MAC–IME–USP

Carlos Eduardo Ferreira
Sala 108C Tel.: 3091 6079
e-mail cef@ime.usp.br

monitor Thiago Coraini
e-mail tcoraini@linux.ime.usp.br

monitor Alvaro Junio Pereira Franco
e-mail alvaro@ime.usp.br

MAC 323 - Estruturas de Dados

Primeiro semestre de 2009

Busca por pontos em janelas – **Entrega: 22 de abril de 2009**

O objetivo deste exercício-programa será implementar e usar árvores de busca binária balanceadas, e resolver com elas um problema da área de geometria computacional. O problema que trataremos é o de busca de pontos em uma janela. Inicialmente descrevemos uma aplicação.

Um grande latifundiário da Amazônia deseja construir um hotel de selva em sua propriedade, mas está muito preocupado com questões que envolvam desmatamento e o corte de árvores de madeira nobre e rara. Dessa forma, ele fez uma relação de todas as árvores de sua propriedade: posição exata da mesma, tipo da madeira, idade aproximada, altura, diâmetro do tronco, etc. De posse destes dados, e agora com a ajuda do arquiteto que vai projetar o hotel, ele deseja encontrar a melhor localização possível do hotel em sua propriedade. Desta forma, seu EP deve receber várias consultas de possíveis localizações e, para cada uma delas, listar quais árvores poderão ser afetadas. A fim de que estas consultas possam ser feitas eficientemente, utilizaremos uma estrutura de dados para armazenar as árvores cadastradas.

Vamos descrever o problema mais formalmente. Dado um conjunto de pontos P no plano (as árvores da propriedade do empresário) e uma sequência de janelas $W_1, W_2, \ldots, W_m$ (as possíveis localizações do hotel de selva), o seu programa deve responder eficientemente quais pontos de P estão em cada janela W_i, ou seja, para cada W_i-consulta, o seu programa deve listar os pontos de P que estão dentro de W_i, para $i = 1, \ldots, m$. Um exemplo de um conjunto de pontos e uma janela $W = (w_1, w_2)$ é ilustrado em seguida.

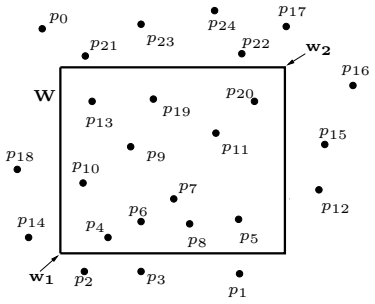

Figura 1: Um conjunto de pontos no plano e uma janela.

Descrição da estrutura de dados

A estrutura de dados que deve ser usada para armazenar os pontos é uma árvore de busca binária balanceada *de dois níveis*, isto é, em um primeiro nível a árvore de busca binária terá os pontos

ordenados por sua coordenada x e no segundo nível, por sua coordenada y. A seguir a estrutura é descrita com mais detalhes.

Nesta estrutura temos uma árvore de busca binária (a de primeiro nível) em que cada nó aponta para uma outra árvore de busca binária (a de segundo nível). A árvore do primeiro nível (ou árvore principal) é construída sobre as x-coordenadas dos pontos de P. Os pontos de P são armazenados nas suas folhas e os nós internos armazenam valores que direcionam uma busca no eixo x. Cada nó da árvore principal aponta agora, adicionalmente, para uma estrutura associada que também é uma árvore de busca binária balanceada. Denotamos por $P(v)$ os pontos armazenados nas folhas da subárvore com raiz em um nó v da árvore principal. A estrutura associada de v, ou seja, a árvore do segundo nível de v, é construída sobre os pontos de $P(v)$. As suas folhas armazenam os pontos de $P(v)$ e seus nós internos armazenam pontos que orientam uma busca no eixo y. Uma estrutura associada a um nó v de T é denotada por $T_y(v)$. Um exemplo parcial desta estrutura pode ser visto na Figura 2.

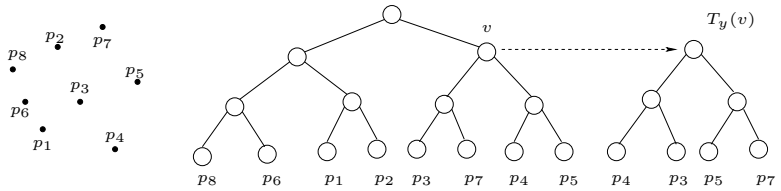

Figura 2: Uma árvore de 2-níveis parcial (somente uma estrutura associada de um nó v de T está ilustrada).

Denotamos o valor armazenado em um nó v de T por $p(v)$. Note que os pontos armazenados à esquerda de um nó v na árvore principal possuem x-coordenada menores ou iguais a $p(v)$ e os pontos armazenados à direita de v possuem x-coordenada maiores (afinal é uma árvore de busca binária). O mesmo ocorre com qualquer $T_y(v)$ com relação à y-coordenada.

Inicialmente você deverá implementar os algoritmos de manipulação de árvores de busca binária balanceadas. Para este EP precisaremos de inserção, mas é um excelente exercício implementar as operações de remoção também. Um bônus na nota será dado a quem implementar a remoção também.

Descrição do algoritmo para a consulta

Dada uma janela $W = (w_1, w_2)$, onde w_1 é o ponto inferior esquerdo da janela e w_2 o ponto superior direito, uma W-consulta pode ser resolvida usando uma estrutura de dados sugerida acima da seguinte forma. Primeiramente, encontramos o nó mais alto da árvore de busca binária T, v_{div}, tal que $x(w_1) \leq p(v_{div}) < x(w_2)$ (este nó é comumente chamado de ancestral comum mais baixo). Em seguida, os pontos na janela serão encontrados em buscas nas subárvores de v_{div}.

Primeiro, percorremos a subárvore esquerda de v_{div} buscando o valor $x(w_1)$ (a busca na subárvore direita é análoga). Observe que os pontos desta subárvore são menores ou iguais (na coordenada x) a $p(v_{div})$. Logo, eles são menores que $x(w_2)$. Suponha que estamos em um nó v descendente na subárvore esquerda de v_{div}. Se $x(w_1) > p(v)$, então nada podemos fazer a não ser continuar a busca na subárvore à direita. No entanto, se $x(w_1) \leq p(v)$ então a busca continua na subárvore à esquerda de v e $x(w_1) < p(r) < x(w_2)$, para todo ponto r armazenado na subárvore à direita de v

(vamos denotar por $d(v)$ o filho direito de v). Sabemos que a árvore de busca binária balanceada de segundo nível $T_y(d(v))$ armazena tais pontos, ordenados agora pela coordenada y. Precisamos, então, determinar quais destes pontos estão dentro da janela. De forma análoga, encontramos em $T_y(d(v))$ o nó mais alto v'_{div} tal que $y(w_1) \leq p(v'_{div}) < y(w_2)$. Percorremos as subárvores à esquerda e à direita de v'_{div} buscando pelos valores $y(w_1)$ e $y(w_2)$ e encontramos alguns pontos que estão em W. Veja a Figura 3.

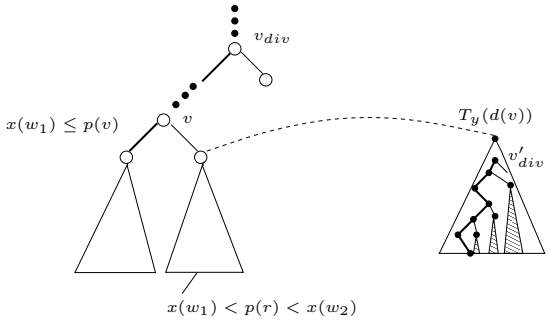

Figura 3: Os pontos nas árvores hachuradas estão em W.

O seu programa deve usar uma lista ligada L para listar os pontos que estão na janela e deve escrever em uma linha da saída padrão os elementos de L para cada W_i-consulta.

As informações necessárias para resolver este problema devem ser lidas de um arquivo chamado `consultas.in` que terá o seguinte formato: a primeira linha possui um número inteiro n entre 1 e 1000000; as próximas n linhas possuem n pontos no plano (o valor das coordenadas de cada ponto é um número inteiro); a linha $n + 2$ possui um número inteiro m entre 1 e 50000 que é o número de consultas; e as m linhas a partir daí trazem as coordenadas dos pontos de cada janela $W = (w_1, w_2)$, com $0 < x(w_1) < x(w_2)$ e $0 < y(w_1) < y(w_2)$.

Exemplo de uma entrada:

```
4
10 10
20 3
5 30
100 50
3
10 3 20 40
30 20 35 25
40 10 150 60
```

Saída correspondente:

```
Pontos na janela 1: (10, 10); (20, 3).
Pontos na janela 2:.
Pontos na janela 3: (100, 150).
```

Observação: No conjunto de pontos P não existem pontos coincidentes porém, podem existir em P pontos com mesma x-coordenada ou y-coordenada.