Tópicos em Geometria Computacional

Terceiro Trabalho

26 de julho de 2021

1 Introdução

O trabalho consiste em fazer uma implementação de um dos problemas descritos abaixo:

- Fecho convexo do $\mathbb{R}^3$
- BSP de um conjunto de triângulos no $\mathbb{R}^3$

Somente um destes problemas deve ser escolhido para a entrega.

2 Resolução do problema

A resolução do problema, ou seja, a descrição do problema, do algoritmo e sua corretude, deve estar em um texto claro em formato de um artigo e em pdf (com nome relatorio.pdf). Deve conter o nome do autor (aluno), uma introdução com o problema, o algoritmo e sua explicação. Todas as referências que forem usadas devem estar citadas corretamente no texto.

3 Especificação da implementação

A implementação pode ser feita em qualquer linguagem, contanto que seja possível rodar no ambiente computacional do DINF.

A entrada de dados deve ser feita pela entrada padrão (stdin) e a saída de dados pela saída padrão (stdout), ou seja, o seu programa lê do teclado e escreve na tela. O objetivo é que seja executado com redirecionamento de arquivos, como o comando abaixo:

$ programa < entrada.txt > saida.txt

Onde programa deve ser convex ou bsp, conforme o problema escolhido for o fecho convexo ou a BSP, respectivamente.

O trabalho deve ser entregue com um makefile de forma que ao digitar o comando make o executável seja construído.

4 Entrega

A entrega deve ser feita por e-mail para `andre@inf.ufpr.br`, com um arquivo tar.gz (compactado) enviado como anexo de uma mensagem com assunto "geocomp-trabalho 3" (exatamente).

O arquivo compactado deve ter nome fulano.tar.gz, onde fulano deve ser substituído pelo "login name" do autor.

O arquivo fulano.tar.gz, uma vez expandido, deve criar (no diretório corrente) os arquivos abaixo:

- makefile (ou Makefile)
- relatorio.pdf
- e os arquivos fontes necessários para a geração do executável.

5 Problemas, entradas e saídas

5.1 Fecho convexo no $\mathbb{R}^3$

Dado um conjunto S de pontos no espaço ($\mathbb{R}^3$), encontrar o fecho convexo de S. A saída deve estar organizada em uma estrutura de triângulos (como no trabalho 1).

5.1.1 Estrutura de Dados para Triangulações

Uma estrutura de dados para representar uma triangulação pode ser a descrita abaixo.

Cada vértice recebe um índice, de 1 a k, e suas coordenadas são armazenadas em um vetor indexado por estes índices.

Cada triângulo recebe um índice, de 1 a m, e um vetor indexado por estes índices deve conter: três campos para os vértices do triângulo, V_1, V_2, e V_3, com os índices dos três vértices; três campos com os índices dos triângulos vizinhos, T_1, T_2 e T_3, de forma que o triângulo T_i seja oposto ao vértice V_i.

Os vértices de um triângulo devem estar em ordem horária (do ponto de vista de um observador externo ao fecho convexo) e o primeiro índice (V_1) deve ser o menor.

5.1.2 Entrada

Considere que todas as coordenadas dadas são números inteiros entre 1 e 99 (inclusive).

A entrada de dados será um texto com a lista de pontos. A primeira linha tem um número, n, de pontos no espaço ($\mathbb{R}^3$). As n linhas seguintes tem as coordenadas x, y e z de cada um dos pontos, separadas por um espaço em branco.

Exemplo (comentários não fazem parte do arquivo):

```
5          # número de pontos
10 10 10   # coordenadas do 1o ponto
10 10 90   # ···
10 90 10
20 20 20
90 10 10
```

5.1.3 Saída

A saída de dados deve ser um texto com a descrição da estrutura de dados que representa a triangulação que representa o fecho convexo. As primeiras linhas tem o mesmo formato da entrada, com a descrição dos vértices do fecho convexo. Após estas primeiras linhas, temos uma linha com o número de triângulos e depois uma linha para cada triângulo, com os 6 campos da estrutura de dados, separados por um espaço em branco.

Exemplo:

```
4
10 10 10
10 10 90
10 90 10
90 10 10
4
1 3 2 4 3 2
1 2 4 4 3 1
1 4 3 4 1 2
2 3 4 3 2 1
```

5.2 BSP no $\mathbb{R}^3$

Dado um conjunto T de triângulos no espaço ($\mathbb{R}^3$), e uma lista L de segmentos de reta:

- construir uma BSP com os triângulos de T;
- usar esta BSP para calcular a lista de triângulos interceptados por cada segmento de reta de L.

5.2.1 Entrada

Considere que todas as coordenadas dadas são números inteiros entre 1 e 99 (inclusive).

A entrada de dados será um texto com a lista de pontos, a lista de triângulos e lista de segmentos de reta. A primeira linha tem os números, n, t e l, que são, respectivamente, o número de pontos, o número de triângulos e o número de segmentos de reta. As n linhas seguintes tem as coordenadas x, y e z de cada um dos pontos, separadas por um espaço em branco. Os pontos recebem índices de 1 a n na ordem em que aparecem. As t linhas seguintes tem 3 números separados por espaços, representando os índices dos vértices de cada triângulo. Os triângulos recebem índices de 1 a t na ordem em que aparecem. As l linhas seguintes tem 6 números representando as coordenadas dos pontos extremos de cada segmento, x_a, y_a, z_a, x_b, y_b, z_b.

Exemplo (comentários não fazem parte do arquivo):

```
12 4 3      # números de pontos, triângulos e segmentos
10 20 20    # coordenadas do 1o ponto
10 20 80    # ···
10 80 20
90 20 20
90 80 20
90 20 80
20 10 20
20 10 80
80 10 20
20 20 10
20 80 10
80 20 10    # coordenadas do 12o ponto
1 2 3       # índices dos vértices do 1o triângulo
4 5 6
7 8 9
```

```
10 11 12
40 5 40 40 95 40   # 1o segmento
40 40 5 40 40 95
5 40 40 95 40 40
```

5.2.2 Saída

A saída de dados deve ser um texto com os índices dos triângulos intersectados por cada segmento. A saída tem l linhas. Cada linha representa a lista de cada segmento. O primeiro número de uma linha é o número k de triângulos intersectados. Em seguida temos os k índices dos triângulos, ordenados do menor para o maior.

Exemplo:

```
1 3
1 4
2 1 2
```