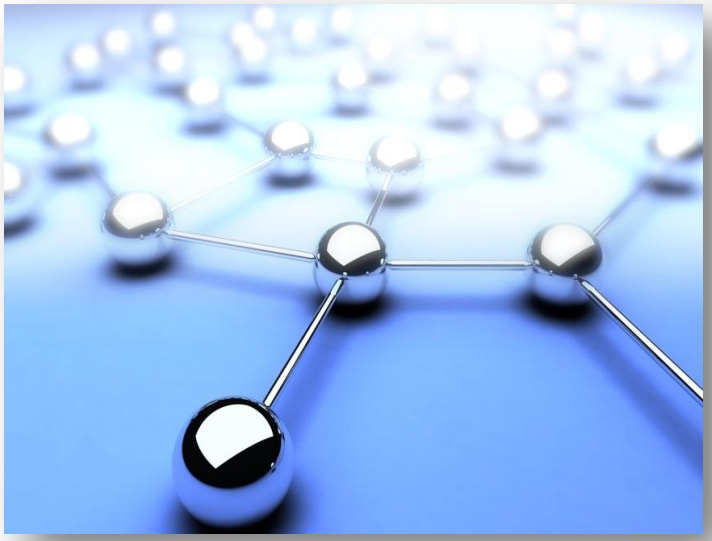
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Vorlesung Algorithmen für hochkomplexe Virtuelle Szenen

Sommersemester 2012



Matthias Fischer
mafi@upb.de

Vorlesung 2
10.4.2012


HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

BSP-Baum = Binary Space Partitions

- Motivation
- Idee
- Anwendungsbeispiel: Painters Algorithmus
- Aufbau und Konstruktion

Zuerst arbeiten wir mit 2D-BSP Bäumen,

anschließend mit dem 3D-Fall

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Binary Space Partitions

- Computational Geometry - Algorithms and Applications;
  Mark de Berg, Otfried Cheong, Marc de Kreveld, Mark Overmars;
  Springer Verlag, 2008.

  Kapitel: Binary Space Partitions

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Verfahren von Fuchs, Kedem & Naylor 1980

Ziel

- Tiefensortierung der Polygone bezüglich der Kameraposition
- Berechnung im Preprocessing, muss jedoch zur Laufzeit für „alle möglichen Kamerapositionen funktionieren“

Anwendung

- Painter‘s Algorithm (HSR = Hidden Surface Removal)
- Verdeckungsberechungen (Occlusion-Culling)
- Darstellung transparenter Objekte / Schatten

Eigenschaften

- Sehr effizient für statische Szenen
- Zeit- und speicherintensives Preprocessing
- „Lineare“ Zeit für die Darstellung, Vorsicht Größe des Baumes beachten!

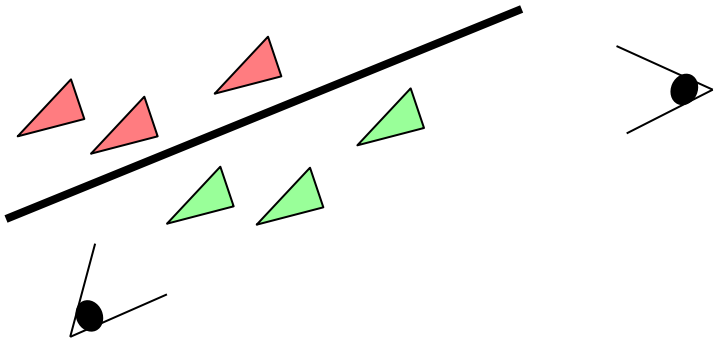
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wir betrachten BSP-Bäume von

- Linien im 2D-Raum
- Dreiecken/Polygonen im 3D-Raum

Grundlegende Idee:

- Tiefensortierung und Gruppierung
- Betrachte eine Szene als Gruppierung von Dreiecken



Wenn eine Gerade/Ebene gefunden werden kann, die eine Gruppierung aufteilt dann gilt:

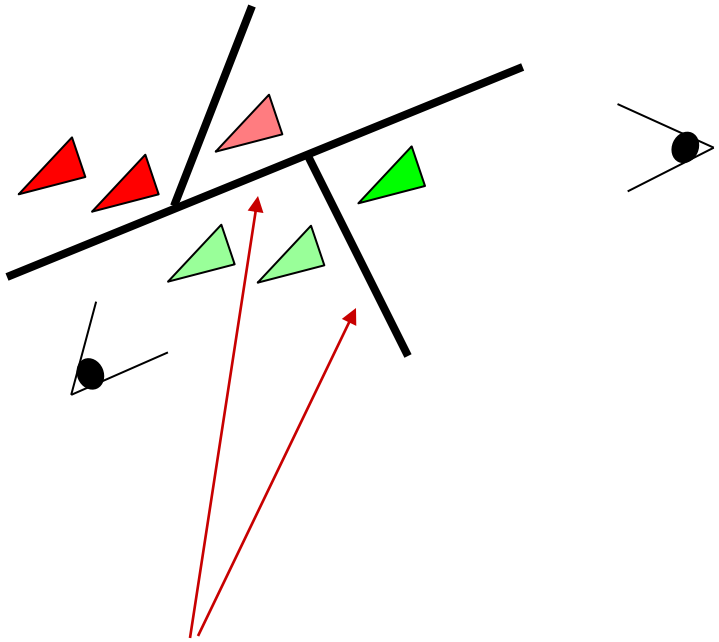
- Linien/Dreiecke auf der gleichen Seite wie der Betrachterstandpunkt
  werden nicht verdeckt von Linien/Dreiecken auf der anderen Seite (umgekehrt ja)

BSP-Baum
Idee

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Jede aufgeteilte Gruppierung kann weiter unterteilt werden:

- es ergibt sich ein Binärbaum, der die Unterteilung repräsentiert



Innere Knoten: Splittinggeraden im 2D / Splittingebenen im 3D

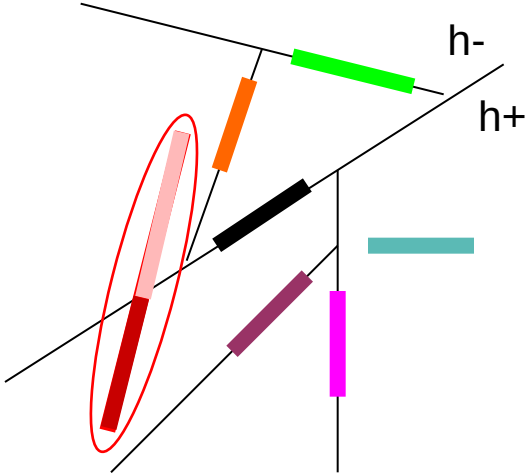
Blätter: Regionen im Raum,
disjunkte vollständige Aufteilung der 2D-Ebene oder des 3D-Raumes

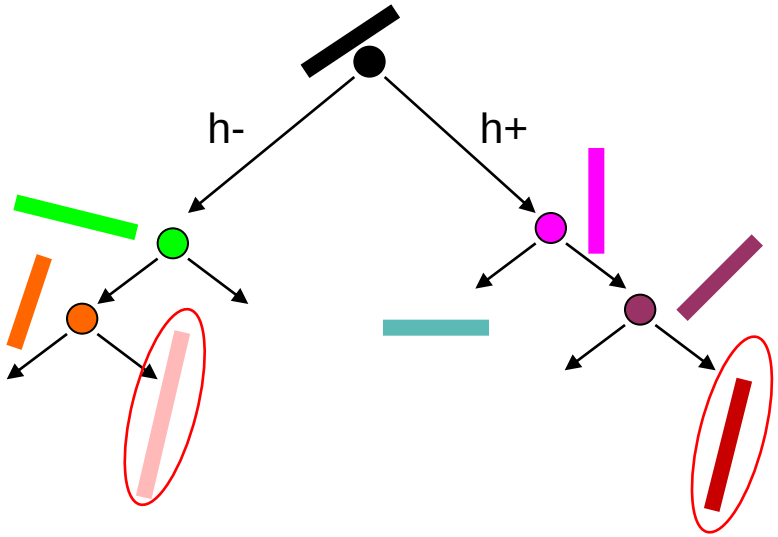
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

h-, h+ := linker bzw. rechter Halbraum (Zuordnung/Bezeichnung frei wählbar)

In diesem Beispiel

- legen wir Splittinggeraden durch ein Linienelement,
- speichern Splittinggerade und Linienelement im inneren Knoten,
- und speichern die in 2 Gruppen aufgeteilten restlichen Linienelemente in den beiden Teilbäumen.



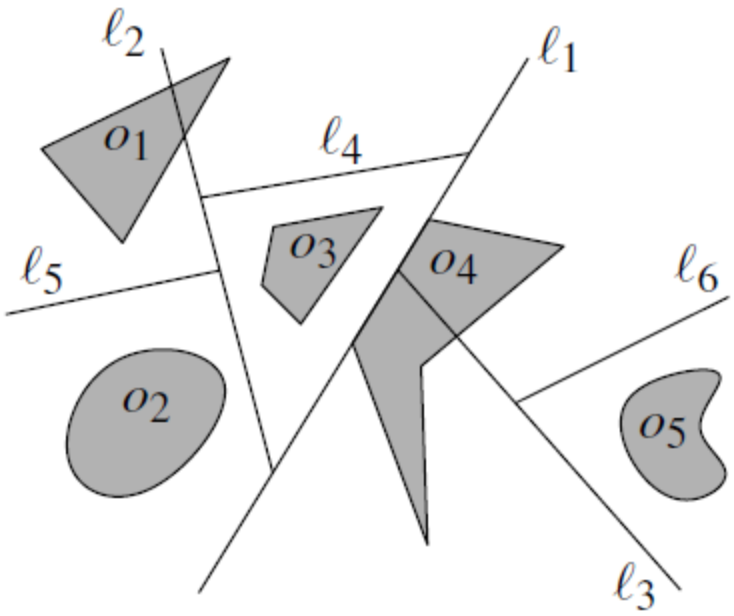


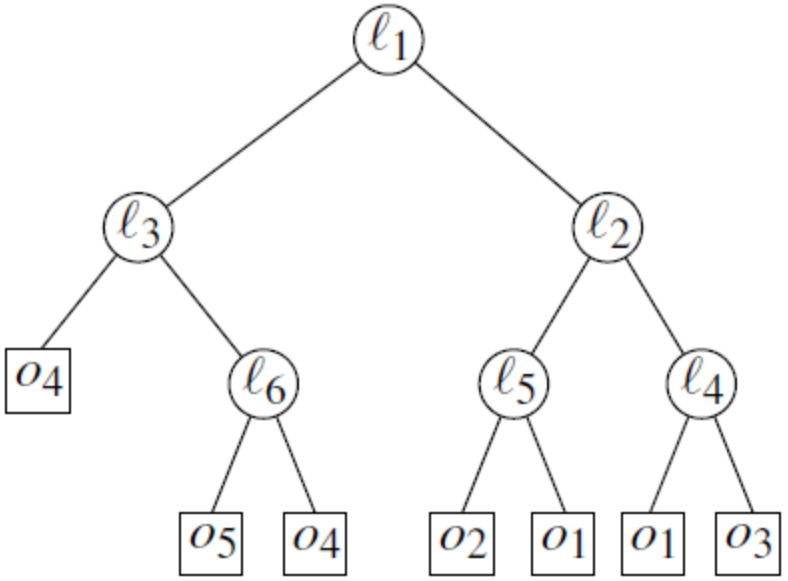
BSP-Baum
Idee

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

In diesem Beispiel

- legen wir die Splittinggeraden unabhängig von der Lage der Objekte,
- die Objekte speichern wir in den Blättern.

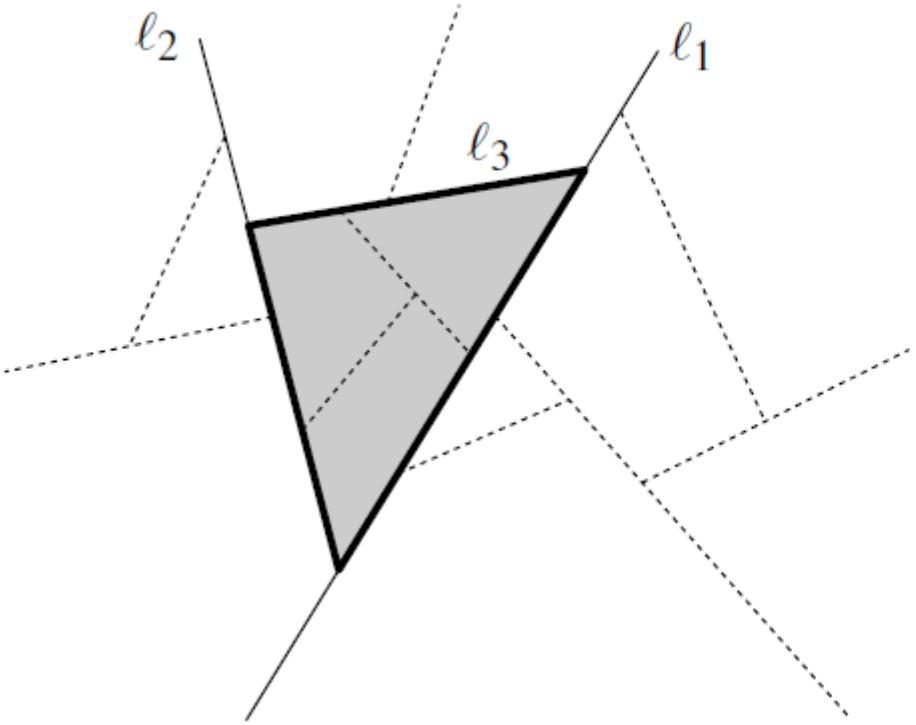


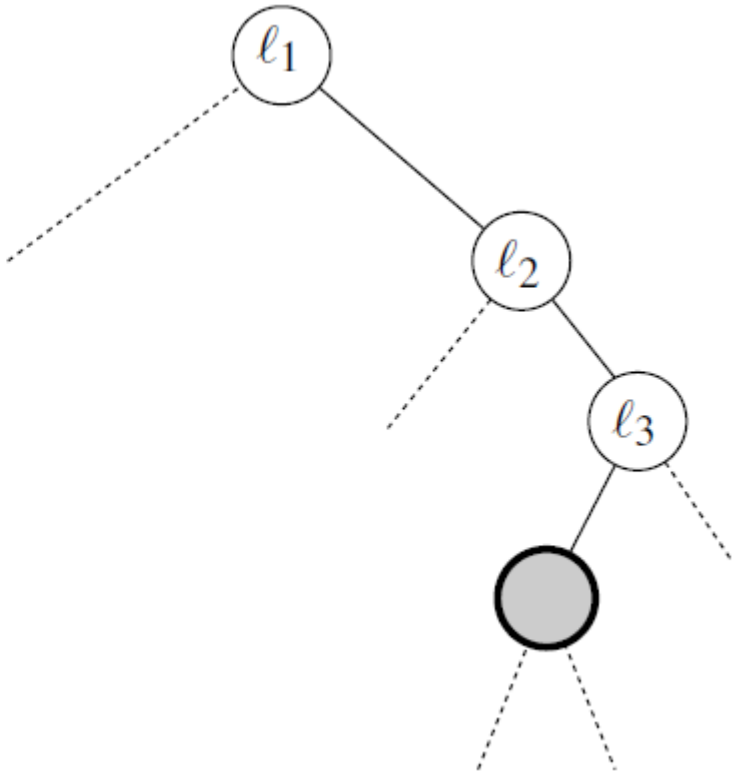


BSP-Baum
Idee

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

- Ein innerer Knoten des BSP-Baums entspricht einer Region (Teilfläche im 2D, Teilraum im 3D).
- Die Region kann geschlossen oder offen sein.





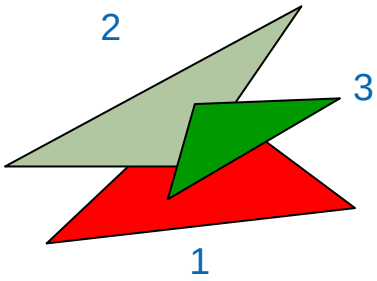
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

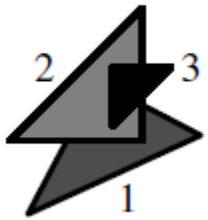
Wie wird eine Menge von Dreiecken gezeichnet?
Wie werden verdeckte Kanten entfernt?

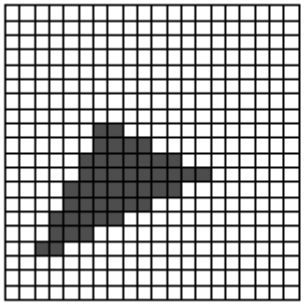
Mit diesem Beispiel demonstrieren wir die Idee der Tiefensortierung.

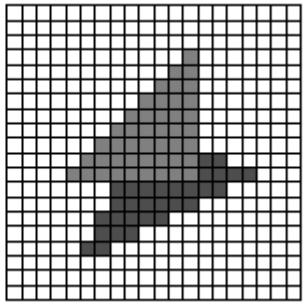
Erster Versuch einer Lösung (Painter‘s Algorithmus)

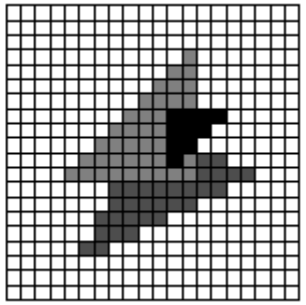
- sortiere Dreiecke nach der Entfernung zum Betrachter
- zeichne von hinten nach vorne









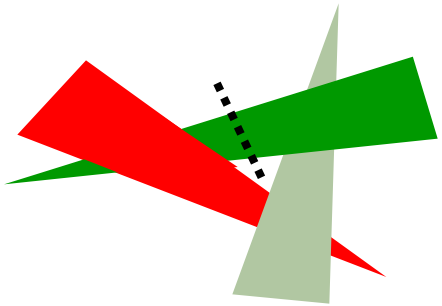


BSP-Baum
Anwendungsbeispiel

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Damit lässt sich nicht jede Anordnung von Dreiecken lösen:

- es können zyklische Überlappungen vorhanden sein
- im Beispiel muss man an der gestrichelten Linie trennen, um den Zyklus aufzulösen
- es entstehen **Fragmente**

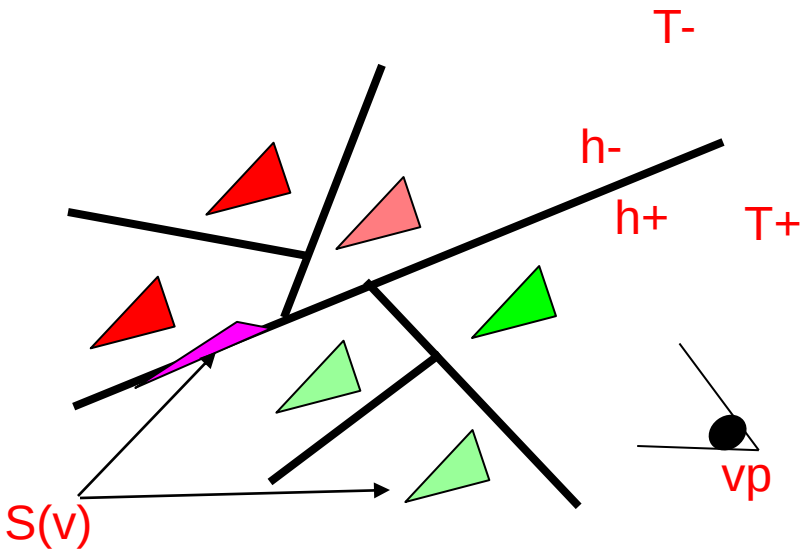


HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Zweiter Versuch einer Lösung (Painter‘s Algorithmus)

- Traversiere den BSP-Baum,
- Dreiecke von hinten nach vorne zeichnen,
- „Dreiecke“ können auch Fragmente sein, die vom BSP-Baum geteilt wurden.

S(v)	:=	Fragmente, die im Knoten des BSP-Baums gespeichert sind
vp	:=	Standort des Betrachters
T-,T+	:=	linker bzw. rechter Teilbaum



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Painters (Tree T, Viewpoint vp)

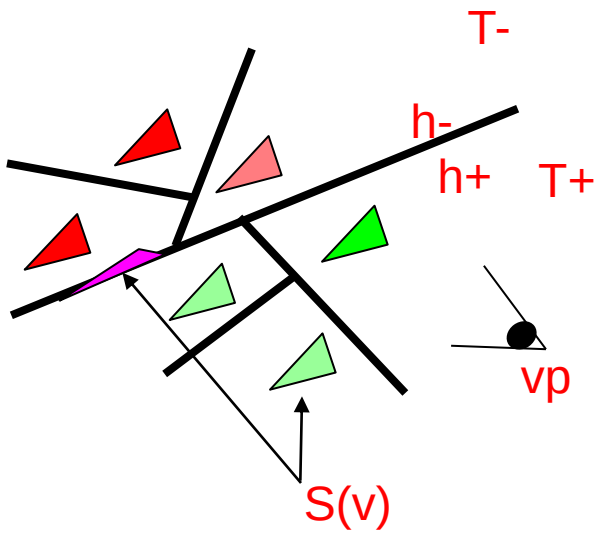
Sei v die Wurzel von T

if v ist ein Blatt
then zeichne die Fragmente S(v) des Knotens v

else if vp in h+
then *Painters* (T-, vp)
Fragmente S(v) des Knotens v zeichnen
Painters (T+, vp)

else if vp in h-
then *Painters* (T+, vp)
Fragmente S(v) des Knotens v zeichnen
Painters (T-, vp)

else *Painters* (T+, vp)
„von der Seite sind Fragmente unsichtbar“
Painters (T-, vp) (oder umgekehrte Reihenfolge der Aufrufe)



BSP-Baum
Aufbau und Konstruktion

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wir berechnen einen 2D BSP-Baum über einer Menge von Liniensegmenten

Eingabe: Eine Menge von *n* Segmenten $S = \{s_1, \dots, s_n\}$

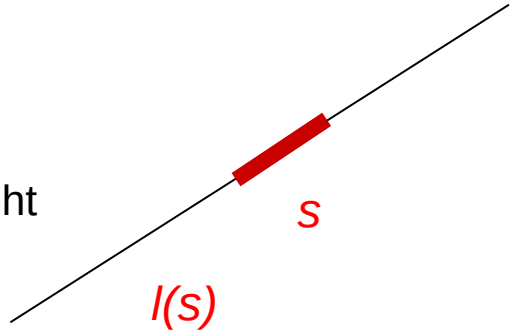
Ausgabe: Ein BSP-Baum für S

Idee:

- Von oben nach unten gehen (top – down),
- Segmente in zwei Hälften teilen,
- rekursiv fortsetzen.

Wie wählen wir die Splittinggerade?

- Gerade geht immer durch ein Segment
- l(s) := eine unendlich lange Linie, die durch das Segment s geht



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Algorithmus 2dBSP(S)

If $|S| \leq 1$

then erzeuge einen Baum T, der aus einem Blatt besteht, das S enthält
return T

else „Wir benutzen $l(s_1)$ als Gerade zum Aufteilen“

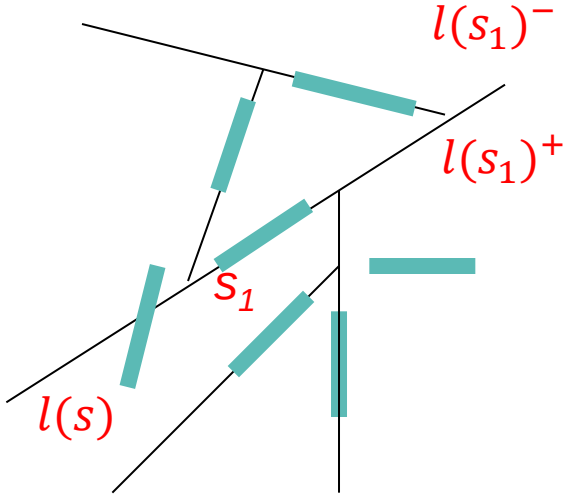
$S^+ := \{s \cap l(s_1)^+ | \forall s \in S\}$; $T^+ := 2dBSP(S^+)$;

$S^- := \{s \cap l(s_1)^- | \forall s \in S\}$; $T^- := 2dBSP(S^-)$;

Erzeuge einen BSP-Baum T mit Knoten v:

- linken Teilbaum T^-
- rechten Teilbaum T^+
- speichere im Knoten v: $S(v) = \{s \in S | \, s \subset l(s_1)\}$

return T

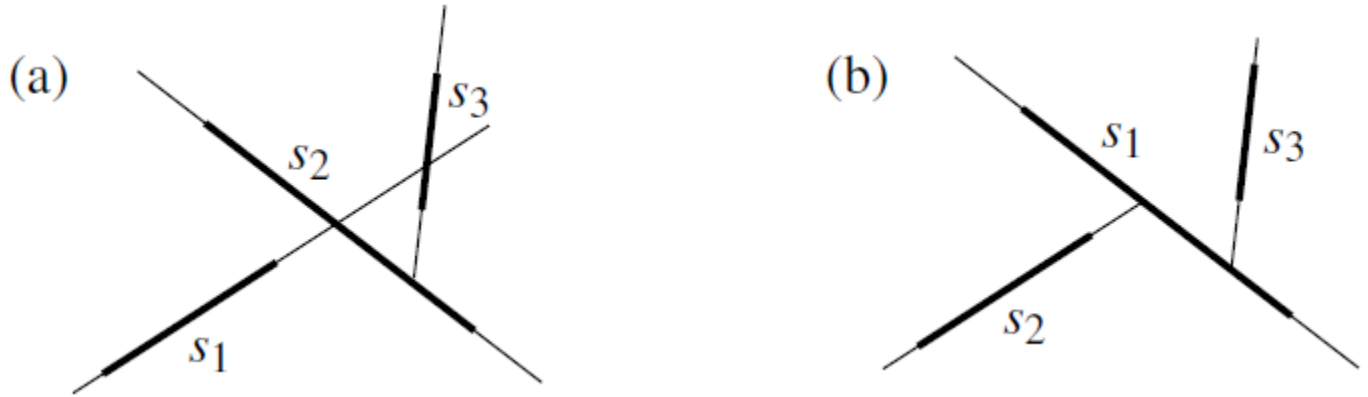


HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Definition
Größe eines BSP-Baums := Anzahl aller erzeugten Segmente

Wie groß kann ein BSP-Baum werden?

- die Größe beeinflusst Laufzeit und Speicherplatz
- unterschiedliche Reihenfolgen der Segmente führen zu unterschiedlich großen Bäumen
- die Größe des Baumes hängt von der Anzahl aufgeteilter Segmente ab



BSP-Baum
Aufbau und Konstruktion

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wie erzeugen wir einen möglichst kleinen Baum?

Lösungsansatz

Wähle das Segment welches möglichst wenig andere Segmente schneidet

- Dieser Greedy-Ansatz verhindert keine großen Bäume
- Berechnung des Segmentes, das wenig Schnitte erzeugt ist zeitaufwendig

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Ansatz Randomisieren

- Wähle die Segmente zufällig
- Zufällig wählen kann gut sein, wenn die Auswahl schwer fällt

Algorithmus 2dRandomBSP (S)

- Erzeuge von der Menge S eine zufällige Permutation $S' = s_1, \ldots, s_n$
- T := 2dBSP(S')
- **return** T

Anders aufgeschrieben lautet der Algorithmus:

- Statt immer das erste Element s_1 der übergebenen Menge S zu wählen
- und die Permutation am Anfang zu berechnen,
- wird in jedem Schritt das Splittingelement s aus der Menge S zufällig gewählt.

BSP-Baum
Aufbau und Konstruktion

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Lemma

Die erwartete Anzahl von Fragmenten,
die der Algorithmus **2dRandomBSP(S)** berechnet, ist $O(n \log(n))$

Warum?

- Wir analysieren die erwartete Größe des BSP-Baums
- Durchschnittliche Größe über n! Permutationen
  (so haben wir S‘ erzeugt)

BSP-Baum
Aufbau und Konstruktion

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

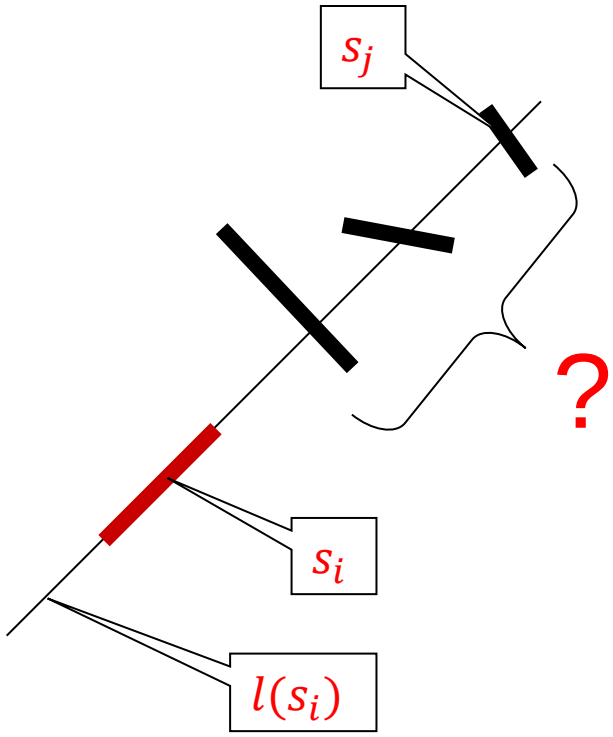
Sei s_i ein festes Segment !

Wir analysieren

- die erwartete Anzahl von Segmenten, die geschnitten werden,
- wenn $l(s_i)$ als Splittingsegment gewählt wird.

Zwei Fragen:

1. Wie viele Segmente liegen „vor“s_i ?
2. Wann wird ein Segment s_j, dass „vor“s_i liegt, geschnitten?

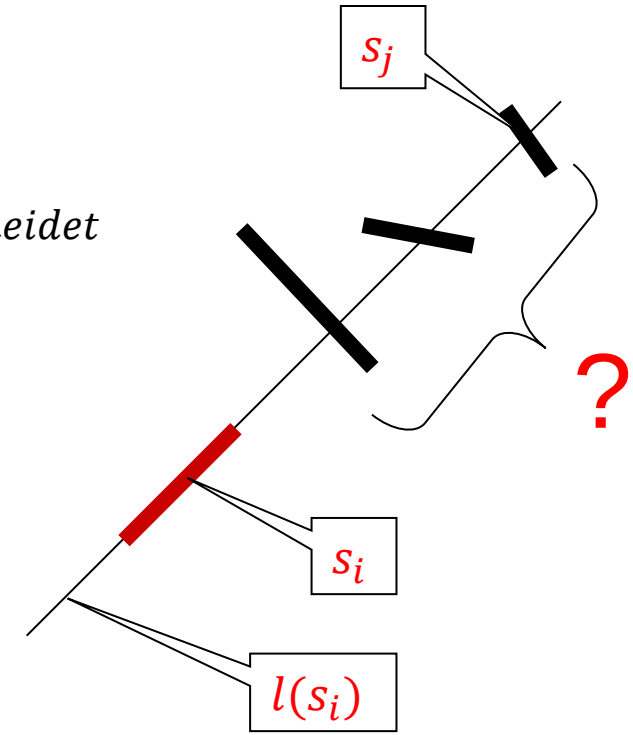


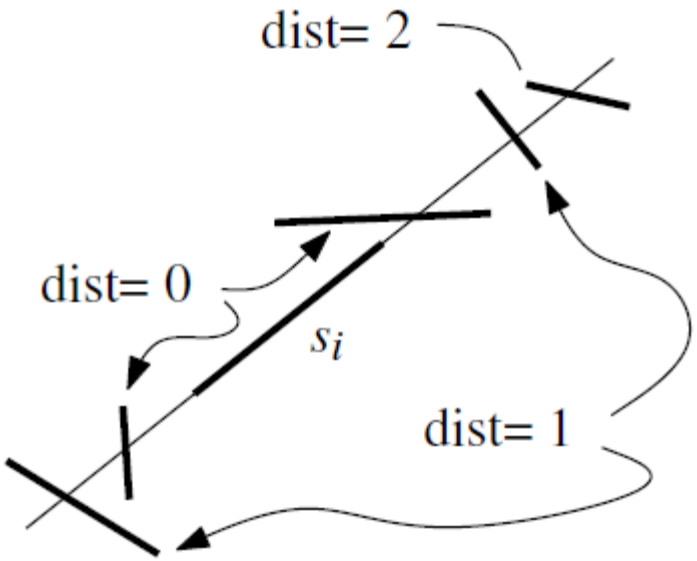
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Frage 1: Wie viele Segmente liegen „vor“ s_i ?

Wir definieren: Distanz eines Segmentes s_j

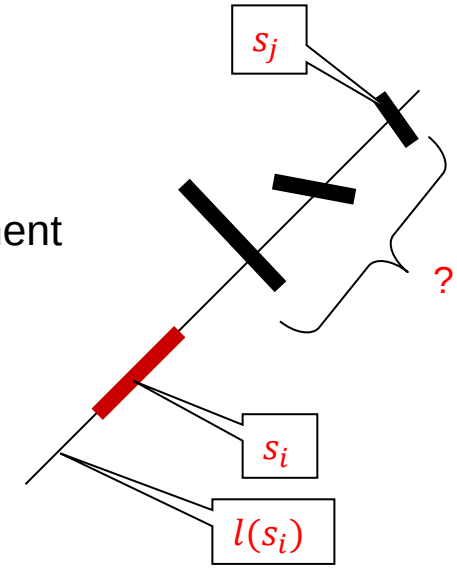
$dist(s_j) := \{Anzahl\ Segmente, die\ l(s_i)\ zwischen\ s_i\ und\ s_j\ schneidet$





HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Frage 2: Wann wird ein Segment s_j , dass „vor“s_i liegt, geschnitten?



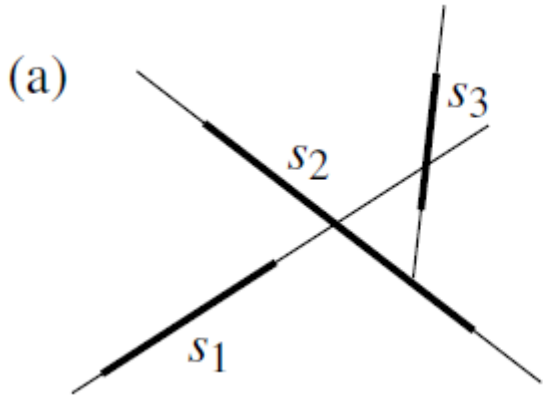
Beobachtung

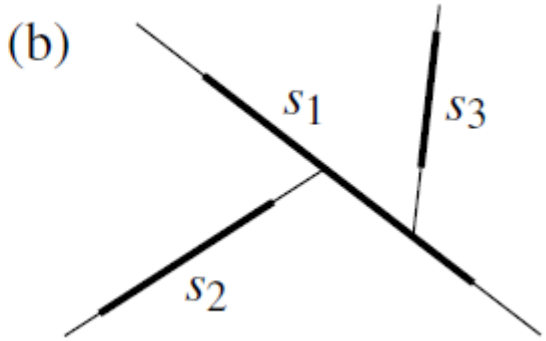
Ein Segment „schützt“s_j, wenn es vor dem Segment s_i als Splittingsegment ausgewählt wurde

Beispiel

(a) s_2 schützt s_3 nicht, da beide nach der Wahl von s_1 geteilt wurden

(b) s_1 schützt s_3 wenn s_2 als Splittingsegment gewählt wird

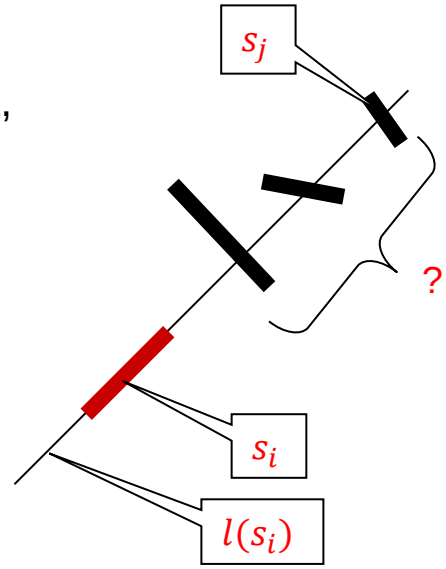




HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wahrscheinlichkeit für einen Schnitt:

Wie hoch ist die Wahrscheinlichkeit, dass $l(s_i)$ das Segment s_j schneidet, wenn s_i als Splittingelement gewählt wird?

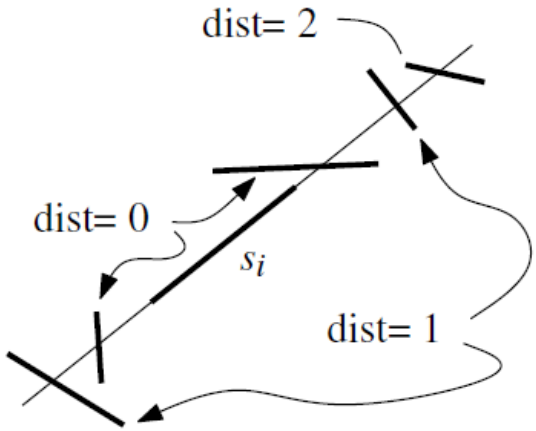


Antwort:

- Sei $k := dist(s_j)$ und seien $s_{j_1}, s_{j_2}, \dots, s_{j_k}$ die Segmente zwischen s_i und s_j
- Bei der Auswahl der Segmente $s_i, s_j, s_{j_1}, s_{j_2}, \dots, s_{j_k}$ muss s_i das erste Segment sein, sonst würde es durch Segmente dazwischen geschützt!

Anders ausgedrückt:

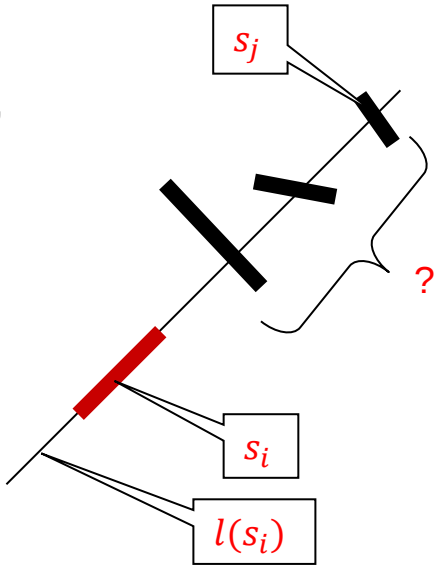
Von der Menge $i, j, j_1, j_2, \dots, j_k$ muss i die kleinste Zahl sein



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wahrscheinlichkeit für einen Schnitt:

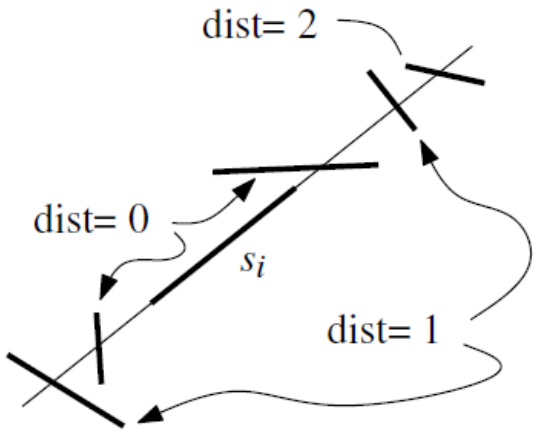
Wie hoch ist die Wahrscheinlichkeit, dass $l(s_i)$ das Segment s_j schneidet, wenn s_i als Splittingelement gewählt wird?



Aus der vorhergehenden Folie folgt:

$$P(l(s_i) \text{ schneidet } s_j) \leq \frac{1}{dist(s_j)+2}$$

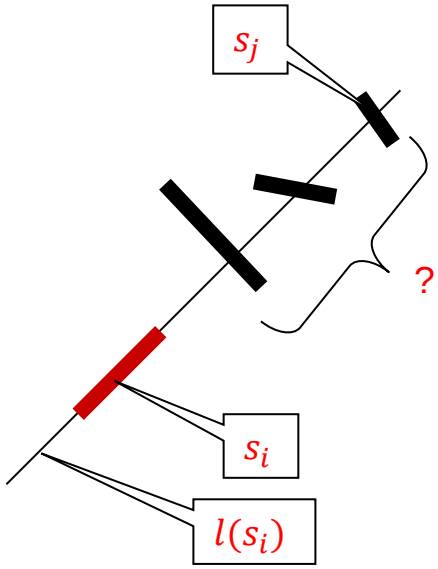
(nicht jedes Segment schneidet die Gerade)



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Erwartete Anzahl Schnitte, erzeugt durch ein Segment:

Aus der Wahrscheinlichkeit, dass ein Segment s_j geschnitten wird, können wir die erwartete Anzahl aller Schnitte, die ein Segment s_i bzw. s erzeugt, berechnen:



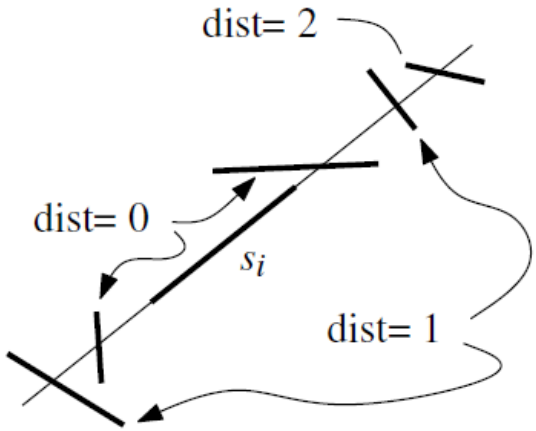
- Dazu summieren wir die evtl. auftretenden Schnitte aller Segmente s_j, die vor s_i liegen auf:

- Die zuvor berechnete Wahrscheinlichkeit multipliziert mit der Ausprägung, hier immer 1, da immer nur ein Schnitt erzeugt wird.

$$E(\text{Anzahl Schnitte erzeugt durch } s) \leq \sum_{s \neq s'} \frac{1}{dist_s(s') + 2} * 1$$

- Die Distanzen treten - evtl. zu beiden Seiten des Segments s_i bzw. s - immer fortlaufend auf, daher können wir schreiben:

$$\leq 2 \sum_{i=0}^{n-2} \frac{1}{i+2}$$
$$\leq 2\ln(n)$$

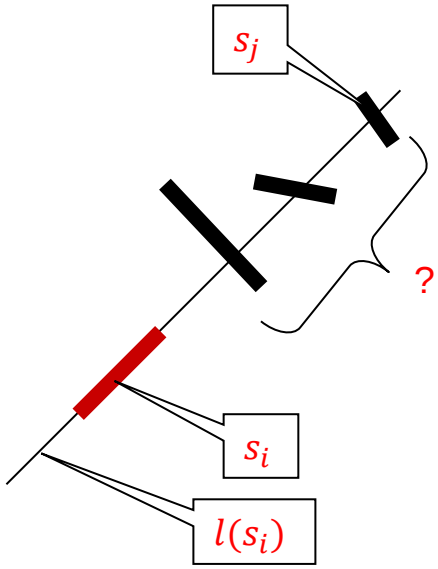


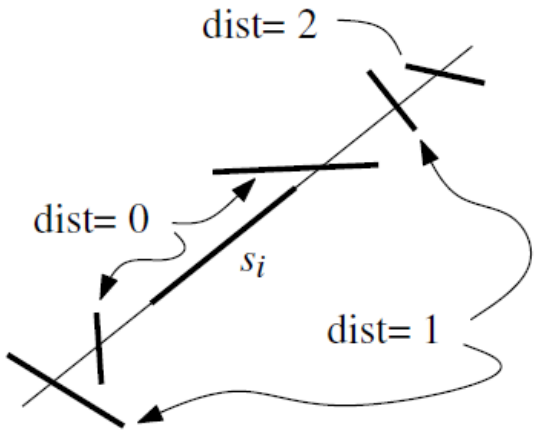
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Erwartete Anzahl Schnitte, erzeugt durch alle Segmente:

Die Anzahl der Schnitte ist dann bei n Segmenten: $2n\ln(n)$

Die Anzahl Fragmente ist dann inkl. der n geg. Segmente: $n + 2nln(n)$






HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Laufzeit des Algorithmus

Hängt von der Größe des BSP-Baums ab:

- Ein 2D-BSP-Baum der Größe $O(n \log(n))$ kann in erwarteter Zeit $O(n^2 \log(n))$ berechnet werden

Warum?

Was passiert bei einem rekursiven Aufruf unseres Algorithmus **2dBSP(S)**?

- Eine Aufteilung in S^+ und S^- kostet $O(n)$ Zeit, da nie mehr als n Fragmente bei einer Aufteilung entstehen
- Die erwartete Anzahl der rekursiven Aufrufe ist gleich der erwarteten Anzahl erzeugter Fragmente, d.h. $O(n \log(n))$

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

3D-BSP-Baum

BSP-Baum
Übersicht

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

3D-BSP-Baum = Binary Space Partition

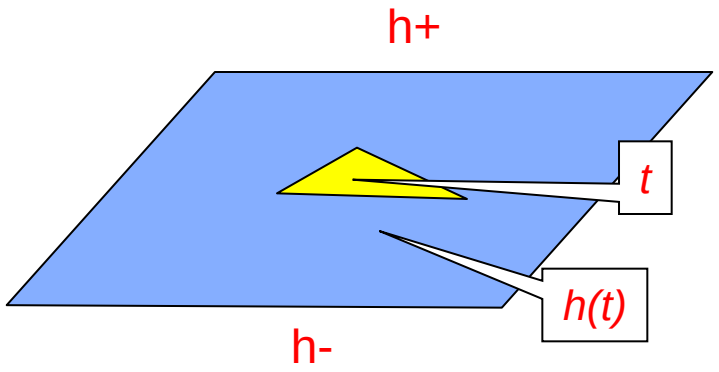
- 3D BSP-Baum
- Randomisierter Algorithmus für 3D
- …und noch ein randomisierter Algorithmus für 3D
- Größe von BSP-Bäumen

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Was ist ein BSP-Baum im 3D?

BSP-Baum im 3D:

- Die Splittinggerade im 2D ist im 3D die Splittingebene.
- Statt der Geradensegmente im 2D, gibt es als Eingabe im 3D die Dreiecke (oder allgemeiner Polygone), also 2D Oberflächen.
- Die Splittingebene kann im 3D automatisch bestimmt werden, d.h. wir legen die Splittingebene in die Ebene der Dreiecke (Polygone).
- Sei t ein Dreieck, h(t) := ist die Ebene, in der das Dreieck t liegt.

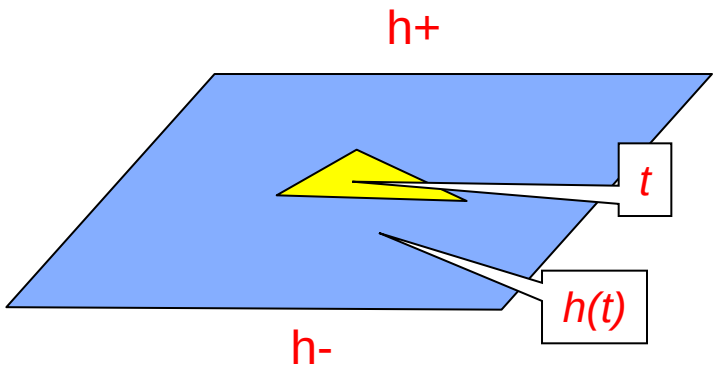


- Die Splittingebene kann aber auch wie im 2D nicht automatisch an beliebigen Stellen den Raum aufteilen.
- BSP-Bäume für noch allgemeinere Eingaben erhalten statt Dreiecken auch ganze Objekte (Würfel, Quader, Baum Haus, ….)

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wie berechnen wir einen BSP-Baum für Dreiecke (Polygone)?

Sei t ein Dreieck, h(t) := ist die Ebene, in der das Dreieck t liegt



Idee für den Aufbau des Baums

- Wir nehmen nacheinander die Dreiecke (oder deren Fragmente) zum Aufteilen des 3D-Raumes.
- Jedes Dreieck (oder dessen Fragmente) definiert eine Splittingebene des BSP-Baums (automatische Aufteilung).
- Die Splittingebene teilt den 3D-Raum in zwei Halbräume h- und h+.

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Eingabe: Eine Menge von Dreiecken $S = \{t_1, \dots, t_n\}$
Ausgabe: Ein BSP-Baum für S

Algorithmus 3dBSP (S)

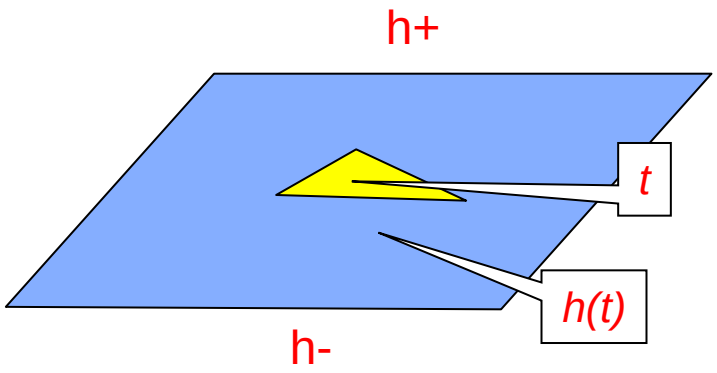
If $|S| \leq 1$
then erzeuge einen Baum T, der aus einem Blatt besteht, das S enthält
return T
else „Wir benutzen h(t1) als Ebene zum Aufteilen“

$S^+ := \{t \cap h(t_1)^+ | \forall t \in S\}$; $T^+ := 3dBSP(S^+)$;
$S^- := \{t \cap h(t_1)^- | \forall t \in S\}$; $T^- := 3dBSP(S^-)$;

Erzeuge einen BSP-Baum T mit Knoten v:

- linken Teilbaum T^- ,
- rechten Teilbaum T^+
- speichere im Knoten v: $S(v) = \{t \in S | \, t \subset h(t_1)\}$

return T



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Definition
Größe eines 3D-BSP-Baums := Anzahl aller Dreiecksfragmente

- Größe beeinflusst Laufzeit und Speicherplatz
  (gleiches Problem, wie bei der 2D-Variante)
- Wie groß kann ein 3D-BSP-Baum werden?

Ansatz Randomisierung: wähle die Segmente zufällig wie im 2D-Fall

Algorithmus 3dRandomBSP (S)

- Erzeuge von der Menge S eine zufällige Permutation $S' = \{t_1, \ldots, t_n\}$
- T = 3dBSP(S')
- **return** T

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Analyse für 3dRandomBSP:

- In der Praxis gutes Verhalten,
- aber bis jetzt keine Analyse möglich!

Daher:

- Wir modifizieren den Algorithmus
- und führen dazu eine neue Operation ein: Free-Split

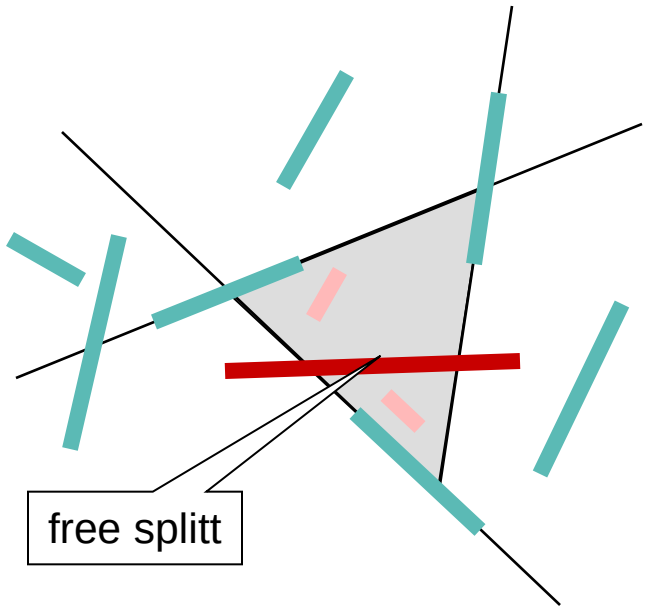
Warum diese Operation?

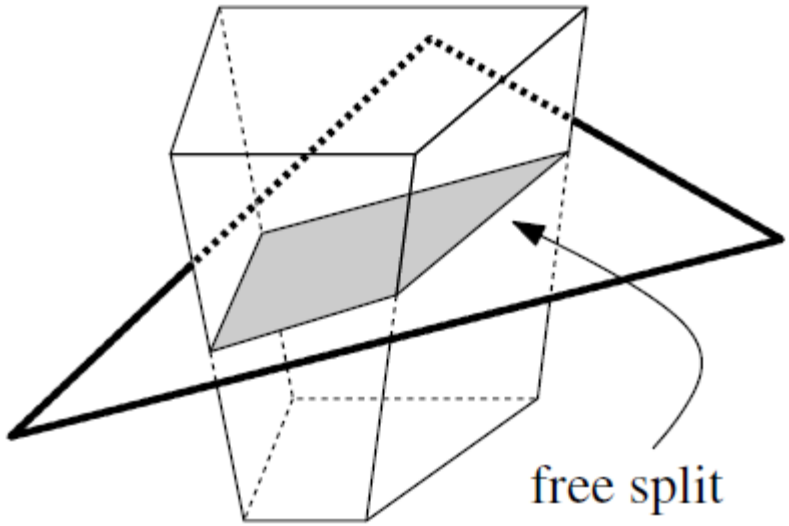
- Sie ermöglicht uns eine einfachere Analyse des veränderten Algorithmus!

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Definition: Free-Split

- Ein Dreieck (Liniensegment) oder dessen Fragment teilt eine Zelle (Region) in zwei nicht zusammenhängende Teile auf, ohne dabei weitere Fragmente der Region weiter zu zerteilen.
- Ein solches Fragment ist einfach zu bestimmen und ermöglicht eine einfache naheliegende Teilung einer Region.





HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Eingabe: Eine Menge von Dreiecken $S = \{t_1, \ldots, t_n\}$
Ausgabe: Ein BSP-Baum für *S*

Algorithmus 3dRandomFreeSplitBSP (S)

Erzeuge von S eine zufällige Permutation $S' = \{t'_1, \ldots, t'_n\}$

for i := 1 **to** n

do

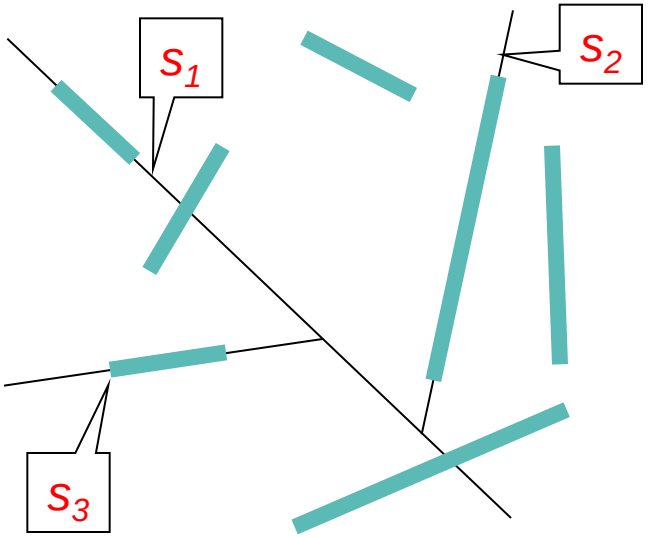
- Verwende $h(t'_i)$ zur Teilung aller Zellen der Patitionierung, die $h(t'_i)$ schneiden (nur „sinnvolle“ Teilungen, s. nachfolgende Skizze)
- Verwende alle möglichen Free-Splits

Dieser Algorithmus arbeitet nach ähnlichem Schema wie **3dRandomBSP(S)**, lässt sich aber nicht mehr rekursiv formulieren.

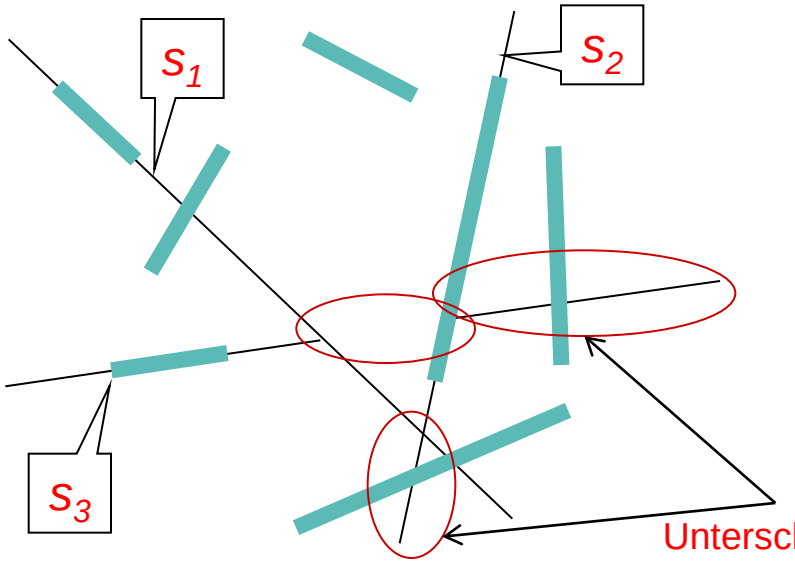
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Unterschiedliche Arbeitsweise:

rekursiver Algorithmus



iterativer Algorithmus



Unterschied zum rekursiven Algorithmus

Für die Größe der berechneten BSP-Bäume gilt:

Die erwartete Anzahl von Fragmenten ist für den randomisierten Algorithmus 3dRandomFreeSplitBSP(S) mit Free-Split $O(n^2)$

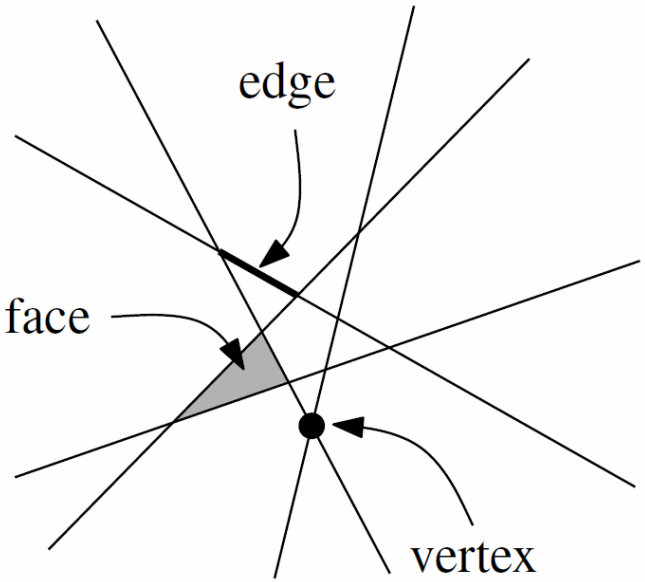
Warum?

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Problem: Arrangement von Linien

Eingabe: n Linien in der Ebene

- Die Linien induzieren eine Einteilung der Ebene in Knoten, Kanten und Flächen (Face).
- Diese Einteilung nennt man ein Arrangement.

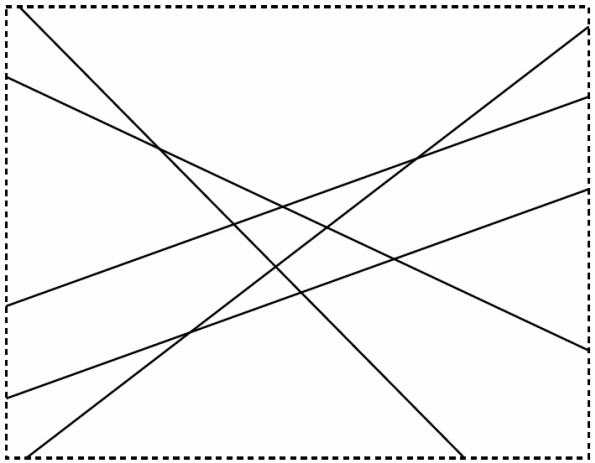


Definition: Komplexität eines Arrangements

- Anzahl aller Knoten, Kanten und Flächen.

Man kann sich überlegen (s. de Berg …)

- Es gibt jeweils $O(n^2)$ Kanten, Knoten und Flächen.
- Insgesamt hat ein Arrangement also eine Komplexität $O(n^2)$.



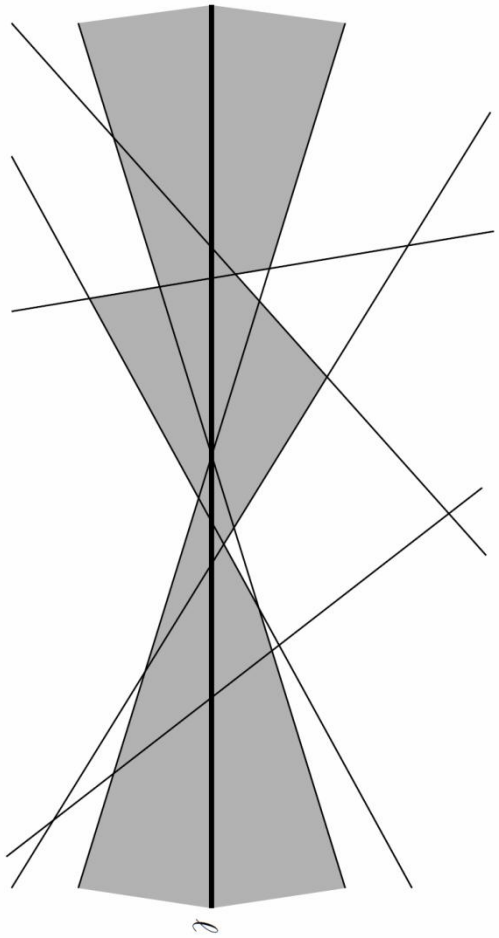
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Zone einer Linie in einem Arrangement

- Wir legen eine Gerade l durch eine Arrangement von n Geraden.
- Die Gerade schneidet eine gewisse Anzahl Flächen des Arrangements.
- Die Menge aller geschnittenen Flächen nennen wir die Zone unserer Geraden l.
- Die Komplexität der Zone ist die Anzahl der geschnittenen Flächen und der zu den Flächen gehörigen Knoten und Kanten.

Man kann sich überlegen (s. de Berg …)

- Die Komplexität einer Zone eines Arrangements mit n Geraden ist $O(n)$.



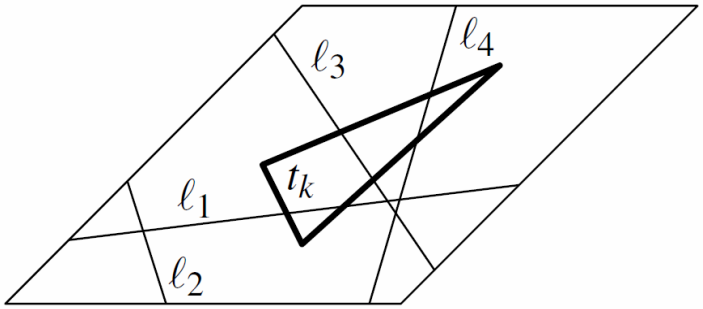
HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Wir berechnen zuerst die erwartete Anzahl Fragmente,
in die ein festes aber zufällig gewähltes Dreieck t_k durch Ebenen anderer Dreiecke geschnitten wird!

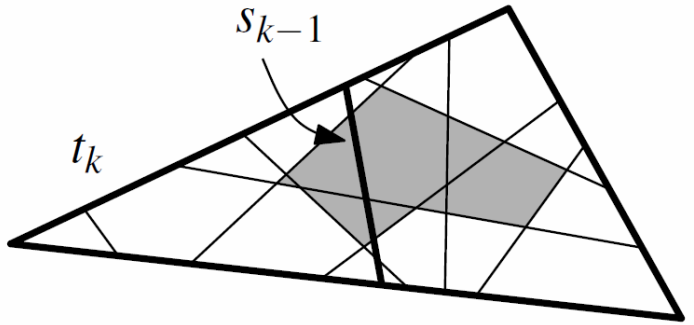
Bezeichnungen

- Für Dreiecke t_i, die vor t_k gewählt wurden $(i < k)$, ist
  $l_i := h(t_i) \cap h(t_k)$die Schnittgerade von der Ebene durch t_k mit der Ebene durch t_i.

- $L := \{l_1, \dots, l_{k-1}\}$ ist die Menge aller solcher Schnittgeraden in der Ebene durch t_k

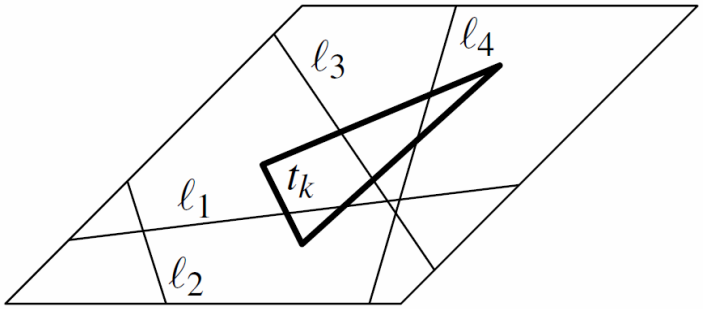


HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide



Bezeichnungen

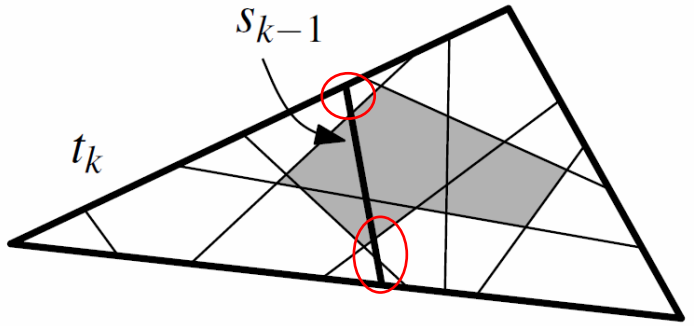
- Einige Geraden aus L schneiden t_k, andere nicht.
- Sei $s_i := l_i \cap t_k$ das Segment der Geraden l_i, welches t_k schneidet und I die Menge aller solcher Geradenstücke s_i.



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Frage

Wie groß ist die Anzahl Fragmente in die t_k geschnitten wird ?



Erste Überlegung:

- … ist gleich der Anzahl Faces des Arrangements, das I in t_k induziert!
- (wir hätten dann bei k Ebenen, die t_k schneiden ein von I induziertes Arrangement der Größe $O(k^2)$. Ungünstig, zu groß abgeschätzt!)

Da wir Free-Splits ausnutzen, gilt ein anderes günstigeres Ergebnis.
Dafür zeigen wir zuerst:

- Die Anzahl der Splits, die $\mathrm{h}(t_{k-1})$ auf t_k erzeugt, ist gleich der Anzahl Kanten, die s_{k-1} zu den „äußeren“ Flächen des Arrangements, das I auf t_k induziert, beiträgt.

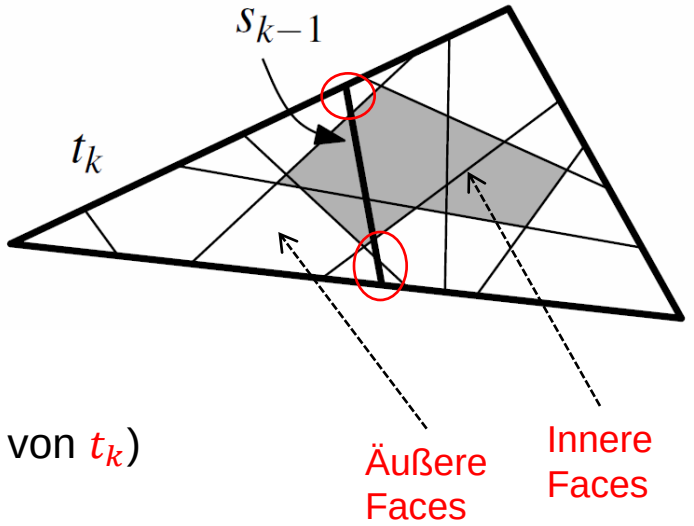
Warum?

Was sind äußere Flächen?

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Betrachte den Augenblick, in dem t_{k-1} gewählt wird.

- Wir nehmen an, dass l_{k-1} Dreieck t_k schneidet.
  (anderenfalls erzeugt t_{k-1} keine Fragmente auf t_k,
  auch gut, müssen wir also nicht zählen)
- Unterscheide:
  - Inneres Face (= Face ohne Begrenzung durch t_k)
  - Äußeres Face (= Face mit einem Teil einer Außenkante von t_k)



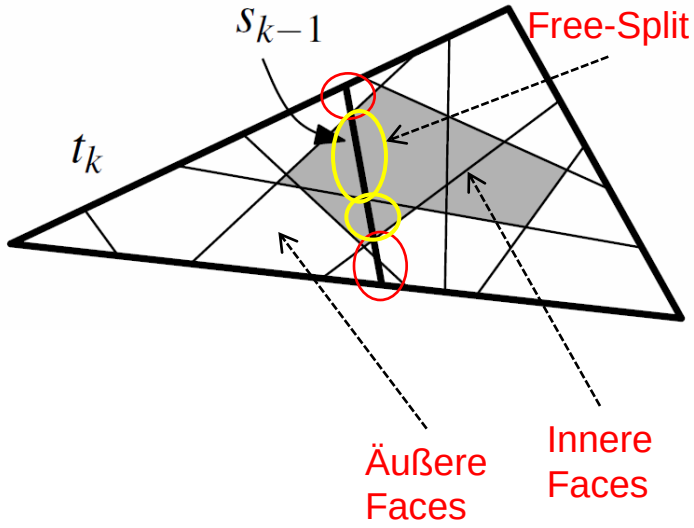
Wir überlegen uns jetzt:

- Segment s_{k-1} schneidet die Faces des Arrangements, das von $l\backslash\{s_k\}$ auf t_k induziert wird.
- Frage:
  führen von den Schnitten alle zu einem Splitt, der gezählt werden muss?
- Antwort: Nein!

Warum?

BSP-Baum
BSP-Baum im 3D Raum

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide



Weil ….

- Es sind nur äußere Faces, die $h(t_{k-1})$ schneidet und deren Schnitt wir zählen müssen.
- Innere Faces erzeugen keinen Splitt der gezählt wird, da diese Teilungen schon als Free-Split verarbeitet werden !!!
- $h(t_{k-1})$ erzeugt also nur Splits (die wir zählen müssen) auf äußeren Faces, die einen Teil der Außenkante von t_k haben.

Also ist die Anzahl der Splits, die $h(t_{k-1})$ auf t_k erzeugt, gleich der Anzahl Kanten, die s_{k-1} zu den „äußeren“ Flächen des Arrangements, das I auf t_k induziert, beiträgt.

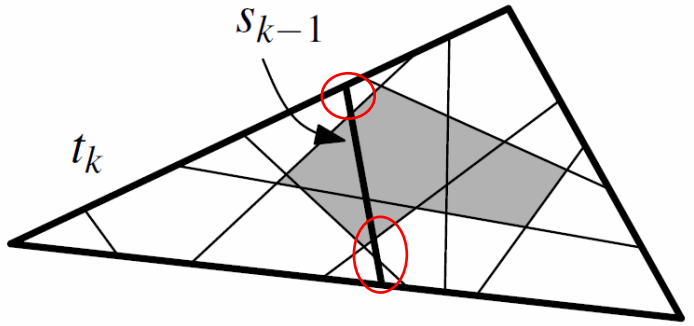
Jetzt sollte man sich überlegen wie viel das sind.

BSP-Baum
BSP-Baum im 3D Raum

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Frage

Wie viel Kanten besitzen die äußeren Faces?



Antwort

Die Anzahl der Kanten von äußeren Faces ist $O(k)$!

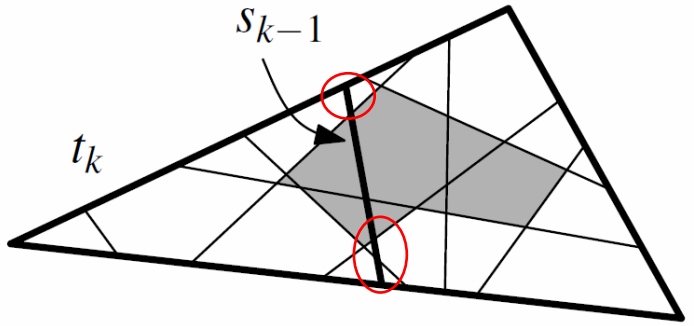
Warum?

- Für ein Arrangement von m Linien ist die Komplexität einer Zone $O(m)$.
- Seien e_1, e_2, e_3 die Kanten von t_k und $l(e_i)$ die jeweiligen Geraden durch e_i.
- Dann liegen die Kanten der äußeren Faces, die wir zählen wollen, in der Zone von $l(e_1)$, $l(e_2)$, oder $l(e_3)$ des Arrangements induziert durch L (alle l_i) auf t_k.
- Da in der Menge L, $k-1$ Geraden liegen, ist die Komplexität der Zone $O(k)$.
- Damit ist die Anzahl der Kanten von äußeren Faces $O(k)$.

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Frage

Wie hoch ist die erwartete Anzahl von Fragmenten in die t_k geteilt wird?



Antwort

Die erwartete Anzahl Fragmente für t_k ist $O(k)$.

Warum?

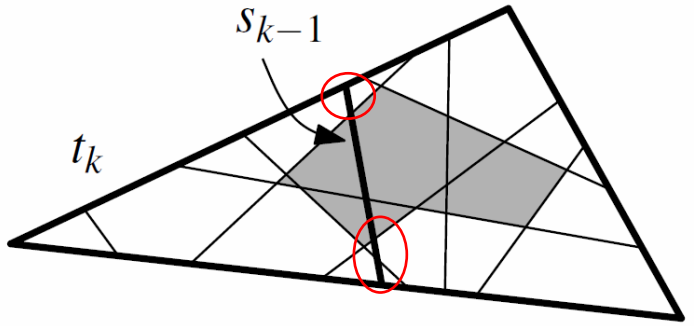
- Da die gesamte Anzahl Kanten von äußeren Faces $O(k)$ ist, liegen im Durchschnitt $O(1)$ Kanten auf einem Segment s_i.
- Da alle t_i zufällig gewählt werden, ist auch die erwartete Anzahl Kanten auf Segment s_{k-1} konstant $O(1)$.
- Damit ist die erwartete Anzahl zusätzlicher Fragmente auf t_k, die durch $h(t_{k-1})$ verursacht werden auch $O(1)$.
- Dieselbe Argumentation gilt auch für alle anderen Ebenen $h(t_1) \dots h(t_{k-2})$.
- Daraus folgt die erwartete Anzahl erzeugter Fragmente auf t_k ist $O(k)$.

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Ergebnis

Die erwartete Anzahl Fragmente erhält man indem man über alle Fragmente zusammenzählt:

$$O\left(\sum_{k=1}^{n} k\right) = O(n^2)$$



HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Aussagen zur Größe von BSP-Bäumen

BSP-Baum
Größe von 2D BSP-Bäumen

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Mit unserem randomisierten Algorithmus zur Konstruktion eines 2D BSP-Baums konstruierten wir Bäume mit erwarteter Größe $O(n \log n)$.

Wie groß können 2D-BSP-Bäume werden?

- Gibt es einen BSP-Baum der Größe $O(n)$ für beliebige Segmente im 2D-Raum?
  Nein!
- Es gibt Anordnungen von Segmenten im 2D-Raum,
  so dass jeder BSP-Baum die Größe $\Omega(n \frac{\log(n)}{\log(\log(n))})$ hat.

Weitere Algorithmen

- Ein 2D BSP-Baum für Liniensegmente der Größe $O(n \log n)$ kann in deterministischer Zeit $O(n \log n)$ berechnet werden.
- Nicht automatische Aufteilung mit Hilfe von Segmentbäumen.


HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Automatischen Aufteilung

Wählt man als Splittinggerade/-ebene immer eine Gerade/Dreieck, spricht man von einer Automatischen Aufteilung.

Frage

Können mit einer automatischen Aufteilung weniger als $O(n^2)$ Fragmente erreicht werden?

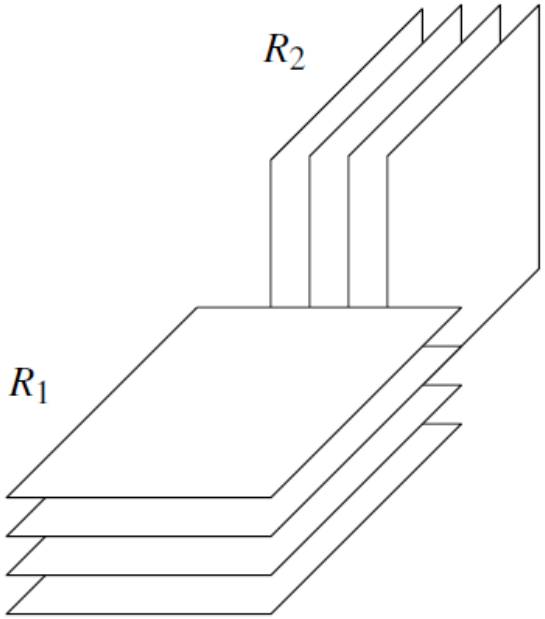
Lemma

Es gibt Mengen von n sich nicht überschneidenden Dreiecken im 3D-Raum, für die jede Automatische Aufteilung $\Omega(n^2)$ Fragmente erzeugt.

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Beweis

- Wir ordnen eine Menge R_1 von n_1 Rechtecken parallel zur xy-Ebene und eine Menge R_2 von n_2 Rechtecken parallel zur yz-Ebene an.
- (funktioniert für Dreiecke genauso)



- $G(n_1, n_2)$:= sei die minimale Größe einer automatischen Aufteilung
- Behauptung: $G(n_1, n_2) = (n_1 + 1)(n_2 + 1) - 1$

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

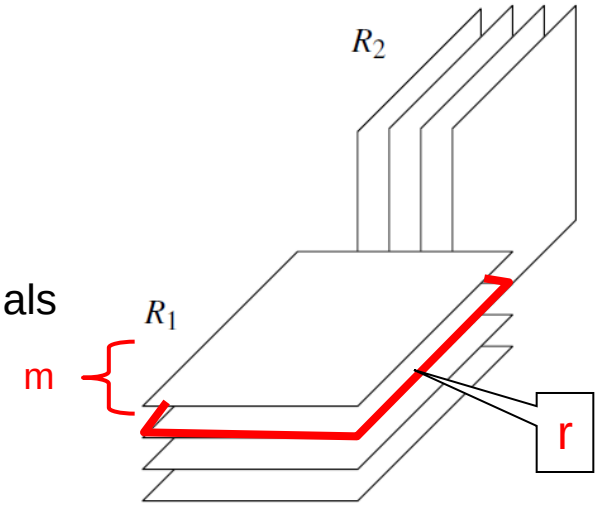
Beweis: Induktion über $n_1 + n_2$

I.A.: $G(1,0) = 1, G(0,1) = 1$

I.S.: $n_1 + n_2 > 1$

O.B.d.A. wählt die automatische Aufteilung ein Rechteck r aus R_1 als Splittingebene:

- das Rechteck r teilt alle Rechtecke in R_2
- die Teilung erzeugt zwei Teilszenen, die so aussehen wie die Ausgangssituation



$m :=$ Anzahl der Dreiecke von R_1, die über r liegen

Dann gilt:

oberhalb von *r* — unterhalb von *r*

$$G(n_1, n_2) = 1 + G(m, n_2) + G(n_1 - m - 1, n_2)$$
$$= 1 + \big((m+1)(n_2+1) - 1\big) + ((n_1 - m)(n_2 + 1) - 1)$$
$$= (n_1 + 1)(n_2 + 1) - 1$$

BSP-Baum
Größe von 3D BSP-Bäumen

HEINZ NIXDORF INSTITUT
Universität Paderborn
Algorithmen und Komplexität
Prof. Dr. Friedhelm Meyer auf der Heide

Weiter kann man für den allgemeinen Fall (nicht automatische Aufteilung) im **3D** zeigen:

- Für jede Menge von n sich nicht überschneidenden Dreiecken existiert ein BSP-Baum der Größe $O(n^2)$.
  (man kann also immer schlechte Bäume bauen)

- Für bestimmte Anordnungen ist die Größe jedes BSP-Baums $\Omega(n^2)$
  (es gibt also ungünstige Szenen/Anordnungen)