DCC207 – Algoritmos 2

Aula 05 – Introdução a Geometria Computacional (Parte 02)

Professor Renato Vimieiro

DCC/ICEx/UFMG

Problema da envoltória convexa

- A computação da envoltória convexa de um conjunto de pontos no plano é um problema bastante estudado em geometria computacional
- Ela tem diversas aplicações reais:
  - Aproximação da forma de um objeto complexo
  - Segmentação de um conjunto de pontos
  - Evitar colisões (navegação de robôs)

Problema da envoltória convexa

- O problema consiste em encontrar o menor polígono (estritamente) convexo que contenha um conjunto de pontos P
- Dessa forma, se H é a envoltória convexa de P, todo ponto de P está dentro ou na borda de H
- Além disso, todos os ângulos internos de H são menores que π
- Logo, a solução para o problema é encontrar os vértices do polígono dentre os pontos de P

Varredura de Graham (Graham Scan)

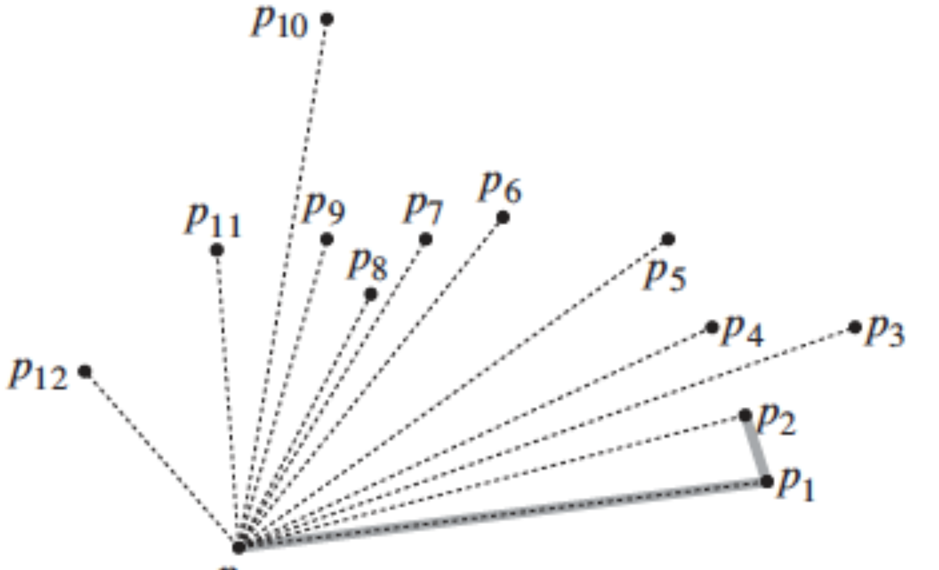
- O primeiro algoritmo que estudaremos foi proposto por Ronald Graham em 1972
  - O artigo em que esse algoritmo foi publicado é creditado por alguns como o primeiro trabalho na área de geometria computacional
  - Segundo O'Rourke (1997), ele foi proposto em resposta a uma aplicação na Bell Labs no fim da década de 1960 que envolvia 10.000 pontos. A solução existente $O(n^2)$ era muito lenta.
- A ideia do algoritmo, contudo, é bastante simples.
- Os vértices são processados um a um, partindo de um vértice âncora, em sentido anti-horário.
- A cada iteração, estende-se a envoltória convexa dos i-1 vértices anteriores com o vértice atual.

Varredura de Graham

- O algoritmo supõe que não há 3 pontos colineares no conjunto; e também que P contém ao menos 3 pontos.
- O algoritmo inicia o processamento a partir do ponto com a menor coordenada y mais à esquerda (em caso de empates).
  - Como esse ponto é o mais abaixo dentre todos, ele é uma escolha segura para ser um vértice da envoltória
- Em seguida, o algoritmo processa os demais pontos ordenados pelo ângulo polar com respeito a p0

Varredura de Graham

- Exemplo de inicialização do algoritmo:
  - p0 é o ponto âncora escolhido
  - Os demais são ordenados segundo o ângulo polar com respeito a p0
- Durante a ordenação, o algoritmo mantém somente o vértice mais distante de p0 dentre aqueles que possuem a mesma inclinação (ângulo) de p0
  - Por quê?



(a)

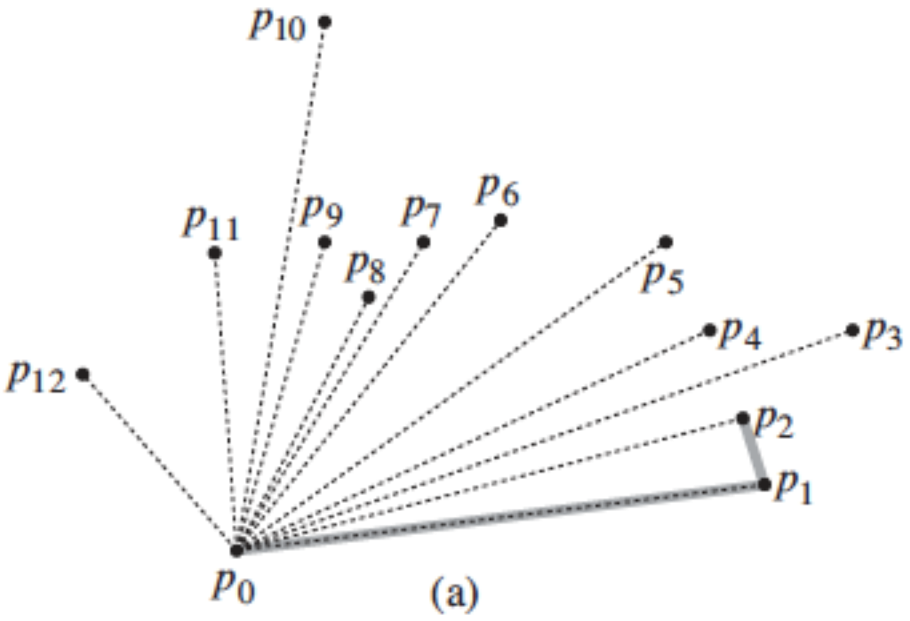
Varredura de Graham

- Tendo terminado o pré-processamento, o algoritmo inicia o processo de extensão da envoltória contendo os i primeiros vértices a partir da envoltória dos i-1 anteriores
- Intuitivamente, como estamos processando os vértices no sentido anti-horário, deve haver uma mudança de direção para a esquerda em p_i quando seguimos pelo caminho p_{i-1}-p_i-p_{i+1}.
    - Se isso ocorrer, então p_{i+1} é adicionado ao resultado
- Caso haja uma mudança para a direita no percurso acima, então p_i deve ser um vértice interno.
    - Nesse caso, p_i é retirado do conjunto solução, e repete-se o teste para p_{i-2}-p_{i-1}-p_{i+1} até que a condição seja satisfeita
- O algoritmo termina quando todos os vértices tiverem sido processados

Varredura de Graham

```
GRAHAM-SCAN(Q)
let p0 be the point in Q with the minimum y-coordinate,
        or the leftmost such point in case of a tie
let ⟨p1, p2, ..., pm⟩ be the remaining points in Q,
        sorted by polar angle in counterclockwise order around p0
        (if more than one point has the same angle, remove all but
        the one that is farthest from p0)
let S be an empty stack
PUSH(p0, S)
PUSH(p1, S)
PUSH(p2, S)
for i = 3 to m
    while the angle formed by points NEXT-TO-TOP(S), TOP(S),
            and pi makes a nonleft turn
        POP(S)
    PUSH(pi, S)
return S
```

- Exemplo:



Algoritmo embrulho para presente (gift wrapping)

- Esse algoritmo foi proposto por R.A Jarvis em 1973 e é muito similar ao da Varredura de Graham (citado no artigo original)
- O algoritmo apresenta uma vantagem em relação ao outro por possuir complexidade O(n) no melhor caso
  - O algoritmo possui complexidade O(nh), h é o número de vértices da envoltória
- A ideia do algoritmo é a seguinte:
  - Imagine que você tenha um conjunto de pregos em um tabuleiro (pontos)
  - Ate uma corda no prego mais ao sul do tabuleiro
  - Puxe a corda para a direita e depois para cima até encontrar o próximo prego
  - Ate a corda a esse prego e repita o processo até retornar ao primeiro

Algoritmo embrulho para presente (gift wrapping)

- Em termos computacionais, a ideia é similar ao algoritmo de Graham.
- Escolha um vértice para iniciar o processamento (vértice âncora)
  - Esse vértice é o de menor coordenada y, mais à esquerda em caso de empates
- Em seguida, repita até que p_i = p0
  - Encontre o vértice pi+1 com o menor ângulo polar à esquerda de pi e mova para ele

Algoritmo embrulho para presente (gift wrapping)

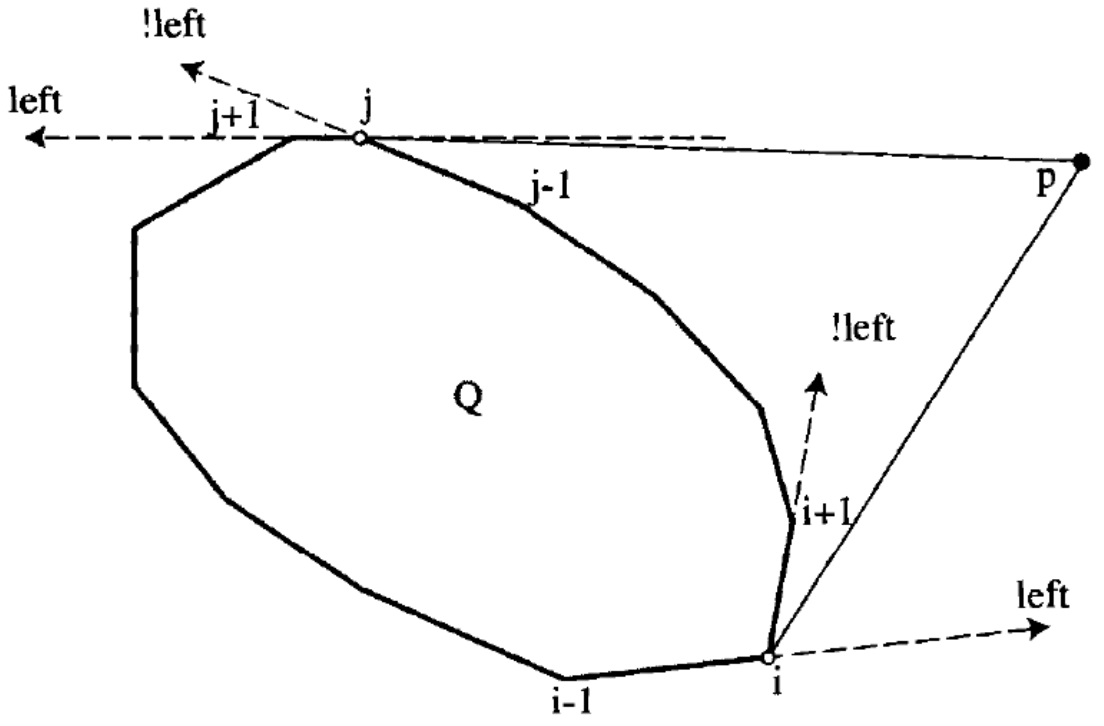
Algorithm: GIFT WRAPPING
Find the lowest point (smallest y coordinate).
Let i_0 be its index, and set $i \leftarrow i_0$.
repeat
    for each $j \neq i$ do
        Compute counterclockwise angle θ from previous hull edge.
    Let k be the index of the point with the smallest θ.
    Output (p_i, p_k) as a hull edge.
    $i \leftarrow k$
until $i = i_0$

Algoritmo Incremental

- Embora os algoritmos anteriores possuam bons desempenhos, eles não podem ser estendidos para 3D
- O algoritmo incremental pode ser tão eficiente quanto os outros e ainda pode ser estendido para 3D!
- A ideia do algoritmo também é consideravelmente simples
  - Escolha três pontos aleatórios. A solução é trivial.
  - Agora, adicione os outros pontos um a um e atualize a envoltória
  - Podem ocorrer dois casos:
    - O ponto está dentro da envoltória. Logo, pode ser descartado.
    - O ponto não está na envoltória. A envoltória precisa ser atualizada.

Algoritmo incremental

- A atualização da envoltória pode ser feita computando-se as retas tangentes que passam pelo polígono e pelo ponto
  - Como não há três pontos colineares, cada reta tangente tocará somente em um ponto do polígono
- A descoberta dos pontos de tangência pode ser feita usando as primitivas vistas
  - Note que p está à esquerda de i com respeito a i-1, e à direita de i+1 com respeito a i
  - O inverso ocorre em relação a j
  - Logo um ponto k é o ponto de tangência se há mudança de direção considerando o antecessor e sucessor de k

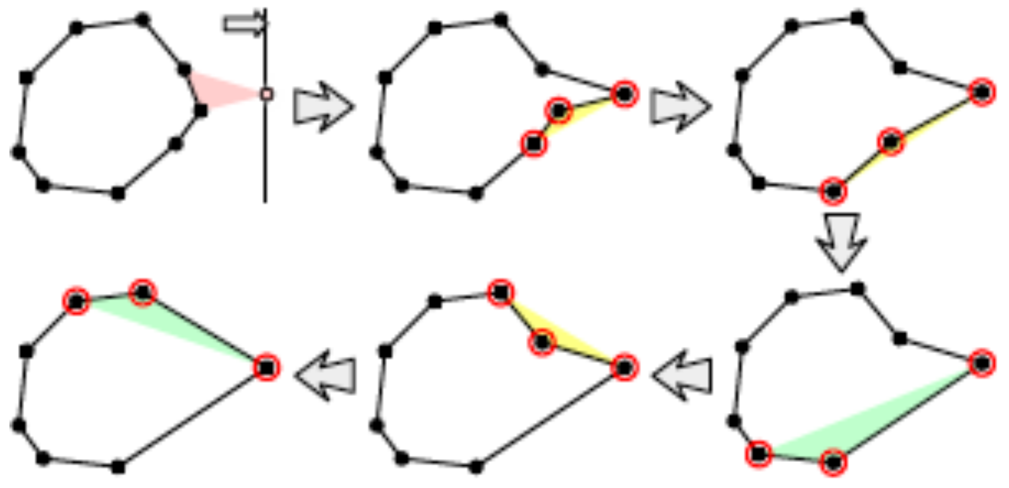


Algoritmo incremental (v2)

- A implementação do algoritmo como ele está descrito leva a um custo $O(n^2)$
- A cada iteração são realizado *i* testes para descobrir o ponto de tangência
- O tempo pode ser melhorado se ordenarmos os pontos e usarmos um mecanismo de varredura
- Os pontos podem ser ordenados pela coordenada x
- A reta varre os pontos da esquerda para a direita, atualizando a envoltória
  - Elimina a necessidade de verificar pertinência

Algoritmo incremental (v2)

- Sempre que um novo ponto p for avaliado:
  - Conecte-o ao ponto mais à direita da envoltória, p_i, e a um de seus vizinhos p_{i+1}
  - (atualizar borda inferior) Enquanto p estiver à direita de p_i com respeito a p_{i-1}:
    - Descarte p_i
    - Repita o processo com p_{i-1}
  - (atualizar borda superior) Enquanto p estiver à direita de p_{i+2} com respeito a p_{i+1}:
    - Descarte p_{i+1}
    - Repita o processo com p_{i+2}
- Como no máximo n pontos são inseridos/removidos, o custo total das atualizações é O(n)
- O custo total do algoritmo é dominado pela ordenação dos pontos O(n lg n)

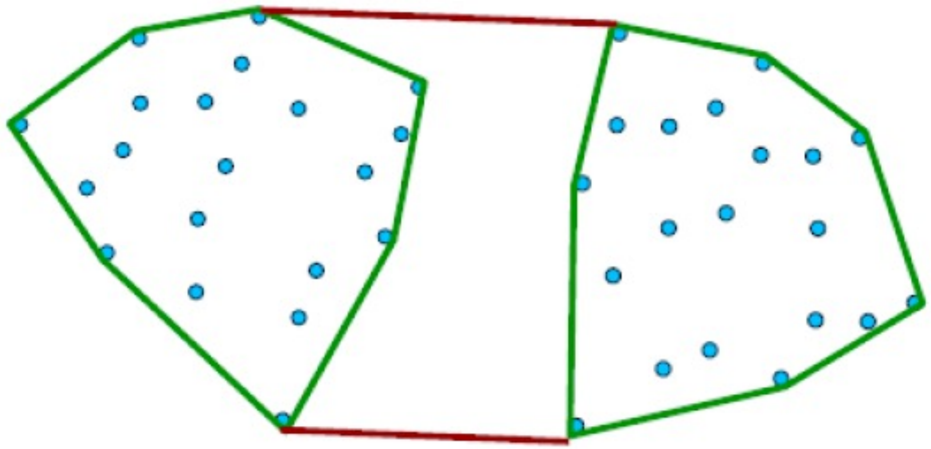


Algoritmo dividir e conquistar

- Outro algoritmo que pode ser estendido para 3D é a abordagem dividir e conquistar
- A ideia geral é dividir o problema até que a solução seja trivial, e depois combinar as soluções parciais
- O algoritmo supõe que, além de não existirem 3 pontos colineares, não existem dois pontos com a mesma coordenada x (essa premissa pode ser garantida fazendo pequenas perturbações nos pontos)
- Para evitar sobreposições dos subproblemas, o algoritmo inicialmente ordena os pontos pela coordenada x

Algoritmo dividir e conquistar

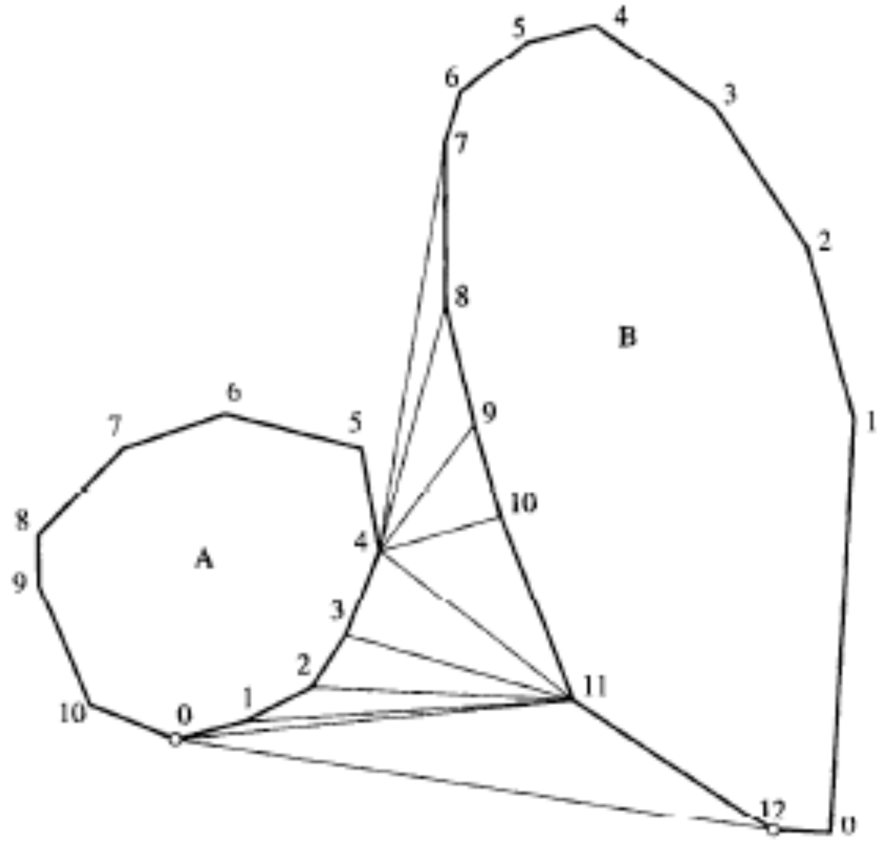
- O segundo passo consiste em dividir os pontos em duas metades A e B, e computar recursivamente as envoltórias convexas desses conjuntos
  - O problema é resolvido de forma trivial se o número de pontos for no máximo 3
- O último passo é combinar as envoltórias de A e B para obter a envoltória do conjunto A U B
- A combinação das envoltórias pode ser feita computando-se as retas tangentes aos dois polígonos nas partes superior e inferior



http://www.faculty.jacobs-university.de/llinsen/teaching/320201/Lecture25.pdf

Algoritmo dividir e conquistar

- A computação das retas tangentes pode ser feita da seguinte forma
  - Considere a reta formada pelo ponto *a* mais à direita de A, e *b* mais à esquerda de B
  - Enquanto a reta ab não for a tangente inferior dos polígonos, mova para baixo avançando em cada polígono por vez
    - Se ab não for tangente inferior a A (ante(a) ou succ(a) não estão acima de ab), então mova a para ante(a) (ante e succ definidos pela ordem da varredura de Graham)
    - Repita o mesmo para B
  - O mesmo procedimento se aplica para encontrar a tangente superior
- A combinação das envoltórias é feita eliminando todos os pontos de A e B entre os de tangência
- O procedimento é executado em tempo O(n)



Leitura

- Seção 33.3 (CLRS)
- Seção 22.2 (Goodrich e Tamassia)
- Capítulo 3 (Computational Geometry in C, 2nd ed., J. O'Rourke)
- Notas de aula Prof. Lars Linsen, Jacobs University Bremen
  http://www.faculty.jacobs-university.de/llinsen/teaching/320201/Lecture25.pdf
- Notas de aula Jeff Erickson, Universty of Illinois
  http://jeffe.cs.illinois.edu/teaching/compgeom/notes/01-convexhull.pdf

DCC207 – Algoritmos 2

Aula 05 – Introdução a Geometria Computacional (Parte 02)

Professor Renato Vimieiro

DCC/ICEx/UFMG