

Algorithmische Geometrie

1	Konvexe Hüllen	1
1.1	Konstruktion einer konvexen Hülle einer Punktmenge im \mathbb{R}^2	1
1.1.1	Definition (<i>konvexe Menge/ konvexe Hülle</i>)	1
1.1.2	Satz	1
1.1.3	Korollar	1
1.1.4	Beispiel	1
1.1.5	Wiederholung	1
1.2	Algorithmus I: Gift Wrapping	2
1.2.1	Idee	2
1.2.2	Korrektheit	2
1.2.3	Laufzeit	2
1.2.4	Satz	2
1.2.5	Bemerkung	2
1.2.6	Details der Implementierung	2
1.2.6.1	Satz	3
1.2.6.2	Definition (<i>orientation(a,b,c)</i>)	3
1.2.7	Übertragung auf höhere Dimensionen	3
1.2.7.1	Satz	3
1.3	Algorithmus II: Graham's Scan	4
1.3.1	Idee	4
1.3.2	Algorithmus	4
1.3.3	Beispiel	5
1.3.4	Korrektheit	5
1.3.5	Laufzeit	6
1.3.6	Satz	7
1.3.7	Bemerkung	7
1.3.8	Varianten von Graham's Scan	7
1.4	Algorithmus III: Devide and Conquer	7
1.4.1	Spezialfall	7
1.4.2	Allgemein	8
1.4.3	Laufzeit	8
1.5	Eine Anwendung von Convex Hull	8
1.5.1	Problem (<i>Schnitt von zwei Halbebenen mit Dualitätsalgorithmus</i>)	8
1.5.1.1	Definition (<i>abgeschlossene Halbebene</i>)	8
1.5.1.2	Anmerkung	8
1.5.1.3	Ziel und Lösungsansatz	8
1.5.1.4	Definition (<i>dualer Punkt/ duale Gerade</i>)	8
1.5.1.5	Lemma	8
1.5.1.6	Folgerung	9
1.5.1.7	Betrachte folgendes Problem	9
1.5.1.8	Beobachtung	9
1.5.1.9	Definition (<i>redundant</i>)	9
1.5.1.10	Lemma	9

1.5.1.11	<u>Lemma</u>	11
1.5.1.12	<u>Algorithmus</u>	11
1.5.1.13	<u>Zwischenresultat</u>	12
1.5.1.14	<u>Berechne S</u>	12
1.5.1.15	<u>Satz (Zusammenfassung)</u>	13
1.5.1.16	<u>Bemerkungen</u>	13
1.5.2	<u>Schnitt von n Halbebenen mit Divide & Conquer</u>	13
1.5.2.1	<u>Beispiel</u>	13
1.5.2.2	<u>Idee</u>	13
1.5.2.3	<u>Definition (Region)</u>	13
1.5.2.4	<u>Algorithmus</u>	14
1.5.2.5	<u>Implementierungsdetails</u>	14
1.5.2.6	<u>Satz</u>	14
1.5.2.7	<u>Fragen</u>	14
1.5.2.8	<u>D & C – Algorithmus für Schnitt von Halbebenen</u>	14
1.5.2.9	<u>Laufzeit</u>	14
1.5.2.10	<u>Frage</u>	14

2 Konvexe Polygone.....15

2.1 Einführung.....15

2.1.1	<u>Ziel</u>	15
2.1.2	<u>Idee</u>	15
2.1.3	<u>Definition (hierarchische Darstellung)</u>	15
2.1.4	<u>Beispiel</u>	15
2.1.5	<u>Bemerkung</u>	15
2.1.6	<u>Eigenschaften</u>	15
2.1.7	<u>Beispiel</u>	15
2.1.8	<u>Alternative Darstellung</u>	16
2.1.9	<u>Beispiel</u>	16
2.1.10	<u>Lemma</u>	16

2.2 Anwendung der hierarchischen Darstellung.....16

2.2.1	<u>Strategie</u>	16
2.2.2	<u>Darstellung im Rechner</u>	17
2.2.3	<u>Beispiel</u>	17
2.2.4	<u>Anwendung : Schnitt mit einer Geraden</u>	17
2.2.4.1	<u>Ziel</u>	17
2.2.4.2	<u>Idee für Algorithmus</u>	17
2.2.4.3	<u>Laufzeit</u>	18
2.2.4.4	<u>Satz</u>	19
2.2.4.5	<u>Bemerkung</u>	19

2.3 Weiteres Problem auf konvexen Polygonen.....19

2.3.1	<u>Ziel</u>	19
2.3.2	<u>Idee</u>	19
2.3.3	<u>Laufzeit</u>	21

3	Das Plane Sweep Verfahren	22
3.1	Einführung	22
3.1.1	<u>Idee</u>	22
3.1.2	<u>Bemerkung</u> ... <i>Sl, S_e, S_a</i>	22
3.1.3	<u>Bemerkung</u> ... <i>sonstige Varianten von Sl-Verfahren</i>	22
3.2	Erste Anwendung: Line Segment Intersection	22
3.2.1	<u>Problem</u>	22
3.2.2	<u>Triviale Lösung</u>	22
3.2.3	<u>Ziel</u>	22
3.2.4	<u>Idee für einen Plane Sweep Algorithmus</u>	22
3.2.4.1	<u>Beobachtung</u>	23
3.2.4.2	<u>Bemerkung/Definition (Event)</u>	23
3.2.4.3	<u>Events beim Segmentschnitt</u>	23
3.2.5	<u>Plane Sweep allgemein</u>	23
3.2.5.1	<u>X-Struktur</u>	23
3.2.5.2	<u>Y-Struktur</u>	23
3.2.6	<u>Operationen beim Segmentschnitt</u>	24
3.2.6.1	<u>Auf Y-Struktur</u>	24
3.2.6.2	<u>Auf X-Struktur</u>	24
3.2.7	<u>Implementierung von X-/Y-Struktur</u> ... <i>lexikante + Annahmen</i>	24
3.2.8	<u>Bemerkung</u> ... <i>Altkbedarf</i>	24
3.2.9	<u>Sweep Algorithmus für Segmentschnitt</u>	26
3.2.10	<u>Laufzeit</u>	27
3.2.11	<u>Geometrische Primitive</u>	27
3.2.12	<u>Bemerkung</u> ... <i>wenn keine Annahmen</i>	27
3.2.13	<u>Varianten des Problems</u>	27
3.2.13.1	<u>Red/Black Intersection Problem</u>	27
3.2.13.2	<u>Kurvensegmente</u>	27
3.2.13.3	<u>Berechnung der planaren Unterteilung der Ebene</u>	27
3.3	Zweite Anwendung: Schnitt von beliebigen Polygonen	27
3.4	Dritte Anwendung: Post Office Problem	28
3.4.1	<u>Einführung</u>	28
3.4.1.1	<u>Voronoi-Diagramm</u>	28
3.4.1.2	<u>Problem</u>	28
3.4.1.3	<u>Mögliche Varianten</u>	28
3.4.1.4	<u>Idee</u>	28
3.4.2	<u>Erster Schritt: Voronoi-Diagramm</u>	28
3.4.2.1	<u>Definition (Voronoi-Region)</u>	28
3.4.2.2	<u>Beispiel</u>	28
3.4.2.3	<u>Bemerkung</u> ... <i>VR(x) := ∩ H</i>	28
3.4.2.4	<u>Definition (Voronoi-Diagramm)</u>	29
3.4.2.5	<u>Definition (Voronoi-Knoten/-Kanten)</u>	29
3.4.2.6	<u>Bemerkung</u> ... <i>VR (Menge S)</i>	29
3.4.2.7	<u>Beispiel</u>	29
3.4.2.8	<u>Definition (Voronoi-Diagramm der Ordnung k)</u>	29
3.4.2.9	<u>Beispiel</u>	29
3.4.2.10	<u>Spezialfälle</u>	30
3.4.2.11	<u>Lemma</u>	30
3.4.2.12	<u>Bemerkung</u> ... <i>Delaunay: Triangulierung</i>	31

3.4.3	<u>Konstruktion von Voronoi-Diagramm</u>	32
3.4.3.1	<u>Ziel</u>	32
3.4.3.2	<u>Problem</u>	32
3.4.3.3	<u>Idee</u>	32
3.4.3.4	<u>Beobachtung</u> <i>Wie kommt man auf L-2 & Planung hier</i>	32
3.4.3.5	<u>Frage</u>	32
3.4.3.6	<u>Beispiel</u>	32
3.4.3.7	<u>Beobachtung</u>	33
3.4.3.8	<u>Idee</u> <i>Y-SH</i>	33
3.4.3.9	<u>Fragen</u>	33
3.4.3.10	<u>Zwei Arten von Events</u>	33
3.4.3.11	<u>Lemma</u>	33
3.4.3.12	<u>Implementierungsdetails</u>	34
3.4.3.13	<u>Ausgabe</u>	35
3.4.3.14	<u>Übung</u> <i>Wichtig</i>	35
3.4.3.15	<u>Satz</u> <i>Langzeit</i>	35
3.4.3.16	<u>Bemerkung</u> <i>Vermutung</i>	36
3.4.3.17	<u>Beispiel</u>	36
3.4.3.18	<u>Bemerkung</u>	36
3.4.4	<u>Zweiter Schritt: Point Location</u>	36
3.4.4.1	<u>Aufgabe</u>	36
3.4.4.2	<u>Idee</u>	36
3.4.4.3	<u>Ziel</u>	36
3.4.5	<u>Point Location allgemein (unabhängig von Voronoi-Diagramm)</u>	37
3.4.5.1	<u>Problem</u>	37
3.4.5.2	<u>Streifenmethode: allgemein</u>	37
3.4.5.3	<u>Streifenmethode: Idee</u>	37
3.4.5.4	<u>Streifenmethode: Ergebnis</u>	38
3.4.5.5	<u>Triangulierungsmethode: allgemein</u>	38
3.4.5.6	<u>Triangulierungsmethode: Idee</u>	38
3.4.5.7	<u>Triangulierungsmethode: Fragen und Antworten</u>	39
3.4.5.8	<u>Triangulierungsmethode: Definition (unabhängig)</u>	39
3.4.5.9	<u>Triangulierungsmethode: Lemma</u>	39
3.4.5.10	<u>Triangulierungsmethode: Lemma</u>	40
3.4.5.11	<u>Triangulierungsmethode: Algorithmus</u>	40
3.4.5.12	<u>Beispiel</u>	41
3.4.5.13	<u>Zusammenfassung</u>	42
3.4.5.14	<u>Algorithmus für Point Location</u>	43

4 Bewegungsplanung in der Ebene

4.1 Einführung.....

4.1.1	<u>Allgemeines Problem</u>	44
4.1.2	<u>Bemerkungen</u>	44

4.2 Problem 1: R ist Kreis und S Menge von Segmenten.....

4.2.1	<u>Idee</u>	44
4.2.2	<u>Frage</u>	44
4.2.3	<u>Antwort</u>	44
4.2.4	<u>Voronoi-Diagramm von Segmenten in der Praxis</u>	44
4.2.5	<u>Definition (Freiheit, frei, FP)</u>	44
4.2.6	<u>Idee für Algorithmus</u>	44
4.2.7	<u>Beispiel</u>	44
4.2.8	<u>Algorithmus</u>	45

4.2.9	<u>Beispiel</u>	46
4.2.10	<u>Lemma</u>	46
4.2.11	<u>Laufzeit</u>	46
4.2.12	<u>Satz</u>	46
4.3	<u>Problem 2: R ist konvexes Polygon und S Menge von konvexen Polygonen</u>	47
4.3.1	<u>Problem</u>	47
4.3.2	<u>Anmerkung</u>	47
4.3.3	<u>Idee</u> ... Reduktion.....	47
4.3.4	<u>Konstruktion von aufgeblähten Hindernis</u>	47
4.3.5	<u>Beispiel</u>	48
4.3.6	<u>Anmerkung</u> ... Größe von FP.....	48
4.3.7	<u>Satz</u> ... über #. P.ken von Konv.....	48
4.3.8	<u>Algorithmus</u> ... Berechnung einzelner Konturen.....	48
4.3.9	<u>Laufzeit</u>	48
4.3.10	<u>Plane Sweep-Algorithmus zum Mischen von zwei Konturen A und B</u>	49
4.3.10.1	<u>Problem</u>	49
4.3.10.2	<u>Definition (sichtbar)</u>	49
4.3.10.3	<u>Idee</u> ... Modifikation.....	49
4.3.10.4	<u>Aktionen</u>	49
4.3.10.5	<u>Bemerkung</u>	50
4.3.10.6	<u>Lemma</u> ... Laufzeit.....	50
4.3.10.7	<u>Bemerkung</u> ... $s = O(n^2)$ i.d.....	50
4.3.10.8	<u>Beispiel</u>	50
4.3.11	<u>Analyse der Laufzeit</u>	50
4.3.11.1	<u>Idee</u>	50
4.3.11.2	<u>Definition (Int(y), usw)</u>	50
4.3.11.3	<u>Satz</u> ... E.M.S.....	51
4.3.11.4	<u>Bemerkung</u>	53
4.3.11.5	<u>Beispiel</u>	53
4.3.11.6	<u>Satz</u>	53
4.3.12	<u>Lösung des Bewegungsplanungsproblems</u>	53
4.3.13	<u>Grober Algorithmus</u>	54
4.3.14	<u>Satz (Zusammenfassung)</u>	54
4.3.15	<u>Bemerkung</u> ... i.d. Praxis.....	54

5 Geometrische Datenstrukturen.....55

5.1	<u>Segmentbaum</u>	55
5.1.1	<u>Definitionen und Bemerkungen (Segmentbaum)</u>	55
5.1.2	<u>Beispiele</u>	55
5.1.3	<u>Lemma</u> ... Komplexität beim Segmentbaum.....	56
5.1.4	<u>Suche in Segmentbäumen</u>	56
5.1.5	<u>Laufzeit</u>	57
5.1.6	<u>Problem</u>	57
5.1.7	<u>Algorithmus zur Berechnung des Problems</u>	57
5.1.8	<u>Laufzeit</u>	57
5.1.9	<u>Realisierung der Knotenlisten</u>	58
5.1.10	<u>Satz</u> ... Zusammenfassung.....	58
5.1.11	<u>Bemerkungen</u> ... \exists voll dynamische Bäume.....	58

5.2	Range-Tree (Bereichsabfrage-Baum)	58
5.2.1	Definition (<i>Range-Tree</i>)	58
5.2.2	Beispiele	58
5.2.2.1	Dimension = 1	} jeweils mit Komplexitätsbehandlung
5.2.2.2	Dimension = 2	
5.2.3	Verallgemeinerung für beliebige Dimensionen	60
5.2.4	Bemerkungen	60
5.3	Priority-Search-Tree	61
5.3.1	Definition (<i>Priority-Search-Tree</i>)	61
5.3.2	Speichern der Punkte	61
5.3.3	Beispiel	61
5.3.4	Problem1 und Lösung	61
5.3.5	Problem2 und Lösung	62
5.3.6	Satz (<i>Zusammenfassung</i>)	62
5.3.7	Anwendung	62
5.4	Das Maßproblem für achsenparallele Rechtecke	63
5.4.1	Problem	63
5.4.2	Idee	63
5.4.3	Beispiel	64
5.4.4	Beobachtung	64
5.4.5	Genauere Betrachtung der Aktionen	65
5.4.6	Satz	65
6	Drei-dimensionale konvexe Hüllen	66
6.1	Einführung	66
6.1.1	Problem	66
6.1.2	Darstellung des planaren Oberflächengraphen	66
6.1.3	Beispiel	66
6.1.4	Geometrische Prädikate	66
6.2	Algorithmen	67
6.2.1	Inkrementeller Algorithmus	67
6.2.1.1	Algorithmus	67
6.2.1.2	Beispiel	67
6.2.1.3	Bemerkung	68
6.2.1.4	Laufzeit	68
6.2.1.5	Bemerkung	68
6.2.2	Divide & Conquer-Algorithmus	68
6.2.2.1	Algorithmus	68
6.2.2.2	Situation	69
6.2.2.3	Problem	69
6.2.2.4	Beobachtungen	69
6.2.2.5	Satz	70
6.3	Anwendung von 3-D konvexen Hülle (<i>Delaney-Triangulierung</i>)	70
6.3.1	Einführung	70
6.3.2	Idee	70
6.3.3	Umsetzung	70

6.3.4	<u>Geometrisches Prädikat</u>	71
6.4	<u>Arrangements und Dualität</u>	71
6.4.1	<u>Problem1</u>	71
6.4.1.1	<u>Problem</u>	71
6.4.1.2	<u>Einfache Lösung</u>	71
6.4.1.3	<u>Bessere Lösung</u>	71
6.4.1.4	<u>Dualität</u>	71
6.4.1.5	<u>Algorithmus</u>	71
6.4.1.6	<u>Laufzeit</u>	72
6.4.1.7	<u>Bemerkung</u>	72
6.4.2	<u>Problem2</u>	72
6.4.2.1	<u>Problem</u>	72
6.4.2.2	<u>Einfache Lösung</u>	72
6.4.2.3	<u>Bessere Lösung</u>	72
6.4.2.4	<u>Dualität</u>	72
6.4.2.5	<u>Laufzeit</u>	73
6.4.2.6	<u>Beispiel</u>	73
6.4.2.7	<u>Berechnung des Arrangements von n Geraden</u>	73
6.4.2.8	<u>Laufzeit</u>	74
6.4.2.9	<u>Lemma</u>	74
6.4.2.10	<u>Beispiel</u>	74
6.4.2.11	<u>Satz (Arrangements)</u>	75
6.4.2.12	<u>Folgerungen</u>	75

Gesellschaft für Informatik - Fachgruppe 0.1.2 Algorithmische Geometrie



Was ist Algorithmische Geometrie?

Bereits im Altertum haben sich Wissenschaftler wie Pythagoras und Euklid mit geometrischen Problemen beschäftigt. Ihr Interesse galt der Entdeckung geometrischer Sachverhalte und deren Beweis. Sie operierten ausschließlich mit geometrischen Figuren (Punkten, Geraden, Kreisen etc.). Erst die Einführung von Koordinaten durch Descartes machte es möglich, geometrische Objekte durch Zahlen zu beschreiben.

Heute gibt es in der Geometrie verschiedene Richtungen, deren unterschiedliche Ziele man vielleicht an folgendem Beispiel verdeutlichen kann. Denken wir uns eine Fläche im Raum, etwa das Paraboloid, das durch Rotation einer Parabel um seine Symmetrieachse entsteht. In der *Differentialgeometrie* werden mit analytischen Methoden Eigenschaften wie die Krümmung der Fläche an einem Punkt definiert und untersucht.

Die *Algebraische Geometrie* faßt das Paraboloid als *Nullstellenmenge* des Polynoms $p(X,Y,Z) = X^2 + Y^2 - Z$ auf; hier würde man zum Beispiel den Durchschnitt mit einer anderen algebraischen Menge, etwa dem senkrechten Zylinder $(X - x_0^2) + (Y - y_0^2) - r^2$ betrachten und sich fragen, durch welche Gleichungen der Durchschnitt beschrieben wird.

Aus der Mathematik sind uns solche Fragestellungen von einfachen Beispielen vertraut: In der Analysis werden Tangenten an Kurven betrachtet, und in der linearen Algebra immerhin Durchschnitte von Objekten, die sich durch *lineare Gleichungen* beschreiben lassen.

Die *Algorithmische Geometrie*, verfolgt andere Ziele. Ihre Aufgaben bestehen in

- der Entwicklung von *effizienten und praktikablen Algorithmen zur Lösung geometrischer Probleme*, und in Existenz v. effiz. Lösungsverfahren + Konkrete Angabe der Algorithmen.
- der Bestimmung der *algorithmischen Komplexität* geometrischer Probleme. Angabe der unteren Schranke + Konstr. v. Alg., die diese Schranke nicht überschreitet.

Die untersuchten Probleme haben meistens sehr *reale Anwendungshintergründe*. Bei der *Bahnplanung für Roboter* geht es darum, eine Bewegung von einer Anfangskonfiguration in eine Endkonfiguration zu planen, die Kollisionen mit der Umgebung vermeidet und außerdem möglichst effizient ist. Wer je eine Leiter durch verwinkelte Korridore getragen hat, kann sich ein Bild von der Schwierigkeit dieser Aufgabe machen. Sie wächst noch, wenn der Roboter seine Umgebung noch gar nicht kennt, sondern sie während der Ausführung erkunden muß.

Beim *computer aided geometric design (CAGD)* kommt es unter anderem darauf an, *Durchschnitt und Vereinigung von dreidimensionalen Körpern schnell zu berechnen*. Oder es sollen interpolierende Flächen durch vorgegebene Stützpunkte konstruiert werden.

Bei der Arbeit mit *geographischen Daten*, die in der Regel in Datenbanken gespeichert sind, müssen

Zusammenfassung von Algorithmische Geometrie.

Kapitel 0: Vorbemerkungen:

Inhalt der Vorlesung: Algorithmen und Datenstrukturen zur Lösung geometrischer Probleme.

Beispiele: Robotik / Bewegungsplanung, Computergrafik, CAD, VLSI

- Typische Probleme:
- konvexe Hülle
 - Zerlegung in einfache konvexe Teile (z.B. Triangulierung).
 - Schnittpunkte (z.B. Segmente, Polygone)
 - Suchprobleme (Memberanfrage in Pktmenge, Nearest-Neighbor, Range-Abfragen).
 - Bewegungsplanung (Roboter)
 - Hidden Line / Surface (Elimination).

Grundlegende Ansätze bzw. Vorgehensweisen:

- Divide & Conquer
- Plane Sweep.
- Hierarchische Darstellung
- Dualität
- Randomisierte Algorithmen.
- Rundungsfehler u. degenerierte Eingaben.

Kapitel I: Konvexe Hüllen.

1.1. Konstruktion der konvexen Hülle einer Pktmenge im \mathbb{R}^2 - Grundlagen.

1.1.1. Def: Sei $S \subset \mathbb{R}^2$ Pktmenge.

S heißt konvex: $\Leftrightarrow \forall p, q \in S$ gilt: $\overline{pq} \subset S$.

Konvexe Hülle einer Menge $S \subset \mathbb{R}^2$ ($CH(S)$) ist die kleinste konvexe Menge, die S enthält.

Wir betrachten nun endliche Pktmengen, d.h. $|S| = n \in \mathbb{N}$.

Ann: Der Rand von $CH(S)$ ist ein konvexes Polygon mit Ecken aus S .

Bew. Übung.

Konvexe Hülle-Problem für endl. Menge $S \subset \mathbb{R}^2$:

Berechne die Folge der Ecken von $CH(S)$ gg. den Uhrzeigersinn (pos. orientiert).

$\rightarrow q_1, \dots, q_k \in S$. (Anordnung definiert Kanten der Fläche).

Degenerierte Fälle: $P=2$ (alle Pkte sind colinear)

$P=1$ (alle Pkte in S sind gleich).

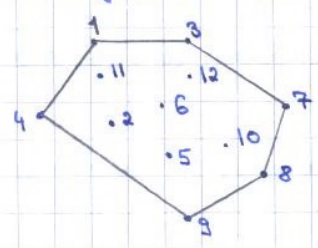
Problem im \mathbb{R}^3 analog, $CH(S)$ ist konvexes Polyeder mit Ecken aus S .

1.1.2. Satz: Berechnung der konvexen Hülle von n Pkten. im \mathbb{R}^2 ist mind. so schwer wie das Sortieren von n Zahlen.

Bew. siehe ÜB. (Idee: Fahre Sortieren von $x_1 \dots x_n$ zurück auf $CH(\{p_1 \dots p_n\})$).

1.1.3. Korollar: Im Allgemeinen braucht die Berechnung von $CH(S)$ Zeit $\Omega(n \log n)$.

1.1.4. Bsp:



$S = \{1..12\}$, $CH(S) = 8, 7, 3, 1, 4, 9$
(Jedes zyklische Shift)

1.1.5. Wdh: Lexikografische Ordnung (siehe Strings) für Pkte im \mathbb{R}^2 :

Lexikogr. Sortierung nach Kartesischen (x, y) -Koordinaten, d.h. für zwei Pkte in der Ebene, also $p, q \in \mathbb{R}^2$ gilt: $p < q \Leftrightarrow p_x < q_x \vee (p_x = q_x \wedge p_y < q_y)$

D.h. Sortierung von links nach rechts bzw. von unten nach oben (bei gleichem x -Koordinate).