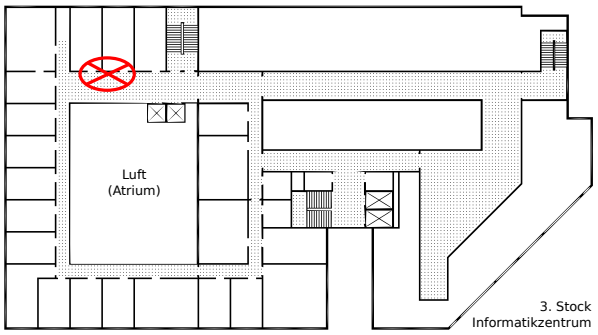
Abteilung Algorithmik **Winter 2018/2019**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Dr. Christian Scheffer
Andreas Haas

Algorithmische Geometrie
Übung 2 vom 11. 11. 2018

Abgabe der Lösungen bis zum Freitag, den 30. 11. 2018 um 11:30 im Hausaufgabenrückgabeschrank.

Bitte die Blätter vorne deutlich mit eigenem Namen sowie Matrikel- und Gruppennummer versehen!



Aufgabe 1 (Jarvis March): Wende Jarvis' March auf der Punktmenge an, die in Abbildung 1 abgebildet ist. Gib hierzu für jede Iteration der REPEAT-Schleife an, welcher neue Eckpunkt der konvexen Hülle gefunden wird. Weiter gib innerhalb jeder Iteration der REPEAT-Schleife nach jeder Aktualisierung von p_{next} an, welcher Punkt p_i der Punkt p_{next} ist. Betrachte für diese Aufgabe folgende Veränderung des Algorithmus Jarvis' March:

- In jeder Iteration der REPEAT − UNTIL-Schleife wird p_{next} initial als der Punkt aus $P \setminus \{p\}$ gewählt, der den kleinsten Index hat.
- In der FOR-Schleife werden die Punkte aus P in der Reihenfolge $\langle p_1, \ldots, p_n \rangle$ abgearbeitet.

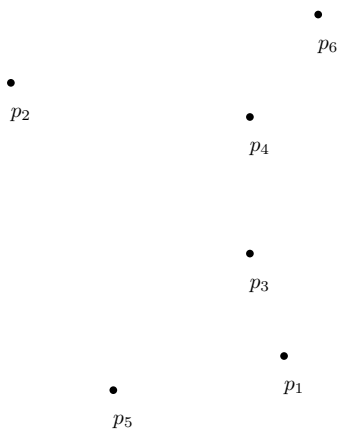


Abbildung 1: Die Punktmenge P für Aufgabe 1.

(5 Punkte)

Aufgabe 2 (Quickhull): In dieser Aufgabe sollen die Details des in Kapitel 2 bearbeiteten Algorithmus QUICKHULL ausgearbeitet werden.

a) Gib an, wie man mit den in der Vorlesung besprochenen elementaren geometrischen Prädikaten das *Pivotelement* h bestimmen kann (verwende nicht explizit die Euklidische Distanzfunktion). Begründe, warum Dein Ansatz korrekt ist.

b) Analysiere die Laufzeit des resultierenden Algorithmus. Gib hierbei je eine Familie von Punktemengen (d.h. eine mit n parametrisierbare Konstruktionsvorschrift zum Erzeugen einer Konfiguration mit n Punkten) an, für die QUICKHULL $\Theta(n \log n)$ Zeit bzw. $\Theta(n^2)$ Zeit benötigt. Begründe dabei jeweils, warum die geforderte Laufzeit auch erreicht wird.

(5 Punkte)

Aufgabe 3 ($O(n)$ Space – Sweep Line): Sei k die Anzahl der Schnittpunkte. Ändere den Sweep-Line Algorithmus `FindIntersections` so, dass der Speicherbedarf $O(n)$ statt $O(n+k)$ ist. Der Speicherbedarf bezeichnet den Speicher, der vom Algorithmus tatsächlich benötigt wird, d.h. bereits ausgegebene Schnittpunkte werden nicht dazu gezählt. Eine kurze Beschreibung deiner Idee/Änderung genügt.

Zeige, dass dein neuer Algorithmus $O(n)$ Speicher benötigt und argumentiere wieso die Laufzeit unverändert bleibt.

Bonusfrage zum Grübeln: Kannst du ein Beispiel finden, bei dem der originale Algorithmus mehr als $O(n)$ Speicher benötigt oder kannst du zeigen, dass es keine solche Eingabe gibt und der Speicherbedarf bereits $O(n)$ war? **(5 Punkte)**

Aufgabe 4 (Visibility Polygon – Angular Sweep): Wir möchten folgendes Problem lösen: Gegeben ein Polygon $\mathcal{P}$ mit n Eckpunkten, möglicherweise mit Löchern, und ein Punkt v im Inneren von $\mathcal{P}$. Berechne das Sichtbarkeitspolygon $\text{Vis}(v)$, d.h., alle Punkte im inneren von $\mathcal{P}$ die von v aus sichtbar sind. $\text{Vis}(v)$ bildet stets ein einfaches sternförmiges Polygon, es reicht daher die Eckpunkte dieses Polygons auszugeben. Ein Beispiel ist in Abbildung 2 gegeben.

Dieses Problem lässt sich mit einem Sweep-Line-Verfahren lösen. Beschreibe das Verfahren und beantworte dabei folgende Fragen:

- Wie sieht die Sweep-Line aus? Tipp: Sie bewegt sich **nicht** von links nach rechts.
- Was sind die Events und in welcher Reihenfolge werden sie bearbeitet?
- Was wird im Status der Sweep-Line gespeichert?
- Welche Änderung im Status hat eine Ausgabe eines neuen Eckpunktes von $\text{Vis}(v)$ zur Folge.
- Was ist die Laufzeit?

Bonus Die untere Laufzeit-Schranke zur Berechnung von $\text{Vis}(v)$ ist $\Omega(n \log n)$ für Polygone mit Löchern. Kannst du dies beweisen? Bemerkung: Diese Schranke gilt nicht

für einfache Polygone, d.h., Polygone ohne Löcher. In diesen lässt sich $\mathrm{Vis}(v)$ in $O(n)$ berechnen.

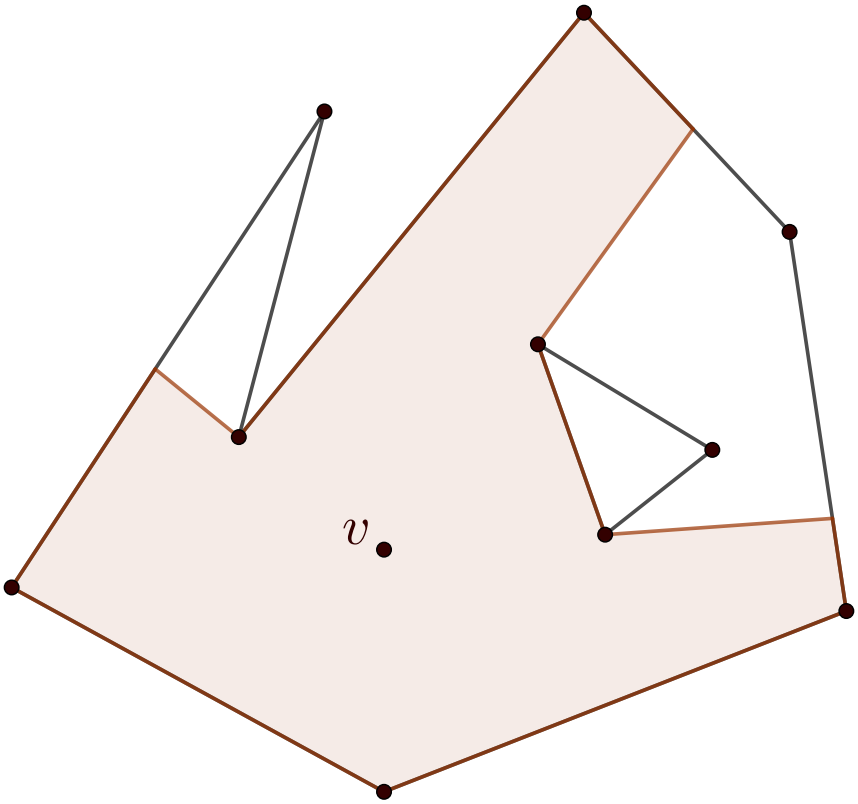


Abbildung 2: Ein Polygon mit einem Loch. Das Sichtbarkeitspolygon von v ist farblich hervorgehoben.

(5 + 5 Bonus Punkte)