

Assignment 2
Advanced Algorithms & Data Structures PS

Christian Müller 1123410
Daniel Kocher, 0926293

March 15, 2016

Theorieaufgabe

Für den Divide-&-Conquer Algorithmus zur Berechnung von Segmentschnittpunkten werden zuerst zwei Datenstrukturen definiert, welche verwendet werden um die Elemente der Liste S darzustellen.

Endpunkt Stellt einen Endpunkt eines horizontalen Segments dar. Jeder Endpunkt besitzt mindestens drei Felder:

- x ... repräsentiert die x-Koordinate des Endpunktes.
- y ... repräsentiert die y-Koordinate des Endpunktes.
- ass ... Pointer zum zugehörigen Partner-Endpunkt.

Beispiel: Das horizontale Segment mit den Endpunkten $(1, 2)$ und $(4, 2)$ wird durch zwei **Endpunkte** $e_{1,2}$ repräsentiert. Die Felder von $e_{1,2}$ sind wie folgt belegt:

e_1	$.x = 1$	$.y = 2$	$.ass = e_2$
e_2	$.x = 4$	$.y = 2$	$.ass = e_1$

VertSegment Stellt ein vertikales Segment dar. Jedes vertikale Segment besitzt mindestens drei Felder:

- x ... repräsentiert die x-Koordinate des vertikalen Segments.
- y_{unten} ... repräsentiert die untere y-Koordinate des vertikalen Segments.
- y_{oben} ... repräsentiert die obere y-Koordinate des vertikalen Segments.

Beispiel: Das vertikale Segment mit den Endpunkten $(1, 6)$ und $(1, 1)$ wird durch ein **VertSegment** v repräsentiert. Die Felder von v sind wie folgt belegt:

v	$.x = 1$	$.y_{unten} = 1$	$.y_{oben} = 6$
-----	----------	------------------	-----------------

Im folgenden sollen mit $L(S_i)$, $R(S_i)$ und $V(S_i)$ die verwendeten einfach verkettete Listen bezeichnet werden. $L(S_i)$ enthält die y-Koordinaten aller linken Endpunkte in S_i , deren rechter Partner (ass) nicht in S_i enthalten ist. Analog beinhaltet $R(S_i)$ die y-Koordinaten aller rechten Endpunkte in S_i , deren linker Partner (ass) nicht in S_i liegt. $V(S_i)$ hingegen enthält die y-Intervalle (y_{unten} bis y_{oben}) der vertikalen Segmente in S_i .

$L(S_i)$ und $R(S_i)$ sind nach steigenden y-Koordinaten sortierte, einfach verkettete Listen. $V(S_i)$ sind nach steigenden unteren Endpunkten (y_{unten}) sortierte, einfach verkettete Listen.

Seien l , l_i einfach verkettete Listen und d ein Wert gespeichert in einer einfach verketteten Liste. Solche Listen haben mindestens folgende Felder:

- $head$... Erster Knoten einer Liste.

Weiters stellen sie folgende Funktionen zur Verfügung:

- $l_1.insertAll(l_2)$... Fügt alle Element von Liste l_2 zu Liste l_1 hinzu.
- $l.search(d)$... Durchsucht die Liste l nach einem Element d und gibt dessen Knoten zurück (falls gefunden).
- $l.insert(d)$... Fügt das Element d zu Liste l hinzu.
- $l.delete(d)$... Entfernt das Element d aus Liste l .

Knoten einer solchen Liste stellen mindestens die folgenden Felder bereit:

- $next$... Nächster Knoten.

Input : Liste S bestehend aus vertikalen Segmenten (**VertSegment**) und Endpunkten (**Endpunkt**) von horizontalen Segmenten.

S sei nach horizontalen Koordinaten sortiert (Algorithmus ist leicht zu adaptieren falls nicht).

Output: Alle Schnittpunkte von vertikalen Segmenten mit horizontalen Segmenten

```

1 Function ReportCuts( $S$ ,  $L(S)$ ,  $R(S)$ ,  $V(S)$ )
2   if  $|S| \leq 0$  then return;
3   if  $|S| = 1$  then
4     // Initialisierung von  $L(S)$ ,  $R(S)$  und  $V(S)$ 
5     Sei  $s$  das einzige Element in  $S$ ;
6     if  $s$  ist linker Endpunkt then  $L(S) \leftarrow \{s\}$ ,  $R(S) \leftarrow \emptyset$ ,  $V(S) \leftarrow \emptyset$ ;
7     else if  $s$  ist rechter Endpunkt then  $L(S) \leftarrow \emptyset$ ,  $R(S) \leftarrow \{s\}$ ,  $V(S) \leftarrow \emptyset$ ;
8     else if  $s$  ist vertikales Segment then  $L(S) \leftarrow \emptyset$ ,  $R(S) \leftarrow \emptyset$ ,  $V(S) \leftarrow \{s\}$ ;
9     return ; // Rekursionsende
10  end
11  if  $S$  ist nicht bezüglich  $x$ -Koordinaten sortiert then Sortiere  $S$  bezüglich  $x$ -Koordinaten;
12  /* Divide Schritt.
13      $m$  sei ein Integer (repräsentiert die Mitte der Liste  $S$ ).
14      $S_1$  und  $S_2$  seien Listen von vertikalen Segmenten und Endpunkten von horizontalen
15     Segmenten, sortiert bezüglich ihrer  $x$ -Koordinaten. */
16  if  $|S|$  ist gerade then  $m \leftarrow |S|/2$ ;
17  else  $m \leftarrow \lceil |S|/2 \rceil$ ;
18   $S_1 \leftarrow$  Liste der ersten  $m$  Elemente aus  $S$ ;
19   $S_2 \leftarrow$  Liste der letzten  $(|S| - m)$  Elemente aus  $S$ ;
20  // Conquer Schritt
21  ReportCuts( $S_1$ ,  $L(S_1)$ ,  $R(S_1)$ ,  $V(S_1)$ );
22  ReportCuts( $S_2$ ,  $L(S_2)$ ,  $R(S_2)$ ,  $V(S_2)$ );
23  /*  $L(S_i)$ ,  $R(S_i)$ ,  $V(S_i)$  für  $i = 1, 2$  bekannt  $\Rightarrow$  Merge Schritt
24     Berichte Segmentschnittpunkte (Paare  $(h, v)$ ) */
25   $h_1 \leftarrow R(S_2)$ ;
26  DeletePartnersOf( $h_1$ ,  $L(S_1)$ );
27  IntersectAndReport( $h_1$ ,  $V(S_1)$ );
28   $h_2 \leftarrow L(S_1)$ ;
29  DeletePartnersOf( $h_2$ ,  $R(S_2)$ );
30  IntersectAndReport( $h_2$ ,  $V(S_2)$ );
31  // Aktualisiere  $L(S)$ ,  $R(S)$  und  $V(S)$  für  $S = S_1 \cup S_2$ 
32   $L(S).insertAll(h_2)$ ;
33   $L(S).InsertAll(L(S_2))$ ;
34   $R(S).insertAll(h_1)$ ;
35   $R(S).InsertAll(R(S_1))$ ;
36   $V(S).insertAll(V(S_1))$ ;
37   $V(S).InsertAll(V(S_2))$ ;

```

Input : Zwei einfach verkettete Listen l_1 und l_2 .

$l_{1,2}$ beinhalten nur Endpunkte von horizontalen Segmenten.

Output: Eine Liste, welche alle Elemente aus l_1 beinhaltet, die keinen Partner in l_2 besitzen.

```
1 Function DeletePartnersOf( $l_1, l_2$ )
2   if  $l_2$  ist leer then return  $l_1$ ;
3    $current \leftarrow l_2.head$ ;
4   while  $current \neq null$  do
5     if  $l_1.search(current.ass)$  hat einen Partner gefunden then  $l_1.delete(current)$  ;
6      $current \leftarrow current.next$ ;
7   end
8   return  $l_1$ ;
```

Input : Zwei einfach verkettete Listen h und v .

h enthält Endpunkte von horizontalen Segmenten.

v enthält vertikale Segmente.

h und v sind aufsteigend gemäß y bzw. y_{unten} sortiert.

Output: Berichtet alle Schnittpunkte zwischen Elementen aus v und Elementen aus h .

```
1 Function IntersectAndReport( $h, v$ )
2   // Keine Schnittpunkte möglich
3   if  $h$  ist leer oder  $v$  ist leer then return;
4    $currHor \leftarrow h.head$ ;
5    $currVert \leftarrow v.head$ ;
6   while  $currHor \neq null$  und  $currVert \neq null$  do
7     if  $currHor.y > currVert.y_{oben}$  then  $currVert = currVert.next$ ;
8     if  $currHor.y \geq currVert.y_{unten}$  then
9        $print(currVert.x, currHor.y)$ ;
10       $tmpHor \leftarrow currHor.next$ ;
11      while  $tmpHor \neq null$  und  $tmpHor.y < currVert.y_{oben}$  do
12         $print(currVert.x, tmpHor.y)$ ;
13         $tmpHor \leftarrow tmpHor.next$ ;
14      end
15       $currVert \leftarrow currVer.next$ ;
16    else
17      if  $currVert \neq null$  then  $currHor \leftarrow currHor.next$ ;
18    end
19  end
```