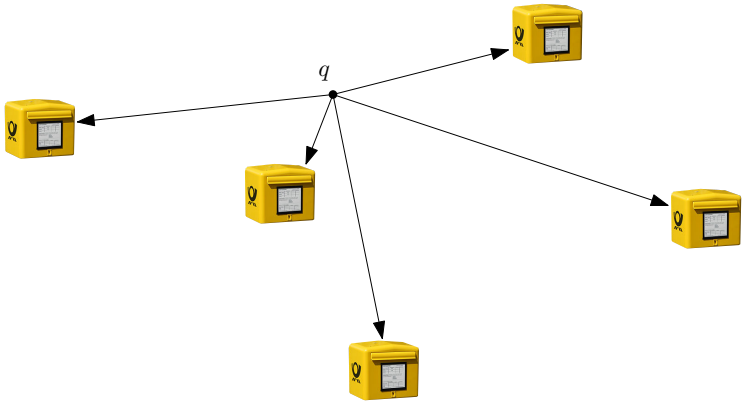
6.2 Nächste-Nachbarn-Suche

Problem des nächsten Postamtes: Welches Postamt ist am nächsten zum Standort q?



Definition (Euklidischer Abstand)

Für $a = (a_1, a_2) \in \mathbb{R}^2$ und $b = (b_1, b_2) \in \mathbb{R}^2$ ist der Euklidische Abstand gegeben als $\|a - b\| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$.

6.2 Nächste-Nachbarn-Suche

Definition (Euklidischer Abstand)

Für $a = (a_1, a_2) \in \mathbb{R}^2$ und $b = (b_1, b_2) \in \mathbb{R}^2$ ist der Euklidische Abstand gegeben als $\|a - b\| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$.

Definition (Nächster Nachbar)

Gegeben eine Menge $S \subseteq \mathbb{R}^2$ und ein Punkt $q \in \mathbb{R}^2$, der Punkt $p \in S$ ist ein nächster Nachbar von q in S wenn $\|q - p\|$ minimal über alle $p \in S$ ist.

Definition (Euklidischer Abstand)

Für $a = (a_1, a_2) \in \mathbb{R}^2$ und $b = (b_1, b_2) \in \mathbb{R}^2$ ist der Euklidische Abstand gegeben als $\|a - b\| = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$.

Definition (Nächster Nachbar)

Gegeben eine Menge $S \subseteq \mathbb{R}^2$ und ein Punkt $q \in \mathbb{R}^2$, der Punkt $p \in S$ ist ein nächster Nachbar von q in S wenn $\|q - p\|$ minimal über alle $p \in S$ ist.

Gesucht ist eine **Datenstruktur** für eine beliebige Punktmenge S, die folgende Operation unterstützt: Gegeben ein Punkt q, finde einen nächsten Nachbarn von q in S.

6.2.1 Voronoi-Diagramm

Sei $S \subseteq \mathbb{R}^2$ eine Menge von n Punkten in der Ebene und sei $p \in S$.

6.2.1 Voronoi-Diagramm

Sei $S \subseteq \mathbb{R}^2$ eine Menge von n Punkten in der Ebene und sei $p \in S$. Für welche Anfragen gibt die Datenstruktur als Antwort p zurück?

6.2.1 Voronoi-Diagramm

Sei $S \subseteq \mathbb{R}^2$ eine Menge von n Punkten in der Ebene und sei $p \in S$. Für welche Anfragen gibt die Datenstruktur als Antwort p zurück?

Betrachte die Menge der Punkte, für die p der eindeutige nächste Nachbar in S ist:

$$\{x \in \mathbb{R}^2 \mid \forall q \neq p \in S : \|x - p\| < \|x - q\|\}$$

6.2.1 Voronoi-Diagramm

Sei $S \subseteq \mathbb{R}^2$ eine Menge von n Punkten in der Ebene und sei $p \in S$. Für welche Anfragen gibt die Datenstruktur als Antwort p zurück?

Betrachte die Menge der Punkte, für die p der eindeutige nächste Nachbar in S ist:

$$\{x \in \mathbb{R}^2 \mid \forall q \neq p \in S : \|x - p\| < \|x - q\|\}$$

Das **Voronoi-Diagramm** ist die Unterteilung der Ebene $\mathbb{R}^2$ in diese Teilmengen der Ebene für alle Punkte in S.

6.2.1 Voronoi-Diagramm

Sei $S \subseteq \mathbb{R}^2$ eine Menge von n Punkten in der Ebene und sei $p \in S$. Für welche Anfragen gibt die Datenstruktur als Antwort p zurück?

Betrachte die Menge der Punkte, für die p der eindeutige nächste Nachbar in S ist:

$$\{x \in \mathbb{R}^2 \mid \forall q \neq p \in S : \|x - p\| < \|x - q\|\}$$

Das **Voronoi-Diagramm** ist die Unterteilung der Ebene $\mathbb{R}^2$ in diese Teilmengen der Ebene für alle Punkte in S.

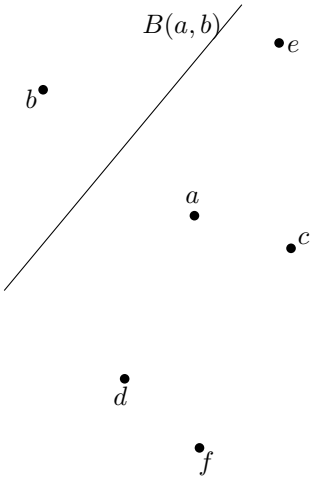
Wir werden Eigenschaften des Voronoi-Diagramms untersuchen und einen direkten Zusammenhang mit der **Delaunay-Triangulation** von S herstellen.

6.2.1 Voronoi-Diagramm

Definition (Bisektor)

Der Bisektor von zwei Punkten $p \in \mathbb{R}^2$ und $q \in \mathbb{R}^2$ ist

$$B(p, q) = \{x \in \mathbb{R}^2 \mid \|p - x\| = \|q - x\|\}$$

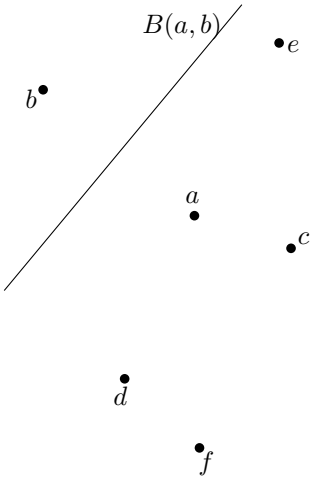


6.2.1 Voronoi-Diagramm

Definition (Bisektor)

Der Bisektor von zwei Punkten $p \in \mathbb{R}^2$ und $q \in \mathbb{R}^2$ ist

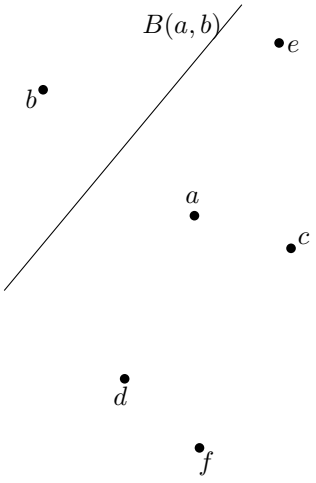
$$B(p, q) = \{x \in \mathbb{R}^2 \mid \|p - x\| = \|q - x\|\}$$



Der Bisektor ist die Menge der Punkte, für die der Abstand zu p gleich dem Abstand zu q ist.

6.2.1 Voronoi-Diagramm

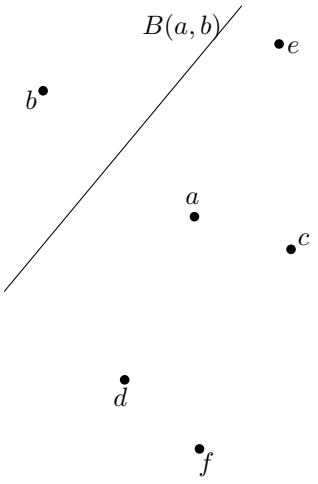
Der Bisektor $B(p, q)$ von zwei Punkten p, q ist eine Gerade.



6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ von zwei Punkten p, q ist eine Gerade.

$$\|p - x\| = \|q - x\|$$

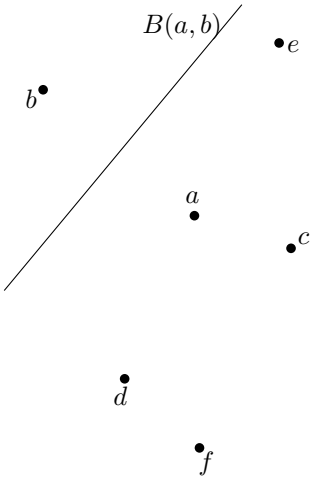


6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ von zwei Punkten p, q ist eine Gerade.

$$\|p - x\| = \|q - x\|$$

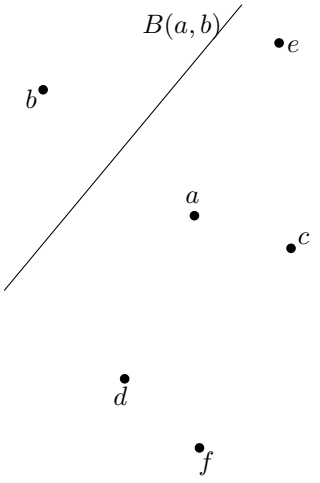
$$\Leftrightarrow \quad \|p - x\|^2 = \|q - x\|^2$$



6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ von zwei Punkten p, q ist eine Gerade.

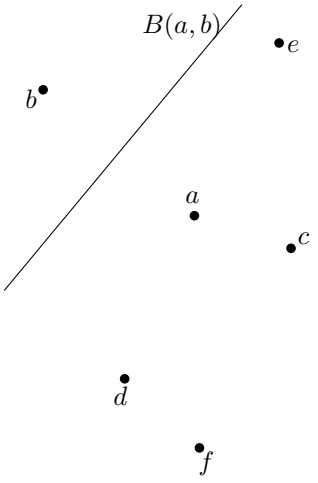
$$\begin{aligned} & \|p - x\| = \|q - x\| \\ \Leftrightarrow \quad & \|p - x\|^2 = \|q - x\|^2 \\ \Leftrightarrow \quad & \langle p - x, p - x \rangle = \langle q - x, q - x \rangle \end{aligned}$$



6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ von zwei Punkten p, q ist eine Gerade.

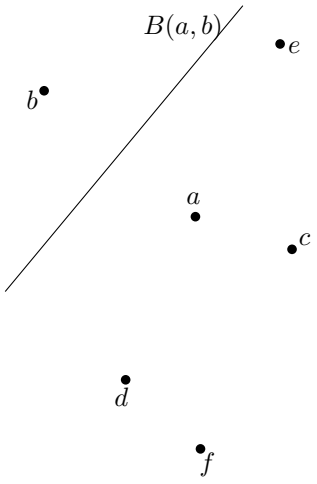
$$
\begin{aligned}
& \|p - x\| = \|q - x\| \\
\Leftrightarrow \quad & \|p - x\|^2 = \|q - x\|^2 \\
\Leftrightarrow \quad & \langle p - x, p - x \rangle = \langle q - x, q - x \rangle \\
\Leftrightarrow \quad & \langle p, p \rangle + \langle x, x \rangle - 2 \langle p, x \rangle = \langle q, q \rangle + \langle x, x \rangle - 2 \langle q, x \rangle
\end{aligned}
$$



6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ von zwei Punkten p, q ist eine Gerade.

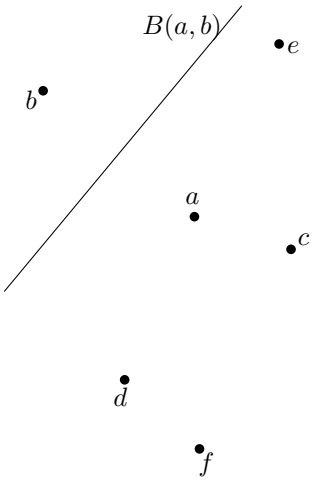
$$
\begin{aligned}
& \|p - x\| = \|q - x\| \\
\Leftrightarrow \quad & \|p - x\|^2 = \|q - x\|^2 \\
\Leftrightarrow \quad & \langle p - x, p - x \rangle = \langle q - x, q - x \rangle \\
\Leftrightarrow \quad & \langle p, p \rangle + \langle x, x \rangle - 2 \langle p, x \rangle = \langle q, q \rangle + \langle x, x \rangle - 2 \langle q, x \rangle \\
\Leftrightarrow \quad & \langle p, p \rangle - 2 \langle p, x \rangle = \langle q, q \rangle - 2 \langle q, x \rangle
\end{aligned}
$$



6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ von zwei Punkten p, q ist eine Gerade.

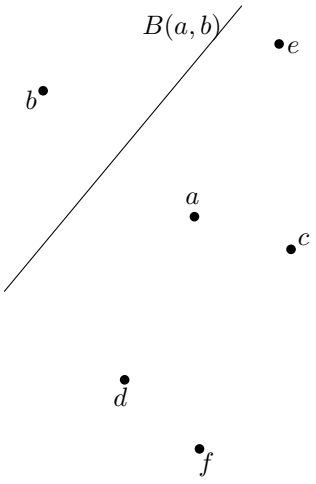
$$
\begin{aligned}
& \|p - x\| = \|q - x\| \\
\Leftrightarrow \quad & \|p - x\|^2 = \|q - x\|^2 \\
\Leftrightarrow \quad & \langle p - x, p - x \rangle = \langle q - x, q - x \rangle \\
\Leftrightarrow \quad & \langle p, p \rangle + \langle x, x \rangle - 2 \langle p, x \rangle = \langle q, q \rangle + \langle x, x \rangle - 2 \langle q, x \rangle \\
\Leftrightarrow \quad & \langle p, p \rangle - 2 \langle p, x \rangle = \langle q, q \rangle - 2 \langle q, x \rangle \\
\Leftrightarrow \quad & 2 \langle q, x \rangle - 2 \langle p, x \rangle = \langle q, q \rangle - \langle p, p \rangle
\end{aligned}
$$



6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ von zwei Punkten p, q ist eine Gerade.

$$
\begin{aligned}
& \|p - x\| = \|q - x\| \\
\Leftrightarrow \quad & \|p - x\|^2 = \|q - x\|^2 \\
\Leftrightarrow \quad & \langle p - x, p - x \rangle = \langle q - x, q - x \rangle \\
\Leftrightarrow \quad & \langle p, p \rangle + \langle x, x \rangle - 2 \langle p, x \rangle = \langle q, q \rangle + \langle x, x \rangle - 2 \langle q, x \rangle \\
\Leftrightarrow \quad & \langle p, p \rangle - 2 \langle p, x \rangle = \langle q, q \rangle - 2 \langle q, x \rangle \\
\Leftrightarrow \quad & 2 \langle q, x \rangle - 2 \langle p, x \rangle = \langle q, q \rangle - \langle p, p \rangle \\
\Leftrightarrow \quad & \langle 2(q - p), x \rangle = \langle q, q \rangle - \langle p, p \rangle
\end{aligned}
$$

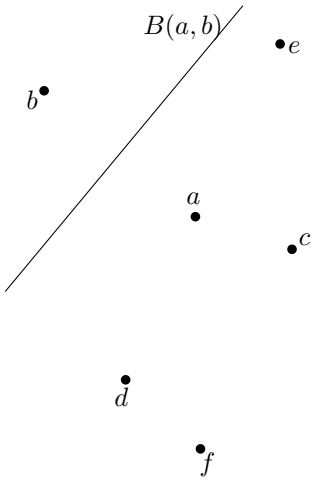


6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ von zwei Punkten p, q ist eine Gerade.

$$
\begin{aligned}
& \|p - x\| = \|q - x\| \\
\Leftrightarrow \quad & \|p - x\|^2 = \|q - x\|^2 \\
\Leftrightarrow \quad & \langle p - x, p - x \rangle = \langle q - x, q - x \rangle \\
\Leftrightarrow \quad & \langle p, p \rangle + \langle x, x \rangle - 2 \langle p, x \rangle = \langle q, q \rangle + \langle x, x \rangle - 2 \langle q, x \rangle \\
\Leftrightarrow \quad & \langle p, p \rangle - 2 \langle p, x \rangle = \langle q, q \rangle - 2 \langle q, x \rangle \\
\Leftrightarrow \quad & 2 \langle q, x \rangle - 2 \langle p, x \rangle = \langle q, q \rangle - \langle p, p \rangle \\
\Leftrightarrow \quad & \langle 2(q - p), x \rangle = \langle q, q \rangle - \langle p, p \rangle \\
\Leftrightarrow \quad & \langle w, x \rangle = u
\end{aligned}
$$

mit $w = 2(q - p) \in \mathbb{R}^2$ und $u = \langle q, q \rangle - \langle p, p \rangle \in \mathbb{R}$. $\square$



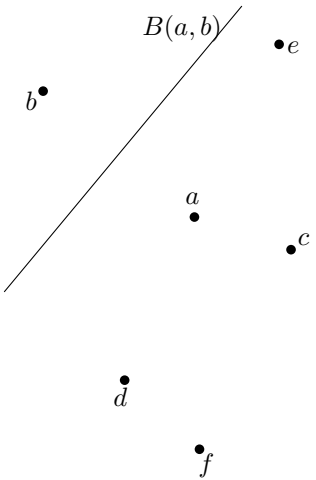
6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ unterteilt die Ebene in die Mengen

$$B_-(p,q) = \{ x \in \mathbb{R}^2 \mid \|p - x\| < \|q - x\| \}$$
$$B_+(p,q) = \{ x \in \mathbb{R}^2 \mid \|p - x\| > \|q - x\| \}$$

und den Bisektor selbst.

$B_-(p, q)$ sind Punkte die näher an p liegen als an q
$B_+(q, p)$ sind Punkte die näher an q liegen als an p



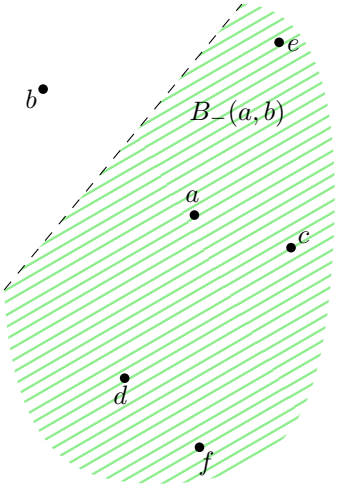
6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ unterteilt die Ebene in die Mengen

$$B_-(p,q) = \{ x \in \mathbb{R}^2 \mid \|p - x\| < \|q - x\| \}$$
$$B_+(p,q) = \{ x \in \mathbb{R}^2 \mid \|p - x\| > \|q - x\| \}$$

und den Bisektor selbst.

$B_-(p, q)$ sind Punkte die näher an p liegen als an q
$B_+(q, p)$ sind Punkte die näher an q liegen als an p



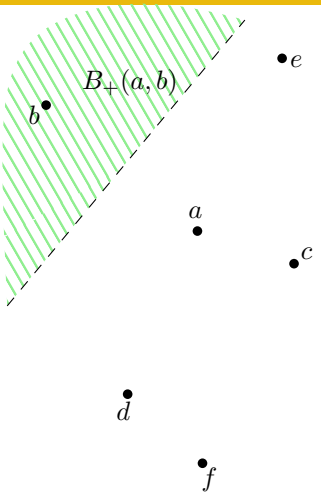
6.2.1 Voronoi-Diagramm

Der Bisektor $B(p, q)$ unterteilt die Ebene in die Mengen

$$B_-(p,q) = \{ x \in \mathbb{R}^2 \mid \|p - x\| < \|q - x\| \}$$
$$B_+(p,q) = \{ x \in \mathbb{R}^2 \mid \|p - x\| > \|q - x\| \}$$

und den Bisektor selbst.

$B_-(p, q)$ sind Punkte die näher an p liegen als an q
$B_+(q, p)$ sind Punkte die näher an q liegen als an p

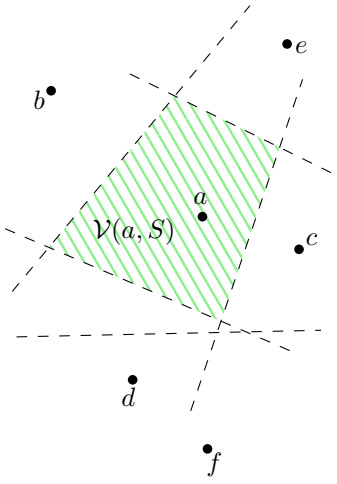


6.2.1 Voronoi-Diagramm

Definition (Voronoi-Region)

Die Voronoi-Region eines Punktes p bezüglich S ist

$$\mathcal{V}(p, S) = \bigcap_{\substack{q \in S \\ p \neq q}} B_{-}(p, q)$$



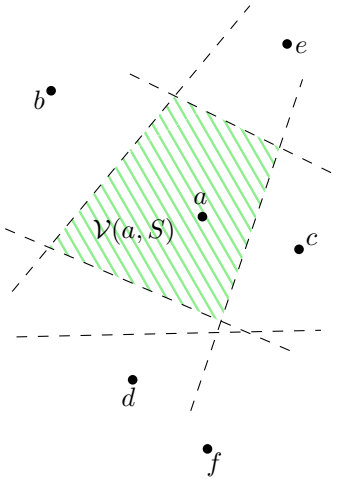
6.2.1 Voronoi-Diagramm

Definition (Voronoi-Region)

Die Voronoi-Region eines Punktes p bezüglich S ist

$$\mathcal{V}(p, S) = \bigcap_{\substack{q \in S \\ p \neq q}} B_{-}(p, q)$$

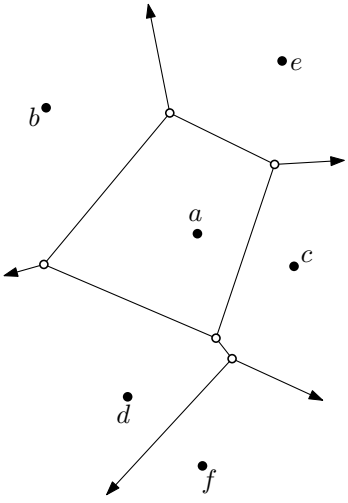
Die Voronoi-Region von p bezüglich S ist genau die Menge der Punkte, für die p der eindeutige nächste Nachbar in S ist.



6.2.1 Voronoi-Diagramm

Voronoi-Diagramm:

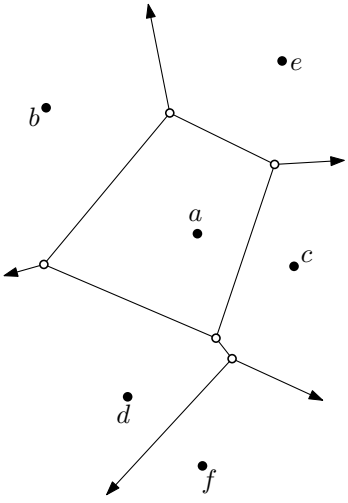
Die **Voronoi-Regionen** sind die Flächen eines kreuzungsfreien geometrischen Graphen G, den wir das **Voronoi-Diagramm** nennen.



6.2.1 Voronoi-Diagramm

Voronoi-Diagramm:
Die **Voronoi-Regionen** sind die Flächen eines kreuzungsfreien geometrischen Graphen G, den wir das **Voronoi-Diagramm** nennen.

Wir nennen die Kanten von G **Voronoi-Kanten** und die Knoten von G **Voronoi-Knoten**.

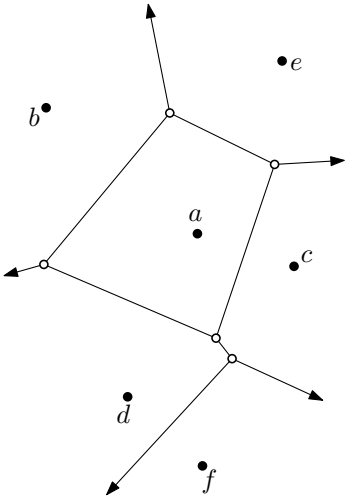


6.2.1 Voronoi-Diagramm

Voronoi-Diagramm:
Die **Voronoi-Regionen** sind die Flächen eines kreuzungsfreien geometrischen Graphen G, den wir das **Voronoi-Diagramm** nennen.

Wir nennen die Kanten von G **Voronoi-Kanten** und die Knoten von G **Voronoi-Knoten**.

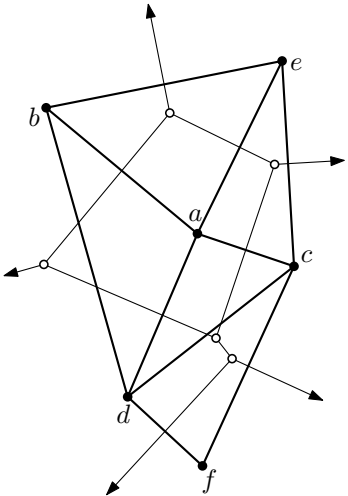
Manche Voronoi-Kanten sind unbeschränkt (Strahlen). Definiere einen **abstrakten Voronoi-Knoten im Unendlichen**, der Endpunkt dieser Kanten ist.



6.2.1 Voronoi-Diagramm

Theorem 6.25

Sei S eine endliche Menge von Punkten in allgemeiner Lage in der Ebene. Für den Graph des **Voronoi-Diagramms** von S und den Graph der **Delaunay-Triangulation** von S gilt:



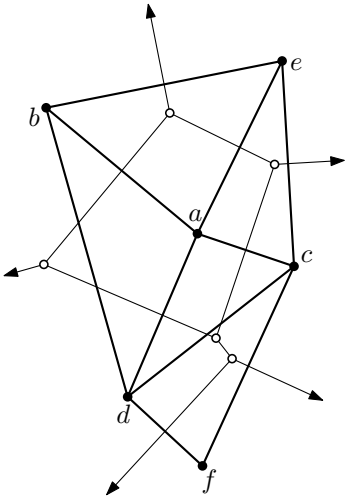
6.2.1 Voronoi-Diagramm

Theorem 6.25

Sei S eine endliche Menge von Punkten in allgemeiner Lage in der Ebene. Für den Graph des **Voronoi-Diagramms** von S und den Graph der **Delaunay-Triangulation** von S gilt:

(i) $p, q, r \in S$ bilden **Delaunay-Dreieck** genau dann wenn es einen **Voronoi-Knoten** gibt, der zu $\mathcal{V}(p, S)$, $\mathcal{V}(q, S)$, $\mathcal{V}(r, S)$ inzident ist.

(ii) $p, q \in S$ sind genau dann durch eine **Delaunay-Kante** verbunden, wenn es eine **Voronoi-Kante** gibt, die zu $\mathcal{V}(p, S)$ und $\mathcal{V}(q, S)$ inzident ist.



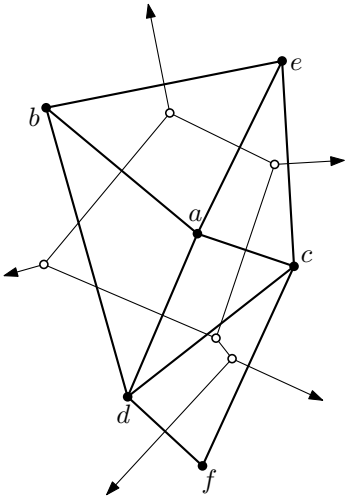
6.2.1 Voronoi-Diagramm

Theorem 6.25

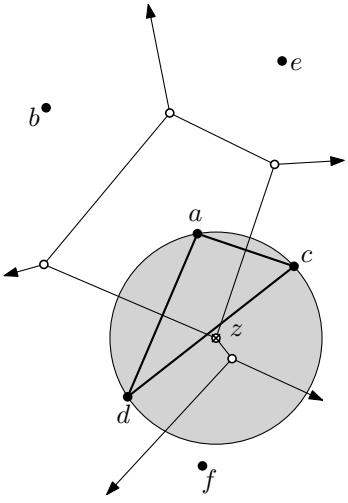
Sei S eine endliche Menge von Punkten in allgemeiner Lage in der Ebene. Für den Graph des **Voronoi-Diagramms** von S und den Graph der **Delaunay-Triangulation** von S gilt:

(i) $p, q, r \in S$ bilden **Delaunay-Dreieck** genau dann wenn es einen **Voronoi-Knoten** gibt, der zu $\mathcal{V}(p, S)$, $\mathcal{V}(q, S)$, $\mathcal{V}(r, S)$ inzident ist.

(ii) $p, q \in S$ sind genau dann durch eine **Delaunay-Kante** verbunden, wenn es eine **Voronoi-Kante** gibt, die zu $\mathcal{V}(p, S)$ und $\mathcal{V}(q, S)$ inzident ist.



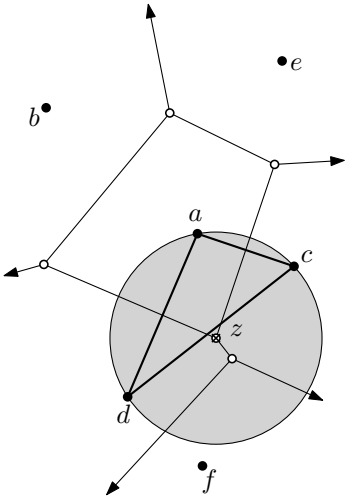
Beweis von Theorem 6.25 (i): Seien $a, d, c \in S$ paarweise verschieden. Der Umkreismittelpunkt z des Dreiecks (a, d, c) ist der gemeinsame Schnittpunkt der Bisektoren $B(a, d)$, $B(a, c)$, und $B(d, c)$.



Beweis von Theorem 6.25 (i): Seien $a, d, c \in S$ paarweise verschieden. Der Umkreismittelpunkt z des Dreiecks (a, d, c) ist der gemeinsame Schnittpunkt der Bisektoren $B(a, d)$, $B(a, c)$, und $B(d, c)$.
Die folgenden beiden Aussagen sind äquivalent:

(1) Die nächsten Nachbarn von z in S sind a, d und c.

(2) Der Kreis mit Mittelpunkt z und a, d, c auf dem Rand hat keinen anderen Punkt aus S in seinem Inneren.

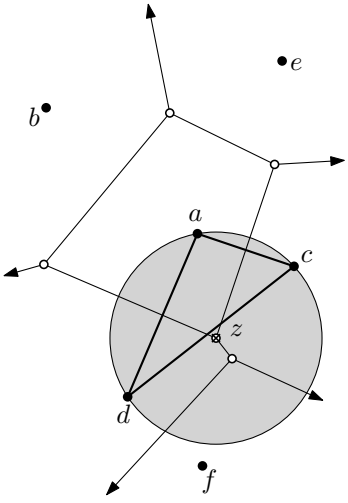


Beweis von Theorem 6.25 (i): Seien $a, d, c \in S$ paarweise verschieden. Der Umkreismittelpunkt z des Dreiecks (a, d, c) ist der gemeinsame Schnittpunkt der Bisektoren $B(a, d)$, $B(a, c)$, und $B(d, c)$.
Die folgenden beiden Aussagen sind äquivalent:

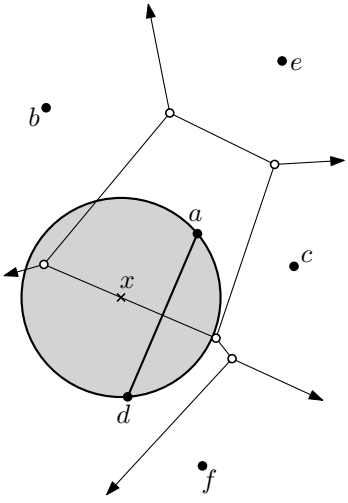
(1) Die nächsten Nachbarn von z in S sind a, d und c.

(2) Der Kreis mit Mittelpunkt z und a, d, c auf dem Rand hat keinen anderen Punkt aus S in seinem Inneren.

Genau dann, wenn beide Aussagen auf z zutreffen, bildet (a, d, c) ein Dreieck in der Delaunay-Triangulation und ebenso existiert ein Voronoi-Knoten, der zu $\mathcal{V}(a, S)$, $\mathcal{V}(d, S)$ und $\mathcal{V}(c, S)$ inzident ist. □



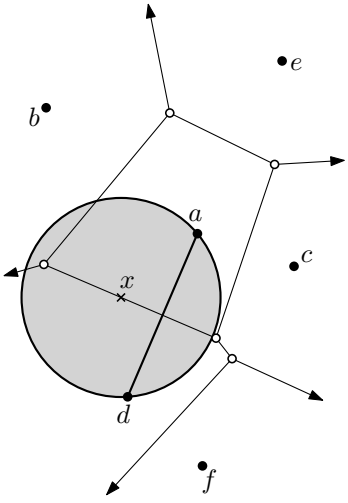
Beweis von Theorem 6.25 (ii): Seien $a, b \in S$ verschieden.
Sei x ein Punkt auf dem Bisektor $B(a, b)$.



Beweis von Theorem 6.25 (ii): Seien $a, b \in S$ verschieden.
Sei x ein Punkt auf dem Bisektor $B(a, b)$.
Die folgenden beiden Aussagen sind äquivalent:

(3) Die nächsten Nachbarn von x in S sind a und b.

(4) Der Kreis mit Mittelpunkt x und Punkten a und b auf dem Rand hat keinen anderen Punkt aus S in seinem Inneren.

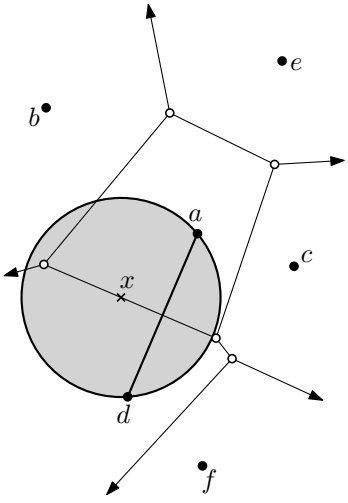


Beweis von Theorem 6.25 (ii): Seien $a, b \in S$ verschieden.
Sei x ein Punkt auf dem Bisektor $B(a, b)$.
Die folgenden beiden Aussagen sind äquivalent:

(3) Die nächsten Nachbarn von x in S sind a und b.

(4) Der Kreis mit Mittelpunkt x und Punkten a und b auf dem Rand hat keinen anderen Punkt aus S in seinem Inneren.

Genau dann, wenn ein Punkt x mit den obigen Eigenschaften existiert, dann ist (a, b) Delaunay-Kante (Theorem 6.7) und ebenso existiert die Voronoi-Kante die zu $\mathcal{V}(a, S)$ und $\mathcal{V}(b, S)$ inzident ist. □



6.2.1 Voronoi-Diagramm

Berechnung des Voronoi-Diagramms:

- Gegeben eine Menge S von n Punkten S in der Ebene in allgemeiner Lage
- Berechne die Delaunay-Triangulation T von S

Berechnung des Voronoi-Diagramms:

- Gegeben eine Menge S von n Punkten S in der Ebene in allgemeiner Lage
- Berechne die Delaunay-Triangulation T von S
- Breitensuche auf Graph von T:
  - Für Kante (p, q) von T berechne Voronoi-Kante e inzident zu $\mathcal{V}(p, S)$ und $\mathcal{V}(q, S)$

Berechnung des Voronoi-Diagramms:

- Gegeben eine Menge S von n Punkten S in der Ebene in allgemeiner Lage
- Berechne die Delaunay-Triangulation T von S
- Breitensuche auf Graph von T:
  - Für Kante (p, q) von T berechne Voronoi-Kante e inzident zu $\mathcal{V}(p, S)$ und $\mathcal{V}(q, S)$
  - Endpunkte von e sind Umkreismittelpunkte der beiden zu (p, q) inzidenten Dreiecke in T
  - Falls die Kante zu der äußeren Fläche inzident ist, ist sie zu dieser Seite unbeschränkt

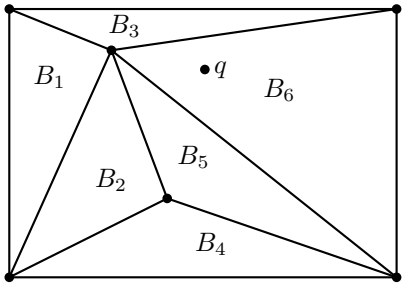
Berechnung des Voronoi-Diagramms:

- Gegeben eine Menge S von n Punkten S in der Ebene in allgemeiner Lage
- Berechne die Delaunay-Triangulation T von S
- Breitensuche auf Graph von T:
  - Für Kante (p, q) von T berechne Voronoi-Kante e inzident zu $\mathcal{V}(p, S)$ und $\mathcal{V}(q, S)$
  - Endpunkte von e sind Umkreismittelpunkte der beiden zu (p, q) inzidenten Dreiecke in T
  - Falls die Kante zu der äußeren Fläche inzident ist, ist sie zu dieser Seite unbeschränkt

Laufzeit: $O(n^2)$

6.2.2 Punktlokalisierung

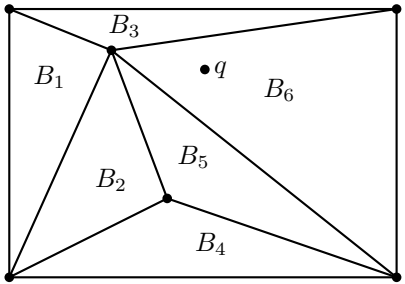
Punklokalisierung: Gegeben eine Triangulation T und ein Punkt q, welche Fläche von T enthält q? Gesucht ist eine Datenstruktur die T speichert und Anfragen mit q beantwortet.



T_0

6.2.2 Punktlokalisierung

Punklokalisierung: Gegeben eine Triangulation T und ein Punkt q, welche Fläche von T enthält q? Gesucht ist eine Datenstruktur die T speichert und Anfragen mit q beantwortet.



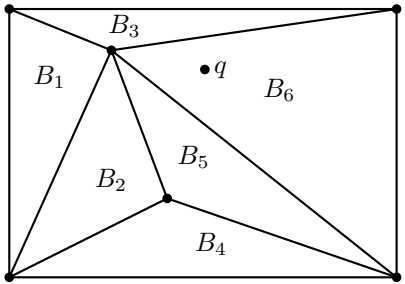
T_0

Lineare Suche:

- Speichere T in Halbkanten-Datenstruktur
- Breitensuche auf den Kanten der Triangulation
- Teste jede inzidente Fläche, ob q enthalten

6.2.2 Punktlokalisierung

Punklokalisierung: Gegeben eine Triangulation T und ein Punkt q, welche Fläche von T enthält q? Gesucht ist eine Datenstruktur die T speichert und Anfragen mit q beantwortet.



T_0

Lineare Suche:

- Speichere T in Halbkanten-Datenstruktur
- Breitensuche auf den Kanten der Triangulation
- Teste jede inzidente Fläche, ob q enthalten

Speicherplatz: $O(n)$

Anfragezeit: $O(n)$

6.2.2 Punktlokalisierung

Datenstruktur für Doppelte Binäre Suche:

- Seien $x_1 < \cdots < x_k$ die x-Koordinaten der Vertikalen an Knoten von T
- Berechne Schnittpunkte der Vertikalen mit den Kanten von T und teile Kanten

6.2.2 Punktlokalisierung

Datenstruktur für Doppelte Binäre Suche:

- Seien $x_1 < \cdots < x_k$ die x-Koordinaten der Vertikalen an Knoten von T
- Berechne Schnittpunkte der Vertikalen mit den Kanten von T und teile Kanten
- Für jeden vertikalen Streifen (x_i, x_{i+1}):
  - Kanten innerhalb des Streifens von links nach rechts orientiert

6.2.2 Punktlokalisierung

Datenstruktur für Doppelte Binäre Suche:

- Seien $x_1 < \cdots < x_k$ die x-Koordinaten der Vertikalen an Knoten von T
- Berechne Schnittpunkte der Vertikalen mit den Kanten von T und teile Kanten
- Für jeden vertikalen Streifen (x_i, x_{i+1}):
  - Kanten innerhalb des Streifens von links nach rechts orientiert
  - Sortiere Kanten von unten nach oben in einem Array

6.2.2 Punktlokalisierung

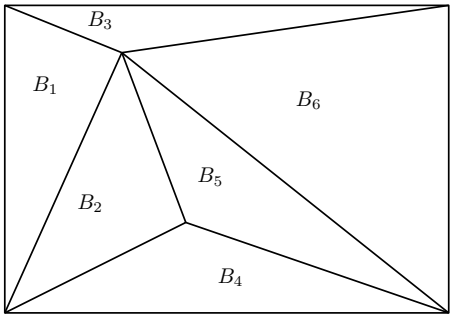
Datenstruktur für Doppelte Binäre Suche:

- Seien $x_1 < \cdots < x_k$ die x-Koordinaten der Vertikalen an Knoten von T
- Berechne Schnittpunkte der Vertikalen mit den Kanten von T und teile Kanten
- Für jeden vertikalen Streifen (x_i, x_{i+1}):
  - Kanten innerhalb des Streifens von links nach rechts orientiert
  - Sortiere Kanten von unten nach oben in einem Array
  - Speichere Zeiger auf inzidente Dreiecke für jede Kante

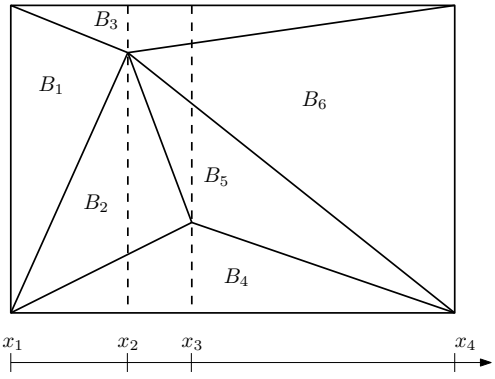
Datenstruktur für Doppelte Binäre Suche:

- Seien $x_1 < \cdots < x_k$ die x-Koordinaten der Vertikalen an Knoten von T
- Berechne Schnittpunkte der Vertikalen mit den Kanten von T und teile Kanten
- Für jeden vertikalen Streifen (x_i, x_{i+1}):
  - Kanten innerhalb des Streifens von links nach rechts orientiert
  - Sortiere Kanten von unten nach oben in einem Array
  - Speichere Zeiger auf inzidente Dreiecke für jede Kante
- Speichere $x_1, \ldots, x_k$ in einem Array a mit Zeigern auf Arrays b_i für Streifen (x_i, x_{i+1})

6.2.2 Punktlokalisierung

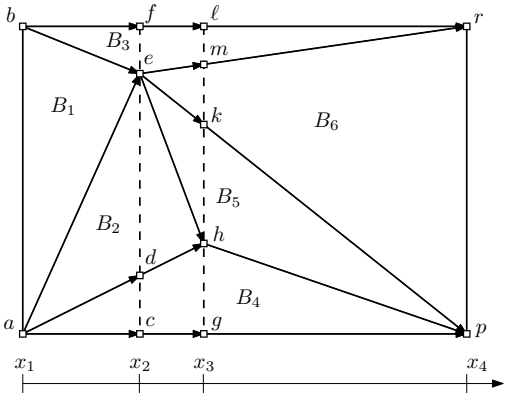


6.2.2 Punktlokalisierung



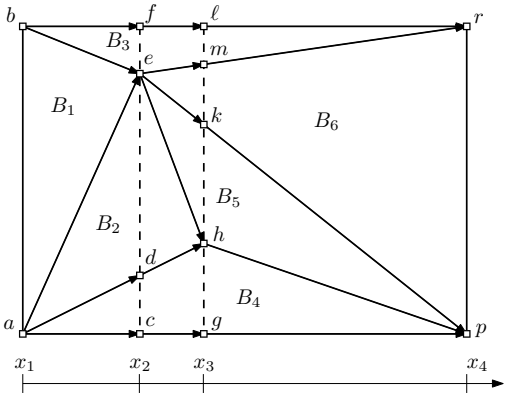
x_1	x_2	x_3	x_4

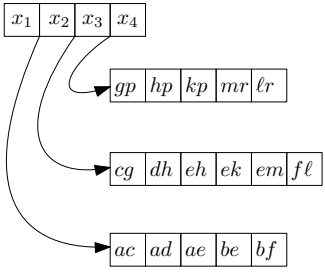
6.2.2 Punktlokalisierung



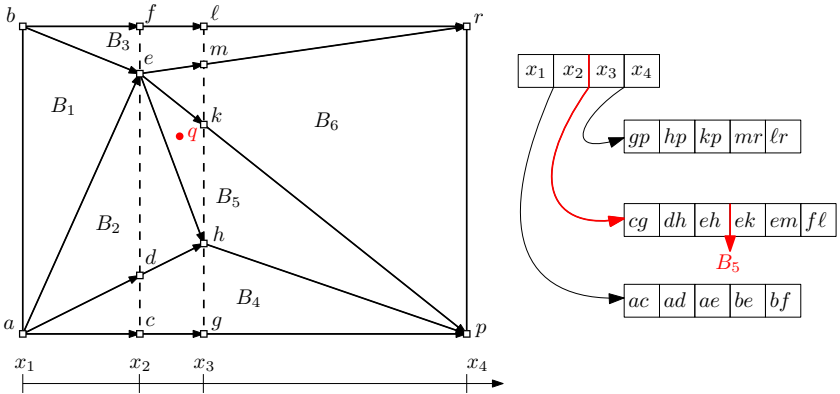
x_1	x_2	x_3	x_4

6.2.2 Punktlokalisierung





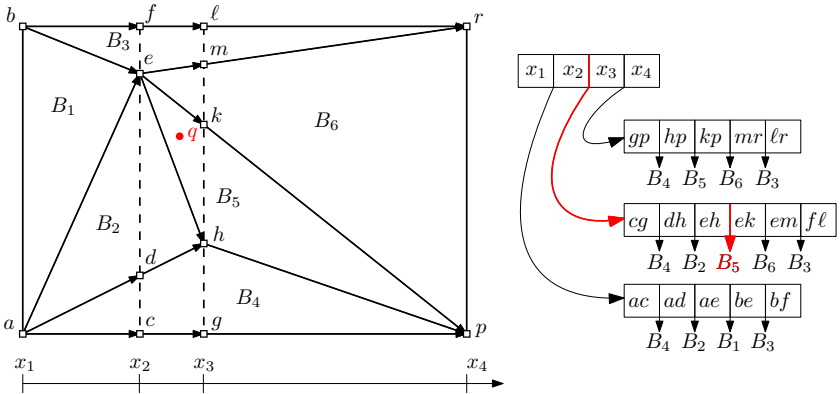
6.2.2 Punktlokalisierung



Suchanfrage mit Punkt q:

- Binäre Suche in a, um den vertikalen Streifen (x_i, x_{i+1}) zu finden, der q enthält
- Binäre Suche in b_i. In jedem Schritt, teste ob q links oder rechts von der Kante liegt.

6.2.2 Punktlokalisierung



Suchanfrage mit Punkt q:

- Binäre Suche in a, um den vertikalen Streifen (x_i, x_{i+1}) zu finden, der q enthält
- Binäre Suche in b_i. In jedem Schritt, teste ob q links oder rechts von der Kante liegt.

Zusammenfassung:

	Anfragezeit	Speicherplatz
Lineare Suche	$O(n)$	$O(n)$
Doppelte Binäre Suche	$O(\log n)$	$O(n^2)$

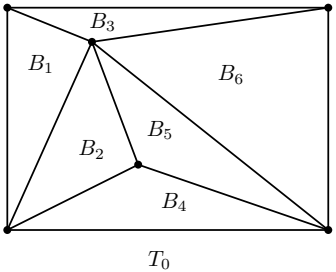
Zusammenfassung:

	Anfragezeit	Speicherplatz
Lineare Suche	$O(n)$	$O(n)$
Doppelte Binäre Suche	$O(\log n)$	$O(n^2)$

Gesucht ist eine Datenstruktur mit Speicherplatz $O(n)$ und Anfragezeit $O(\log n)$.

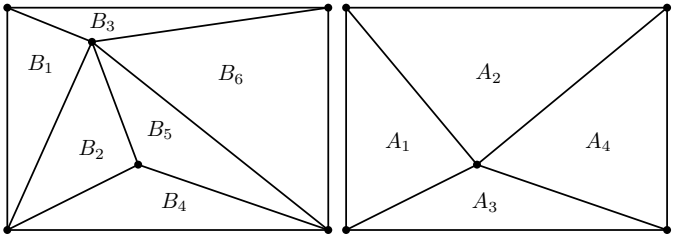
6.2.3 Triangulationshierarchie

Triangulationshierarchie: Sequenz von Triangulationen $T_0, \ldots, T_h$, in der die Anzahl der Dreiecke, ausgehend von $T_0 = T$, in jedem Schritt reduziert wird.



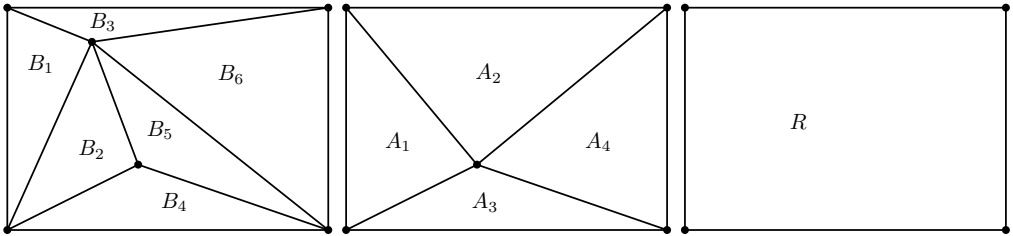
6.2.3 Triangulationshierarchie

Triangulationshierarchie: Sequenz von Triangulationen $T_0, \ldots, T_h$, in der die Anzahl der Dreiecke, ausgehend von $T_0 = T$, in jedem Schritt reduziert wird.



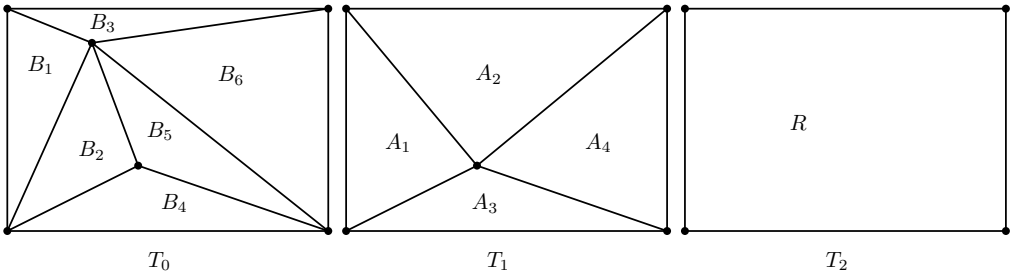
6.2.3 Triangulationshierarchie

Triangulationshierarchie: Sequenz von Triangulationen $T_0, \ldots, T_h$, in der die Anzahl der Dreiecke, ausgehend von $T_0 = T$, in jedem Schritt reduziert wird.



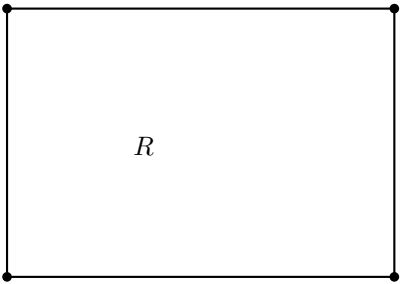
6.2.3 Triangulationshierarchie

Triangulationshierarchie: Sequenz von Triangulationen $T_0, \ldots, T_h$, in der die Anzahl der Dreiecke, ausgehend von $T_0 = T$, in jedem Schritt reduziert wird.



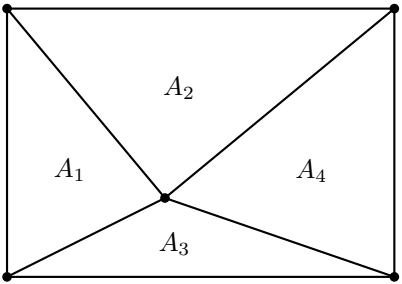
Suchstruktur: Graph G. Jeder Knoten in G ist Dreieck in einer der Triangulationen. Zwei Knoten aus T_i und T_{i+1} sind durch Kante verbunden, wenn die Dreiecke sich überlappen.

6.2.3 Triangulationshierarchie

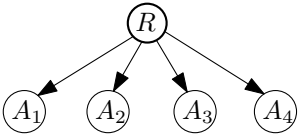


R

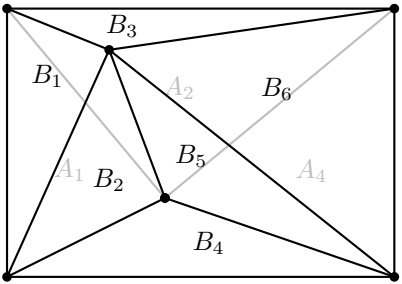
6.2.3 Triangulationshierarchie



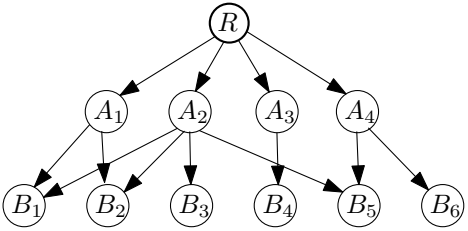
T_1



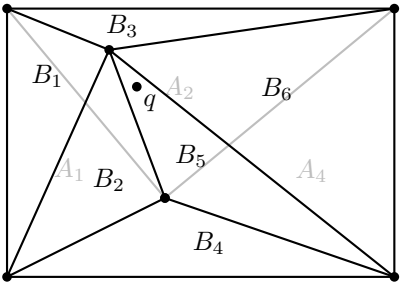
6.2.3 Triangulationshierarchie

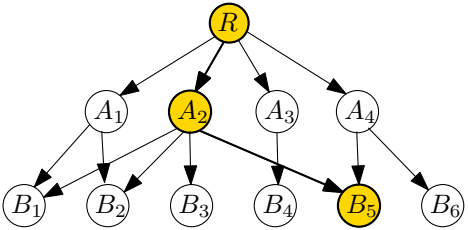


T_0



6.2.3 Triangulationshierarchie





Suchanfrage mit Punkt q: Starte im Wurzelknoten R von der Suchstruktur. Gehe rekursiv zu dem Kind-Knoten, dessen Dreieck q enthält, bis an einem Blattknoten angekommen.

6.2.3 Triangulationshierarchie

Theorem 6.26

Sei G der Graph der Suchstruktur einer Triangulationshierarchie für eine Triangulation mit n Knoten. Sei d der maximale Outgrad von G und sei h die Länge des längesten Weges von der Wurzel zu einem Knoten auf der untersten Ebene. Die Anfragezeit mit einem Punkt $q \in \mathbb{R}^2$ in G ist in $O(dh)$.

6.2.3 Triangulationshierarchie

Theorem 6.26

Sei G der Graph der Suchstruktur einer Triangulationshierarchie für eine Triangulation mit n Knoten. Sei d der maximale Outgrad von G und sei h die Länge des längesten Weges von der Wurzel zu einem Knoten auf der untersten Ebene. Die Anfragezeit mit einem Punkt $q \in \mathbb{R}^2$ in G ist in $O(dh)$.

Beweis: In jedem rekursiven Aufruf werden höchstens d ausgehende Kanten des aktuellen Knotens getestet.

6.2.3 Triangulationshierarchie

Theorem 6.26

Sei G der Graph der Suchstruktur einer Triangulationshierarchie für eine Triangulation mit n Knoten. Sei d der maximale Outgrad von G und sei h die Länge des längesten Weges von der Wurzel zu einem Knoten auf der untersten Ebene. Die Anfragezeit mit einem Punkt $q \in \mathbb{R}^2$ in G ist in $O(dh)$.

Beweis: In jedem rekursiven Aufruf werden höchstens d ausgehende Kanten des aktuellen Knotens getestet. Das Testen, ob ein Dreieck einen Punkt enthält kann in konstanter Zeit durchgeführt werden.

6.2.3 Triangulationshierarchie

Theorem 6.26

Sei G der Graph der Suchstruktur einer Triangulationshierarchie für eine Triangulation mit n Knoten. Sei d der maximale Outgrad von G und sei h die Länge des längesten Weges von der Wurzel zu einem Knoten auf der untersten Ebene. Die Anfragezeit mit einem Punkt $q \in \mathbb{R}^2$ in G ist in $O(dh)$.

Beweis: In jedem rekursiven Aufruf werden höchstens d ausgehende Kanten des aktuellen Knotens getestet. Das Testen, ob ein Dreieck einen Punkt enthält kann in konstanter Zeit durchgeführt werden. Die Anzahl der rekursiven Aufrufe entspricht der Länge des Pfades von der Wurzel zu einem Knoten auf der untersten Ebene der Triangulationshierarchie, entspricht also h. □

6.2.3 Triangulationshierarchie

Definition (Unabhängige Knotenmenge)

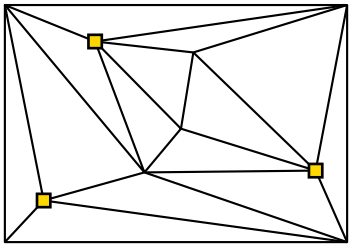
Eine unabhängige Knotenmenge eines Graphen $G = (V, E)$ ist eine Menge $S \subseteq V$, mit der Eigenschaft dass keine zwei Kanten aus S durch eine Kante in G verbunden sind.

6.2.3 Triangulationshierarchie

Definition (Unabhängige Knotenmenge)

Eine unabhängige Knotenmenge eines Graphen $G = (V, E)$ ist eine Menge $S \subseteq V$, mit der Eigenschaft dass keine zwei Kanten aus S durch eine Kante in G verbunden sind.

Beispiel:



6.2.3 Triangulationshierarchie

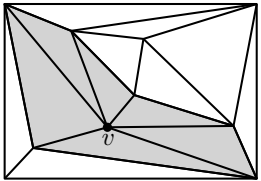
```
KIRKPATRICK(Triangulation T, int n)
1   i = 0;
2   T_0 = COPY(T);
3   while(n > C){
4       S = INDEPENDENTSET(T);
5       foreach(v ∈ S){
6           while(Knotengrad(v) > 3){
7               Führe zulässigen Flip auf einer Kante von v aus.
8           }
9           Entferne v mit seinen Kanten aus T.
10      }
11      n = n − |S|;
12      i = i + 1;
13      T_i = COPY(T);
14  }
```

6.2.3 Triangulationshierarchie

```
IndependentSet(Triangulation T)
 1    Sei W die Teilmenge der Knoten von T mit Knotengrad(v) ≤ 11,
 2    U = ∅;
 3    while(|W| ≥ 5){
 4        Sei v ∈ W innerer Knoten in T.
 5        Füge v zu U hinzu.
 6        Entferne v und seine benachbarten Knoten aus W.
14    } return U;
```

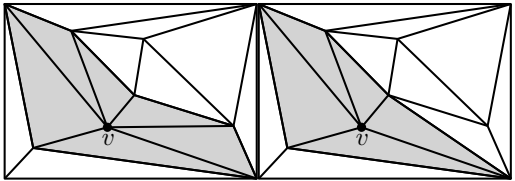
6.2.3 Triangulationshierarchie

Beispiel für Flips in Zeile 7:



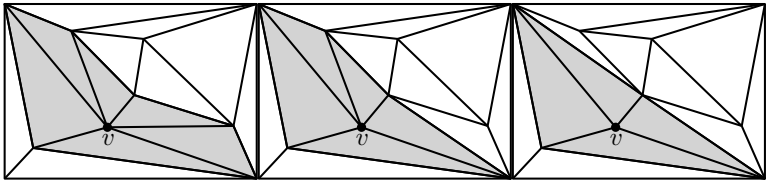
6.2.3 Triangulationshierarchie

Beispiel für Flips in Zeile 7:

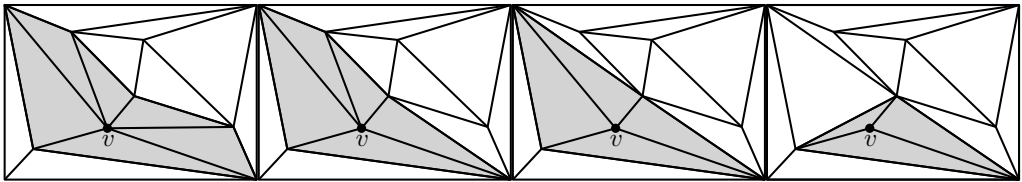


6.2.3 Triangulationshierarchie

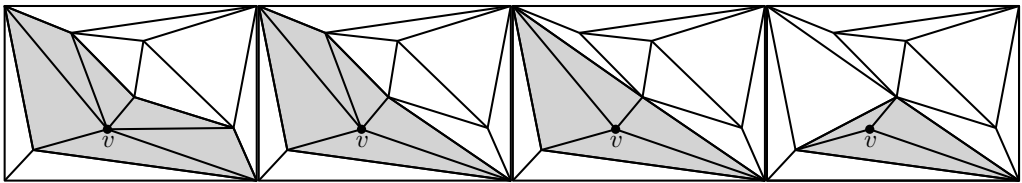
Beispiel für Flips in Zeile 7:



Beispiel für Flips in Zeile 7:



Beispiel für Flips in Zeile 7:



Man kann mithilfe der Winkelsummen zeigen, dass immer ein zulässiger Flip existiert, solange der Knotengrad von v größer 3 ist.

6.2.3 Triangulationshierarchie

Lemma 6.28

Sei $k \geq 5$, eine Triangulation T mit $n \geq 4$ Knoten hat mindestens $\lceil \frac{k-5}{k+1} n \rceil$ Knoten mit Outgrad höchstens k.

6.2.3 Triangulationshierarchie

Lemma 6.28

Sei $k \geq 5$, eine Triangulation T mit $n \geq 4$ Knoten hat mindestens $\lceil \frac{k-5}{k+1} n \rceil$ Knoten mit Outgrad höchstens k.

Beweis: Sei m die Anzahl der Knoten in der Triangulation T, die Knotengrad größer als k haben, für ein $k \geq 5$.

6.2.3 Triangulationshierarchie

Lemma 6.28

Sei $k \geq 5$, eine Triangulation T mit $n \geq 4$ Knoten hat mindestens $\lceil \frac{k-5}{k+1} n \rceil$ Knoten mit Outgrad höchstens k.

Beweis: Sei m die Anzahl der Knoten in der Triangulation T, die Knotengrad größer als k haben, für ein $k \geq 5$. Sei d_i der Knotengrad des $i$ten Knotens, wobei die Knoten in einer beliebige Reihenfolge seien. Es gilt

$$\sum_{i=1}^{n} d_i \geq m(k+1)$$

Lemma 6.28

Sei $k \geq 5$, eine Triangulation T mit $n \geq 4$ Knoten hat mindestens $\lceil \frac{k-5}{k+1} n \rceil$ Knoten mit Outgrad höchstens k.

Beweis: Sei m die Anzahl der Knoten in der Triangulation T, die Knotengrad größer als k haben, für ein $k \geq 5$. Sei d_i der Knotengrad des $i$ten Knotens, wobei die Knoten in einer beliebige Reihenfolge seien. Es gilt

$$\sum_{i=1}^{n} d_i \geq m(k+1)$$

Gleichzeitig wissen wir, dass jede Kante zu genau 2 Knoten inzident ist. Also gilt $\sum_{i=1}^{n} d_i = 2e$.

Lemma 6.28

Sei $k \geq 5$, eine Triangulation T mit $n \geq 4$ Knoten hat mindestens $\lceil \frac{k-5}{k+1} n \rceil$ Knoten mit Outgrad höchstens k.

Beweis: Sei m die Anzahl der Knoten in der Triangulation T, die Knotengrad größer als k haben, für ein $k \geq 5$. Sei d_i der Knotengrad des $i$ten Knotens, wobei die Knoten in einer beliebige Reihenfolge seien. Es gilt

$$\sum_{i=1}^{n} d_i \geq m(k+1)$$

Gleichzeitig wissen wir, dass jede Kante zu genau 2 Knoten inzident ist. Also gilt $\sum_{i=1}^{n} d_i = 2e$. Aus der Analyse im Beweis von Theorem 6.2 folgt, dass $e \leq 3(n-2)$.

6.2.3 Triangulationshierarchie

Zusammen implizieren diese Ungleichungen, dass

$$6n - 12 \geq 2e$$

6.2.3 Triangulationshierarchie

Zusammen implizieren diese Ungleichungen, dass

$$6n - 12 \geq 2e = \sum_{i=1}^{n} d_i$$

6.2.3 Triangulationshierarchie

Zusammen implizieren diese Ungleichungen, dass

$$6n - 12 \geq 2e = \sum_{i=1}^{n} d_i \geq m(k+1)$$

6.2.3 Triangulationshierarchie

Zusammen implizieren diese Ungleichungen, dass

$$6n - 12 \geq 2e = \sum_{i=1}^{n} d_i \geq m(k+1)$$

und somit gilt

$$m \leq \frac{6n - 12}{k+1}$$

6.2.3 Triangulationshierarchie

Zusammen implizieren diese Ungleichungen, dass

$$6n - 12 \geq 2e = \sum_{i=1}^{n} d_i \geq m(k+1)$$

und somit gilt

$$m \leq \frac{6n - 12}{k+1}$$

Es gibt $n - m$ Knoten, die Knotengrad höchstens k haben.

6.2.3 Triangulationshierarchie

Zusammen implizieren diese Ungleichungen, dass

$$6n - 12 \geq 2e = \sum_{i=1}^{n} d_i \geq m(k+1)$$

und somit gilt

$$m \leq \frac{6n - 12}{k+1}$$

Es gibt $n - m$ Knoten, die Knotengrad höchstens k haben. Einsetzen ergibt

$$n - m \geq n - \frac{6n - 12}{k+1}$$

6.2.3 Triangulationshierarchie

Zusammen implizieren diese Ungleichungen, dass

$$6n - 12 \geq 2e = \sum_{i=1}^{n} d_i \geq m(k+1)$$

und somit gilt

$$m \leq \frac{6n-12}{k+1}$$

Es gibt $n - m$ Knoten, die Knotengrad höchstens k haben. Einsetzen ergibt

$$n - m \geq n - \frac{6n-12}{k+1} = \frac{k-5}{k+1}n + \frac{12}{k+1}$$

6.2.3 Triangulationshierarchie

Zusammen implizieren diese Ungleichungen, dass

$$6n - 12 \geq 2e = \sum_{i=1}^{n} d_i \geq m(k+1)$$

und somit gilt

$$m \leq \frac{6n-12}{k+1}$$

Es gibt $n - m$ Knoten, die Knotengrad höchstens k haben. Einsetzen ergibt

$$n - m \geq n - \frac{6n-12}{k+1} = \frac{k-5}{k+1}n + \frac{12}{k+1} \geq \frac{k-5}{k+1}n$$

Zusammen implizieren diese Ungleichungen, dass

$$6n - 12 \geq 2e = \sum_{i=1}^{n} d_i \geq m(k+1)$$

und somit gilt

$$m \leq \frac{6n-12}{k+1}$$

Es gibt $n - m$ Knoten, die Knotengrad höchstens k haben. Einsetzen ergibt

$$n - m \geq n - \frac{6n-12}{k+1} = \frac{k-5}{k+1}n + \frac{12}{k+1} \geq \frac{k-5}{k+1}n$$

Da die Anzahl der Knoten eine natürliche Zahl ist, dürfen wir aufrunden. □

6.2.3 Triangulationshierarchie

Lemma 6.28

Sei $k \geq 5$. Eine Triangulation T mit $n \geq 4$ Knoten hat mindestens $\lceil \frac{k-5}{k+1} n \rceil$ Knoten mit Knotengrad höchstens k.

Einsetzen mit $k = 11$ ergibt das folgende Korollar.

Korollar 6.29

Eine Triangulation T mit $n \geq 4$ Knoten hat mindestens $\lceil \frac{n}{2} \rceil$ Knoten mit Knotengrad höchstens 11.

6.2.3 Triangulationshierarchie

Lemma 6.30

Sei $C = 251$ und sei T eine Triangulation mit $n > C$ Knoten, von denen höchstens 4 auf der äußeren Fläche liegen. Der Algorithmus IndependentSet berechnet für T eine unabhängige Knotenmenge S mit $|S| = \lceil \frac{n}{25} \rceil$ mit Knotengrad höchstens 11, wobei keiner der Knoten von S auf der äußeren Fläche von T liegt. Die Laufzeit ist in $O(n)$.

6.2.3 Triangulationshierarchie

Lemma 6.30

Sei $C = 251$ und sei T eine Triangulation mit $n > C$ Knoten, von denen höchstens 4 auf der äußeren Fläche liegen. Der Algorithmus IndependentSet berechnet für T eine unabhängige Knotenmenge S mit $|S| = \lceil \frac{n}{25} \rceil$ mit Knotengrad höchstens 11, wobei keiner der Knoten von S auf der äußeren Fläche von T liegt. Die Laufzeit ist in $O(n)$.

Beweis: Mithilfe Breitensuche kann die Menge W in $O(n)$ Zeit berechnet werden werden.

6.2.3 Triangulationshierarchie

Lemma 6.30

Sei $C = 251$ und sei T eine Triangulation mit $n > C$ Knoten, von denen höchstens 4 auf der äußeren Fläche liegen. Der Algorithmus IndependentSet berechnet für T eine unabhängige Knotenmenge S mit $|S| = \lceil \frac{n}{25} \rceil$ mit Knotengrad höchstens 11, wobei keiner der Knoten von S auf der äußeren Fläche von T liegt. Die Laufzeit ist in $O(n)$.

Beweis: Mithilfe Breitensuche kann die Menge W in $O(n)$ Zeit berechnet werden werden. Aus Korollar 6.29 folgt $|W| \geq \lceil \frac{n}{2} \rceil$. Sei $k = 11$.

6.2.3 Triangulationshierarchie

Lemma 6.30

Sei $C = 251$ und sei T eine Triangulation mit $n > C$ Knoten, von denen höchstens 4 auf der äußeren Fläche liegen. Der Algorithmus IndependentSet berechnet für T eine unabhängige Knotenmenge S mit $|S| = \lceil \frac{n}{25} \rceil$ mit Knotengrad höchstens 11, wobei keiner der Knoten von S auf der äußeren Fläche von T liegt. Die Laufzeit ist in $O(n)$.

Beweis: Mithilfe Breitensuche kann die Menge W in $O(n)$ Zeit berechnet werden werden. Aus Korollar 6.29 folgt $|W| \geq \lceil \frac{n}{2} \rceil$. Sei $k = 11$. In jedem Schritt entfernt der Algorithmus höchstens $k + 1$ Knoten aus W und erweitert die unabhängige Knotenmenge U um genau einen Knoten.

6.2.3 Triangulationshierarchie

Lemma 6.30

Sei $C = 251$ und sei T eine Triangulation mit $n > C$ Knoten, von denen höchstens 4 auf der äußeren Fläche liegen. Der Algorithmus IndependentSet berechnet für T eine unabhängige Knotenmenge S mit $|S| = \lceil \frac{n}{25} \rceil$ mit Knotengrad höchstens 11, wobei keiner der Knoten von S auf der äußeren Fläche von T liegt. Die Laufzeit ist in $O(n)$.

Beweis: Mithilfe Breitensuche kann die Menge W in $O(n)$ Zeit berechnet werden werden. Aus Korollar 6.29 folgt $|W| \geq \lceil \frac{n}{2} \rceil$. Sei $k = 11$. In jedem Schritt entfernt der Algorithmus höchstens $k + 1$ Knoten aus W und erweitert die unabhängige Knotenmenge U um genau einen Knoten. Sei w_i die Größe von W nach i Schritten. Es gilt

$$w_i \geq w_0 - i(k + 1)$$

6.2.3 Triangulationshierarchie

Lemma 6.30

Sei $C = 251$ und sei T eine Triangulation mit $n > C$ Knoten, von denen höchstens 4 auf der äußeren Fläche liegen. Der Algorithmus IndependentSet berechnet für T eine unabhängige Knotenmenge S mit $|S| = \lceil \frac{n}{25} \rceil$ mit Knotengrad höchstens 11, wobei keiner der Knoten von S auf der äußeren Fläche von T liegt. Die Laufzeit ist in $O(n)$.

Beweis: Mithilfe Breitensuche kann die Menge W in $O(n)$ Zeit berechnet werden werden. Aus Korollar 6.29 folgt $|W| \geq \lceil \frac{n}{2} \rceil$. Sei $k = 11$. In jedem Schritt entfernt der Algorithmus höchstens $k + 1$ Knoten aus W und erweitert die unabhängige Knotenmenge U um genau einen Knoten. Sei w_i die Größe von W nach i Schritten. Es gilt

$$w_i \geq w_0 - i(k + 1)$$

Sei $m = |U|$ wenn der Algorithmus terminiert. Es gilt $w_m < 5$.

6.2.3 Triangulationshierarchie

Daraus folgt

$$5 > w_m \geq w_0 - m(k+1)$$

6.2.3 Triangulationshierarchie

Daraus folgt

$$5 > w_m \geq w_0 - m(k+1)$$

Wegen Korollar 6.29 gilt $w_0 \geq \frac{n}{2}$ und da $k = 11$ gewählt wurde, folgt

$$5 > \frac{n}{2} - 12m$$

6.2.3 Triangulationshierarchie

Daraus folgt

$$5 > w_m \geq w_0 - m(k+1)$$

Wegen Korollar 6.29 gilt $w_0 \geq \frac{n}{2}$ und da $k = 11$ gewählt wurde, folgt

$$5 > \frac{n}{2} - 12m$$

Diese Ungleichung können wir nach m umstellen. Für $n \geq C$ gilt

$$m > \frac{n}{24} - \frac{5}{12} > \frac{n}{25}$$

Daraus folgt

$$5 > w_m \geq w_0 - m(k+1)$$

Wegen Korollar 6.29 gilt $w_0 \geq \frac{n}{2}$ und da $k = 11$ gewählt wurde, folgt

$$5 > \frac{n}{2} - 12m$$

Diese Ungleichung können wir nach m umstellen. Für $n \geq C$ gilt

$$m > \frac{n}{24} - \frac{5}{12} > \frac{n}{25}$$

Da m eine natürliche Zahl ist, dürfen wir aufrunden. Es gilt also $m \geq \left\lceil \frac{n}{25} \right\rceil$ für hinreichend große n. □

6.2.3 Triangulationshierarchie

Theorem 6.31

Sei T eine Triangulation mit $n \geq 4$ Knoten in der Ebene, von denen höchstens 4 auf der äußeren Fläche liegen. Wir können in Laufzeit $O(n)$ eine Datenstruktur bauen, die in Zeit $O(\log n)$ das Dreieck bestimmt, das den gegebenen Anfragepunkt enthält. Die Datenstruktur benutzt Speicherplatz in $O(n)$.

6.2.3 Triangulationshierarchie

Theorem 6.31

Sei T eine Triangulation mit $n \geq 4$ Knoten in der Ebene, von denen höchstens 4 auf der äußeren Fläche liegen. Wir können in Laufzeit $O(n)$ eine Datenstruktur bauen, die in Zeit $O(\log n)$ das Dreieck bestimmt, das den gegebenen Anfragepunkt enthält. Die Datenstruktur benutzt Speicherplatz in $O(n)$.

Beweis: Wieviele Triangulationen werden vom Algorithmus erstellt?

6.2.3 Triangulationshierarchie

Theorem 6.31

Sei T eine Triangulation mit $n \geq 4$ Knoten in der Ebene, von denen höchstens 4 auf der äußeren Fläche liegen. Wir können in Laufzeit $O(n)$ eine Datenstruktur bauen, die in Zeit $O(\log n)$ das Dreieck bestimmt, das den gegebenen Anfragepunkt enthält. Die Datenstruktur benutzt Speicherplatz in $O(n)$.

Beweis: Wieviele Triangulationen werden vom Algorithmus erstellt?Aus Lemma 6.30 folgt, dass die Anzahl der Knoten in jedem Schritt um mindestens einen Faktor $\frac{1}{25}$ kleiner wird.

6.2.3 Triangulationshierarchie

Theorem 6.31

Sei T eine Triangulation mit $n \geq 4$ Knoten in der Ebene, von denen höchstens 4 auf der äußeren Fläche liegen. Wir können in Laufzeit $O(n)$ eine Datenstruktur bauen, die in Zeit $O(\log n)$ das Dreieck bestimmt, das den gegebenen Anfragepunkt enthält. Die Datenstruktur benutzt Speicherplatz in $O(n)$.

Beweis: Wieviele Triangulationen werden vom Algorithmus erstellt?Aus Lemma 6.30 folgt, dass die Anzahl der Knoten in jedem Schritt um mindestens einen Faktor $\frac{1}{25}$ kleiner wird. Der Prozess terminiert sobald höchstens $C = 251$ Knoten übrig sind.

6.2.3 Triangulationshierarchie

Theorem 6.31

Sei T eine Triangulation mit $n \geq 4$ Knoten in der Ebene, von denen höchstens 4 auf der äußeren Fläche liegen. Wir können in Laufzeit $O(n)$ eine Datenstruktur bauen, die in Zeit $O(\log n)$ das Dreieck bestimmt, das den gegebenen Anfragepunkt enthält. Die Datenstruktur benutzt Speicherplatz in $O(n)$.

Beweis: Wieviele Triangulationen werden vom Algorithmus erstellt?Aus Lemma 6.30 folgt, dass die Anzahl der Knoten in jedem Schritt um mindestens einen Faktor $\frac{1}{25}$ kleiner wird. Der Prozess terminiert sobald höchstens $C = 251$ Knoten übrig sind. Wir suchen also nach der kleinsten natürlichen Zahl h, sodass gilt

$$\left(\frac{24}{25}\right)^h n \leq C$$

6.2.3 Triangulationshierarchie

$$\left(\frac{24}{25}\right)^h n \leq C$$

6.2.3 Triangulationshierarchie

$$\left(\frac{24}{25}\right)^h n \leq C$$

Das ist äquivalent zu

$$h \cdot \log_2\left(\frac{25}{24}\right) \geq \log_2\left(\frac{n}{C}\right)$$

6.2.3 Triangulationshierarchie

$$\left(\frac{24}{25}\right)^h n \leq C$$

Das ist äquivalent zu

$$h \cdot \log_2\left(\frac{25}{24}\right) \geq \log_2\left(\frac{n}{C}\right)$$

Die kleinste natürliche Zahl h, die diese Bedingungen erfüllt ist

$$h = \left\lceil \frac{\log_2\left(\frac{n}{C}\right)}{\log_2\left(\frac{25}{24}\right)} \right\rceil \in O(\log n)$$

6.2.3 Triangulationshierarchie

Sei n_i die Anzahl der Knoten in der Triangulation T_i.

6.2.3 Triangulationshierarchie

Sei n_i die Anzahl der Knoten in der Triangulation T_i. Die Konstruktion dauert in jedem Schritt $O(n_i)$ Zeit.

6.2.3 Triangulationshierarchie

Sei n_i die Anzahl der Knoten in der Triangulation T_i. Die Konstruktion dauert in jedem Schritt $O(n_i)$ Zeit. Die Gesamtlaufzeit ist also asymptotisch beschränkt durch

$$\sum_{i=0}^{h} n_i$$

6.2.3 Triangulationshierarchie

Sei n_i die Anzahl der Knoten in der Triangulation T_i. Die Konstruktion dauert in jedem Schritt $O(n_i)$ Zeit. Die Gesamtlaufzeit ist also asymptotisch beschränkt durch

$$\sum_{i=0}^{h} n_i \leq \sum_{i=0}^{h} \left(\frac{24}{25}\right)^h n$$

6.2.3 Triangulationshierarchie

Sei n_i die Anzahl der Knoten in der Triangulation T_i. Die Konstruktion dauert in jedem Schritt $O(n_i)$ Zeit. Die Gesamtlaufzeit ist also asymptotisch beschränkt durch

$$\sum_{i=0}^{h} n_i \leq \sum_{i=0}^{h} \left(\frac{24}{25}\right)^h n \leq n \cdot \sum_{i=0}^{\infty} \left(\frac{24}{25}\right)^i$$

6.2.3 Triangulationshierarchie

Sei n_i die Anzahl der Knoten in der Triangulation T_i. Die Konstruktion dauert in jedem Schritt $O(n_i)$ Zeit. Die Gesamtlaufzeit ist also asymptotisch beschränkt durch

$$\sum_{i=0}^{h} n_i \leq \sum_{i=0}^{h} \left(\frac{24}{25}\right)^h n \leq n \cdot \sum_{i=0}^{\infty} \left(\frac{24}{25}\right)^i \leq 25n$$

wobei der letzte Schritt aus der Formel für die geometrische Reihe folgt.

6.2.3 Triangulationshierarchie

Sei n_i die Anzahl der Knoten in der Triangulation T_i. Die Konstruktion dauert in jedem Schritt $O(n_i)$ Zeit. Die Gesamtlaufzeit ist also asymptotisch beschränkt durch

$$\sum_{i=0}^{h} n_i \leq \sum_{i=0}^{h} \left(\frac{24}{25}\right)^h n \leq n \cdot \sum_{i=0}^{\infty} \left(\frac{24}{25}\right)^i \leq 25n$$

wobei der letzte Schritt aus der Formel für die geometrische Reihe folgt. Diese Schranke und Analyse gelten gleichermaßen auch für den Speicherplatz.

6.2.3 Triangulationshierarchie

Sei n_i die Anzahl der Knoten in der Triangulation T_i. Die Konstruktion dauert in jedem Schritt $O(n_i)$ Zeit. Die Gesamtlaufzeit ist also asymptotisch beschränkt durch

$$\sum_{i=0}^{h} n_i \leq \sum_{i=0}^{h} \left(\frac{24}{25}\right)^h n \leq n \cdot \sum_{i=0}^{\infty} \left(\frac{24}{25}\right)^i \leq 25n$$

wobei der letzte Schritt aus der Formel für die geometrische Reihe folgt. Diese Schranke und Analyse gelten gleichermaßen auch für den Speicherplatz.

Aus Theorem 6.26 folgt, dass die Anfragezeit linear in der maximalen Länge eines Pfades von der Wurzel zu einem Dreieck in T_0 ist, da der Outgrad der Suchstruktur konstant ist.

6.2.3 Triangulationshierarchie

Sei n_i die Anzahl der Knoten in der Triangulation T_i. Die Konstruktion dauert in jedem Schritt $O(n_i)$ Zeit. Die Gesamtlaufzeit ist also asymptotisch beschränkt durch

$$\sum_{i=0}^{h} n_i \leq \sum_{i=0}^{h} \left(\frac{24}{25}\right)^h n \leq n \cdot \sum_{i=0}^{\infty} \left(\frac{24}{25}\right)^i \leq 25n$$

wobei der letzte Schritt aus der Formel für die geometrische Reihe folgt. Diese Schranke und Analyse gelten gleichermaßen auch für den Speicherplatz.

Aus Theorem 6.26 folgt, dass die Anfragezeit linear in der maximalen Länge eines Pfades von der Wurzel zu einem Dreieck in T_0 ist, da der Outgrad der Suchstruktur konstant ist. Nach der Analyse oben ist die Länge des Pfades höchstens $h + 1$ und liegt somit $O(\log n)$. □