Übungsblatt 6

Julius Auer, Alexa Schlegel

Aufgabe 1 (Voronoi-Diagramme von Strecken):

Wir betrachten als erstes Voronoi Kanten zwischen Punkt P und Strecke s (siehe Abbildung 1). Diese bestehen grundsätzlich aus drei Abschnitten. (i) und (iii) sind Strecken bzw. Strahlen. Es ist die Voronoi Kante zwischen dem Punkt P und dem linken A bzw. rechten Streckenende B. (ii) ist eine Parabel. Deren Form ist abhängig vom Abstand zwischen P und s. In welchem Abschnitt P liegt ist egal, die Zusammensetzung der Voronoi Kante bleibt gleich.

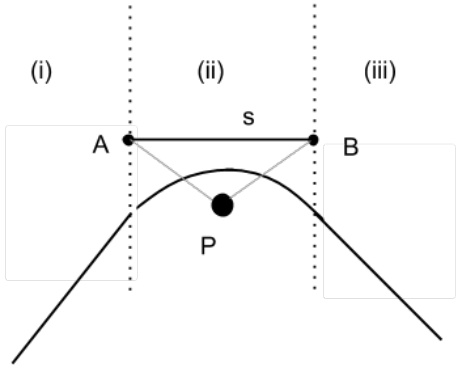

Abbildung 1: Bisektor Punkt und Strecke

Betrachten wir nun die Bisektoren von zwei Strecken s und t, dabei unterscheiden wir in (a) parallele Strecken und (b) nicht parallele Strecken.

(a) parallele Strecken:

- (1) s und t liegen auf einer Geraden (Spezialfall, siehe Abbildung 2)
- (2) s und t „überschneiden“ sich nicht (siehe Abbildung 3)
- (3) s und t „überschneiden“ sich zum Teil oder ganz (siehe Abbildung 4 und 5)

Im Fall (2) und (3) entstehen fünf Abschnitte:
(i) Strecke: Bisektor der linkesten Punkte A und C
(ii) Parabel: s und C
(iii) Strecke: Gerade dazwischen
(iv) Parabel: t und B
(v) Strecke: Bisektor der rechtesten Punkte B und D

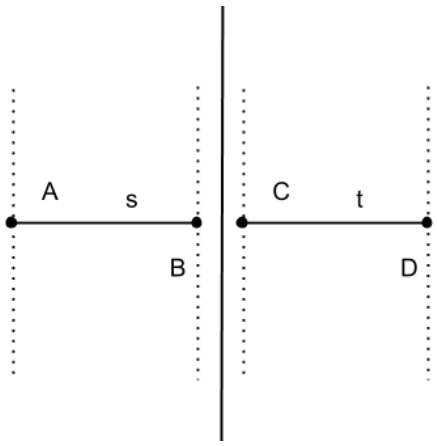

Abbildung 2: (1) Bisektor von zwei parallelen Strecken, die auf der gleichen Gerade liegen.

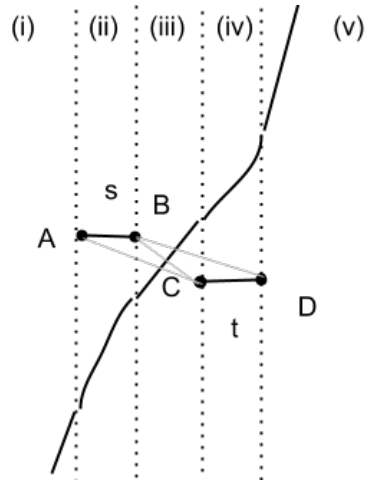

Abbildung 3: (2) Bisektor von zwei parallelen Strecken, keine Überschneidung.

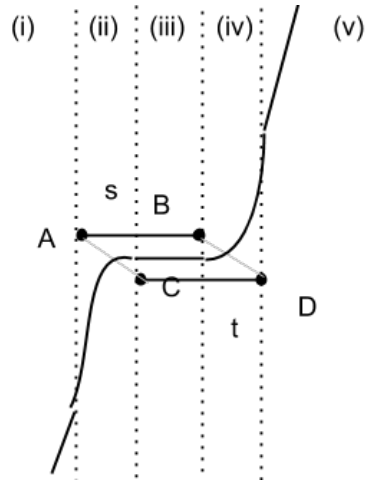

Abbildung 4: (3) Bisektor von zwei parallelen Strecken, Überschneidung.

(b) nicht parallele Strecken

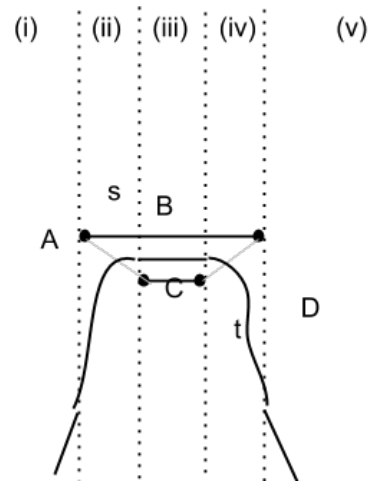

Abbildung 5: (3) Bisektor von zwei parallelen Strecken, komplett innerhalb.

- (1) s und t „überschneiden“ sich nicht
- (2) s und t „überschneiden“ sich zum Teil oder ganz

Hier entstehen im Fall (1) vier Abschnitte (siehe Abbildung 6), im Fall (2) sieben Abschnitte (siehe Abbildung 7):

Fall (1): s und t „überschneiden“ sich nicht
(i) Strecke
(ii) winkelhalbierende Strecke
(iii) Parabel
(iv) Strecke

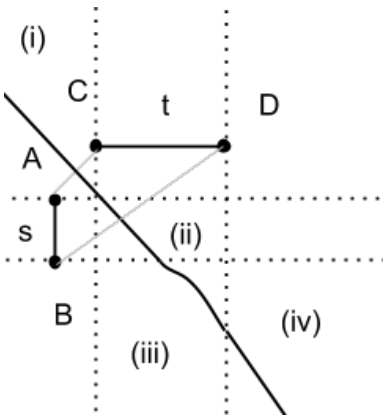

Abbildung 6: (3) Bisektor von zwei nicht parallelen Strecken, keine Überschneidung.

Fall (2): s und t „überschneiden“ sich zum Teil oder ganz
(i) Parabel
(ii) 2×winkelhalbierende Strecken
(iii) und (iv) Parabel
(v) Strecke
(vi) Parabel

(vii) Strecke

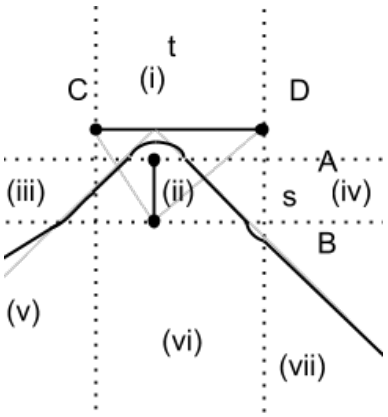

Abbildung 7: (3) Bisektor von zwei nicht parallelen Strecken, mit Überschneidung.

Anzahl der Ecken, Kanten und Zellen Die Anzahl der Zellen entspricht der Anzahl der Strecken n. Ein Bisektor zwischen zwei Strecken besteht höchstens aus 7 Abschnitten, d.h aus 6 Knoten ...

Aufgabe 2 (Fortune-Sweep):

Die Sweepline sweept von links nach rechts, also entlang er x-Achse.

site ist ein Punkt in der Ebene.

beach line teilt die Ebene in zwei Bereiche. Oberhalb der beach line sind alle Bereiche die potentiell zu den Voronoi Regionen der jeweiligen Punkte gehören werden. Unterhalb der beach line alle Punkte zu denen noch keine Aussage getroffen werden kann. Die Punkte auf der beach line sind equidistant zum zugehörigen Punkt und zur sweep line.

spikes zukünftige Voronoi Kanten. An Schnittpunkten von Spikes entstehen zukünftige Voronoi Knoten.

breakpoint sind die Schnittpunkte der Parabeln, dort befinden sich auch die Spikes, also durch die zwei Schnittpunkte verläuft genau ein Spike.

site Event tritt auf, wenn die Sweepline einen Punkt („site“) trifft.

circle Event tritt auf, wenn Kreisbögen verschwinden. D.h. ein circle event entsteht aus einem site event. Wenn sich zwei Spikes schneien, der Kreis durch die zugehörigen Punkte und dann ganz unten auf dem Kreis ist eine neues circle event.

sweep line status (SLS) enthält die Topologie der Beachline in Form einer Liste, sozusagen die Kreisbögen und die dazugehörigen Punkte und eine Information darüber, durch welches Event ein Kreisbogen verschwindet.

event point schedule (EPS) enthält site events und circle events. Die Priorität ist durch die x-Koordinate gegeben, d.h. tritt das Event weiter links auf, so ist die Priorität höher. Initialisiert wird die priority queue mit allen zu Beginn gegebenen Punkten.

Initialisierung

EPS $(0,0)$[se1], $(1,2)$[se2], $(2,3)$[se3], $(4,3)$[se4]

beachline $\emptyset$

EVENT 1 - site event $(0, 0)$

EPS $(1, 2)$[se2], $(2, 3)$[se3], $(4, 3)$[se4]

beachline $a_1 \rightarrow (0, 0)$

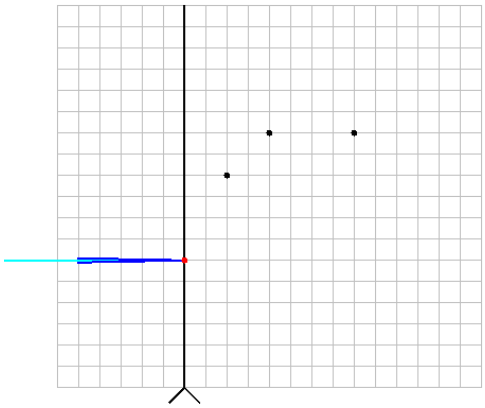

Abbildung 8: Schritt 1

Kein circle event und keine neuen Kanten.

EVENT 2 - site event $(1, 2)$

EPS $(2, 3)$[se3], $(4, 3)$[se4]

beachline $a_{1.1} \rightarrow (0, 0)$,
$a_2 \rightarrow (1, 2)$,
$a_{1.2} \rightarrow (0, 0)$

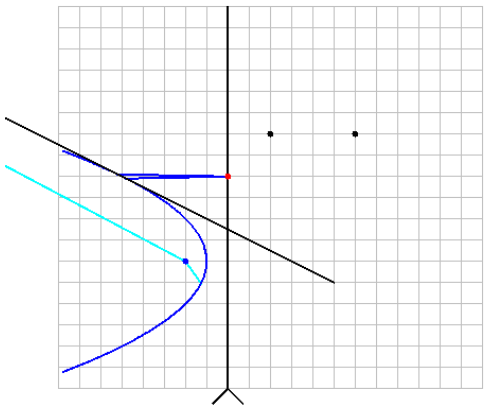

Abbildung 9: Schritt 2

Kein circle event, eine neue Kante, die $(0, 0)$ von $(1, 2)$ trennt.

EVENT 3 - site event $(2, 3)$

EPS $(4, 3)[\text{se}4], (11 + \sqrt{130}, -3)[\text{ce}1]$

beachline $a_{1.1} \rightarrow (0, 0)$
$a_{2.1} \rightarrow (1, 2)$
$a_3 \rightarrow (2, 3)$
$a_{2.2} \rightarrow (1, 2)[\text{delete at ce}1]$
$a_{1.2} \rightarrow (0, 0)$

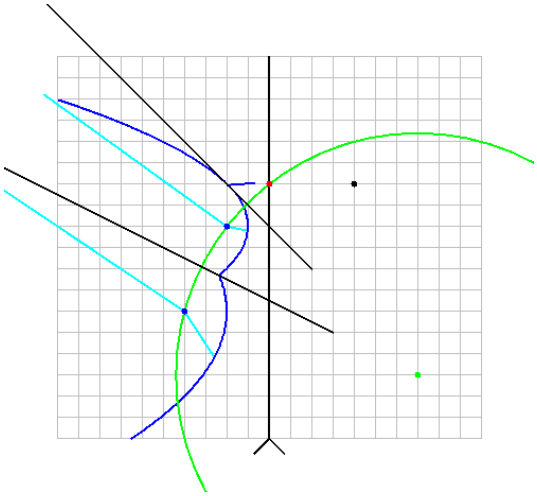

Abbildung 10: Schritt 3

Der Schnittpunkt $(11, -3)$ der beiden Spikes ist ein evtl. zukünftiger Voronoi Knoten. Der Fußpunkt des grünen Kreises wird als neues circle event $(11 + \sqrt{130}, -3)[ce1]$ für $a_{2.2}$ der EPS hinzugefügt.

Für $a_{2.1}$ ensteht kein valides circle event, da der Kreis keine „Rechtsdrehung“ beschreibt.

EVENT 4 - site event $(4, 3)$

EPS $(6 + \sqrt{20}, 2)$[ce2], $(11 + \sqrt{130}, -3)$[ce1]

beachline $a_{1.1} \rightarrow (0, 0)$
$a_{2.1} \rightarrow (1, 2)$
$a_{3.1} \rightarrow (2, 3)$
$a_4 \rightarrow (4, 3)$
$a_{3.2} \rightarrow (2, 3)$[delete at ce2]
$a_{2.2} \rightarrow (1, 2)$[delete at ce1]
$a_{1.2} \rightarrow (0, 0)$

Der Schnittpunkt der beiden letzen Spikes $(6, 2)$ ist ein evtl. zukünftiger Voronoi Knoten. Der Fußpunkt dieses Kreises wird als neues circle event $(6 + \sqrt{20}, 2)$[ce2] der EPS hinzugefügt und rutscht in der priority queue ganz nach vorne.

Für $a_{3.1}$ ensteht kein valides circle event.

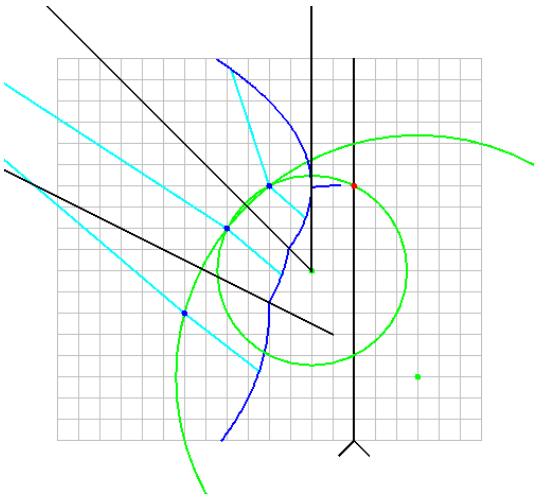

Abbildung 11: Schritt 4

EVENT 5 - circle event $(6 + \sqrt{20}, 2)$

EPS $(11 + \sqrt{130}, -3)$[ce3]

beachline $a_{1.1} \rightarrow (0, 0)$
$a_{2.1} \rightarrow (1, 2)$
$a_3 \rightarrow (2, 3)$
$a_4 \rightarrow (4, 3)$
$a_{2.2} \rightarrow (1, 2)$[delete at ce3]
$a_{1.2} \rightarrow (0, 0)$

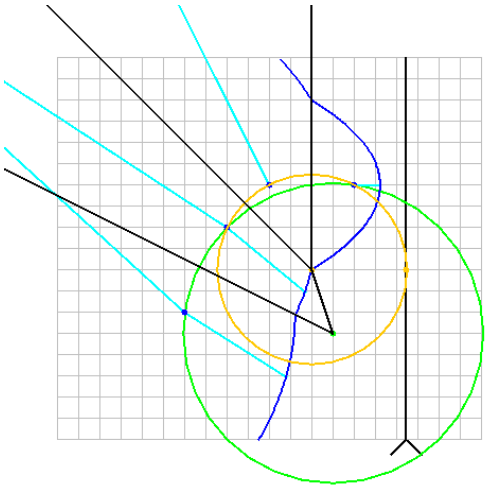

Abbildung 12: Schritt 5

Ein Parabelbogen verschwindet, der Voronoi Knoten bei $(6, 2)$ ist nun fest. Für $a_{2.2}$ wird ein neues circle event mit Mittelpunkt $(7, -1)$ bei $(7 + \sqrt{50}, -1)$[ce3] erzeugt, das $ce1$ obsolet macht.

EVENT 6 - circle event $(7 + \sqrt{50}, -1)$

EPS $\emptyset$

beachline $a_{1.1} \rightarrow (0, 0)$
$a_2 \rightarrow (1, 2)$
$a_3 \rightarrow (2, 3)$
$a_4 \rightarrow (4, 3)$
$a_{1.2} \rightarrow (0, 0)$

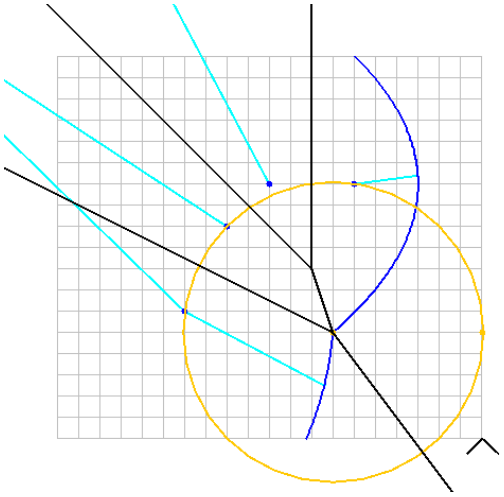

Abbildung 13: Schritt 6

Ein Parabelbogen verschwindet, der Voronoi Knoten bei $(7, -1)$ ist nun fest.

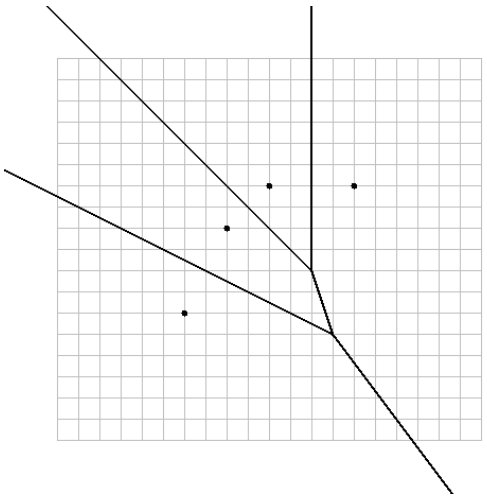

Abbildung 14: Schritt 7

Abschließend wird geprüft, ob mit den Parabelsegmenten in der beachline noch ungebundene Kanten verknüpft sind, die sich zwischen den Segmenten schneiden. Dies ist hier nicht der Fall, womit das Voronoi-Diagramm feststeht.

Btw: die Bilder sind aus unserer *Implementierung* ;)

Aufgabe 3 (Durchschnitt einfacher Polygone):

Das Verfahren ähnelt ein wenig dem in der Vorlesung vorgestellten zur Berechnung der Triangulierung einer Punktwolke mit einem Sweep-Line Verfahren: die Front besteht aus Intervallen die innerhalb von keinem, einem oder beiden Polygonen liegen. An Ereignispunkten werden Kanten des Schnitts hinzugefügt und Intervalle hinzugefügt/entfernt. Die bereits zu Beginn des Algorithmus bekannten Ereignispunkte sind die Koordinaten der Ecken der Polygone (Punktereignisse), die dynamischen Ereignispunkte sind Schnittpunkte zwischen Kanten der beiden Polygone (Schnittereignisse).

Es ist viel anschaulicher und rechnet sich oft leichter (siehe Parabeln bei Fortune) die Sweepline „von oben nach unten“ laufen zu lassen und die Front „von links nach rechts“ zu organisieren. Als totale Ordnung der Punkte der Ebene sei hier deshalb eine Sortierung vorgeschlagen, die zunächst absteigend nach Y-Koordinate und danach aufsteigend nach X-Koordinate sortiert.

Es seien im Folgenden $P = (V_p, E_p), R = (V_q, E_q)$ zwei Polygone für die $S = P \cap Q$ berechnet werden soll. Benötigt werden auch hier eine Prioritätswarteschlange Q und eine geeignete Datenstruktur (z.B. binärer Suchbaum) um die Front F zu verwalten.

Ein Intervall $i = (l, r, e_l, e_r, o)$ besteht aus Verweisen l/r auf das linke/rechte Nachbar-Intervall, den Kanten e_l/e_r die das Intervall nach links/rechts abgrenzen (eine würde reichen, beide zu haben erleichert aber das Aufschreiben ...) und einem Hinweis $o \in \{0, P, Q, S\}$ ob das Intervall zu keinem Polygon / nur P / nur Q / S gehört.

Für ein Schnittereignis $s = (p, e_l, e_r)$ werden neben der Position p an der es auftritt auch die Kanten e_l, e_r gespeichert, die sich geschnitten haben.

Algorithmus:
(1) Füge alle Ecken aus P, R zur Prioritätswarteschlange Q hinzu
(2) Solange Q nicht leer ist: bearbeite & entferne das erste Ereignis aus Q

(3) Gebe Ergebnis S zurück

Mit Punktereignissen bzw. Schnittereignissen wird nun wie folgt verfahren:

Punktereignis p:
Zu p gehören stets zwei ausgehende Kanten $e_1 = (p, (x_l, y_l)), e_2 = (p, (x_r, y_r))$ wobei $x_l < x_r$, sowie ein Polygon $P(p)$.
(P1) Suche das Intervall $i = (l, r, e_l, e_r, o)$ in F, für das die X-Koordinate von p zwischen den X-Koordinaten der Knoten von e_l, e_r liegt, welche die größere Y-Koordinate haben.
(P2): Prüfe ob ein Schnittpunkt s zwischen e_1 und e_l existiert. Wenn ja, setzte $e_1 = (p, s)$ und füge das Schnittereignis $e = (s, e_l, e_1)$ zu Q hinzu.
Es muss ebenfalls geprüft werden, ob ein Schnittpunkt zwischen e_1 und e_r existiert. Ebenso müssen Schnitte mit e_2 berücksichtigt werden - in allen Fällen wird analog verfahren.
(P3):

- Fall 1: F ist leer
  Füge zu F das Intervall $j = (null, null, e_1, e_2, P(p))$ hinzu

- Fall 2: Die Endpunkte von e_1 und e_2 liegen beide unterhalb von p:
  Dann wird i aufgespalten und ein neues Intervall $j = (i, i, e_1, e_2, o_j)$ zwischen die aufgespaltenen Teile eingefügt (Zeiger werden entsprechend umgelegt).

  $$o_j = \begin{cases} 0 & \text{, falls } o = P(p) \\ P(p) & \text{, falls } o = 0 \\ S & \text{, sonst} \end{cases}$$

  Falls $o_j = S$ werden e_1, e_2 mit den dazugehörigen Kanten zu S hinzugefügt.

- Fall 3: Die Endpunkte von e_1 und e_2 liegen beide oberhalb von p:
  Dann wird der Polygonzug an dieser Stelle geschlossen. Falls $o_j = S$ werden e_1, e_2 mit den dazugehörigen Kanten zu S hinzugefügt. i wird aus F entfernt und l mit r vereinigt (es sollten bzgl. der Polygonzugehörigkeit hier keine Uneindeutigkeiten auftreten). Für die begrenzenden Kanten des vereinigten Intervalls muss nun erneut auf Schnittereignisse geprüft werden.

- Fall 4:
  Es handelt sich um eine "Durchgangskante". Die Kante $e \in \{e_l, e_r\}$ die mit p verbunden ist wird durch die "nach unten zeigende" Kante ersetzt. Es muss auf Schnittereignisse geprüft werden und falls $o = S$ die neue Kante zu S hinzugefügt werden.

Schnittereignis s:
(S1): Es wird verfahren wie bei (P1)-(P3), wobei e_1, e_2 die Kanten sind, bei denen ein Endpunkt eine kleinere Y-Komponente hat als s. In (P3) greift somit Fall 2 - es unterscheidet sich hier nun aber die Belegung von o_j:

$$o_j = \begin{cases} 0 & \text{, falls } o = S \\ S & \text{, falls } o = 0 \\ P & \text{, falls } o = Q \\ Q & \text{, sonst} \end{cases}$$

Zeitkomplexität:

- Sortieren in (1) benötigt $O(n \cdot \log n)$

- Jeder Schnittpunkt zwischen P und R ist immer auch Ecke von S (Berührpunkte zählen nicht als Schnitt). Es gibt insgesamt also genau k Schnittereignisse. Die Schleife in (2) wird folglich $\Theta(n+k)$ mal ausgeführt.

- Jedes Mal wenn ein Intervall zur Front hinzugefügt wird, wird ein bestehendes aufgespalten. Im entarteten Fall kann dies $O(n+k)$ mal erforderlich sein, es können also maximal $2 \cdot (n+k)$ Elemente in F vorhanden sein. Steht für F eine passende DS (Baum) zur Verfügung, kann somit bei (P1) in $O(\log(n+k))$ gesucht werden (amortisiert vermutlich besser).

- (P2) kann $O(\log(n+k))$ erfordern, um neue Ereignisse einzusortieren.

- (P3) kann $O(\log(n+k))$ erfordern, um neue Intervalle abzulegen.

$$\rightarrow O((n+k) \cdot \log(n+k))$$