MAC 5747 GEOMETRIA COMPUTACIONAL

LISTA DE EXERCÍCIOS 7

1. Instruções

Entregue os Exercícios 1, 3, 7, 8 e 9 até a quinta-feira dia 22 de junho.

2. Exercícios

1. (Exercício 3.4.1.1 de [1] — `Embrulho-para-Presente`: melhor caso) Encontre o melhor caso para o Algoritm `Embrulho-para-Presente`: conjuntos de n pontos no plano tal que a complexidade de tempo assintótica do algoritmo seja o menor possível em função de n. Qual é esta complexidade de tempo?
2. (Exercício 3.4.1.2 de [1] — `Embrulho-para-Presente`: melhorias) Durante a execução do algoritmo `Embrulho-para-Presente` algumas vezes é possível determinar alguns pontos que não podem ser vértices do fecho convexo e portanto podem ser eliminado *"on the fly"*. Determine algumas regras para identificar tais pontos. Qual é o conjunto de pontos que exige mais trabalho do seu novo algoritmo?
3. (Exercício 3.4.1.4 de [1] — `QuickHull`: pior caso) Construa um conjunto de pontos genérico que força o Algoritmo `QuickHull` a ter um comportamente quadrático. Por 'genérico' (como sempre) queremos dizer uma construção que seja o pior caso para o `QuickHull` para valores arbitrários de n.
4. (Exercício 3.4.1.5 de [1] — `QuickHull`: análise do pior caso) Argumente que o `QuichHull`, como o Algoritmo `Embrulho-para-Presente`, tem complexidade de tempo *output-sensitive* de $O(nh)$, onde n é o número de pontos e h é o número de pontos na fronteira do fecho convexo.
5. (Exercício 3.4.1.7 de [1] — `QuickHull`: complexidade no caso médio) Argumente que o `QuichHull` tem complexidade de tempo $O(n)$ para n pontos escolhidos uniformemente ao acaso no quadrado de dimensão 1.
6. (Exercício 3.8.5.4 de [1] — `MergeHull`: *merge* sem ordenação). Se o pré-porcessamento do Algoritmo `MergeHull` não é feito, os fechos convexos podem se intersectar. Descreva um algoritmo que faz o 'merge' de dois fechos convexos arbitrários com k e m vértices em tempo $O(k + m)$. Este seu algoritmo fornece um outro algoritmos por divisão-e-conquista de complexidade de tempo $O(n \log n)$ para construir o fecho convexo de um conjunto de n pontos.

Date: 9 de junho de 2000.

7. (Exercício 3.5.6.1 de [1] — `Graham-Scan`: pior caso). Construa um conjunto de pontos para o qual o laço **while** da Linha 6 do algoritmo de Graham faz o maior número de iterações.
8. (Polígono passando por conjunto de pontos) Descreva um algoritmo de complexidade de tempo $O(n \log n)$ que dado um conjunto S de n pontos no plano contrói um polígono que tem os pontos em S como vértices.
9. (Algoritmo `Incremental` ótimo) Modifique o Algoritmo `Incremental` visto em sala de aula de tal forma que a sua complexidade de tempo seja reduzida para $O(n \log n)$.

   [**Sugestão.** Em um pré-processamento ordene os pontos de S em relação a x-coordenada de tal forma que $p \notin Q$ em cada passo. Examine os pontos nesta ordem.]
10. (Exercício 3.9.1.1 de [1] — fecho convexo de polígonos monótonos) Descreva um algoritmo que controi o fecho convexo de um polígono monótono em tempo linear.
11. (Exercício 3.9.1.2 de [1] — fecho convexo de um polígono arbitrário) Descreva um algoritmo que controi o fecho convexo de um polígono arbitrário em tempo linear. este execício é *tricky* e vários algoritmos publicados para este problema estavam errados.

    [**Observação.** A cota inferio de $\Omega(n \log n)$ é para um conjunto de pontos sem estrutura alguma, não para os pontos que são os vértices de um polígono. Os vértices de um polígono são dados na ordem em que eles ocorem ao percirrermos a fronteira do polígono.]

Referências

[1] J. O'Rourke, *Computational geometry in C*, Cambridge University Press, Cambridge, 1993, Second Edition, 1998.