EXERCÍCIO PROGRAMA 3

Data de entrega: 28/6/2002

Considere o seguinte problema, o *Problema da Localização de Pontos no Plano*:

Problema PLPP. *São dadas n retas no plano e dois pontos X e Y. Determine se X e Y pertencem ou não à mesma região definida pelas n retas*

O objetivo deste EP é desenvolver uma boa estruturas de dados para resolver o problema PLPP acima. Você deve escrever um programa que, dadas as n retas, constrói uma estrutura de dados para o conjunto dado que permita que, dado um par de pontos (X, Y), rapidamente possa se determinar se X e Y são separados por alguma das retas.

A estratégia de força bruta aplicada a este problema implica em testar se ambos os pontos estão do mesmo lado de cada uma das n retas. *O seu programa deve ser tal que o número de retas a serem testadas é, em geral, $O(\log n)$.*

Por simplicidade, você pode supor que todas as retas cruzam o quadrado unitário (onde ambas as coordenadas estão entre 0 e 1), e que você tem apenas que trabalhar com pontos e retas neste quadrado. Faça a leitura de um arquivo, de acordo com o seguinte formato:

```
5
 0.00 0.12 0.23 1.00
 1.00 0.41 0.00 0.52
 1.00 0.20 0.30 1.00
 0.00 0.40 0.10 0.00
 1.00 0.35 0.10 0.00

 0.25 0.80 0.60 0.50
 0.95 0.10 0.11 0.50
```

O arquivo de entrada começa com um inteiro n e seguem então n retas, cada uma dada por 4 números (`float`) que representam dois pontos da reta. Já que estamos interessados apenas no quadrado unitário, cada um dos extremos é um ponto na borda do quadrado, ou seja, tem pelo menos uma coordenada 0 ou 1. Após as n retas, vem uma seqüência de pares de pontos (cada par também dado por 4 números). No endereço

`http://www.ime.usp.br/~yoshi/2002i/CompII/EPs/EP3/Dados`,

você pode encontrar quatro arquivos, de tamanhos variados, em que você deve testar o seu programa.

Uma maneira de resolver este problema é construir uma árvore binária com nós internos correspondendo a retas e nós externos correspondendo a regiões do plano (veja Figura 1). Uma busca na árvore corresponde a comparar um ponto com a reta que está na raiz da árvore, e então continuar para a esquerda se ele está de um lado ou a direita se ele está do outro lado. Se a busca por dois pontos termina no mesmo nó externo, os dois pontos estão na mesma região. É um exercício mostrar

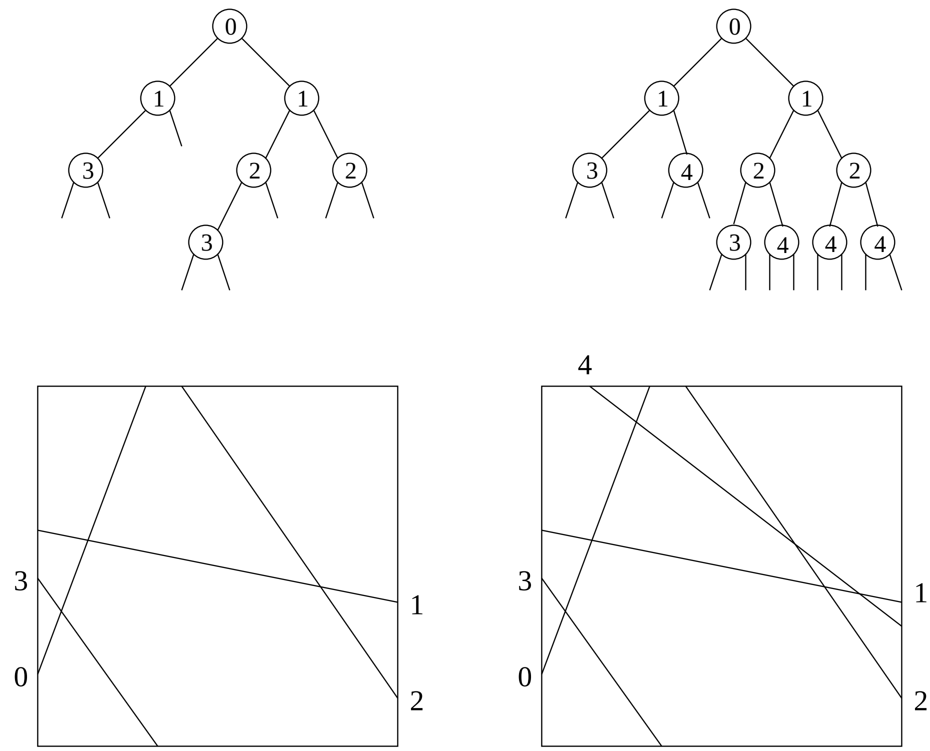

Figura 1. Um exemplo

que n retas determinam no máximo

$$\frac{1}{2}n(n+1)+1 \leq n^2$$

regiões, onde para a última desigualdade supomos que $n \geq 2$. Cada folha corresponde a uma região, e assim podemos esperar que o tempo deva ser proporcional a $\log n^2 = 2\log n$ para uma entrada 'aleatória' (e uma árvore construída de forma ingênua).

Para a fase de implementação e teste do seu programa, é importante que você possa ver graficamente o que há no arquivo de entrada. Você pode usar uma saída gráfica qualquer para isso. Uma sugestão é usar o formato `PS` (*PostScript*). No endereço

`http://www.ime.usp.br/~yoshi/2002i/CompII/EPs/EP3/PSs`,

você encontra um exemplo bem simples de arquivo *PostScript*, que corresponde ao exemplo da Figura 1. Desse exemplo, é fácil ver como escrever uma rotina que gera, a partir de um arquivo de entrada no formato acima, um arquivo `PS` correspondente às retas e pontos dados neste arquivo de entrada. (Você pode visualizar o que está codificado em um arquivo `PS` com o aplicativo *ghostview* (`gv`)). Você vai também precisar de um pouco de geometria: descubra uma forma simples de determinar se o ponto está de um lado ou de outro de uma reta e de calcular a intersecção de duas retas.

Inicialmente não se preocupe com casos degenerados (por exemplo, três retas que se intersectem em um ponto, ou problemas de precisão). Garanta que seu programa funcione para os casos não-degenerados. Após ter conseguido isso, passe a trabalhar nos casos degenerados. Concentre-se nos que lhe parecerem mais comuns. Escreva em seu relatório, além da descrição do arquivo de entrada e de como utilizar o seu programa, quais casos degenerados o seu programa trata e quais ele não trata. Os arquivos-teste disponíveis são de tamanhos variados. Comece testando o seu programa com o arquivo menor, progredindo para os testes maiores a medida que o seu programa funciona para os menores.

Seu programa, além de resolver o problema PLPP, deve imprimir os seguintes dados sobre a árvore montada: o número de nós da árvore, o número de nós externos da árvore (que deve ser exatamente o número de regiões), o comprimento médio de um caminho da raiz a um nó externo e a razão entre este último número e o log na base 2 do número de nós da árvore. Imprima uma tabela com estes números: cada arquivo-teste deve corresponder a uma linha da tabela.

Entregue não só os programas como também um relatório, por correio eletrônico para `yoshi@ime.usp.br` e `daniel@linux.ime.usp.br`. Não deixe de entregar uma listagem de seu EP e de seu relatório em papel.

Observações finais:

1. Para os testes de seu programa, use entradas variadas, além daquelas fornecidas. O ideal é você escrever um programa para gerar mais entradas.
2. Explique no relatório o funcionamento do seu sistema e discuta seus resultados.
3. No caso em que o conjunto de retas dadas é 'aleatória', você pode esperar que o comportamento de um algoritmo ingênuo para construir sua árvore tenha EPL pequeno. Você pode entregar um programa que implementa um tal algoritmo.
4. *Você consegue escrever um programa que constrói uma árvore de altura $O(\log n)$ para qualquer conjunto de n retas?*
5. *Você consegue justificar que o algoritmo que você usou em (4) acima de fato constrói uma árvore de altura $O(\log n)$?*