leftarrow \{y\}$, $V(S) \leftarrow \emptyset$;
6 **else if** *S enthält nur das vertikale Segmente v* **then** $L(S) \leftarrow \emptyset$, $R(S) \leftarrow \emptyset$, $V(S) \leftarrow \{[y_1, y_2]\}$;
7 **return** ; `// Rekursionsende`
8 **end**

`// Divide Schritt`
9 Teile S per vertikaler Trennlinie in zwei (nahezu) gleich große Teile S_1 und S_2 auf;

`// Conquer Schritt`
10 ReportCuts(S_1);
11 ReportCuts(S_2);

`//` $L(S_i)$, $R(S_i)$, $V(S_i)$ `für` $i = 1, 2$ `bekannt` $\Rightarrow$ `Merge Schritt`
`// Berichte Segmentschnittpunkte (Paare` (h, v)`)`
12 IntersectAndReport($R(S_2) \setminus L(S_1)$,$V(S_1)$);
13 IntersectAndReport($L(S_1) \setminus R(S_2)$,$V(S_2)$);

`// Aktualisiere` $L(S)$, $R(S)$ `und` $V(S)$ `für` $S = S_1 \cup S_2$
14 $L(S) = (L(S_1) \setminus R(S_2)) \cup L(S_2)$;
15 $R(S) = (R(S_2) \setminus L(S_1)) \cup R(S_1)$;
16 $V(S) = V(S_1) \cup V(S_2)$;

```
1  Function IntersectAndReport(H(S),V(S))
       // Sei V̇ = (v_1, v_2, .., v_n) mit v_i = (y_unten, y_oben) eine einfach verkettete aufsteigend
          sortierte (gemäß y_unten) Liste die alle Element aus V(S) enthält.
       // Sei Ḣ = (h_1, h_2, .., h_n) mit h_i = (x, y) eine einfach verkettete aufsteigend sortierte
          (gemäß y) Liste die alle Element aus V(S) enthält.
2      currHor ← h_1;
3      currVert ← v_1;
4      while currVert ≠ null do
5          if intersects(currVert,currHor) then
6              print(currHor,currVert);
7              tmpHor ← currHor.next;
8              while tmpHor ≠ null and intersects(currVert,tmpHor) do
9                  print(tmpHor,currVert);
10                 tmpHor ← tmpHor.next;
11             end
12             currVert ← currVer.next;
13         else
14             if currHor.y < currVert.y_unten then
15                 currHor ← currHor.next;
16             end
17         end
18     end
```