GEOMETRIA COMPUTACIONAL

Esta introdução foi escrita pelo professor José Coelho de Pina, do Departamento de Ciência da Computação do IME-USP, que ministrou essa disciplina em 1997, 2000, 2001, 2002 e 2004, e apenas adaptada/atualizada para essa edição da disciplina

1. Introdução

A procura de algoritmos para resolver problemas geométricos vem desde a época da antiguidade. (Algumas motivações práticas para a busca por tais algoritmos foram os impostos sobre o uso da terra e construções de edificações.) São bem-conhecidas as construções geométricas de Euclides que usavam como instrumentos régua e compasso e consistiam de algumas *operações primitivas* que podiam ser realizadas com esses instrumentos. Um dos problemas algorítmicos[1] em geometria foi o chamado Problema de Apollonius (cerca de 200 A.C.) no qual três circunferências arbitrárias no plano eram dadas e pedia-se uma quarta circunferência que fosse tangente às três circunferências dadas. Euclides apresentou um algoritmo que resolve este problema.

Dentre todos os problemas algorítmicos em geometria (usando construções geométricas de Euclides) um que atraiu grande atenção foi o problema da construção de um polígono regular de n lados. Para alguns valores de n (e.g. $n = 3, 4, 5, 6$) a solução é conhecida desde a antiguidade. Entretanto, para heptágonos regulares ($n = 7$) prova-se que o problema não tem solução.[2]

Em 1902, Emile Lemoine introduziu uma medida de *simplicidade* para os algoritmos que usam as construções de Euclides. Esta medida é baseada no número de operações primitivas realizadas pelo algoritmo. Para Lemoine, o algoritmo mais simples é aquele que faz menos operações primitivas. A solução de Euclides para o Problema de Apollonius requer 508 dessas operações enquanto que um algoritmo proposto por Lemoine requer menos de duzentas. Estava portanto introduzido em geometria um conceito que é, pelo menos em essência, o que hoje chamamos de complexidade de um algoritmo.

Em geometria computacional também estamos interessados em desenvolver algoritmos eficientes para resolvermos problemas geométricos. Pelo que

[1]Diremos que um problema é *algorítmico* se este problema pede como resposta um algoritmo para resolver um determinado problema. Em geometria clássica esses problemas são conhecidos como *Problemas de Construções Geométricas.*

[2]Aos 17 anos Carl Friedrich Gauss (1777-1855) mostrou que *não* existe um algoritmo que usando somente as operações primitivas de Euclides construa um heptágono regular. Na realidade Gauss mostrou mais que isso, ele mostrou que existe um algoritmo para construir um p-gon (polígono regular com p lados), p primo se e somente se p é um primo da forma $2^{2^n} + 1$.

foi exposto acima vemos que não é algo novo. A diferença é que as operações primitivas usam um instrumento diferente da régua e do compasso: usam um computador. Um pouco mais precisamente, em geometria computacional estamos interessados em encontrar algoritmos eficientes, ou procedimentos computacionais, para resolver problemas geométricos. Muitos desses problemas têm sua origem em outras áreas como computação gráfica, robótica, computer-aided design e processamento de imagens. No desenvolvimento de tais algoritmos são comumente utilizados resultados em geometria euclidiana, combinatória, teoria dos grafos, estruturas de dados e análise de algoritmos.

Geometria computacional é um termo usado por diversos grupos[3]. Entretanto, o termo tem sido mais utilizado para descrever a sub-área da teoria de algoritmos trata do projeto e análise de algoritmos eficientes para problemas envolvendo objetos geométricos, principalmente, em espaços de dimensão 2, 3 ou de dimensão constante. As entradas para os problemas são primordialmente objetos simples (pontos, retas, segmentos de retas, polígonos, planos e poliedros). É neste sentido que usaremos o termo geometria computacional nesta disciplina.

Se a tese de Shamos [20] em 1978 for aceita como o início da geometria computacional (pelo menos da maneira como ela será tratada nesta disciplina), então a área tem apenas cerca de 30 anos. Apesar disso existem pelo menos 7 livros na área, 4 revistas.

A área desenvolveu rapidamente nos anos 70's, 80's e 90's, e ainda continua a se desenvolver. Por causa da área a partir da qual cresceu (desenvolvimento de algoritmos discretos), geometria computacional tem sempre enfatizado problemas de natureza matemática discreta. Na maioria dos problemas em geometria computacional as instâncias dos problemas são um conjunto finito de pontos ou outro objeto geométrico, e a saída exigida é algum tipo de estrutura consistindo de um conjunto finito de pontos ou segmentos de retas.

De acordo com O'Rourke [18] (página *xi*):

> "...Not all open problems [em geometria computacional] are necessarily difficult; some are simply awaiting the requisite attention ...".

Este pode ser um bom motivo para ficarmos de olhos abertos durante a disciplina e talvez tentar fazer alguma contribuição para a área.

Divirtam-se!

[O que foi brevemente tratado nesta introdução foi extraído de Capítulo 3 de Courant [4], Graham [12], Mount [15], e no Capítulo 1 de Preparata e Shamos [19].]

[3] O termo "geometria computacional" tem sido usado com várias conotações distintas. Por exemplo, geometria computacional também foi usado para se referir a modelagem geométrica através de splines e superfícies (cf. Capítulo 1 de Preparata e Shamos [19]).

2. Objetivos da disciplina

O objetivo desta disciplina é apresentar técnicas, algoritmos e estruturas de dados empregados no projeto e análise de algoritmos eficientes para resolução de problemas geométricos. Pretendemos mostrar estratégias clássicas de solução de problemas geométricos, assim como apresentar possivelmente temas de pesquisa.

3. Pré-requisitos

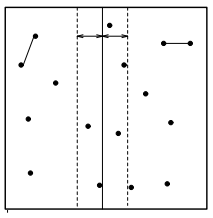
Para esta disciplina, os pré-requisitos são: conhecimento de técnicas básicas de projeto de algoritmos, como divisão-e-consquista, algoritmo guloso, programação dinâmica; notação e técnicas básicas de análise de algoritmos, como notação assintótica, resolução de somatórios e recorrências; e conhecimento de estruturas de dados básicas, como filas de prioridades (heaps) e árvores balanceadas de busca binária.

4. Tópicos que pretendemos cobrir

Alguns dos tópicos que pretendemos cobrir nesta disciplina são: fechos convexos; problemas de proximidade; partições convexas; busca geométrica; e problemas de intersecção.

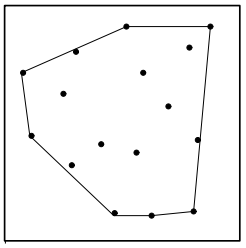
Abaixo encontra-se uma breve descrição de alguns problemas que estudaremos nesta disciplina.

Problema do par mais próximo (closest pair problem)



Dados n pontos, queremos encontrar dois deles que estejam a distância mínima. Uma aplicação prática deste problema é em controle de tráfego aéreo: os dois aviões que estão em maior perigo de colisão são aqueles que estão mais próximos. Este problema pode ser resolvido facilmente em $O(dn^2)$, onde d é a dimensão do espaço. O problema do par mais próximo pode ser resolvido por um algoritmo do tipo divisão-e-conquista em tempo $O(dn \log n)$ (cf. Capítulo 5 de Preparata e Shamos [19]) .

Fecho convexo de um conjunto de pontos

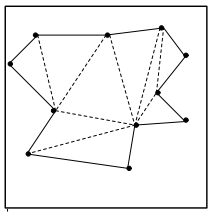


Convexidade é uma propriedade geométrica bastante importante. Um conjunto de pontos é *convexo* se, para cada par de pontos no conjunto, o segmento de reta entre eles está inteiramente contido no conjunto. Segundo O'Rourke (cf. O'Rourke [18], pg. 80) talvez o primeiro artigo na área de geometria computacional tenha sido sobre fechos convexos. O Problema do Fecho Convexo consiste em, dados n pontos, encontrar o fecho convexo desses pontos. Uma das aplicações práticas deste problema se encontra em

robótica. Se o fecho convexo de um robô não colide com obstáculos então o robô também não colide.

Nos anos 60 uma aplicação da Bell Labs necessitava computar o fecho convexo de aproximadamente 10.000 pontos no plano e os algoritmos de complexidade de tempo $O(n^2)$ foram considerados muito lentos. Tendo essa aplicação como motivação, no começo do anos 70, Graham [11] projetou o primeiro algoritmo de complexidade de tempo $O(n \log n)$. O fecho convexo também pode ser construído em $O(n \log n)$ por um algoritmo de divisão-e-conquista (cf. Capítulo 3 de Preparata e Shamos [19]).

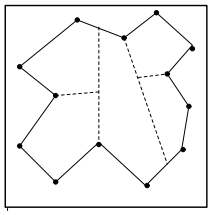
Triangularização de polígonos



O interesse aqui é particionar um certo 'domínio complexo' em uma coleção de objetos 'simples'. A região mais simples na qual podemos decompor um objeto planar é um triângulo (um tetraedro em 3-d e um 'simplex' em geral). Dado um polígono P, queremos adicionar a P o maior número possível de diagonais que não se cruzem de tal forma que o interior de P fique particionado em triângulos. Chazelle [1] projetou um algoritmo linear para este problema.

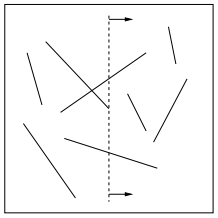
Um algoritmo para triangularizar polígonos pode ser utilizado em problemas do tipo *Art Gallery* (cf. O'Rourke [17]). Imagine que as salas de uma galeria de arte formem um polígono. Considerando que cada guarda fica parado em um local da galeria, qual é o menor número de guardas que são necessários para tomar conta das salas?

Partição de polígonos



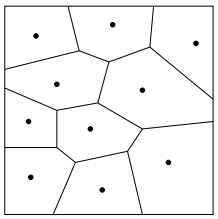
Além de algoritmos eficientes para particionar um polígono em triângulos, também são de interesse algoritmos que particionem um polígono em (digamos) polígonos monótonos, trapezóides e polígonos convexos. Uma motivação para particionar um polígono em polígonos convexos é o reconhecimento de caracteres: um caractere pode ser representado como um polígono particionado em partes convexas.

Intersecções



Um dos problemas geométricos mais básicos é o de determinar quando dois objetos se intersectam. A determinação se dois objetos complexos se intersectam é freqüentemente reduzida ao problema de determinar quais pares de entidades primitivas (e.g., segmentos de retas) se intersectam. Veremos algoritmos eficientes para computar a intersecção de um conjunto de segmentos de retas.

Diagramas de Voronoi

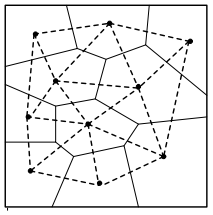


Dado um conjunto S de n pontos no plano, queremos determinar para cada ponto p em S qual é a região $V(p)$ dos pontos do plano que estão mais perto de p do que de qualquer outro ponto em S. As n regiões $V(p)$ formam uma partição do plano chamada de *Diagrama de Voronoi*.

Imagine uma vasta floresta contendo vários pontos de observação de incêndio. O conjunto das árvores que estão mais próximas de um determinado posto p determina a região $V(p)$ das árvores que são de responsabilidade do ponto p.

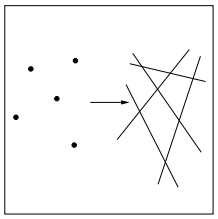
O diagrama de Voronoi de um conjunto de n pontos pode ser construído em $O(n \log n)$ por um (complicado) algoritmo de divisão-e-conquista (cf. Shamos [21]). Em 1985, Fortune [10] projetou um algoritmo de varredura (*plane-sweep algorithm*) muito elegante e simples cuja complexidade de tempo é $O(n \log n)$.

Triangularização de Delaunay



O dual geométrico (usando retas) de um diagrama de Voronoi para um conjunto S de pontos forma uma triangularização do conjunto S, chamada de *triangularização de Delaunay*. A triangularização de Delaunay tem várias propriedades geométricas interessantes. Por exemplo, ela contém todas as "árvores geradoras mínimas" de S (cf. Capítulo 6 de Preparata e Shamos [19]).

Arranjos e dualidade



Talvez uma das estruturas matemáticas mais importantes em geometria computacional seja um arranjo de retas (e em geral, arranjos de curvas e superfícies). Dadas n retas no plano, um arranjo é simplesmente o grafo que tem como vértices as intersecções das retas e como arestas os segmentos de retas ligando estas intersecções. Veremos que uma tal estrutura pode ser construída em tempo $O(n^2)$. A razão para está estrutura ser tão importante é que muitos problemas envolvendo pontos podem ser transformados em problemas envolvendo retas através do método de dualidade. Por exemplo, suponha que desejemos determinar se existem três pontos colineares entre um conjunto de n pontos no plano. Isto pode ser determinado por um algoritmo do tipo força-bruta em tempo $O(n^3)$. Entretanto, se os pontos são dualizados em retas, então (como veremos mais tarde nesta semestre) a questão é reduzida a decidir se existe um vértice de grau pelo menos 4 neste arranjo de retas.

5. Bibliografia

Para preparar as aulas desta disciplina tenho consultado as notas de aula do professor José Coelho de Pina [6], os livros de O'Rourke [18] e de Berg, van Kreveld, Overmars, e Schwarzkopf [5].

O livro de Preparata e Shamos [19] é um texto clássico em geometria computacional (foi primeiro livro sobre o assunto) que coloca bastante ênfase na análise dos algoritmos apresentados. Este livro contém basicamente todos os tópicos que serão tratados nesta disciplina. Outros livros que também podem ser encontrados na biblioteca são: Edelsbrunner [8] ("The art of counting and estimating is at heart of combinatorics—and it is a necessary prerequisite for analyzing algorithms ..."; cópiado da introdução da Parte I de [8]); Figueiredo e Carvalho [9] (um livro muito claro e introdutório); e Resende e Stolfi [7] (descreve varias técnicas e algoritmos em geometria computacional). Outros livros sobre geometria computacional são: Laszlo [14] (um livro que descreve vários algoritmos em geometria computacional e apresenta trechos de implementações em `C++`); Mulmuley [16] (como o próprio título diz, este livro trata de algoritmos aleatórios em geometria computacional).

Cormen, Leiserson, Rivest & Stein [3] é um livro enciclopédico sobre análise de algoritmos que trata de geometria computacional no Capítulo 33.

Na biblioteca também podem ser encontrados alguns surveys sobre geometria computacional, veja por exemplo: Chazelle [2]; Graham e Yao [12]; Guibas e Stolfi [13]; e Yao [22].

Artigos em geometria computacional podem ser encontrados em várias revistas, incluindo *ACM Transactions on Graphics*, *Algorithmica*, *Journal of Algorithms*, *Journal of the ACM*, e *SIAM Journal on Computing*. Uma revista que é particularmente dedicada à área é *Discrete and Computational Geometry* e mais recentemente temos *International Journal of Computational Geometry & Applications* e *Computational Geometry, Theory and Applications*.

Existe uma conferência anual em geometria computacional, a *ACM Annual Conference on Computational Geometry* (alguns dos proceedings podem ser encontrados na biblioteca; veja QA758.C S989). Além desta, várias outras conferências apresentam trabalhos em geometria computacional: por exemplo, *STOC* (QA800.C S989), *FOCS* (QA800.C S989), *SODA* (QA758.C S989), e *ICALP*.

6. Implementações de algoritmos

Algumas das implementações que vocês verão durante as aulas foram feitas por Cassio Polpo de Campos (`cassio@ime.usp.br`, `http://www.ime.usp.br/~cassio/`) e por Eduardo Garcia de Freitas (`freitas@ime.usp.br`, `http://www.ime.usp.br/~freitas/`). O Cassio fez suas implementações no `Turbo C` e o Eduardo fez *applets* em `Java`. Todos os programas estão disponíveis no URL

http://www.ime.usp.br/~freitas/gc/.

7. Geometria computacional na Internet

Existe **muito** material **muito** bom de Geometria Computacional na Internet. Durante o andamento da disciplina manterei na página

http://www.ime.usp.br/~cris/geocomp2007/,

uma lista de alguns sítios de Geometria Computacional. Durante o andamento da disciplina está página deverá ser atualizada e expandida. Se você encontrar algum sítio de Geometria Computacional (ou de qualquer outra coisa) que você ache interessante, por favor, não deixe de me avisar.

8. Monitor

O monitor desta disciplina é Rafael Cosentino (cosen@ime.usp.br, http://www.ime.usp.br/~cosen/.

9. Outras informações

A minha sala é a 107-C, o número do meu telefone é 3091-5709 e meu endereço eletrônico é cris@ime.usp.br.

Manterei uma página de MAC 331 / MAC 5747 no URL

http://www.ime.usp.br/~cris/geocomp2007/.

Nessa página eu colocarei o material da disciplina (como, por exemplo, listas de exercícios, notas de aula, programação das aulas, etc). Por favor, dê uma olhada nesta página regularmente.

Há uma lista de discussão que tem como objetivo servir de suporte para a disciplina. Recomenda-se que você mande para esta lista suas dúvidas, sugestões, críticas ou observações sobre o andamento da disciplina. Assim, se você pretende cursar geometria computacional, por favor, inscreva-se na lista, que estará acessível a partir da página da disciplina. Sinta-se a vontade para me escrever e fazer perguntas ou comentários sobre a disciplina.

Outros professores do Departamento de Ciência da Computação que estudam geometria computacional são Carlos Eduardo Ferreira (sala 108-C, cef@ime.usp.br, http://www.ime.usp.br/~cef/), José Augusto Ramos Soares (sala 102-C, jose@ime.usp.br, http://www.ime.usp.br/~jose/), e José Coelho de Pina (sala 4-C, jose@ime.usp.br, http://www.ime.usp.br/~coelho/).

Se seu interesse é Computação Gráfica, converse com o professor Antonio Elias Fabris (e-mail aef@ime.usp.br, http://www.ime.usp.br/~aef/). Se você quer saber o que é Processamento de Imagens, Visão Computacional, etc, então converse com os professores Carlos Hitoshi Morimoto (sala 209-C, hitoshi@ime.usp.br, http://www.ime.usp.br/~hitoshi/), Júnior Barreira (sala 290-A, jb@ime.usp.br, http://www.ime.usp.br/~jb/), e Roberto Marcondes Cesar Júnior (sala 297-A, cesar@ime.usp.br, http://www.ime.usp.br/~cesar/).

Referências

1. B. Chazelle, *Triangulating a simple polygon in linear time*, Discrete and Computational Geometry **6** (1991), 485–524.
2. ———, *Computational geometry: A retrospective*, Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing (Montréal, Québec, Canada), The ACM Special Interest Group for Algorithms and Computation Theory, May 1994, pp. 75–94.
3. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to algorithms*, The MIT Electrical Engineering and Computer Scienece Series, The MIT Press, MacGraw-Hill Book Company, 1990, QA758 C811i.
4. R. Courant and H. Robbins, *What is mathematics?*, Oxford University Press, New York, 1941.
5. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational geometry, algorithms and applications*, Springer Verlag, 1997, second edition, 2000.
6. J.C. de Pina, *Geometria computacional*, Notas de aula, 2000.
7. P.J. de Resende and J. Stolfi, *Fundamentos de geometria computacional*, IX Escola de Computação, 1994.
8. H. Edelsbrunner, *Algorithms in combinatorial geometry*, EATCS Monographs on Theoretical Computer Science, no. 10, Springer-Verlag, Berlin, 1987, QA758 E21a.
9. L.H. Figueiredo and P.C.P. Carvalho, *Introdução à geometria computacional*, 18º Colóquio Brasileiro de Matemática, IMPA, 1991, QA758 F475i.
10. S. Fortune, *A sweepline algorithm for Voronoi diagrams*, Algorithmica **2** (1987), 153–174.
11. R.L. Graham, *An efficient algorithm for determining the convex hull of a finite planar set*, Information Processing Letters **1** (1972), 132–133.
12. R.L. Graham and F. Yao, *A whirlwind tour of computational geometry*, The American Mathematical Monthly **97** (1990), no. 8, 687–701.
13. L.J. Guibas and J. Stolfi, *Ruler, compass and computer: The design and analysis of geometric algorithms*, Theoretical Foundations of Computer Graphics and CAD (R.A. Earnshaw, ed.), NATO ASI Series, vol. F40, Springer-Verlag, 1988, pp. 111–165.
14. M.J. Laszlo, *Computational geometry and computer graphics in C++*, Prentice Hall, Upper Saddle River, NJ, 1996.
15. D. Mount, *Cmsc 754: Computational geometry*, Spring 2000, Course Syllabus.
16. K. Mulmuley, *Computational geometry: An introduction through randomized algorithms*, Prentice Hall, Englewood Cliffs, NJ, 1994.
17. J. O'Rourke, *Art gallery theorems and algorithms*, The International Series of Monographs on Computer Science, Oxford University Press, New York, 1987, QA830 O74a.
18. ———, *Computational geometry in C*, Cambridge University Press, Cambridge, 1993.
19. F.P. Preparata and M.I. Shamos, *Computational geometry: An introduction*, Texts and Monographs in Computer Science, Springer-Verlag, New York, 1985, QA758 P927c.
20. M.I. Shamos, *Computacional geometry*, Ph.D. thesis, Yale University, New Haven, 1978.
21. M.I. Shamos and D. Hoey, *Closest point problems*, Proc. 16th Annual IEEE Symposium in Foundations of Computer Science, 1975, pp. 151–162.
22. F.F. Yao, *Computational geometry*, Handbook of Theoretical Computer Science (J. van Leeuwen, ed.), vol. A, The MIT Press/Elsevier, Amsterdam, 1990, QA810.C3 V259h v.1A, pp. 343–389.