**Algorithmische Geometrie**
**Wintersemester 2020/2021**
Thomas Bläsius & Marcus Wilhelm

Karlsruher Institut für Technologie

# Musterlösung zum Übungsblatt 5

Erstellt von Emil und Christian

## Aufgabe 1: Größe des Voronoi-Diagramms

**3 Punkte**

Sei $n$ die Anzahl der Punkte der Eingabe. Wir bezeichnen in einem Graph die Anzahl der Kanten mit $m$, die Anzahl der Knoten mit $nodes$ und die Anzahl der Facetten mit $f$. In planaren Graphen gilt dann: $m \leq 3nodes - 6$ (vgl. Algorithmen für planare Graphen) und es gilt die Euler-Formel: $nodes - m + f = 2$.
Betrachten wir nun Voronoi-Diagramme. Wenn es keine 4 Punkte gibt, die auf einem Kreis liegen, so ist der Dualgraph die Delaunay-Triangulierung. Zwischen den Kanten bzw. Halbkanten des Voronoi-Diagrams und seinem Dualgraph besteht, durch die Konstruktion des Dualgraphen, eine Eins-zu-Eins Beziehung. Durch die Konstruktion des Voronoi-Diagrams ist jeder Punkt der Eingabe ein Knoten des Dualgraphs. Daher genügt es die Delaunay-Triangulierung anzusehen, welche als Triangulierung $3n - 6$ Kanten hat. Im Fall, dass 4 Punkte auf einem Kreis liegen, ist der Dualgraph des Voronoi-Diagramms keine Triangulierung mehr. Es gibt jedoch immernoch, dass jeder Punkt der Eingabe ein Knoten im Dualgraph ist. Da dieser nun keine Triangulierung mehr ist, hat er $\leq 3n - 6$ Kanten und die obere Schranke gilt für beliebige Voronoi-Diagramme.
Nun zur Abschätzung der Knoten. Wir sehen uns das Voronoi-Diagramm erweitert mit einer

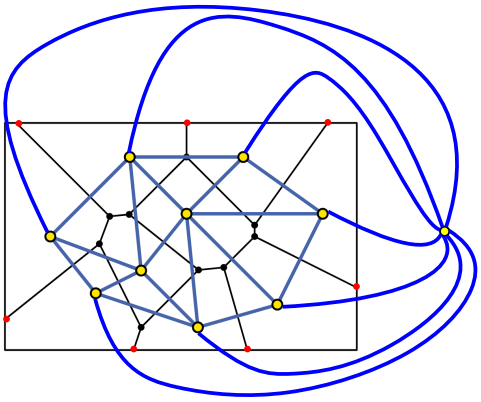

Abbildung 1: Ein erweitertes Voronoi-Diagramm und sein Dualgraph, eine Delaunay-Triangulierung, mit äußerer Facette

Bounding Box an (siehe Abb. 1). Für das erweiterte Voronoi-Diagramm gilt die Euler Formel umgestellt und mit $f = n + 1$ da eine weitere Facette hinzukommt: $nodes = m - n - 1 + 2$. Da wir an einer oberen Schranke interessiert sind, müssen wir uns nur den Fall ansehen, dass keine 4 Punkte

auf einem Kreis liegen, sonst gäbe es weniger Kanten und nach der gerade umgestellten Formel auch weniger Knoten. Für unser erweitertes Voronoi-Diagramm hat nun die ursprünglichen Knoten $nodes_{old}$ und die in der Abb. roten Knoten $nodes_{red}$. Es gibt $nodes_{red}$ neue Kanten zwischen den roten Knoten. Es gilt also $nodes = nodes_{old} + nodes_{red} = m - n - 1 + 2 = 3n - 6 + nodes_{red} - n - 1 + 2$ Nach subtraktion von $nodes_{red}$ folgt: $nodes_{old} = 2n - 5$. Was damit eine obere Schranke für die Anzahl der Knoten des Voronoi-Diagramms ist.

## Aufgabe 2: Untere Schranke: Voronoi-Diagramm **3 Punkte**

Gegeben die zu sortierenden Zahlen $z_1, \dots, z_n$, konstruiert man das Voronoi-Diagramm der Punkte $(z_1,0), \dots, (z_n,0)$ und $p_\infty = (42 \max |z_i|, 42 \max |z_i|)$ über dem rechtesten der $p_i$ (siehe Abb. 2). Der Punkt $p_\infty$ verhindert den degenerierten Fall, dass alle Punkte kolinear sind. Daher liegt jeder Punkt $(z_i,0)$ in einer halb-abgeschlossenen Voronoi-Zelle. Außerdem gibt es keine vier Punkte auf einem Kreis. Wenn $z_i < z_j$ in der Sortierung aufeinanderfolgen, dann ist der Punkt $(\frac{1}{2}(z_i + z_j), 0)$ auf dem Rand der Voronoizellen von $(z_i, 0)$ und $(z_j, 0)$; diese sind also benachbart. Sind umgekehrt zwei Voronoizellen der Punkte $(z_i, 0)$ und $(z_j, 0)$ benachbart, dann gibt es einen Punkt $(p_x, p_y)$ mit $|p_x - z_i| = |p_x - z_j| < |p_x - z_k|$ für $k \neq i,j$. Folglich sind $z_i, z_j$ in der Sortierung benachbart.

Die Sortierung kann damit in $\mathcal{O}(n)$ aus dem Voronoi-Diagramm berechnet werden. Somit ist die untere Laufzeitschranke $\Omega(n \log n)$ für das Sortieren von $n$ Zahlen auch eine untere Schranke für die Konstruktion des Voronoi-Diagramms von $n$ Punkten.

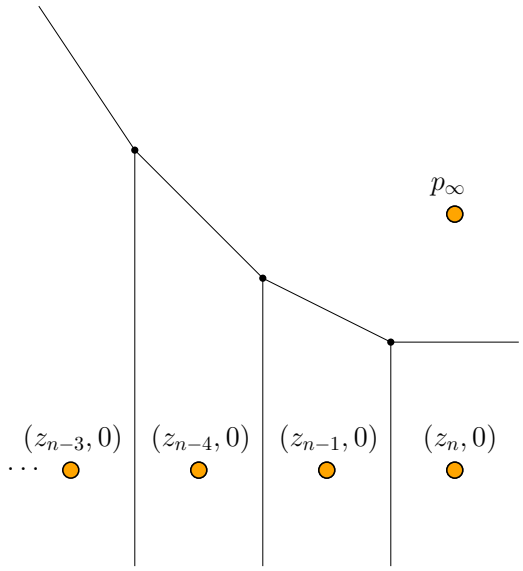


Abbildung 2: Voronio-Diagramm der konstruierten Punktmenge

## Aufgabe 3: Da geht mir ein Knicklicht auf! **6 Punkte**

Gegeben eine Instanz von *planar monotone 3SAT*, modellieren wir einen Graph, der genau dann *knickfrei orthogonal zeichenbar (KOZ)* mit *Winkelkomplexität* 6 ist, wenn die Instanz erfüllbar ist. Für ein Beispiel der, im folgenden beschriebenen Konstruktion siehe Abb. 5.

Als Erstes betrachten wir das Variablen-Gadget in Abb. 3. Es dient dazu, die Orientierung des

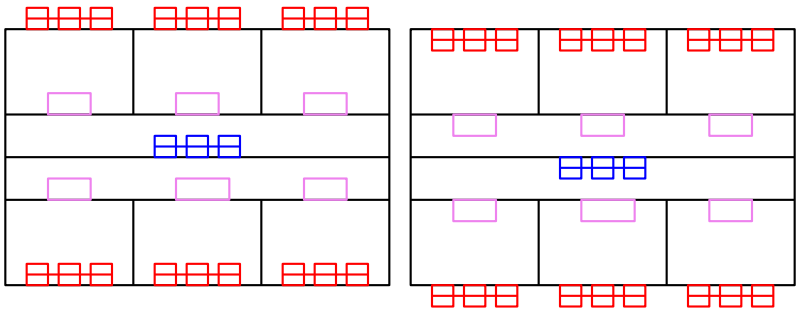

Abbildung 3: Die zwei möglichen Orientierungen eines Variablen-Gadget

blauen Teils in der Mitte zu vervielfältigen. Dabei bedeutet die Orientierung des blauen Teils nach *oben*, dass das positive Literal der Variable *false* ist, die Orientierung nach *unten*, dass das negative Literal der Variable *false* ist.

Wenn der blaue Teil nach oben gedreht ist, so sind in seiner Facette bereits 6 Winkel mit 270°. Damit müssen alle oberen violetten Teile des Gadgets auch nach oben gedreht sein, was wiederum die Orientierung der oberen roten Teile nach oben erzwingt. Dabei führt die horizontale Verbindung der *drei roten Quader* dazu, dass sie entweder alle nach oben oder alle nach unten zeigen. Somit haben wir mit dem blauen Teil die Orientierung von drei gleichen Teile erzwungen. Wenn wir die Variable häufiger benötigen, kann diese Vervielfältigung für jeden der roten Teile wiederholt werden. Für den unteren Teil des linken Gadgets in Abb. 3, ist durch die Orientierung des blauen Teils noch nichts gegeben. Theoretisch könnten die violetten Teile nach unten gerichtet sein oder die unteren roten Teile nach unten orientiert sein, dies hat jedoch keine Auswirkungen auf die Erfüllbarkeit.

Es muss nun an die Klauseln übertragen werden, welches Literal sicher *false* ist. Dafür kann man Tunnel (siehe Teil a) in Abb. 4) verwenden. Dieser überträgt die Orientierung des roten Teils des Variablen-Gadgets an den orangenen Teil. Auch hier ist die Orientierung nur dann erzwungen, wenn der blaue Teil des Variablen-Gadgets in die Richtung dieses Tunnels zeigt.

Das Clause-Gadget kombiniert nun die Orientierungen von drei Literalen. Dabei dürfen nicht alle drei ankommenden Literale *false* sein, also nicht alle drei orangen Teile in Abb. 4 b) in das Clause-Gadget hinein ragen, da sonst 8 Winkel mit 270° in der Facette lägen.

Eine Instanz von *planar monotnoe 3SAT* ist nun genau dann erfüllbar, wenn der, wie in Abb. 5 erzeugte Graph *KOZ* mit *Winkelkomplexität* 6 ist:

$\Rightarrow$: Gegeben eine erfüllende Belegung der Variablen, orientiere die blauen, roten und orangen Teile

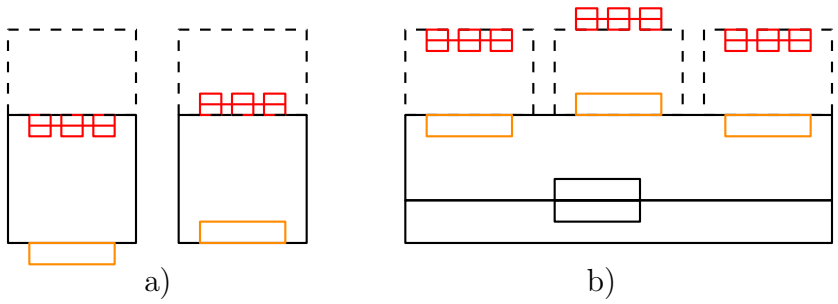


Abbildung 4: a) Tunnel *false* und *true* und b) Clause-Gadget mit *false true false*

der Variablen-Gadgets und der darauf folgenden Tunnel und Clause-Gadgets nach oben, wenn die Variable *false* ist, sonst orientere sie nach unten. Durch die Konstruktion sind in keiner Facette der Variablen-Gadgets oder der Tunnel mehr als 6 Winkel mit 270°. Da die Variablen-Belegung erfüllend ist, sind in jeder Clausel-Gadget Facette maximal zwei Literale *false*, also sind nur 6 Winkel mit 270° erzwungen. Die Winkel mit 270° an der äußeren Facette können alle mit einem weiteren Knoten versehen werden, wodurch sie nicht mehr relevant sind, siehe Abb. 5.
⟸: Gegeben die *KOZ* Zeichnung der Konstruktion zu einer Instanz von *planar monotnoe 3SAT*. Lese aus den Orientierungen der blauen Teile die Belegung der Variablen ab. Da Literale, die *false* sind bis an die Clause-Gadgets übertragen werden und keine der Clause-Gadget Facetten mehr als 6 Winkel mit 270° beinhaltet, gibt es in jeder Klausel ein erfülltes Literal und die Instanz ist erfüllbar.

## Aufgabe 4: Triangulierung Konzyklischer Punkte 3 + 3 + 2 = 8 Punkte

**Teilaufgabe (a)** Betrachte eine beliebige Triangulierung des Polygons $P = (p_1, \dots, p_n)$. Aus dem verallgemeinertem Satz des Thales folgt, dass für jedes Dreieck $p_i p_j p_k$ der Winkel an $p_k$ nur von der Länge $|p_i p_j|$ der Seite abhängt. Jeder Winkel liegt gegenüber einer Kante und jede Kante hat einen (Polygonkante) oder zwei (Sehne) Winkel gegenüberliegend. Daher lässt sich der Winkel-Vektor einer Triangulierung nur anhand des jeweiligen Längen-Vektors bestimmen.

Die Polygonkanten sind in jeder Triangulierung vorhanden. Daher kommen die Winkel gegenüber der Polygonkanten im Winkel-Vektor jeder Triangulierung vor. Für eine Sehne gilt, dass die Größe des kleineren der beiden gegenüberliegenden Winkel monoton in der Länge der Sehne ist (Mit dem Sinussatz erhält man $\alpha = \arcsin \frac{l}{2R}$, wobei $l$ die Länge der gegenüberliegenden Seite und $R$ den Radius des Kreises bezeichnet. Der arcsin ist monoton wachsend; also ist auch $\alpha$ monoton wachsend in $l$).

Damit ist auch die Abbildung der Längen-Vektoren auf die zugehörigen Winkel-Vektoren monoton bzgl. der Ordnung auf den Längen- bzw. Winkel-Vektoren. Daraus folgt, dass der Längen-Vektor einer Triangulierung genau dann maximal ist, wenn der Winkel-Vektor maximal ist.

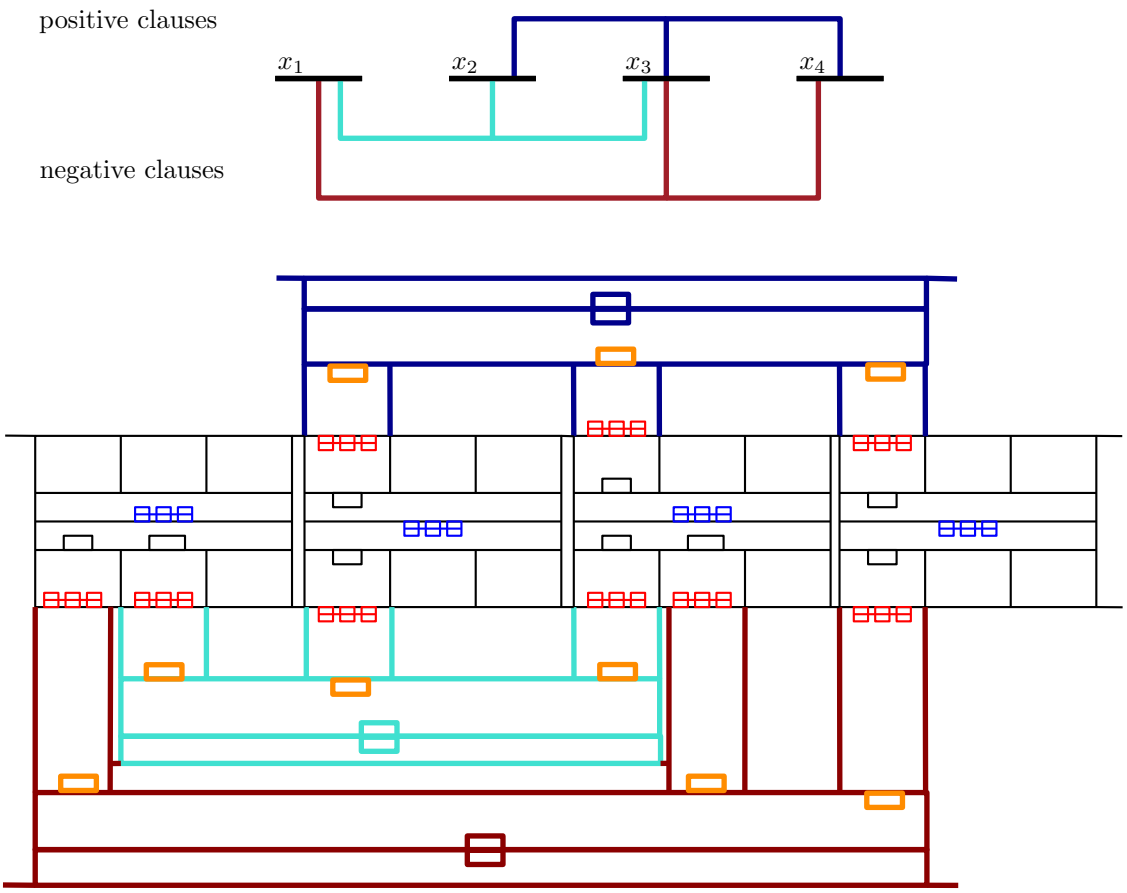


Abbildung 5: Modellierung einer Instanz von *planar monotnoe 3SAT* in *KOZ*

**Teilaufgabe (b)** Jede Triangulierung eines konvexen Polygons mit $n$ Ecken besteht aus genau $n-2$ Dreiecken und genau $n-3$ Sehnen (beides lässt sich leicht per Induktion über $n$ zeigen). Daher ist der schwache Dualgraph $G^*$ ein zusammenhängender Graph mit $n-2$ Knoten und $n-3$ Kanten; also ein Baum. Es bleibt zu zeigen, dass eine maximale Triangulierung genau zwei Blätter hat.

Ein Blatt in $G^*$ ist dadurch charakterisiert, dass das zugehörige Dreieck im Primalgraphen $G$ genau zwei Polygonkanten $p_ip_{i+1}$, $p_{i+1}p_{i+2}$ und eine Sehne $p_ip_{i+2}$, im Folgenden als *kurze Sehne* bezeichnet, enthält. Jede Sehne einer Triangulierung ist die längste Seite mindestens eines der beiden angrenzenden Dreiecke. Damit folgt, dass die kürzeste Sehne einer Triangulierung Teil eines Blattes sein muss, da sie Teil eines Dreiecks mit zwei kürzeren Seiten, also Polygonkanten, ist.

Für jedes Paar von sich nicht kreuzenden kurzen Sehnen $s_1 = p_ip_{i+2}$, $s_2 = p_jp_{j+2}$ gibt es eine

Triangulierung deren schwacher Dualgraph ein Pfad ist und in der $s_1$ und $s_2$ an den beiden Blättern liegt. Eine solche Triangulierung kann konstruiert werden, indem jeder Knoten auf dem Kreisbogen zwischen $p_{i+2}$ und $p_j$ mit dem Knoten $p_{j+2}$ und jeder Knoten auf dem Kreisbogen zwischen $p_{j+2}$ und $p_i$ mit dem Knoten $p_{i+2}$ verbunden wird. Außerdem wird die Sehne $p_{i+2}p_{j+2}$ eingefügt. Exemplarisch ist dies in Abb 6 dargestellt. Da alle eingefügten Sehnen zwischen zwei durch jeweils drei Knoten getrennten Kreisbögen verlaufen, werden keine kurzen Sehnen eingefügt. Es entstehen also keine zusätzlichen Blätter.

Gegeben sei nun eine Triangulierung mit mindestens drei Blättern und zugehörigen kurzen Sehnen $s_1$,$s_2$,$s_3$ wobei $|s_1| < |s_2| < |s_3|$. Dann gibt es aber eine Pfad-Triangulierung mit den kurzen Sehnen $s_2$,$s_3$, die $s_1$ nicht enthält. In dieser Triangulierung ist jede weitere Sehen länger als $s_2$ und damit länger als $s_1$. Nach der a) hat sie also einen größeren Winkel-Vektor. Insgesamt folgt, dass eine maximale Triangulierung keine drei Blätter haben kann und damit ein Pfad sein muss.

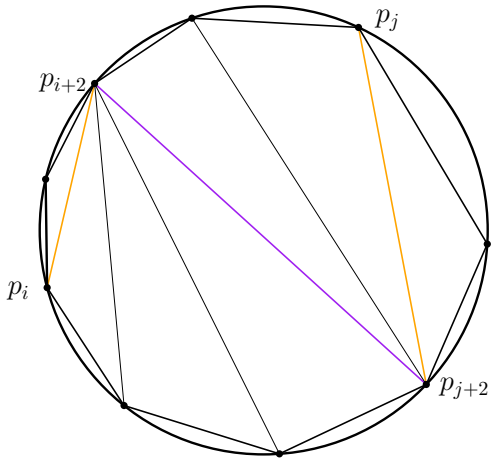


Abbildung 6: Schema einer Pfad-Triangulierung mit zwei kurzen Sehnen (orange)

**Teilaufgabe (c)** Der Algorithmus fügt iterativ Kanten ein und liefert nach $n - 3$ Schritten eine Triangulierung. In jedem Schritt sucht der Algorithmus das Paar von sich nicht kreuzenden kurzen Sehnen, bei dem die Länge der kürzeren Sehne maximal ist. Die kürzere Sehne wird in die Triagulierung eingefügt. Im ersten Schritt findet der Algorithmus das Paar, indem er in die drei längsten kurzen Sehnen bestimmt ($\mathcal{O}(n)$). Unter diesen dreien gibt es mindestens ein Paar, welches sich nicht kreuzt. Nach dem Einfügen einer kurzen Sehne $s_1$ entstehen zwei neue kurze Sehnen $s_2$,$s_3$ (siehe Abb. 7). Diese sind die längsten Seite von Dreiecken, die $s_1$ enthalten und damit länger als $s_1$ und alle verbleibenden kurzen Sehnen. Das heißt, in jedem Folgeschritt können die drei längsten kurzen Sehnen in konstanter Zeit bestimmt werden. Insgesamt benötigt der Algorithmus damit $\mathcal{O}(n)$ Zeit.

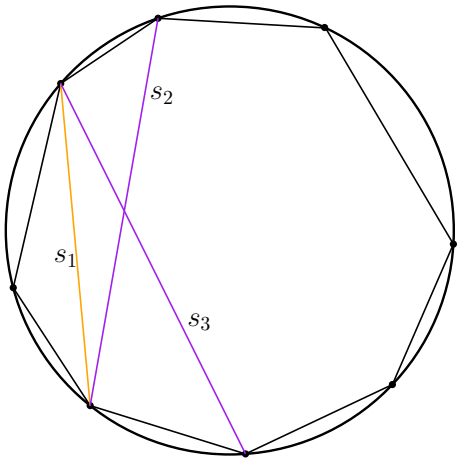


Abbildung 7: Durch Einfügen der kurzen Sehne $s_1$ entstehen zwei neue kurze Sehnen $s_2$ und $s_3$, die beide länger sind als $s_1$