Punktlokalisation

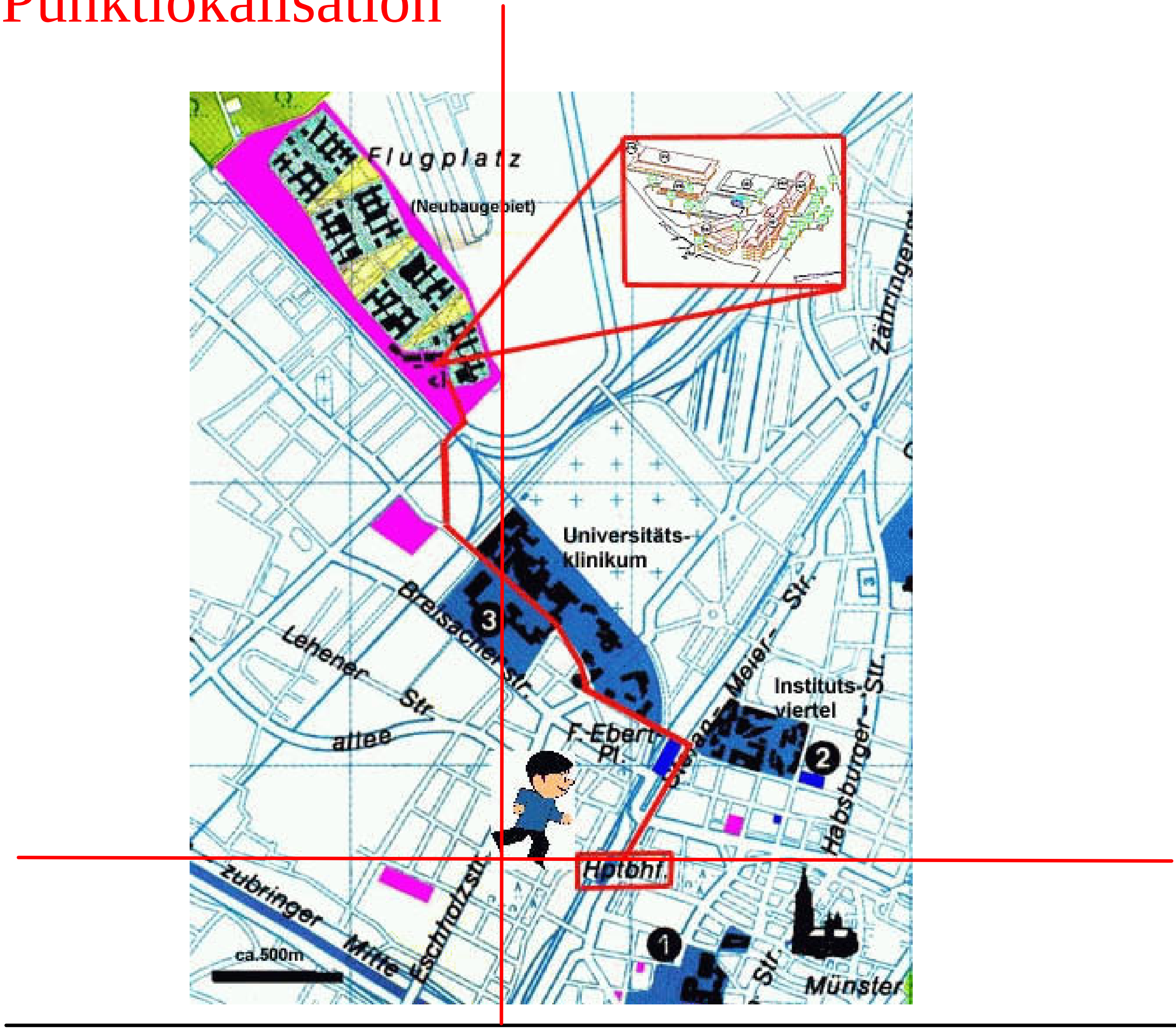
1. Trapez-Zerlegungen

2. Eine Suchstruktur

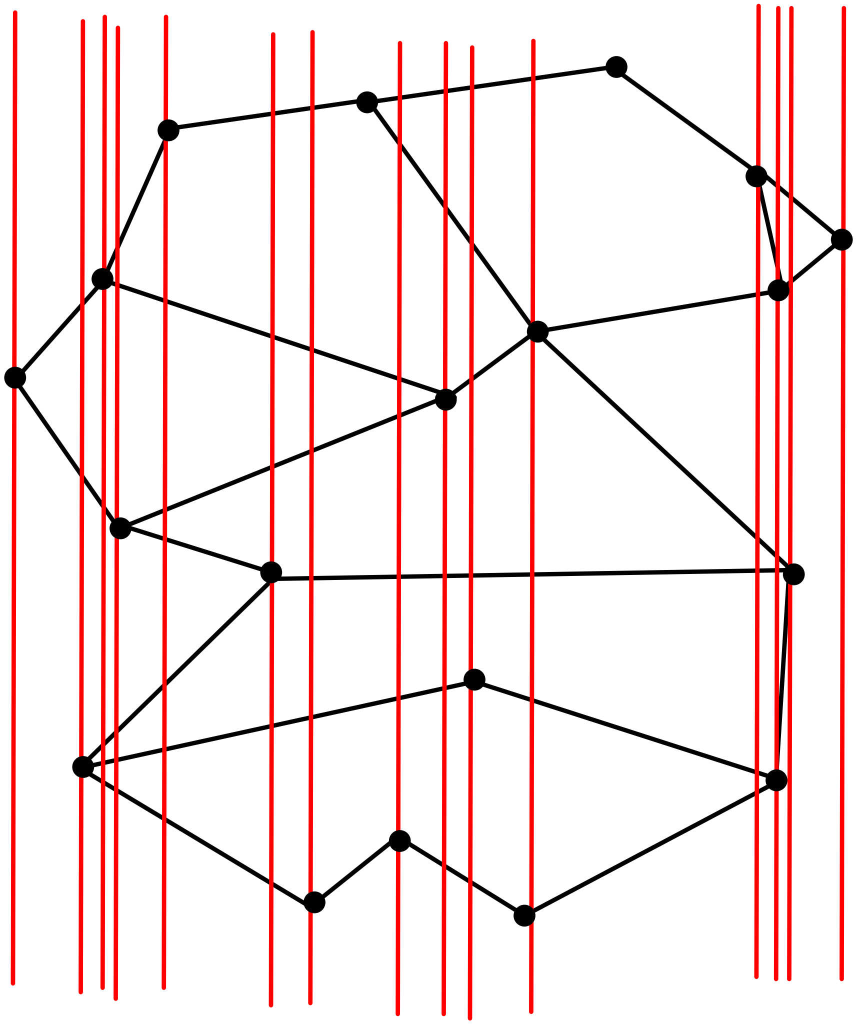
3. Randomisierter, inkrementeller Algorithmus zur Konstruktion der Trapez-Zerlegung

4. Analyse

Punktlokalisation



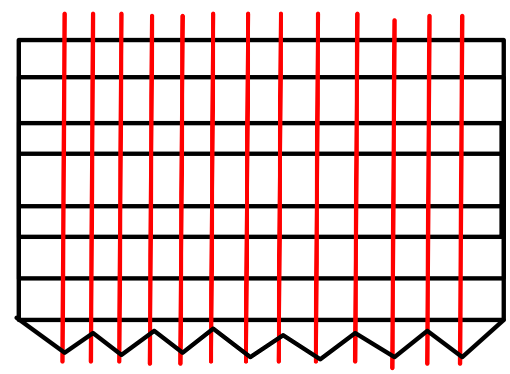
Einteilung in Streifen



Anfragezeit: $O(\log n)$

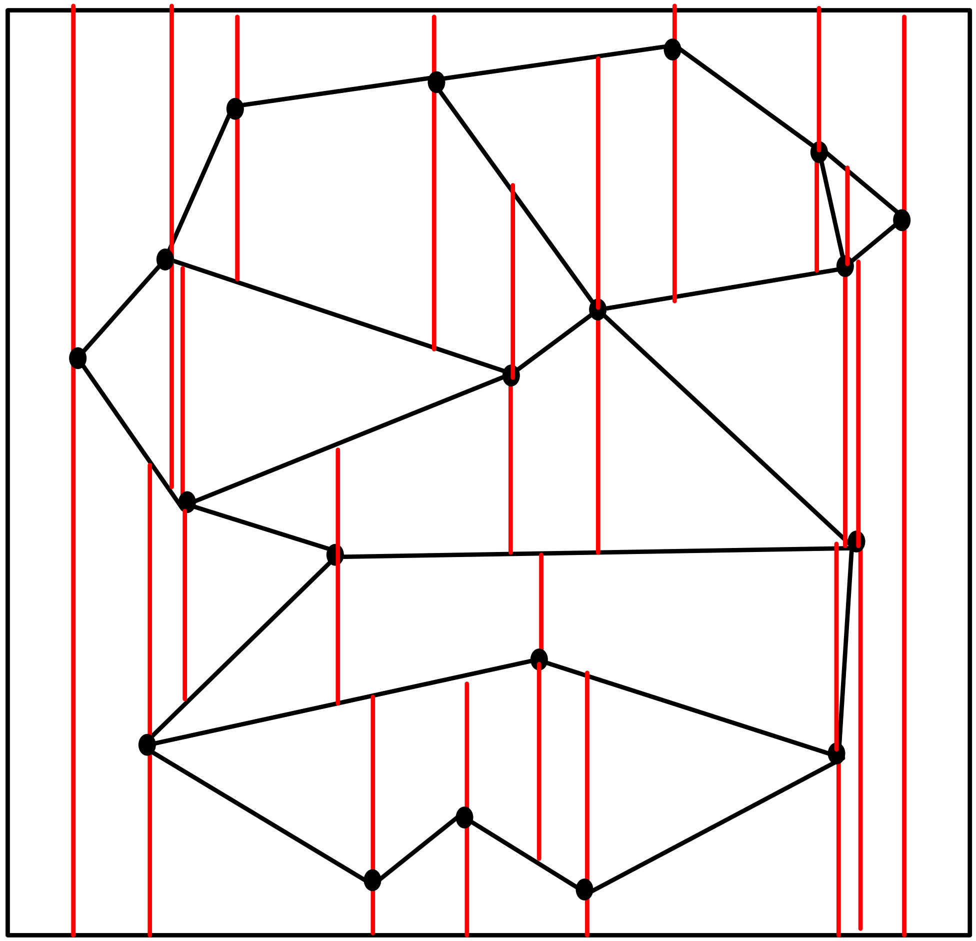
binäre Suche in x und dann binäre Suche in y Richtung.

Leider: Speicherplatz $\Theta(n^2)$



planare Aufteilung der Ebene

Einteilung in Trapeze



Geg: n Segmente mit Bounding Box R
(nicht-schneidend)

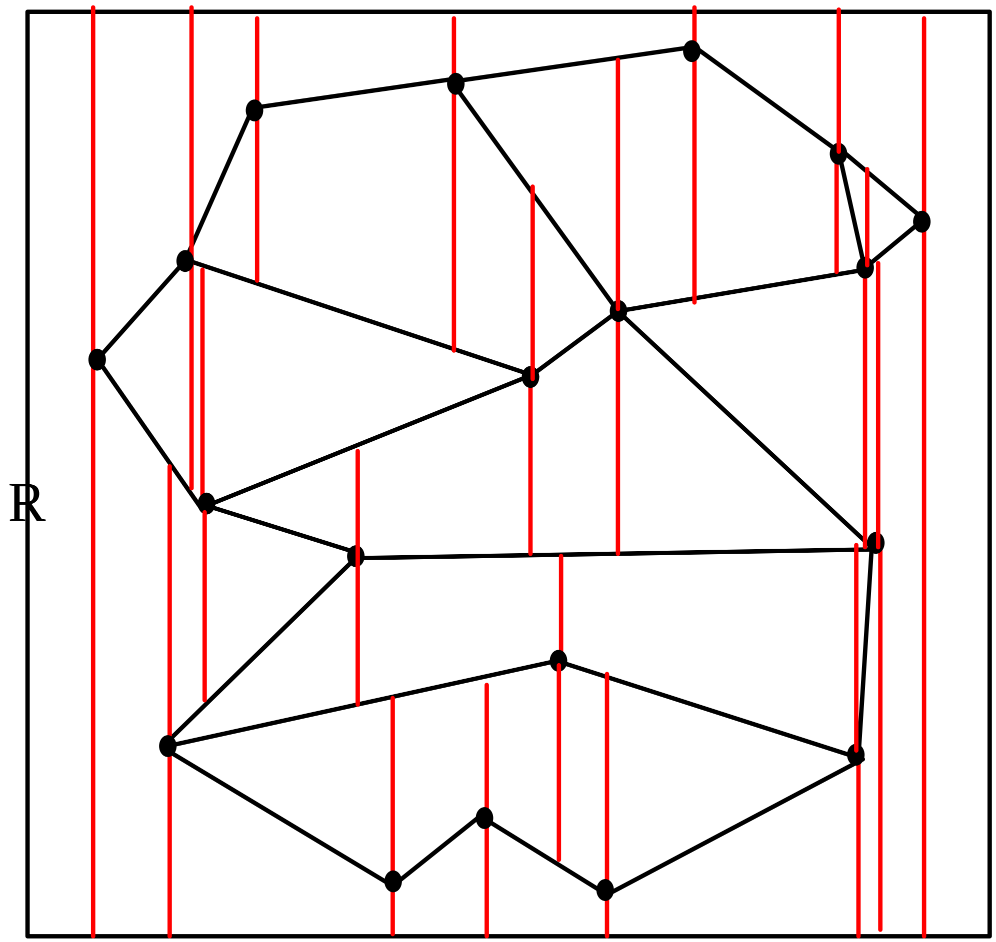
Ges: vertikale Dekomposition T(S)

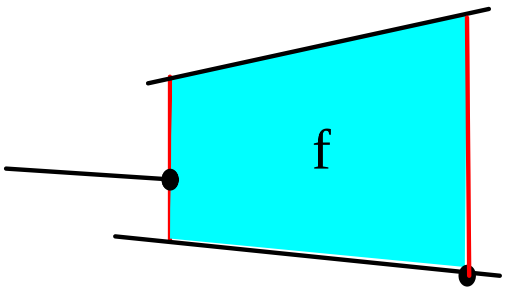
Annahme:
x-Koordinaten paarweise verschieden

Beobachtungen:

Jede vert. Kante hat mindestens einen Punkt mit einem Segmentende gemein.

Beobachtungen:



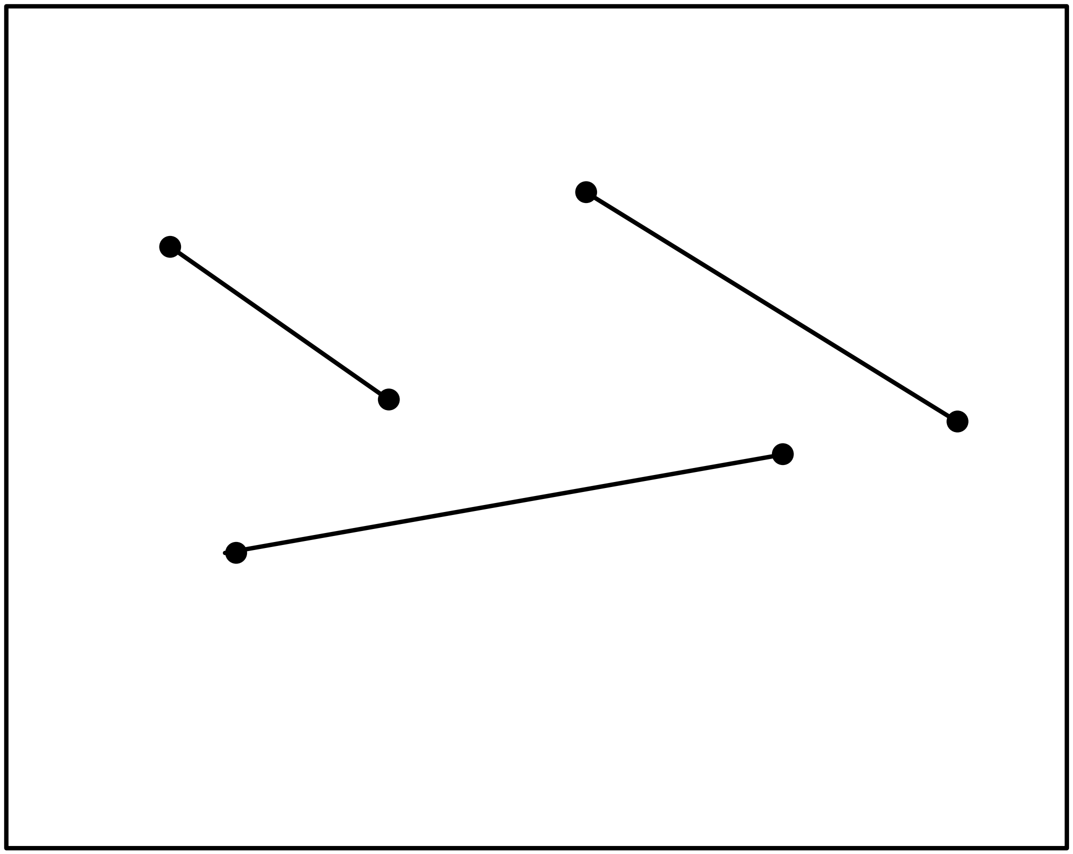


f ist konvex

f ist beschränkt

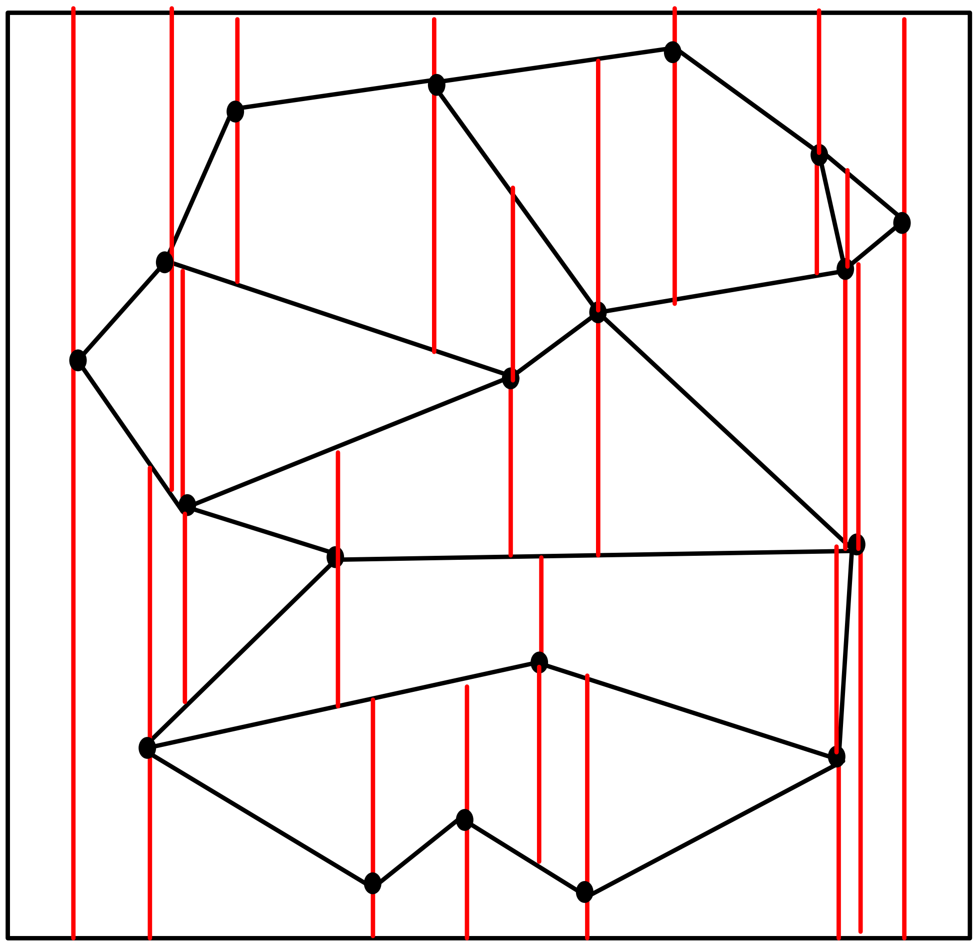
Jede nicht-vertikale
Kante von f ist Teil eines
Segmentes von S oder
des Recktecks R

Trapez-Zerlegung für Menge von Liniensegmenten



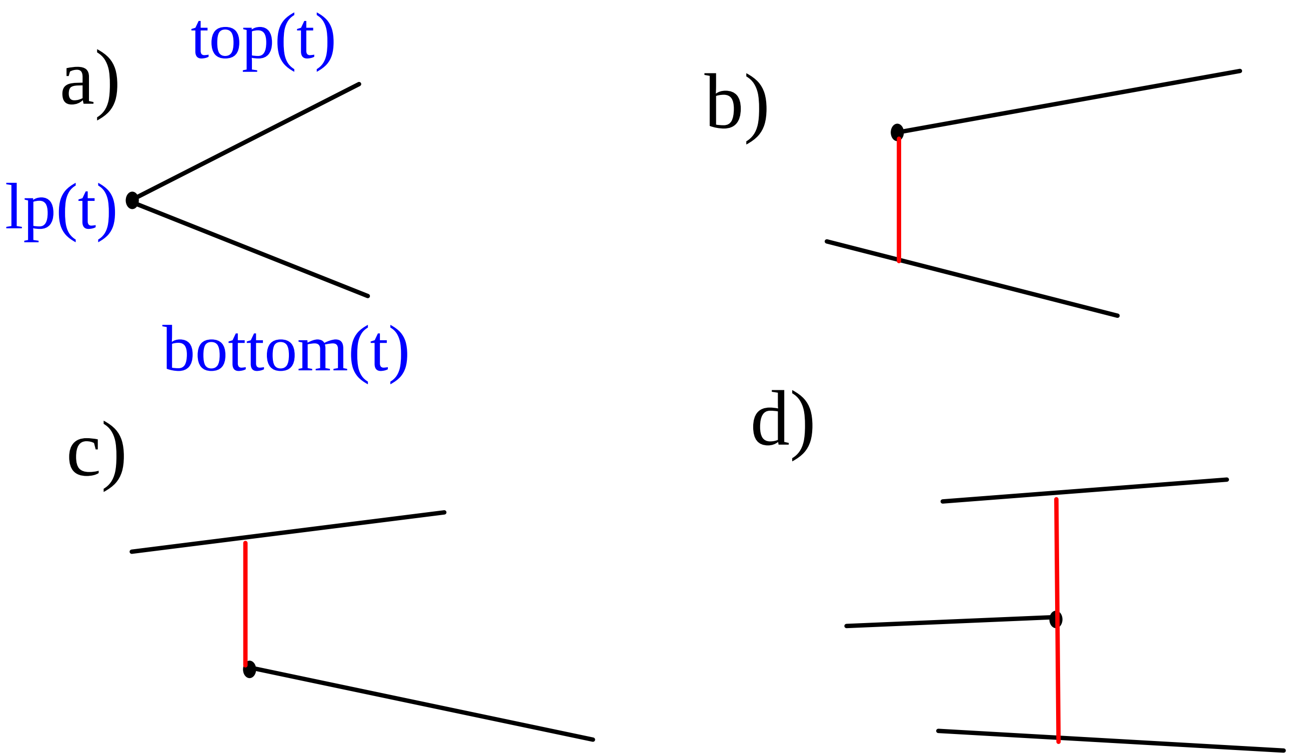
Jede Fläche der Trapez-Zerlegung einer Menge S von Liniensegmenten in allgemeiner Lage hat eine oder zwei vertikale und genau zwei nicht-vertikale Seiten

Für jedes Trapez t in T(S) ist die linke vertikale Begrenzung definiert durch den Endpunkt lp(t) eines Segmentes aus S.



rp(t) ist analog definiert.

5 Fälle (für linke Trapezkante)



e) linke Kante ist gleich bounding box-Kante

(Gilt für das linkeste Trapez!)

Satz: Die Trapez-Zerlegung T(S) einer Menge S von n Liniensegmenten in allgemeiner Lage besteht aus höchstens 6 n + 4 Ecken und 3 n + 1 Trapezen.

Bew.: (1) Jede Ecke von T(S) ist

- Ecke von R oder	4
- Endpunkt eines Segments von S oder	2 n
- Punkt, an dem eine Vertikale durch Endpunkt eines Segments von S auf Element von S oder auf R trifft.	2 * (2 n)
	6 n + 4

Satz: Die Trapez-Zerlegung T(S) einer Menge S von n Liniensegmenten in allgemeiner Lage besteht aus höchstens 6 n + 4 Ecken und 3 n + 1 Trapezen.

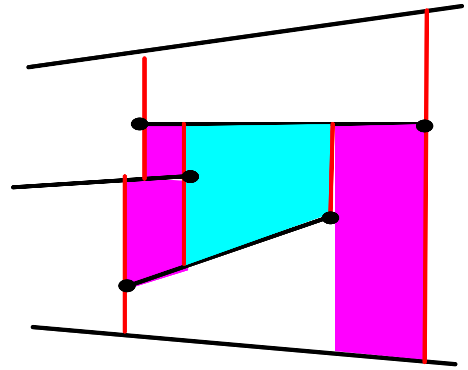
Bew.: (2) Betrachte für jedes t in T(S) den Punkt lp(t):

- Für genau ein t ist eine Ecke von R der Punkt lp(t). 1
- Jeder linke Endpunkt eines Segmentes aus S ist lp(t) für höchstens zwei t aus T(S). 2n
- Jeder rechte Endpunkt eines Segmentes aus S ist lp(t) für höchstens ein t aus T(S). n

---

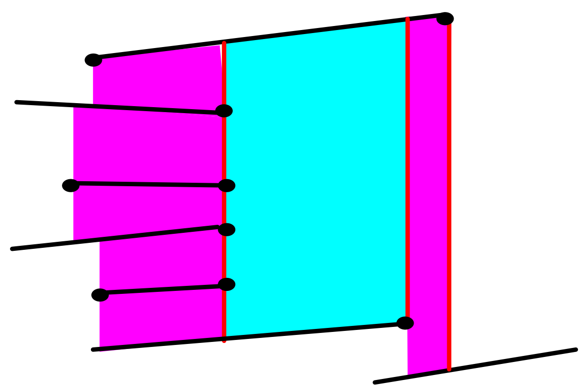
3 n + 1

Nachbartrapeze Längs vertikaler Kante benachbart



(a) Segmente in allgemeiner Lage:

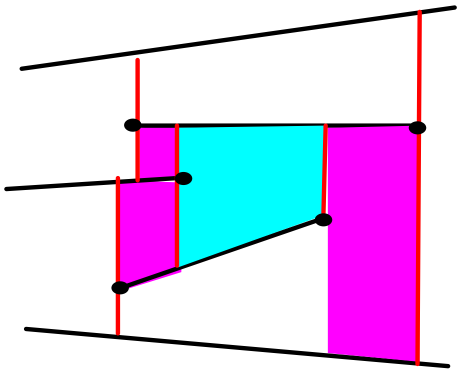
Jedes Trapez hat höchstens
4 Nachbartrapeze.



(b) Beliebige Menge S von Segmenten

Anzahl Nachbartrapeze unbeschränkt.

Vertikale Nachbarschaften, Segmente in allgemeiner Lage



Trapez t’ ist (vertikaler) Nachbar von t
gdw
top(t) = top(t’) oder bottom(t) = bottom(t’)

t’ = ul_neighbor(t)
gdw
t’ ist oberer linker Nachbar von t
gdw
top(t) = top(t’) und t’ links von t.

bl_neighbor, ur_neighbor, br_neighbor ......

Speicherung einer Trapez-Zerlegung T(S):

Speichere die Menge S und für jedes Trapez t:
- Zeiger auf top(t), bottom(t)
- Zeiger auf lp(t), rp(t)
- Zeiger auf die maximal 4 Nachbarn ul, bl, ur, br

Beob.: t ist durch top(t), bottom(t), lp(t), rp(t) eindeutig bestimmt!

Eine Suchstruktur

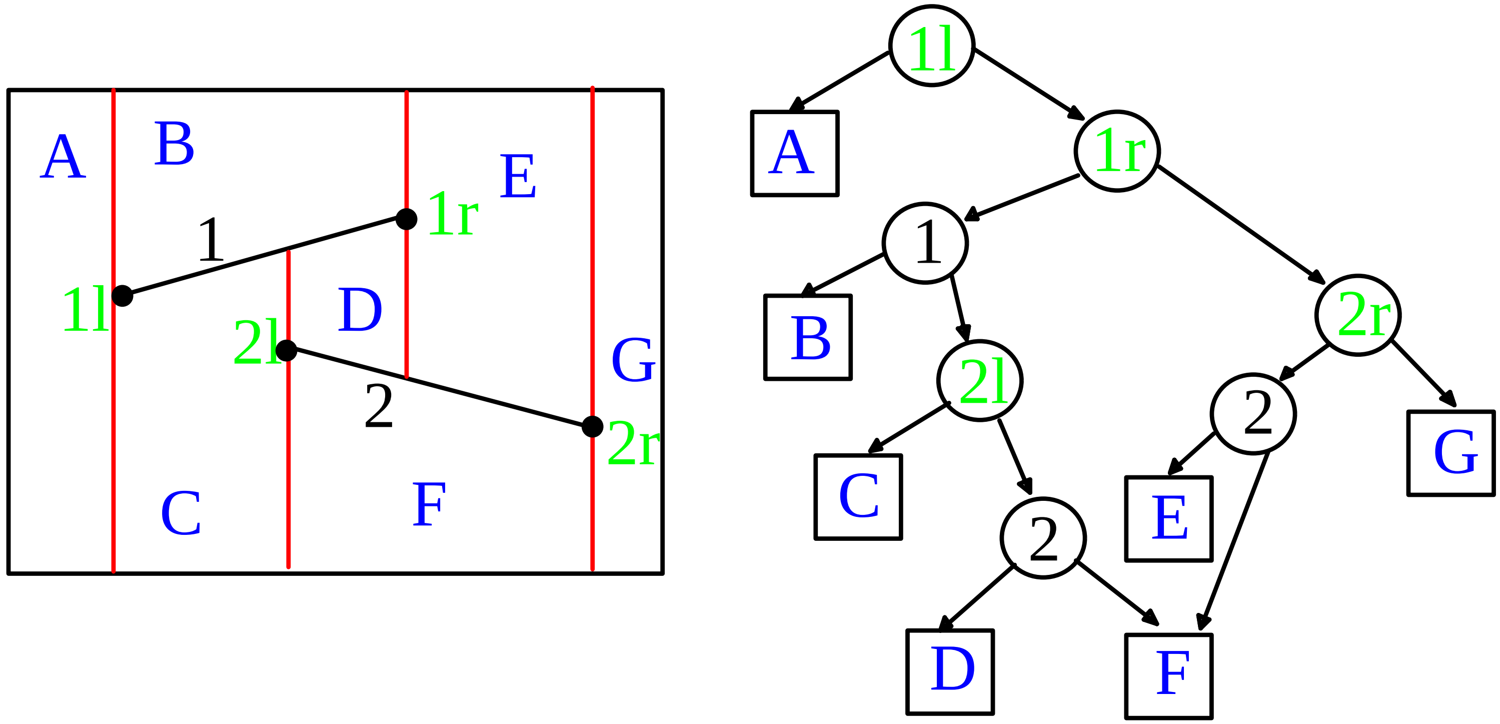
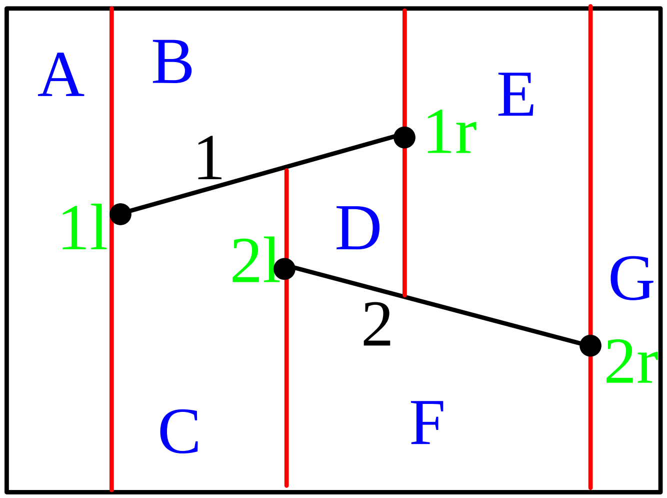


Diagramm:
Endpunkte entscheiden links, rechts
Segmente entscheiden oben, unten


A
B
1
1r
E
1l
D
2l
G
2
2r
C
F

Ein randomisierter inkrementellerAlgorithmus

Eingabe: Menge S von n sich nicht-schneidenen Segmenten
Ausgabe: Trapez-Zerlegung T(S) und Suchstruktur D

Berechne R, initialisiere T(S) und D

Bringe die Elemente s1,...,sn von S in zufällige Reihenfolge

for i = 1,...,n do

> Nehme si hinzu und verändere
>
> T(Si-1) => T(Si) und D(Si-1) => D(Si)

Invariante: Im Schritt i ist T(Si) korrekte Trapez-Zerlegung von Si und D(Si) eine zugehörige Suchstruktur.

Ein randomisierter inkrementellerAlgorithmus

Eingabe: Menge S von n sich nicht-schneidenen Segmenten

Ausgabe: Trapez-Zerlegung T(S) und Suchstruktur D

Berechne R, initialisiere T(S) und D

Bringe die Elemente s1,...,sn von S in zufällige Reihenfolge

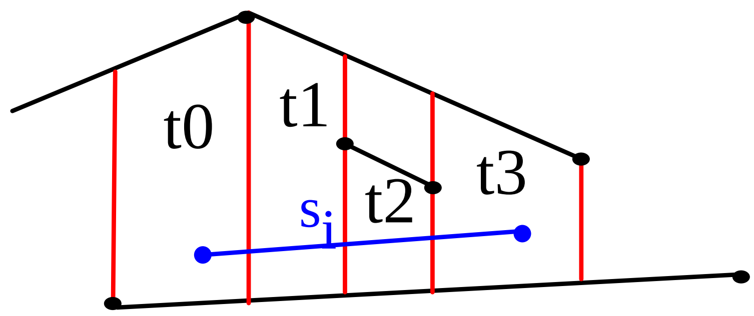
for i = 1,..,n

- Finde Menge t1,t2,...,tk Trapezen, die si schneidet
- Ersetze t0,t1,..,tk in T durch neue Trapeze
- Entferne t0,t1,..,tk in D und erzeuge neue Blätter
- Verbinde neue Blätter mit inneren Knoten

Fragen:

- Wie finde ich die Schnitttrapeze?
- Wie werden T und D aktualisiert
  - (a) falls neues Segment keine früheren schneidet
  - (b) falls neues Segment frühere schneidet ?

Finden der Schnitttrapeze



t_{j+1} ist rechter Nachbar von t_j

In T(Si) werden exakt diejenigen Trapeze verändert, die von s_i geschnitten werden.

Falls $rp(t_j)$ liegt oberhalb von s_i, dann ist

t_{j+1} = lr_neighbor(t_j),

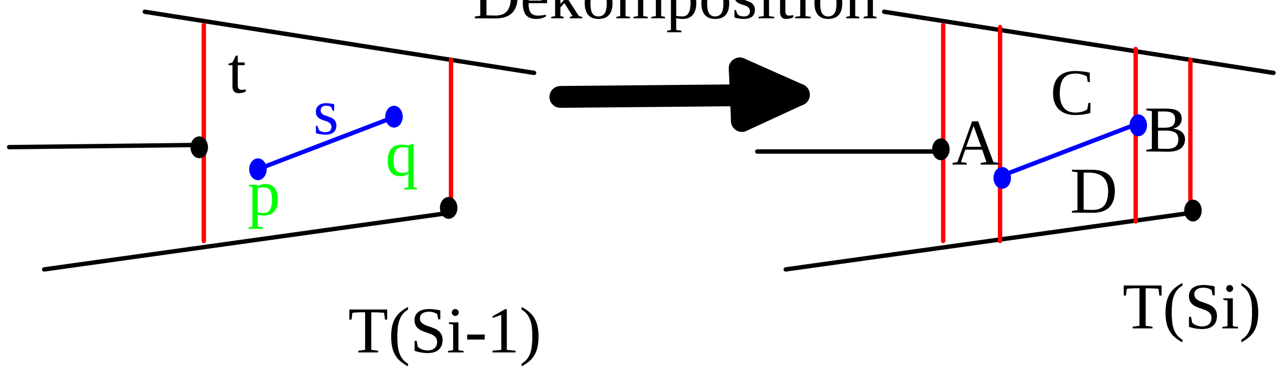
sonst ist t_{j+1} = ur_neighbor(t_j).

Clou:

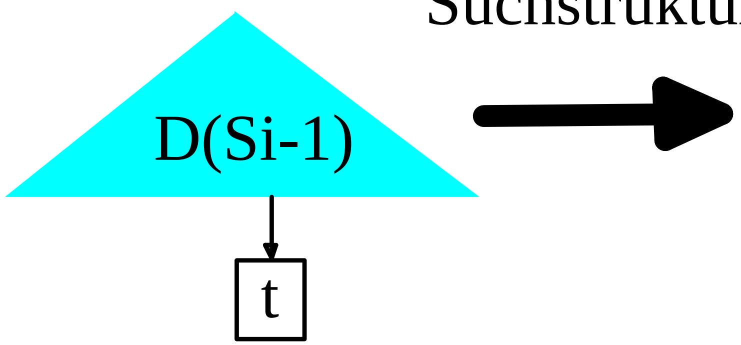
t0 kann durch eine Anfrage in der Suchstruktur D(Si-1) der Iteration i-1 gefunden werden.

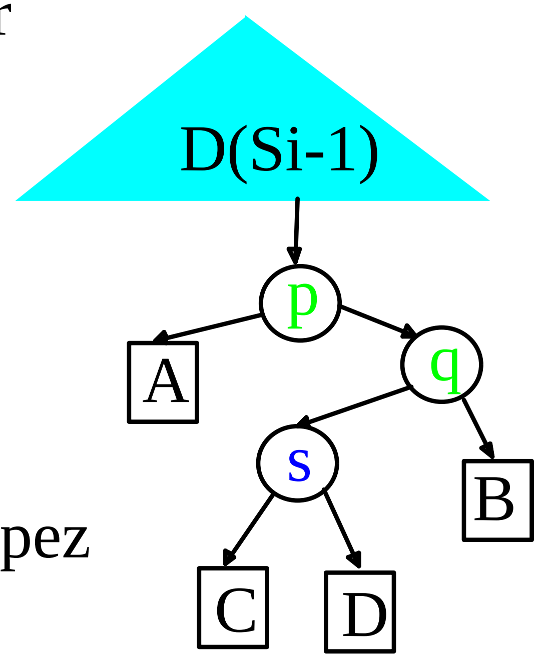
Segment komplett enthalten

Dekomposition



Suchstruktur



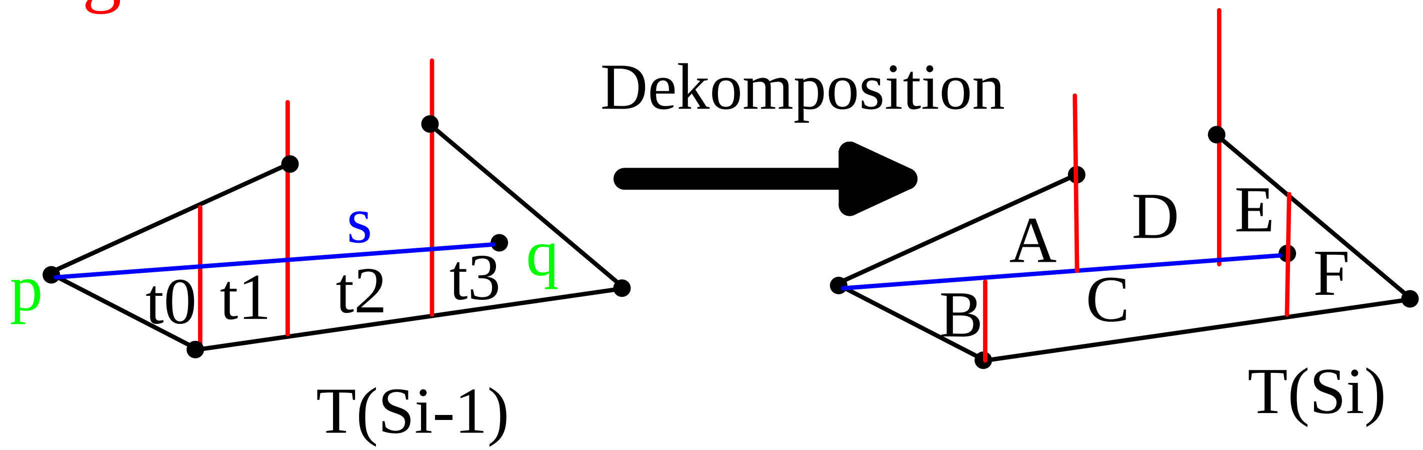


4 neue Trapeze - 1 altes Trapez

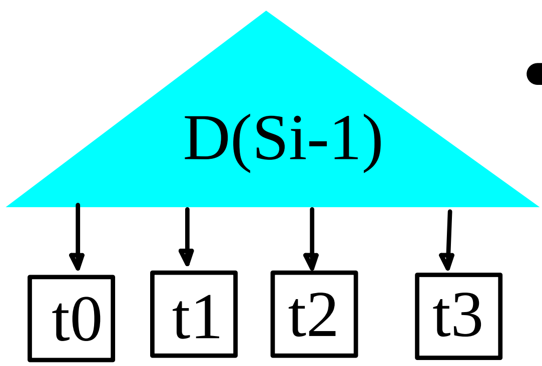
Suchtiefe + 3

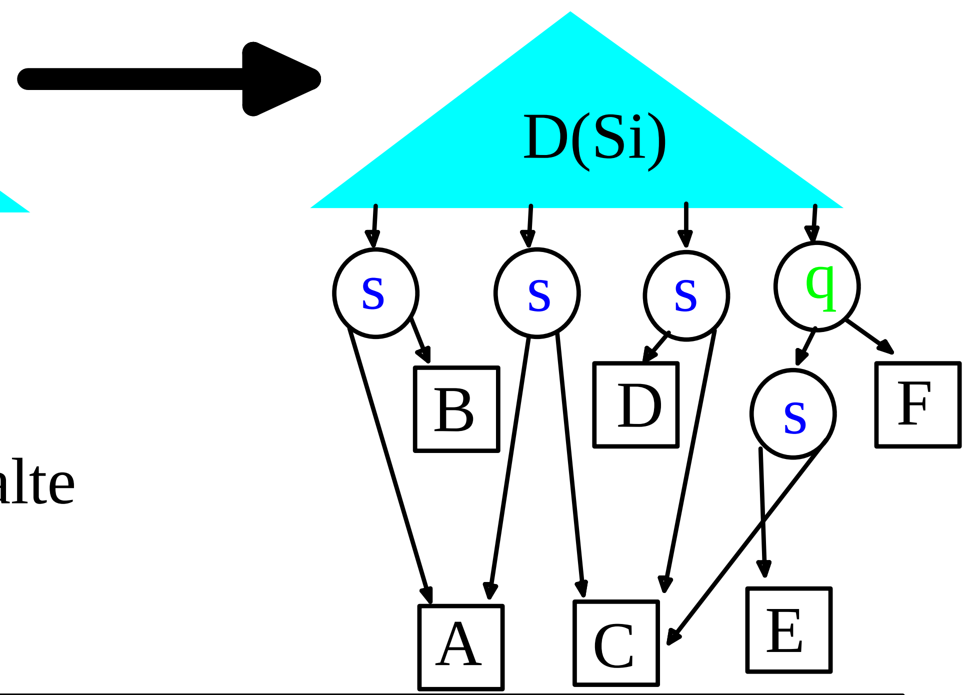
Segment schneidet

Dekomposition



Suchstruktur





6 neue Trapeze - 4 alte

Tiefe +2

Abschätzung der Tiefe der Suchstruktur D

S Menge von n Segmenten in allgemeiner Lage,
q beliebiger, fester Anfragepunkt.

Tiefe von D: im worst case: 3n, im Mittel: O(log n)

Betrachte Anfrage mit q in D
X_i = # in Iteration i erzeugter Knoten auf dem Suchpfad

$X_i \le 3$

P_i = Wahrscheinlichkeit dafür, dass im Schritt i ein Knoten auf dem Suchpfad nach q erzeugt wird

$E[X_i] \le 3 P_i$

Beobachtung: Im Schritt i wird Knoten auf Suchpfad nach q erzeugt, gdw

Trapez t(q, i-1) in $T(S_{i-1})$, das q enthält, ist nicht dasselbe wie das Trapez t(q, i), das q in $T(S_i)$ enthält.

Sei t(q,i) das Trapez, das q in der zu Schritt i gehörenden Zerlegung T(Si) enthält.

=>   E[Xi] <= 3 Prob[t(q,i) <> t(q,i-1)]

Beob.: t(q,i) hängt nur von Si, nicht von der Reihenfolge ab,
in der die Elemente von Si eingefügt wurden.

Rückwärtsanalyse: Pi ist die Wahrscheinlichkeit dafür, dass t(q,i) aus Zerlegung T(Si) verschwindet, wenn si aus T(Si) verschwindet.

Beob.: t=t(q,i) verschwindet mit si aus T(Si)
gdw.   si definiert eine der Seiten von t
gdw.   top(t(q,i)) = si   oder bottom(t(q,i)) = si oder
(lp(t(q,i)) oder rp(t(q,i)) verschwindet mit si)

Prob[top(t(q,i)=si] = Prob[bottom(t(q,i))=si] = 1/i

Prob[lp(t(q,i)) verschwindet mit si aus T(Si)] = 1/i

Prob[rp(t(q,i)) verschwindet mit si aus T(Si)] = 1/i

$$P_i = \mathrm{Prob}[t(q,i) <> t(q,i-1)] = \mathrm{Prob}[t(q,i) \text{ nicht in } T(S_{i-1})]$$

$$<= 4/i$$

$$\Rightarrow \quad E\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} 12/i = 12\, H_n$$

$$= O(\log n)$$

Analyse Größe der Suchstruktur D

D hat O(n) Blätter, weil jedes Blatt genau ein Trapez der Zerlegung T(S) repräsentiert.

\# Knoten von D

$$<= O(n) + \underbrace{\sum_{i=1}^{n} (\text{Anzahl innerer Knoten erzeugt in Schritt } i)}_{O(i)}$$

$$= O(n\hat{\ }2)$$

Im worst case hat D insgesamt O(n^2) Knoten

Analyse Größe der Suchstruktur D

Satz: D hat im Mittel O(n) viele Knoten

Bew. o.B.d.A betrachte innere Knoten, da Blätter in O(n)

Xi = # der in Schritt i erzeugten inneren Knoten

Es gilt: $$E[\sum_{i=1}^{n} Xi] = \sum_{i=1}^{n} E[Xi]$$

Sei d(t,s) = 1, falls t in T(Si) durch s gelöscht wird, und 0 sonst

Da maximal 4 Segmente ein gegebenes Trapez löschen, folgt:

$$\sum_{s \text{ in } Si} \; \sum_{t \text{ in } T(Si)} d(t,s) <= 4 \;\; \#(T(Si)) = O(i)$$

$$E[Xi] = 1/i \sum_{s \text{ in } Si} \; \sum_{t \text{ in } T(Si)} d(t,s)$$

$$<= 1/i \;\; O(i) = O(1)$$

Der Erwartungswert für die Zahl der im Schritt neu erzeugten inneren Knoten der Suchstruktur D ist in O(1).

$$=> \quad E[\sum_{i=1}^{n} Xi] = O(n)$$

Zusammenfassung:
Sei S eine planare Flächeneinteilung der Ebene mit n Kanten. Dann kann man in erwarteter Zeit O(n log n) eine Suchstruktur der erwarteten Grösse O(n) konstruieren, sodass für einen beliebigen Anfragepunkt q eine Point-Location Anfrage in erwarteter Zeit O(log n) beantwortet werden kann.