Proposta de 1ª Prova

MO619/MC948— Geometria Computacional

Prof. Pedro J. de Rezende

1º Semestre de 2018

Elaborada por: Gustavo Yudi Bientinezi Matsuzake, yudi.matsuzake@gmail.com

1. Uma empresa te contratou para desenvolver algoritmos para envoltória convexa. Porém, ela tem diversas situações diferentes onde os algoritmos deverão ser utilizados. Dadas as seguintes situações, escolha um ou mais algoritmos para construção da envoltória convexa mais adequado. Justifique suas respostas.

   (a) A envoltória convexa precisa ser feita em paralelo;

   (b) O algoritmo irá receber pontos pela rede em diferentes momentos, e cada ponto $p_i = (t_i, y_i)$ será recebido no instante de tempo t_i, onde $t_i < t_{i+1}$, e a ordenada de p_i é irrestrita em valor. O algoritmo deve manter a envoltória convexa atualizada após a inclusão de cada novo ponto recebido;

   (c) Pelas propriedades do problema a ser resolvido, sabe-se a priori que a envoltória convexa de cada conjunto a ser processado terá no máximo dez vértices.

2. Dados dois pontos p e q, descreva um processo para encontrar o bissetor deles. Evite cálculos de ângulo e rotações que utilizem funções trigonométricas.

3. Você tem um trabalho em dupla da disciplina Geometria Computacional para entregar. O trabalho é a implementação de uma estrutura DCEL. Seu colega implementou grande parte do trabalho, porém está faltando a implementação de duas funções: connect_orbit e disconnect_orbit que são usadas, respectivamente, como sub-rotinas para adicionar e remover uma aresta em uma DCEL. Usando seus conhecimentos sobre a estrutura DCEL e o código do seu colega abaixo, complete as funções apropriadamente.

```
connect_orbit(edge e1, edge e2)
{
        /*
         * todo
         */
}

connect(edge a, edge b)
{
        e3 = new edge(a, b)

        /*
```

```
        * identifica arestas e1 e e2 apropriadas
        */
       e1, e2 = find_edges(a, b)

       connect_orbit(e1, e3)
       connect_orbit(e2, e3.twin)
}

disconnect_orbit(edge e1, edge e2)
{
       /*
        * todo
        */
}

disconnect(point a, edge b)
{
       /*
        * encontra aresta com pontos em a e b
        */
       e3 = find_edge(a, b)

       /*
        * identifica arestas e1 e e2 apropriadas
        */
       e1, e2 = find_edges(a, b)

       disconnect_orbit(e1, e3)
       disconnect_orbit(e2, e3.twin)
}
```

4. Descreva um algoritmo para encontrar o menor círculo envolvente de um conjunto de pontos S que passa por *exatamente* dois pontos de S. Analise a complexidade do seu algoritmo.

5. Dado um conjunto S de pontos, onde $|S| = n$, sabemos que a construção de todo o Diagrama de Voronoi de S, $\mathrm{Vor}(S)$, requer tempo $\Omega(n \log n)$. Considere que um único ponto no infinito ζ é usado como vértice extremo das arestas ilimitadas de $\mathrm{Vor}(S)$. Justifique a seguinte afirmação: dado um conjunto S, onde $|S| = n$, a determinação da ordem circular das arestas incidentes em ζ em order circular em torno desse vértice também requer tempo $\Omega(n \log n)$, em pior caso.