Lehrgebiet Kooperative Systeme

Die konvexe Hülle und die Winkelhülle

Aufgabenbeschreibung für das Programmierpraktikum 1580/1582/1584 im Sommersemester 2015

mathematik und informatik

FernUniversität in Hagen

Inhalt

Vorwort

Herzlich willkommen zum *Programmierpraktikum* 1580/1582/1584 im Sommersemester 2015!

Ziel dieses Praktikums ist es, eine größere Programmieraufgabe selbständig zu lösen. Ihre konkrete Aufgabe besteht darin, ein Programm für zwei anschauliche Optimierungsaufgaben zu erstellen, wobei das Programm möglichst *effizient* arbeiten soll, damit es auch größere Eingaben in annehmbarer Zeit bewältigen kann. Um Programme dieser Art schreiben zu können, benötigen Sie effiziente Datenstrukturen und Algorithmen, die Sie teilweise schon aus den Grundkursen kennen und zum anderen Teil durch diesen Text kennen lernen werden.

Ein gutes Gelingen und viel Freude an der Bearbeitung dieser Aufgabe wünschen Ihnen die Betreuer!

Betreuung

Newsgruppen unbedingt nutzen!

Für das Praktikum haben wir auf dem News-Server `feunews.fernuni-hagen.de` zwei *Newsgruppen* eingerichtet. Bei allen Fragen und Problemen rund um das Praktikum sind diese das nützlichste und effizienteste Kommunikationsmedium für alle Teilnehmer, hier zumindest immer mitzulesen ist Pflicht!

Die Newsgruppe

Organisatorisches

```
feu.informatik.kurs.1580+82+84.betreuung.ss
```

ist das Forum für Ankündigungen und Organisatorisches im Praktikum. Darin werden wir aktuelle organisatorische Informationen bekanntgeben. Hier *müssen* Sie regelmäßig mitlesen, um nichts zu verpassen.

Die Gruppe

Diskussionen

```
feu.informatik.kurs.1580+82+84.diskussion.ss
```

ist das Diskussionsforum für alle Teilnehmer. Hier wird unter den Teilnehmern und mit den Betreuern über die Aufgabe, Lösungsmöglichkeiten, die Programmierumgebung usw. diskutiert.

Bitte benutzen Sie zur Kommunikation möglichst diese Foren, damit immer alle anderen Teilnehmer mitlesen und von den Antworten profitieren können.

Für Angelegenheiten in Zusammenhang mit Ihrer persönlichen Teilnahme wenden Sie sich bitte an:

```
Andrea.Frank@fernuni-hagen.de
```

Webseite

Die Webseite zum Praktikum ist:

```
http://www.fernuni-hagen.de/mathinf/studium/lehre/praktika/
                                    programmierpraktikum/2015_ss.shtml
```

Wettbewerb

Die abgegebenen Programme werden von den Betreuern über das vorgeschriebene Test-Interface auf Korrektheit überprüft. Dazu wird noch die Programm-Laufzeit für eine ziemlich große Eingabe ermittelt. Wer das Programm abgibt, das in der kürzesten Zeit das korrekte Ergebnis zurückgibt, erhält zur Belohnung ein Informatik-Fachbuch.

Preis zu gewinnen

Zur Verbesserung der Laufzeit ist es möglich, *alternative Algorithmen* (entweder selbst erfundene oder aus anderen Quellen ermittelte) als die in diesem Text vorgestellten zu verwenden, aber natürlich nur selbst programmierte. Man sollte allerdings beachten, dass die hier vorgestellten Verfahren schon wegen ihrer Einfachheit und Robustheit ausgesucht wurden.

alternative Algorithmen

Zeitlicher Ablauf

Empfohlen wird ein zügiger Start in die Praktikumsarbeit gleich nach Erhalt dieser Aufgabe. Insbesondere die Unklarheiten sollten möglichst bald durch Inanspruchnahme der Betreuung beseitigt werden. Kurz vor den Abgabeterminen wird es erfahrungsgemäß auch mit der Betreuungsmöglichkeit sehr eng. Die angegebenen *Termine* sind unbedingt einzuhalten. Eine vorzeitige Abgabe ist zu jeder Zeit möglich.

Termine einhalten!

Arbeitsbeginn: 30. März 2015
Abgabe des ersten Teils spätestens bis: 15. Juni 2015
Abgabe des zweiten Teils spätestens bis: 16. August 2015
Nachbesserungsfrist für den zweiten Teil bis zum: 6. September 2015
Bekanntgabe des Wettbewerbs-Siegers: ca. 15. September 2015
Präsenztag in Hagen: Freitag, 18. September 2015
Zusendung der Leistungsnachweise: ca. 30. September 2015

Kontrollen

Ein ernst gemeinter Appell an alle Teilnehmerinnen und Teilnehmer: Seien Sie fair und kein Plagiator, nutzen Sie auch keine Ghostwriter, die eigene Leistung zählt! Nur jeweils selbständig erstellte Programme dürfen abgegeben werden, Übernahmen aus fremden Quellen werden als Täuschung gewertet und führen zum sofortigen Nichtbestehen bzw. bei nachträglichem Bekanntwerden mindestens zur Aberkennung der Leistung.

nur eigene Programme einreichen!

Die abgegebenen Programme werden intensiv kontrolliert. So werden zum Beispiel alle abgegebenen Programme automatisch untereinander und mit verschiedenen anderen Quellen verglichen und auf Ähnlichkeiten untersucht.

automatische Kontrollen

Programmierumgebung

Computer

Für das Praktikum benötigen Sie einen *Computer* mit Internetanbindung. Als Betriebssysteme sind auf jeden Fall Windows, Linux (diverse Distributionen) und OS X (Mac) geeignet.

Java

Die zu verwendende Programmiersprache ist *Java*, Vorkenntnisse zur Java-Programmierung sind unbedingt erforderlich. Wenn Sie noch kein Java auf Ihrem Computer installiert haben, dann installieren Sie bitte von

`http://www.oracle.com/technetwork/java/javase/downloads/`

JDK

die aktuelle Version des zu Ihrem System passenden *Java SE Development Kits (JDK)*.

Sie dürfen nur eigene Klassen und solche aus der Java-Standard-Bibliothek verwenden. Die Nutzung anderer fremder Klassenbibliotheken ist nicht zulässig.

Eclipse

Die zu verwendende Programmierumgebung ist *Eclipse*, das man sich von der Webseite `www.eclipse.org` herunterladen kann. Empfohlen wird, die aktuelle Version (4.4 Luna) von *Eclipse IDE for Java Developers* zu benutzen.

Subversion (SVN)

Versionskontrolle mit SVN

Verwendet wird außerdem die *Versionskontrolle mit SVN* auf einem zentralen SVN-Server. Größere Programmierprojekte sind heute ohne Versionskontrolle kaum vorstellbar und sollten jedem Informatiker geläufig sein.

Die normale Arbeitsweise mit Versionskontrolle sollte so ablaufen, dass Sie regelmäßig, z. B. täglich, Ihre Änderungen, versehen mit einem sinnvollen Kommentar, mittels des `Commit`-Befehls zum SVN-Server hochladen. Damit haben Sie zunächst eine Sicherung Ihrer Dateien auf einem externen System, zum anderen stehen aber auch noch alle früheren Versionen zum Vergleich und zur Nachvollziehbarkeit zur Verfügung. Außerdem können Sie mit Hilfe des `Update`-Befehls mehrere eigene Arbeitsplätze miteinander synchronisieren und Ihre Software in der Gruppe (hier den Betreuern) bereitstellen.

Die Dateien, die Sie in Ihrem SVN-Bereich verwalten, sind nur für Sie selbst und für die Praktikumsbetreuer lesbar. Angeschaut werden aber normalerweise nur die von Ihnen abgegebenen (mit `Tag`[1] versehenen) Programmversionen. Ein `Tag` steht also für einen bestimmten Versionsstand, der unter einem gewählten Namen eingefroren wird und so einfach zu nutzen ist, während Sie schon weiter arbeiten und auch weitere Versionen erzeugen.

[1] tag (engl.): Etikett, Kennzeichen

Subversive-Plugin

Um innerhalb von Eclipse SVN benutzen zu können, sollten Sie das *Subversive-Plugin* installieren. Dieses finden Sie in Eclipse unter dem Menü-Befehl *Help→Install New Software* auf der (`Work with:`) *Luna*-Site in der Abteilung *Collaboration* unter dem Namen *Subversive SVN Team Provider*. Danach müssen Sie noch einen sogenannten *SVN Connector* installieren, dazu werden Sie spätestens bei der ersten Benutzung automatisch aufgefordert, wählen Sie hier möglichst den aktuellsten (SVN 1.8).

SVN-Funktionen

Die meisten *SVN-Funktionen* von Eclipse+Subversive finden sich in den mit der rechten Maustaste erreichbaren Untermenüs `Team` (`Share Project`, `Commit`, `Update`, `Tag`, ...) und `Compare with...`

Java-Projekt

Zu Beginn sollten Sie ein neues *Java-Projekt* mit dem Namen `q<Matr-Nr>` erzeugen (mit `File->New->Java Project`) und mit Ihrem SVN-Bereich verknüpfen durch `Team->Share Project->SVN`:

```
URL:
https://propra1.fernuni-hagen.de/propra_ss15/students/q<Matr-Nr>
User: q<Matr-Nr>
Password: <Ihr FernUni-LDAP-Kennwort>
```

Klicken Sie nun auf `Next` und achten Sie bitte dann noch darauf, im `Specify...`-Fenster den `Advanced Mode` auszuwählen (nicht `Simple Mode`!).

Auch die Projektdatei (`.project`) von Eclipse wird mit eingecheckt (passiert normalerweise automatisch).

Nun können Sie direkt aus der Programmierumgebung den Stand Ihrer Arbeit sichern, so oft Sie wollen, (mit `Team->Commit`, Versionenkommentare zum eigenen Nutzen nicht vergessen) und später die fertigen Programmversionen abgeben (mit `Team->Tag`).

zweites Projekt

Für das vorgeschriebene Java-Interface und zum Testen benötigen Sie unbedingt noch unser vorbereitetes Modul als zweites Projekt in Ihrem Eclipse-Workspace aus einem zweiten SVN-Repository. Dazu wählen Sie `New->Project...SVN->Project from SVN` und dann `Create a new repository location` mit folgenden Angaben:

```
URL: https://propra1.fernuni-hagen.de/propra_ss15/Tester
User: q<Matr-Nr>
Password: <Ihr FernUni-LDAP-Kennwort>
```

Über `Find projects in the children...` finden Sie in diesem Repository das Projekt `Tester` und checken es aus.

Verknüpfen Sie die beiden Projekte, indem Sie die *Properties* von Ihrem Projekt `q<Matr-Nr>` aufrufen, darin den *Java Build Path* und *Libraries*, dann *Add JARs* anklicken und die Datei `Tester/ProPraTester.jar` aus dem Tester-Projekt auswählen. Überprüfen Sie außerdem noch, ob zu dieser `jar`-Datei das `doc`-Verzeichnis innerhalb der Datei als *Javadoc Location* konfiguriert ist, dann kann Eclipse die passende Dokumentation im jeweiligen Kontext anzeigen.

Testmöglichkeiten

Das Tester-Projekt sollten Sie regelmäßig und insbesondere nach einer entsprechenden Aufforderung in der Newsgruppe aktualisieren (mit `Team->Update`), um zusätzliche *Testmöglichkeiten* zu bekommen.

fertige Programme abgeben

Das *fertige Programm* zu Teil 1 kennzeichnen Sie bitte in Ihrem SVN-Bereich mit dem Tag `Teil1`. Sie können also in Ruhe an Teil 2 weiterarbeiten und weitere Versionen generieren, ohne die abgegebene Version zu verändern. Das fertige Programm zu Teil 2 kennzeichnen Sie später bitte entsprechend mit dem Tag `Teil2`.

Test-Interface

Interface

In `Tester/doc/de/feu/propra15/interfaces` ist die Beschreibung des zu implementierenden *Interfaces* `IHullCalculator` zu finden.

Ihr Programm sollte nun folgendermaßen arbeiten. Wird ihm kein Parameter übergeben, soll die normale Oberfläche mit der graphischen Benutzeroberfläche (siehe weiter unten) erscheinen. Wird dem Programm der Parameter `-t` übergeben, sollte die Testklasse der Bibliothek `ProPraTester.jar` verwendet werden, um die Rückgabewerte zu testen. Um dies zu erreichen, könnten die ersten Zeilen der `main`-Methode wie folgt aussehen:

```
if (args.length > 0 && "-t".equals(args[0])) {
   IHullCalculator calculator = new HullCalculator();
   Tester tester = new Tester(args, calculator);
   System.out.println(tester.test());
}
else // Benutzeroberfläche
```

Dabei ist der Name `HullCalculator` durch den Namen Ihrer Implementierung des Interfaces `IHullCalculator` zu ersetzen.

Teil 1

Die konvexe Hülle

Stellen wir uns eine Menge von einzelnen Punkten in der Ebene anschaulich als *Nägel* vor, die aus der Ebene herausragen. Nun legen wir ein *Gummiband* um diese Nägel herum und lassen es sich zusammenziehen, so dass es so kurz wie möglich wird. Diese Figur, die das Gummiband annimmt, ist der kürzeste Rundweg, der alle Punkte einschließt. Wir nennen ihn die *konvexe Hülle* der Punktmenge; siehe Abbildung 1.1. Einige der Nägel werden vom Gummiband berührt, so dass es dort einen Knick macht; diese Punkte nennen wir *Extrempunkte*.

Nägel

Gummiband

konvexe Hülle

Extrempunkte

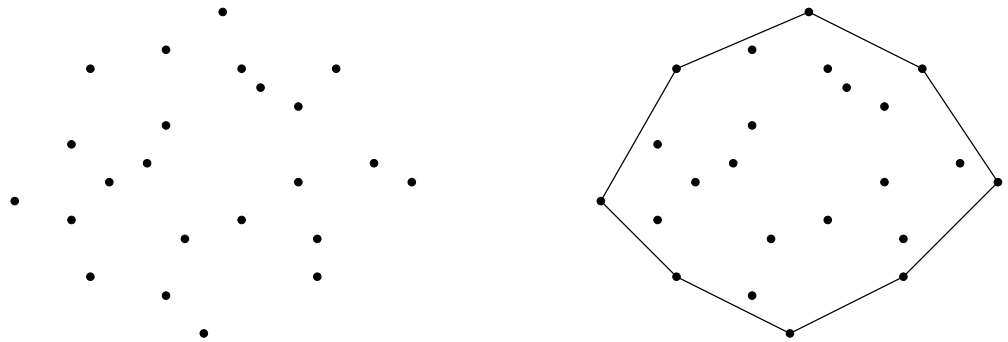

Abbildung 1.1: Eine Menge von Punkten (links) und dieselben Punkte mit ihrer konvexen Hülle (rechts).

Zur *Berechnung der konvexen Hülle* $KH(S)$ einer Punktmenge S von n Punkten müssen alle Extrempunkte gefunden und in der Reihenfolge entlang des Randes (des Gummibands) ausgegeben werden; dies ist die erste Aufgabe im Praktikum.

Berechnung der konvexen Hülle

1.1 Das Konturpolygon-Verfahren

einfaches und schnelles Verfahren

Genauere und formal korrekte Definitionen verschieben wir auf Abschnitt 1.2, zunächst wollen wir aus der großen Auswahl von bekannten Algorithmen zur Konstruktion der konvexen Hülle einer ebenen Punktmenge ein besonders einfaches, leicht verständliches und *worst-case*-optimales Verfahren beschreiben, das *Konturpolygon*-Verfahren. Hier und im Folgenden verzichten wir auf Beweise und beschränken uns darauf, die nötigen Fakten kurz und anschaulich zusammenzutragen. Wer sich für die Hintergründe und weitere interessante Probleme dieser Art interessiert, sei auf den FernUni-Kurs 1840 *Algorithmische Geometrie* [1] verwiesen.

zwei Phasen

Das Verfahren läuft in *zwei Phasen* ab:

Konturpolygon

- Zuerst wird das sogenannte *Konturpolygon* von S konstruiert, dessen Ecken zu S gehören. Das Konturpolygon stellt bereits einen Rundweg dar, der jeden Punkt von S einschließt; dies ist aber nicht unbedingt schon der kürzeste Rundweg und daher nicht unbedingt konvex.

Bereinigung

- Im zweiten Schritt wird die konvexe Hülle des Konturpolygons gebildet, siehe Abbildung 1.2, dazu müssen nur noch einige Ecken weggelassen werden, deswegen nennen wir dies auch die *Bereinigungsphase*.

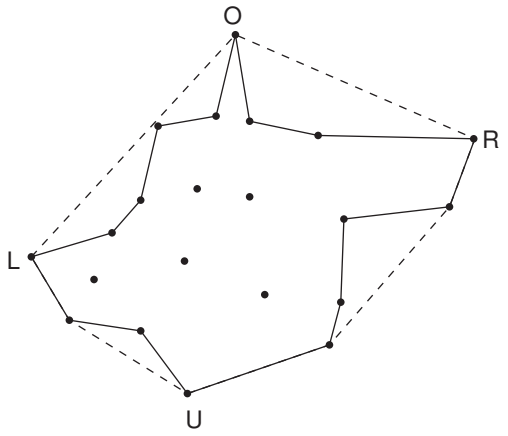

Abbildung 1.2: Zur Konstruktion der konvexen Hülle der Punktmenge wird zunächst ihr Konturpolygon bestimmt.

Das Konturpolygon ist von besonders einfacher Gestalt. Anschaulich entsteht es folgendermaßen: Wir stellen uns wieder vor, dass die Punkte in S als Nägel aus der Ebene herausragen. Links von S liegt ein senkrechter Wollfaden. Er wird nun von einem Gebläse nach rechts gegen die Punkte getrieben. Zuerst trifft er dabei auf den linken Extrempunkt L von S, d. h. auf den Punkt mit minimaler X-Koordinate. Dann schmiegt sich der Faden den Nägeln in der in Abbildung 1.2 gezeigten Weise an, bis er den oberen Extrempunkt O und den unteren Extrempunkt U erreicht hat; dort wird er abgeschnitten. Dasselbe Vorgehen wird von rechts wiederholt. Man nennt die beiden Fäden in ihrer

endgültigen Lage die *linke* und die *rechte Kontur der Punktmenge* S, beide zusammen das Konturpolygon. Übrigens sind die Extrempunkte L, R, U, O auf jeden Fall schon Ecken der konvexen Hülle.

linke Kontur
rechte Kontur

Abbildung 1.3 zeigt, dass sich die beiden Konturen außer an ihren Endpunkten O und U auch an anderen Stellen berühren und dass manche Extrempunke zusammenfallen können; das stört uns aber nicht.

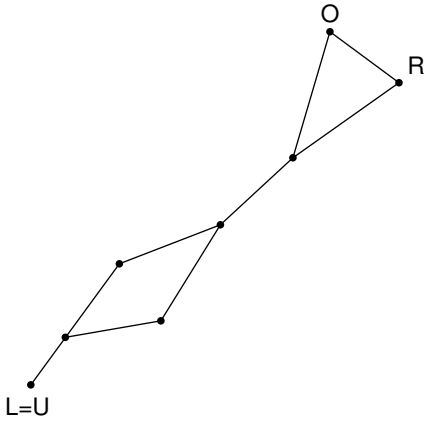

Abbildung 1.3: Ein Konturpolygon mit flachen Stellen.

Das Konturpolygon wird folgendermaßen bestimmt. Zuerst sortiert man die Punktmenge nach aufsteigenden X-Koordinaten, bei Gleichheit nach Y-Koordinaten (lexikographische Sortierung). In dieser Reihenfolge werden die Punkte durchlaufen, so als ob eine senkrechte Gerade (*sweep line*) von links nach rechts über die Ebene bewegt wird und die Punkte der Reihe nach aufsammelt, und merkt sich die Punkte *MinYSoFar* und *MaxYSoFar* mit minimaler und mit maximaler Y-Koordinate, die bisher angetroffen wurden.

sortieren

sweep line

Am Anfang stimmen beide Punkte mit dem linken Extrempunkt L überein, am Ende ist $MaxYSoFar = O$ und $MinYSoFar = U$. Wann immer ein neuer Punkt *MaxYSoFar* entdeckt wird, verbindet man ihn durch eine Kante mit seinem Vorgänger; dasselbe geschieht mit *MinYSoFar*.

Auf diese Weise lässt sich die linke Kontur berechnen. Mit der rechten Seite verfährt man analog, durch einen *sweep* von rechts nach links.

Aus dem Konturpolygon können wir die konvexe Hülle leicht bestimmen. Dazu nehmen wir uns jedes der Fadenstücke zwischen den vier Extrempunkten vor und ziehen es stramm. Es genügt, dieses Verfahren am Beispiel des oberen Teils der linken Kontur zwischen L und O zu demonstrieren; siehe Abbildung 1.4.

Die Ecken dieses Randstücks seien von links nach rechts mit $L = p_1, p_2, \ldots$ bezeichnet. Wir durchlaufen sie in dieser Reihenfolge. Wann immer wir an eine hereinragende Ecke p_j kommen, laufen wir so weit zurück, bis wir auf p_1 stoßen oder vorher auf einen Eckpunkt p_i, für den p_{j+1} rechts von der Geraden durch $p_{i-1}p_i$ liegt; vergleiche Abbildung 1.4. Das gesamte Randstück von p_1 bzw. p_i bis p_{j+1} wird durch das Liniensegment zwischen diesen Punkten ersetzt. Damit ist die spitze Ecke p_j verschwunden, und bis zum Punkt p_{j+1} haben wir eine konvexe polygonale Kette mit Ecken in S erzeugt, die die alte Kontur von oben

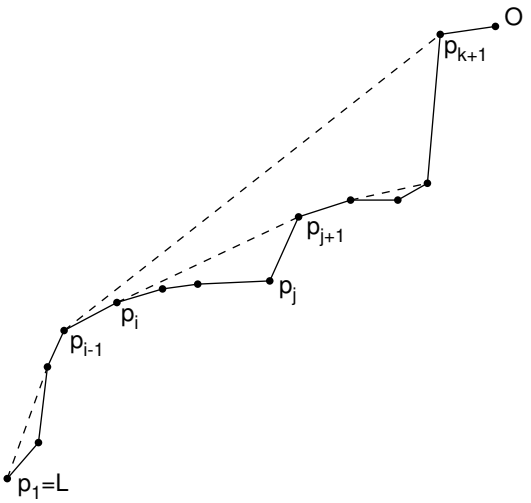

Abbildung 1.4: Berechnung der konvexen Hülle durch Bereinigung des oberen Teils der linken Kontur.

Links-Rechts-Test

umschließt. Bei jedem Test benutzen wir den sogenannten *Links-Rechts-Test*, der die Frage beantwortet, ob ein Punkt links oder rechts einer Geraden durch zwei Punkte liegt; dieser Test wird in Abschnitt 1.3 genauer beschrieben.

Im Beispiel in Abbildung 1.4 wird das Liniensegment $p_i p_{j+1}$ wieder verschwinden, wenn später $p_{i-1}p_{k+1}$ eingeführt wird.

Laufzeit

optimal

worst case

Abschließend (nur für die, die es wirklich wissen wollen) können wir über die *Laufzeit* des Konturpolygonverfahrens sagen, dass es $O(n \log n)$ Zeit benötigt, und zwar schon durch das Sortieren im ersten Schritt, nach dem Sortieren nur noch $O(n)$ Zeit. Dies ist (asymptotisch, für $n \to \infty$) *optimal* im schlimmsten Fall (*worst case*), was aber natürlich nicht garantiert, dass es in konkreten Fällen nicht noch schneller geht.

1.2 Begriffe

Jetzt wollen wir aber doch ein paar Begriffe genau erklären, damit eindeutig und nicht nur anschaulich klar ist, wovon geredet wird.

Ebene

Alles in diesem Text spielt sich im zweidimensionalen Raum ab, also dem $\mathbb{R}^2$ bzw. der *Ebene*, höhere Dimensionen kommen nicht vor.

Punkt

Ein *Punkt* $A = (A_x, A_y)$ ist ein Paar bestehend aus einer X- und einer Y-Koordinate aus den reellen Zahlen ($\mathbb{R}$). Im Programm werden wir nur Punkte mit ganzzahligen Koordinaten behandeln.

Gerade

Eine *Gerade* durch zwei verschiedene Punkte A und B ist anschaulich die gerade Verbindung zwischen den beiden Punkten und deren gerade Verlängerung in beide Richtungen. Mathematisch exakter ausgedrückt ist es die Menge

$$\{\, tA + (1-t)B;\ t \in \mathbb{R} \,\}$$

von Punkten in der Ebene, die Gerade ist natürlich durch die beiden Punkte eindeutig bestimmt.

Drei Punkte sind *kollinear*, wenn Sie auf derselben Geraden liegen. kollinear

Ein *Polygon* ist ein Vieleck (Dreieck, Viereck, Fünfeck, ...), also eine geschlossene Kurve in der Ebene, die nur aus endlich vielen geraden Stücken besteht. Da, wo zwei solche Stücke zusammenstoßen, hat das Polygon seine *Ecken*, die geraden Stücke zwischen den Ecken heißen *Kanten*. Eindeutig bestimmt ist ein Polygon durch die Folge seiner Ecken. Wir betrachen meist nur sogenannte *einfache Polygone*, d. h. zwei nicht benachbarte Kanten schneiden sich nicht, siehe Abbildung 1.5. Polygon, Ecke, Kante, einfaches Polygon

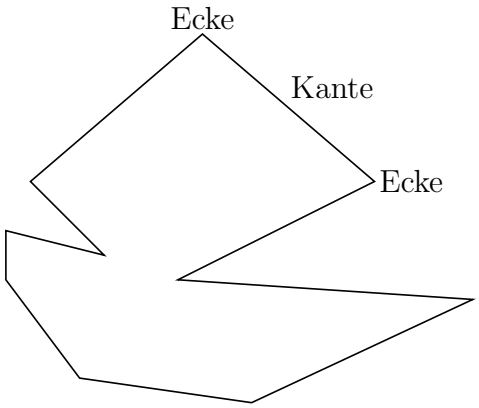

Abbildung 1.5: Ein einfaches Polygon.

Ein *Dreieck* ist ein Polygon mit drei Ecken und wird kurz $\triangle(A, B, C)$ geschrieben, wenn A, B und C die Ecken sind. Dreieck, $\triangle(A, B, C)$

Eine Menge von Punkten in der Ebene heißt *konvex*, wenn für je zwei Punkte der Menge die gerade Verbindung ganz in der Menge liegt. konvex

Ein einfaches Polygon heißt konvex, wenn sein Inneres (das eingeschlossene Gebiet) konvex ist. Ein Dreieck z. B. ist immer konvex, wie auch jede Kreisfläche. konvexes Polygon

Die *konvexe Hülle* $KH(S)$ einer Punktmenge S von n Punkten ist die kleinste konvexe Menge, die S enthält; $KH(S)$ ist immer ein *konvexes Polygon* mit höchstens n Ecken und stellt auch den kürzesten Rundweg dar, der S einschließt. konvexe Hülle, konvexes Polygon

1.3 Links-Rechts-Test

Gegeben sind drei Punkte A, B, C in der Ebene. Man will wissen, ob C links oder rechts von der Geraden g durch A und B liegt, die in Richtung von A nach B orientiert (gerichtet) ist, siehe Abbildung 1.6.

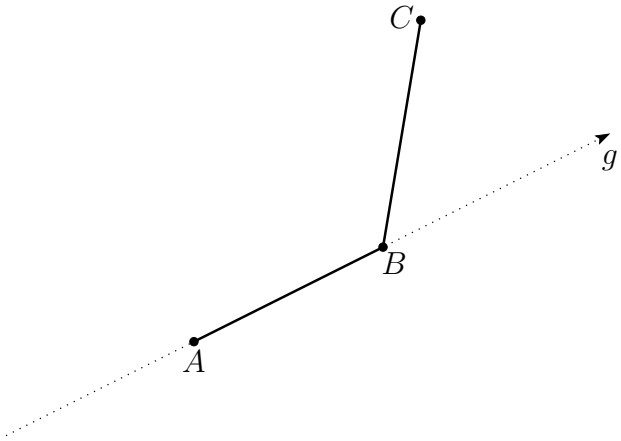

Abbildung 1.6: Hier liegt Punkt C links von der Geraden g durch A und B, die Determinante ist positiv.

Determinante

Dafür benötigen wir die Formel für die *Determinante*:

$$\det(A,B,C) = \begin{vmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ C_x & C_y & 1 \end{vmatrix}$$
$$= (C_x - A_x)(C_y + A_y) + (B_x - C_x)(B_y + C_y) + (A_x - B_x)(A_y + B_y)$$

Hat die Determinante einen positiven Wert, so liegt C links von der orientierten Geraden g, bei einem negativen Wert rechts. Der Wert ist genau dann gleich 0, wenn C auf g liegt, also genau dann, wenn die drei Punkte kollinear sind.

Dreiecksfläche mit Vorzeichen

Wir bezeichnen diese Formel auch als *Dreiecksfläche mit Vorzeichen*, denn sie ergibt den doppelten Flächeninhalt des Dreiecks $\triangle(A,B,C)$, falls A, B, C in dieser Reihenfolge entgegen dem Uhrzeigersinn auf dem Dreiecksrand liegen, bzw. den negativen doppelten Flächeninhalt im Uhrzeigersinn.

ganzzahlige Operationen

große ganze Zahlen

Wichtig ist, dass dieser Test nur Rechenoperationen im Bereich der ganzen Zahlen benutzt, wenn die eingegebenen Koordinaten selbst auch ganze Zahlen sind. Aber Vorsicht: Hier treten schon recht *große ganze Zahlen* auf; bei Verwendung von `int`-Koordinaten sollte die Berechnung der Determinante (inklusive der Zwischenrechnungen!) mindestens als `long` erfolgen. Ein Überlauf von `int`-Zahlen wird von Java nicht bemerkt!

1.4 Anforderungen an das Programm, Teil 1

Java-Application

SVN-Projekt

Packages

Präfix

Zur Lösung dieser Praktikumsaufgabe schreiben Sie bitte eine *Java-Application*. Das *Projekt* soll den Namen `q<Matrikelnummer>` haben und mit SVN unter diesem Namen verwaltet werden, siehe auch den Abschnitt über SVN ab Seite 6. Bitte organisieren Sie Ihr Projekt in *Packages* und verwenden nicht das namenlose sogenannte Default-Package. Für alle eigenen Java-Packages sollen bitte Namen mit dem *Präfix*

```
de.feu.propra15.q<Matrikelnummer>.
```

verwendet werden, so können die Packages immer eindeutig zugeordnet werden.

`main`-Methode

Die Java-Klasse mit der eigentlichen `main`-Methode soll

```
q<Matrikelnummer>Main
```

heißen, das macht es bei der Betreuung und der Korrektur viel einfacher, Ihr Programm zu starten. Sie können z. B. für Tests weitere Klassen mit `main`-Methoden anlegen, um das Programm oder Programmteile anders zu starten, aber das eigentliche Programm soll den genannten festen Namen verwenden.

Test-Interface

Die Application implementiert das vorgegebene *Test-Interface*, siehe Seite 8, und verfügt über folgende Funktionen.

Hauptfenster

Kopfzeile

Beim normalen Programmstart erscheint das *Hauptfenster* Ihrer GUI-Anwendung, dessen *Kopfzeile* neben einem Programmnamen auch Ihren Namen und Ihre Matrikelnummer anzeigen soll.

Zeichenfläche

Punkte editieren

Der Benutzer sieht zunächst eine leere *Zeichenfläche* und kann durch Mausklicks nach Belieben Punkte (der Punktmenge S) setzen, löschen und verschieben. Beim Löschen und Verschieben sollen Mausklicks in der Nähe eines Punktes ausreichen.

Bildschirm-koordinaten

Die Eingabekoordinaten sind vom Typ `int`, es können also die *Bildschirmkoordinaten* von Mausklicks ohne jede Umrechnung direkt übernommen werden. Bildschirmkoordinaten sind allerdings üblicherweise in der Vertikalen von oben nach unten orientiert, so dass sich auch die Rollen von Links und Rechts vertauschen.

keine Rechen-ungenauigkeiten

Alle Berechnungen benutzen ganze Zahlen, so dass das Ergebnis *keine Rechenungenauigkeiten* enthalten kann.

konvexe Hülle

dynamische Anzeige

Das Programm berechnet für jede Konstellation sofort die *konvexe Hülle* und zeigt sie an. Das gilt auch für das Verschieben: Während mit der Maus ein Punkt verschoben wird, geht die Hülle immer mit, falls sich daran durch die Verschiebung etwas ändert.

zufällige Eingaben

Das Programm kann *zufällige Eingaben* von 10, 50, 100, 500 und 1000 Punkten erzeugen, die im Bereich der Zeichenfläche liegen sollen, diese Punkte sollen zur aktuellen Menge hinzugefügt werden.

Dateien einlesen und abspeichern
File Chooser

Das Programm kann Punktmengen aus *Dateien einlesen und abspeichern*. Dabei sollten die üblichen Menübefehle `Datei->Neu`, `->Öffnen` `->Speichern` und `->Speichern unter...` sowie Dateiauswahldialoge (*File Chooser* oder entsprechendes) verwendet werden. Die Daten werden im Textformat abgelegt, wobei in jeder Zeile ein Punkt mit durch Leerzeichen getrennten ganzzahligen X- und Y-Koordinaten steht. Zum Beispiel bedeutet

```
123 453
237 135
42 357
```

die Punktmenge $\{(123, 453), (237, 135), (42, 357)\}$. Alle Zeilen, die sich nicht in diesem Format einlesen lassen, werden ignoriert (Kommentarzeilen).

selbst programmieren
Java-Library

Der Algorithmus zur Berechnung der konvexen Hülle (Konturpolygon-Verfahren oder auch ein anderes nach Wahl) wird selbst programmiert. Für das Sortieren können allerdings fertige Funktionen aus der *Java-Library* benutzt werden. Es wird kein Programmcode aus anderen Quellen verwendet.

Bedienungsanleitung

Das Programm enthält eine *Bedienungsanleitung*, die über ein `Hilfe`-Menü erreichbar ist.

Dokumentation mit Javadoc

Überlegen Sie sich einen einheitlichen Kommentarkopf mit Javadoc-Elementen, den Sie vor jeder Klasse und Methode einfügen, die kommentiert werden soll. In diesem Kommentarkopf beschreiben Sie den Zweck der verwendeten Parameter, die Aufgabe, die von der Klasse bzw. Methode erfüllt wird, und die Einordnung der Klasse/Methode in den Gesamtkontext des Programmes. Benutzen Sie das Javadoc-Werkzeug, um die Parameter und Rückgabewerte auszuzeichnen. Halten Sie die Beschreibung knapp, zwei bis drei prägnante Sätze zur Aufgabe und Einordnung der Klasse bzw. Methode sollten ausreichen.

Eclipse unterstützt Javadoc

Nutzen Sie die Unterstützung von *Eclipse* für Javadoc, das macht es für Sie bequemer. Am Ende muss mittels des Javadoc-Systems eine HTML-Dokumentation des Programms erzeugt werden können, die so ausführlich ist, dass auf einen separaten Text als Programmdokumentation verzichtet werden kann.

(Teil 2 in Kürze)

Literatur

[1] R. Klein, C. Icking. *Algorithmische Geometrie*. Kurs 1840 der FernUniversität in Hagen, 2015.

[2] F. Hoffmann, C. Icking, R. Klein, K. Kriegel. The polygon exploration problem. *SIAM J. Comput.*, 31:577–600, 2001.

[3] F. Hoffmann, C. Icking, R. Klein, K. Kriegel. The polygon exploration problem II: The angle hull. Technical Report 245, FernUniversität Hagen, Department of Computer Science, Germany, 1998.