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

Karlsruher Institut für Technologie

# Übungsblatt 3

Abgabe bis 23. Dezember 2020

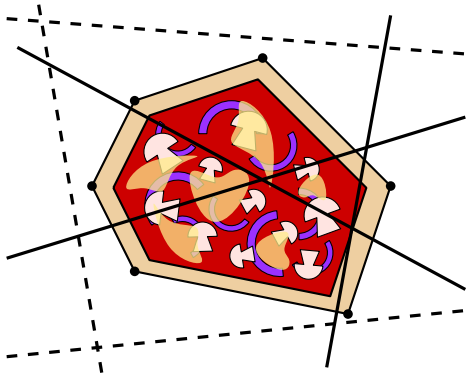
## Aufgabe 1: Pizza-Guillotine

**5 + 5 + 5 = 15 Punkte**

Viele Studenten beschweren sich darüber, dass die örtliche Pizzeria ihre Backwaren ungeschnitten ausliefert. Um die Kundenzufriedenheit zu verbessern, soll das Pizzaschneiden automatisiert werden. Ein erster Prototyp ist eine Pizza-Guillotine. Sie enthält eine unendlich lange Klinge, die prinzipiell frei in der Ebene verschoben und rotiert werden kann und dann auf die Pizza herabfällt. Da der Prototyp noch im Anfangsstadium ist, kann die Ausrichtung der Klinge jedoch nur mit dem Befehl *Reset* geändert werden, welcher die Klinge in eine zufällig gewählte Ausrichtung überführt.

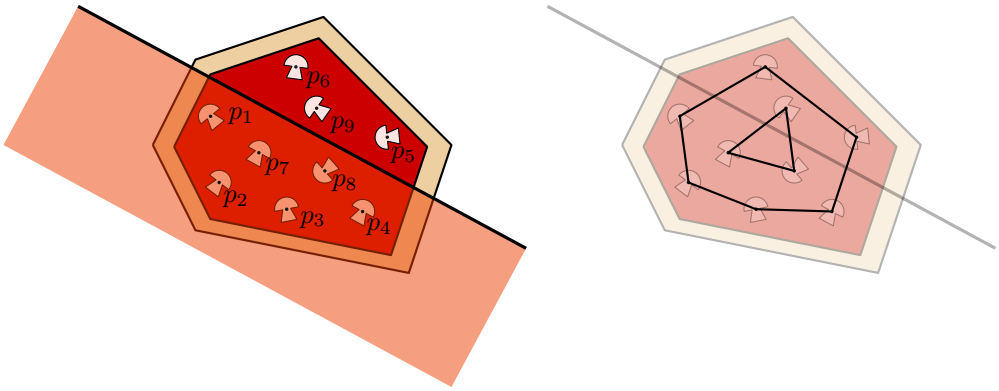
Um trotzdem mit dem Prototypen arbeiten zu können, hat man sich auf folgendes Vorgehen geeinigt. Es wird getestet, ob die aktuelle Ausrichtung der Klinge ungünstig ist oder nicht. Falls ja, wird ein Reset ausgeführt und der Vorgang wiederholt. Andernfalls wird geschnitten. Da dieses Vorgehen extrem ineffizient ist, muss der (womöglich häufig ausgeführte) Test sehr schnell sein!

**Teilaufgabe (a)** Unglücklicherweise kann es passieren, dass die Klinge in ihrer aktuellen Ausrichtung sehr ungünstig oder sogar neben der Pizza landen würde. In diesem Fall muss mittels Reset eine neue Ausrichtung probiert werden. Eine Pizza ist dabei durch ein konvexes Polygon mit $n$ Punkten gegeben. Gebt einen Algorithmus an, der nach $\mathcal{O}(n)$ Vorberechnungszeit jedes Mal in $\mathcal{O}(\log n)$ Zeit bestimmen kann, an welchen Punkten die Klinge in ihrer aktuellen Ausrichtung auf den Rand der Pizza fallen würde.



**Teilaufgabe (b)** Um mehr Zeit zu sparen, sollen nun Ofen und Klinge kombiniert werden, sodass die Pizza während des Schnitts auch gleich gebacken wird! Aus Kostengründen kann allerdings nur eine Seite der Klinge mit der Ofenfunktion ausgestattet werden.

Damit die $n$ Zutaten auf der Pizza, deren Positionen durch die Punkte $p_1, \dots, p_n$ definiert sind, der Hitze nicht mehrfach ausgesetzt werden, muss bei jedem möglichen Schnitt festgestellt werden, welche $k$ Zutaten auf der Ofen-Seite der Klinge liegen.

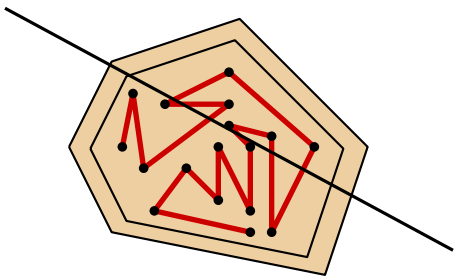



Gebt einen Algorithmus an, der (unter Zuhilfenahme des Algorithmus aus Teilaufgabe (a)) nach $\mathcal{O}(n)$ Vorberechnungszeit jedes Mal in $\mathcal{O}(\log(n) + k)$ Zeit die $k$ Zutaten bestimmen kann, die auf der Ofen-Seite der Klinge liegen.

***Hinweis**: Ihr dürft annehmen, dass ihr als Eingabe die **verschachtelte konvexe Hülle** der Punkte $p_1, \dots, p_n$ bekommt. Diese erhält man, wenn man erst die konvexe Hülle aller Punkte und danach iterativ die konvexe Hülle der Punkte bestimmt, die nicht Teil einer vorherigen konvexen Hülle waren.*

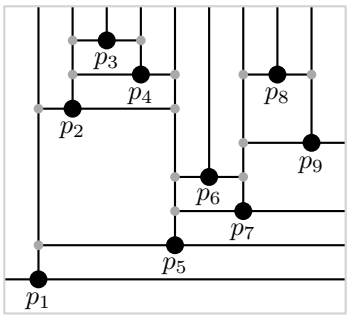
**Teilaufgabe (c)** Ein letzte Baustelle in diesem (sonst optimalen) Vorgehen ist die Lebensdauer der Klinge, welche durch den Säuregehalt der Tomatensoße stark beeinträchtigt wird. Es soll nun ausgenutzt werden, dass die Tomatensoße auf der Pizza "verteilt" wird, indem ein Roboter die Soße in Form eines Polygonzuges, bestehend aus $n$ Punkten, auf den Teig gibt. Wenn die Klinge zu viele Berührungen mit der Soße hätte, muss mittels Reset eine neue Ausrichtung probiert werden.

Gebt einen Algorithmus an, der (unter Zuhilfenahme des Algorithmus aus Teilaufgabe (a)) nach $\mathcal{O}(n \log n)$ Vorberechnungszeit in $\mathcal{O}((k+1)\log(n/(k+1)))$ Zeit die $k$ Schnittpunkte der Klinge mit dem Polygonzug bestimmt.



## Aufgabe 2: Halbe 2D-Bereichsanfragen **5 Punkte**

Gegeben seien die nach $x$-Koordinaten sortierten Punkte $p_1, \ldots, p_n \in \mathbb{R}^2$. Eine Datenstruktur, mit der halbe Bereichsanfragen schnell beantwortet werden können, unterteilt die Ebene in Zellen. Dabei wird von jedem Punkt $p_i$ ein Strahl vertikal nach oben geschossen und anschließend in beiden Richtungen eine horizontale Kante von $p_i$ zum nächsten Strahl eingefügt, dessen Startpunkt eine kleinere $y$-Koordinate hat, als $p_i$ selbst.




Gebt einen Algorithmus an, der in $\mathcal{O}(n)$ Zeit den geometrischen Graphen erstellt, der diese Datenstruktur repräsentiert. Beweist, dass euer Algorithmus die vorgegebene Laufzeitschranke nicht überschreitet.

## Aufgabe 3: Implementierung 2D-Bereichsanfragen **5 Bonuspunkte**

Implementiert ein Programm, welches eine Menge von Punkten $p_1, \ldots, p_n \in \mathbb{R}^2$ einliest und anschließend eine Reihe von Bereichsanfragen ausführt. Zur Ein- und Ausgabe sind dabei folgende Formate vorgesehen.

**Eingabe:** Die Eingabe besteht aus zwei Textdateien. Die erste enthält $n$ Zeilen: eine für jeden der $n$ Punkte. Eine Zeile besteht dabei aus drei, durch ein Tab-Symbol (`\t`) getrennten, ganzen Zahlen, welche die ID $i$ des Punktes $p_i$ und dessen $x$- und $y$-Koordinaten repräsentieren.

Die zweite Datei besteht aus $k$ Zeilen. Eine einzelne Zeile $j$ enthält vier, durch ein Tab-Symbol (`\t`) getrennte, ganze Zahlen $x_j, x'_j, y_j, y'_j$ welche ein zwei-dimensionales Intervall $[x_j, x'_j] \times [y_j, y'_j]$ beschreiben.

**Ausgabe:** Die Ausgabe besteht aus $k$ Zeilen. Eine Zeile $j$ enthält $m_j$, durch ein Tab-Symbol (`\t`) getrennte, Zahlen, welche die $m_j$ IDs der Punkte in aufsteigender Reihenfolge repräsentieren, die in $[x_j, x'_j] \times [y_j, y'_j]$ liegen.

Beispiel Ein- und Ausgaben befinden sich auf der Rückseite.

| Beispiel Eingabe: $p_1 \ldots p_n$ | | |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 2 | 2 |
| 2 | 3 | 2 |
| 3 | 3 | 3 |
| 4 | 2 | 5 |
| 5 | 5 | 4 |
| 6 | 6 | 1 |

| Beispiel Eingabe: Anfragen | | | |
|---|---|---|---|
| 2 | 3 | 2 | 3 |
| 1 | 5 | 3 | 5 |
| 0 | 6 | 0 | 1 |

| Zugehörige Ausgabe | | |
|---|---|---|
| 1 | 2 | 3 |
| 3 | 4 | 5 |
| 0 | 6 | |