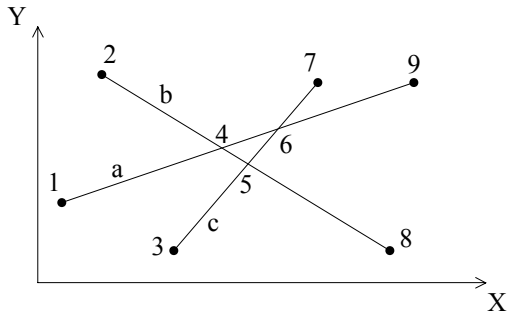
Kap. 13 Sweep-Line Algorithmen
Kap. 13.1 Schnittprobleme



Menge von Linien-Segmenten der Form (a, 1, 9) oder
L: $(a, (x_1, y_1), (x_9, y_9))$

1

X-Struktur: Anfangspunkte
Endpunkte
manche Schnittpunkte
sortiert nach aufsteigenden X-Koordinaten

Y-Struktur: **Status** an Sweep-Line (SL),
i.e. Linien die SL schneiden sortiert nach
Y-Koordinaten des Schnittpunktes mit SL.
(Y-Koord. selbst nicht benötigt)

2

Interessante Vorgänge bei nächstem Punkt bzgl. X-Struktur;

Vorstellung: SL wird bis zum nächsten Punkt in X-Struktur verschoben:

- Anfangspunkt von Linie d:

  nehme d in Y-Struktur, i.e. einsortieren

- Endpunkt von Linie e:

  entferne e aus Y-Struktur

- Schnittpunkt von f,g:

  vertausche f, g in Y-struktur

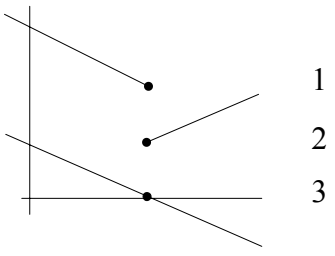
Schnittpunkte in X-Struktur:

Nur Schnittpunkte von benachbarten Linien bzgl. Y-Struktur, die rechts von SL liegen

3

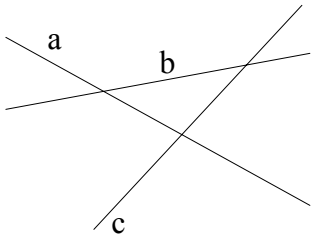
Lemma: Wenn kleinster "interessanter" Punkt rechts von SL ein Schnittpunkt ist, dann ist er ein Schnittpunkt benachbarter Linien, also in X-Struktur.

Bew: interessante Vorgänge:



bei 3 müssen Linien schon in Y-Struktur sein!

Linie a muß erst Nachbar b schneiden, bevor a c schneiden kann!



4

X-Struktur enthält: Punkte **rechts** von SL, wenn sie:

1. Anfangspunkt einer Linie
2. Endpunkt einer Linie
3. Schnittpunkt zweier **benachbater** Linien bzgl. Y-Struktur sind.

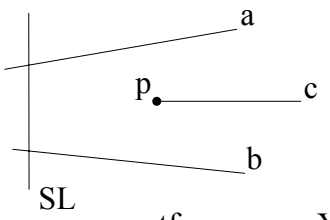
⇒ X-Struktur enthält "interessante" Punkte sortiert nach X-Koordinate plus Zusatzinfo über Linien, zu denen Punkt gehört

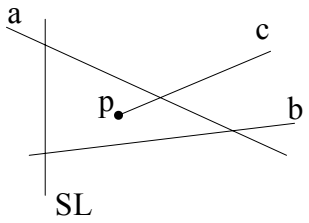
5

Allg. Struktur des Algorithmus:

Nächster Punkt in X-Struktur

1. Anfangspunkt p von c:

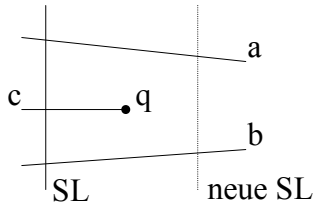




- entferne p aus X-Struktur
- füge Linie c in Y-Struktur ein zwischen a und b, d.h. ändere Nachbarschaften in Y-Struktur
- ersetze Schnittpunkte in X-Struktur, z.B. entferne a∩b aus X, weil (a,b) nicht mehr benachbart, füge c∩a und c∩b ein (liegen rechts von SL, falls existent)

6

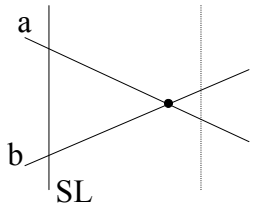
2. Endpunkt: q von c



- entferne c aus Y
- enferne q aus X
- a∩b in X eintragen, falls rechts von SL

Hinw: Nachbarschaften in Y-Struktur ändern sich automatisch, da Linien nach Y-Koord. sortiert sind

3. Schnittpunkt: a∩b



- entferne a∩b aus X
- vertausche a, b in Y
- neue Schnittpunkt a∩c und b∩d mit neuen Nachbarn c, d falls existent und rechts von SL

7

Sweep-Line Alg:

initialisiere X-Struktur mit Anfangs und Endpunkten;

initialisiere Y-Struktur = ∅;

while X-Struktur ≠ ∅ **do**

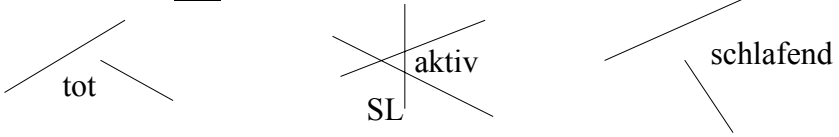
begin P:= Min (X-Struktur);

{P ist Anfang, Ende oder Schnittpunkt}

transformiere X-Struktur und Y-Struktur

abhänging von P

<u>end</u>



Y-Struktur repräsentiert nur aktive Linien

8

Operationen auf Y: fallen an, wenn nächster Punkt in X-Struktur bearbeitet wird:

Find (P): Finde SL-Intervall, in dem P liegt

Insert (L): Einfüge Linie L in Y-Struktur

Delete (L): Entferne Linie L aus Y-Struktur

Pred (L), Succ (L): beide Nachbarn von L

Interchange (L,L´): vertausche Nachbarn L,L´

Datenstruktur für Y: balanzierter Baum, AVL oder B-Baum

Ordnungsrelation für Linien:

$$L_1 <_y L_2 \iff y(L_1 \cap SL) < y(L_2 \cap L)$$

wobei: $y(L_i \cap SL) =$ y-Koordinate des Schnittpunktes von L_i mit SL

9

Hinw: $<_y$ muß bei Suche in Y-Struktur ausgewertet werden.

SL: X = c

y-koordinate des Punktes P ist bekant

$$y(P) \gtrless^{?} y(L_i \cap SL)$$

Analyse: n Linien, s Schnittp.

In Y-Struktur: sind O(n) Linien, deshalb Einzeloperationen erfordert:

suchen
einfügen
löschen
vertauschen

} je O(log n) Kosten

Ingesamt 2n+s Einzeloperationen (Durchläufe durch **while** – Schleife)

$\Rightarrow O$ ((n+s) log n) Kosten

10

In X-Struktur: sind jeweils höchstens:

2n + n-1

Anfangs – n Endpunkte

Schnittpunkte benachbarter Linien, weil:

1	1∩2
2	2∩3
3	3∩4
...	...
n	n-1∩n

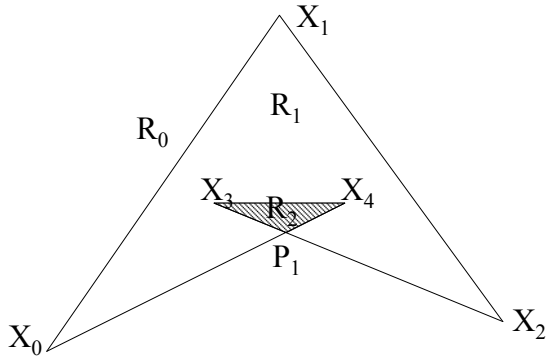
$\Rightarrow O(\log n)$ pro Einzeloperation:
Min, einfügen, löschen

Insgesamt auch 2n+s Einzeloperationen

$\Rightarrow O((n+s) \log n)$ Kosten

11

Verwandtes Problem: Zerlegung eines geschlossenen Polygonzuges in zusammenhängende Polyangebiete



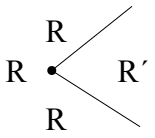
Schnittpunkte nach vorherigen Schema

X-Struktur wie bisher

Y-Struktur erweitert um Namen für Gebiet zwischen 2 benachbarten Linien und begrenzendem Polygonzug

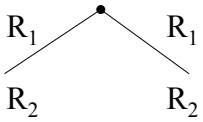
12

Fall 1:



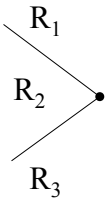
neue Punktfolge für R´ starten

Fall 2:



Punkt in Folgen aufnehmen

Fall 3:



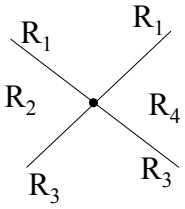
$R_1 \equiv R_3$

Punktfolgen für R_1, R_3 konkatenieren

Folge für R3 “umkehren”, z.B. durch dopellte Verkettung beider Listen. Details? Gebiet liegt links bw. rechts von Streckenzug?

13

Fall 4:

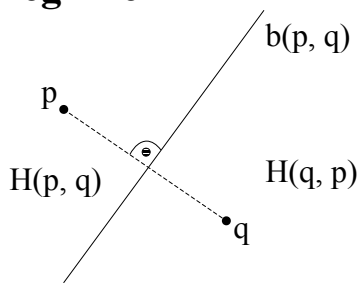


Gesamtkosten: $O((n+s) \log n)$

Details für Wartung der Punktfolgen?

14

Kap. 13.2 Sweep-Line für Nearest Neighbour Begriffe



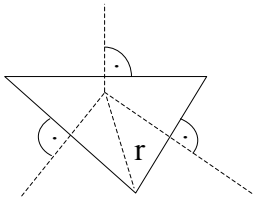
b(p, q) : Bisektor für p, q } teilt Ebene in
b(p, q) = b(q, p) } 2 Halbebenen

H(p, q) : Halbebene, die p enthält

H(q, p) : " " q "

$H(p, q) \neq H(q, p)$

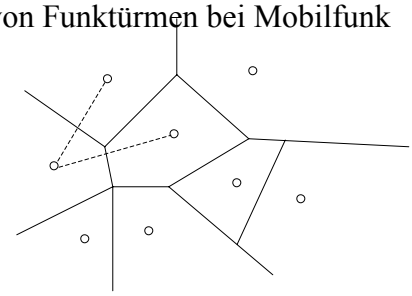
15



die Mittelsenkrechten im Dreieck schneiden sich im einem Punkt = Umkreis Mittelpunkt

1. Teil des Alg: Bestimmung des nächsten Nachbarn nach links:

Voronoi Diagramme für Punktmengen, z.B. für Einzugsbereich von Funktürmen bei Mobilfunk



konvexe Polygone
für einzelne Punkte

16

Sweep-Line Invariante:

Schnitt von SL mit Voronoi-Diagramm der Punkte links von SL

$\Rightarrow$ Unterteilung von SL in Intervalle, die zu Voronoi-Polygonen von Punkten links von SL gehören

Interessante Ereignisse:

(E1) neuer Punkt p

(E2) Schnittpunkt von **aktiven** Bisektoren

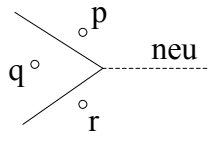
17

Aufgabe bei (E1) :

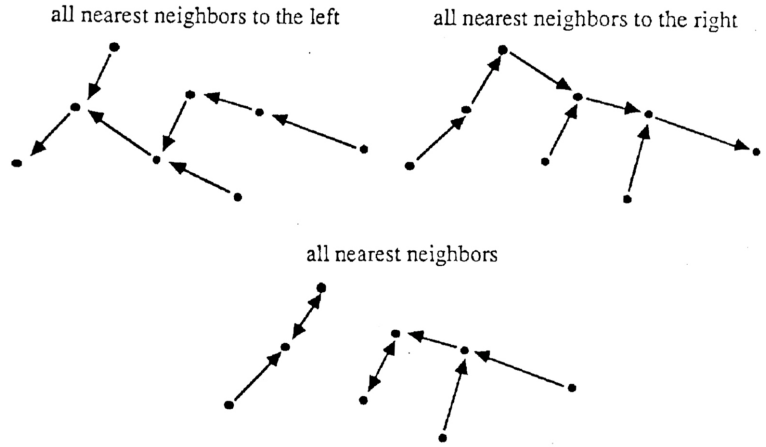
1. Bestimme nächsten linken Nachbarn von p, i.e. das q, in dessen Voronoi-Polygon p liegt.
2. Schreibe Voronoi-Polygon fort, wobei p zu Punktmenge links von SL hinzugenommen wird. Aktiviere und deaktiviere Bisektoren
3. Berechne Unterteilung von SL neu, aber inkrementell.

Aufgabe bei (E2) :

1. Aktiviere neuen Bisektor
2. Deaktiviere alte Bisektoren
3. Berechne Unterteilung von SL neu.



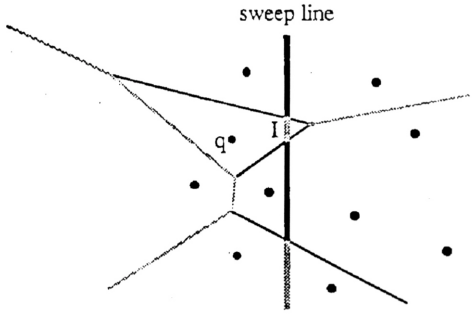
18



3. Choosing and updating the invariant

During the sweep we maintain the intersection of the sweep line with the *Voronoi*

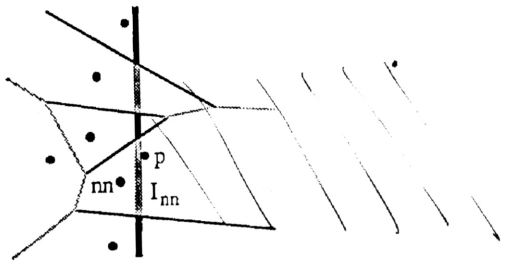
diagram of the points of S to the left of the sweep line. This intersection is a partitioning of the sweep line into disjoint intervals. Each interval I belongs to a point $q \in S$ to the left of the sweep line, in the sense that I is the intersection of the Voronoi polygon of q with the sweep line. Hence all points in I have q as their nearest neighbor to the left. The boundary separating two adjacent intervals is determined by the bisector of the two points corresponding to the two intervals. The edges of the Voronoi polygons intersecting the sweep line are called *active bisectors*, the corresponding points of S are called *active points*. To each active point there corresponds an *upper* and a *lower* active bisector. The intersections of the active bisectors with the sweep line define a total order on the active bisectors.



The intersection of the Voronoi diagram with the sweep line must be updated during the sweep whenever it changes, i.e. when a new interval is inserted or an interval is deleted. This happens when the sweep line encounters a new point or an intersection of two active bisectors; we discuss both cases.

New point p

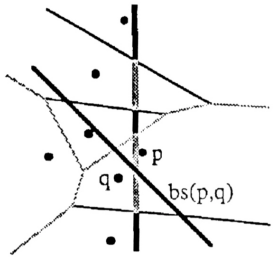
Find p's nearest neighbor nn to the left by determining the interval I_{nn} into which p falls.



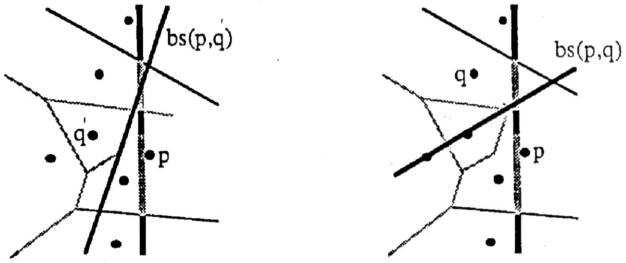
Updating the invariant requires computing the new interval I_p by determining the

lower and upper bisector bounding I_p. We describe how the upper bound of I_p is computed; the lower bound is handled in an analogous way. In order to determine which part of an interval I_q will become part of I_p we compute the bisector bs(p,q) of p and q (which is initially nn). We consider the two cases whether or not bs(p,q) intersects the sweep line between p and the upper bisector of q. H(p,q) is the half-plane which is defined by the perpendicular bisector of the line segment between p and q and which contains p.

1) If bs(p,q) intersects the sweep line outside the subinterval between p and the upper active bisector of q, this subinterval lies in H(p,q) and therefore will become part of I_p. Hence we deactivate the upper active bisector of q:



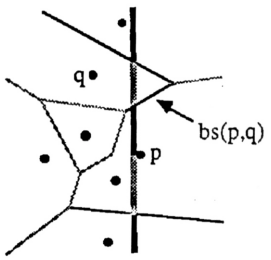
If q has an upper neighbor q' we replace q with q' and bs with the bisector bs(p,q') of p and q' since part of $I_{q'}$ may also become part of I_p. For this new configuration we determine again whether case 1 or 2 applies, and take the corresponding action recursively. If q has no upper neighbor, I_p is unbounded from above and the process terminates.



2) If bs(p,q) intersects the sweep line inside the subinterval between p and the upper active bisector of q, the subinterval between p and the intersection point of bs(p,q) with the sweep line belongs to H(p,q), and will therefore become part of I_p. The part

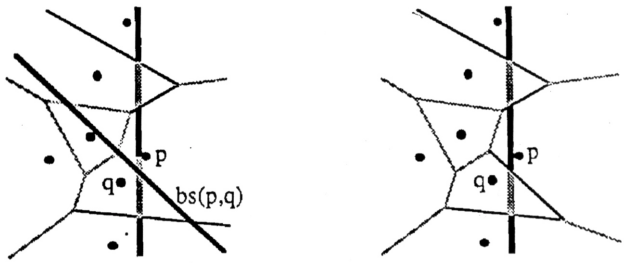
23

of I_q lying above bs(p) is closer to q than to p, therefore bs(p,q) becomes the active bisector bounding I_p from above and the process terminates:



24

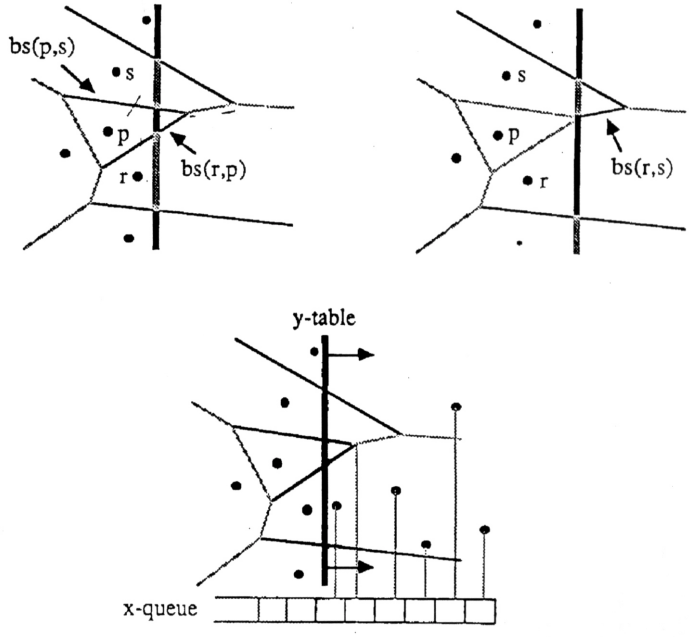
The following figure shows how the lower active bisector of I_p is computed.



Intersection point

An intersection point of two active bisectors is processed by removing an interval I_p, then separating its two former adjacent intervals I_r and I_s. As shown in the figure below, let r be the lower neighbor of p, s the upper neighbor of p. The two bisectors bs(r, p) and bs(p,s) are removed, the new bisector bs(r,s) is inserted.

25



26