Relatório do Projeto de Geometria Computacional MAC0331

Breno Helfstein Moura 9790972
Lucas Carvalho Daher 8991769

Dezembro 2018

1 Introdução

O problema que foi resolvido nesse projeto foi o problema de localização de ponto com decomposição por faixas (*Slabs*). Nesse problema são dados, de maneira offline, polígonos internamente disjuntos, e de maneira online, pontos. O objetivo é determinar em qual dos polígonos cada ponto está ou se não está em nenhum dos polígonos. Um dos exemplos de aplicação do algoritmo é a localização do mouse em um navegador web [1]. O algoritmo executa uma varredura da esquerda para direita. Os pontos eventos são os vértices dos dos polígonos. Mantemos em cada momento as arestas dos polígonos que atravessam a faixa vertical entre dois pontos eventos consecutivos. Essas arestas ficam armazenadas em uma árvore *AVL* de modo que possam ser efetuadas inserções e remoções de arestas. O algoritmo implementado tem complexidade $O(n^2)$ no tempo e no espaço para a construção da estrutura de dados e $O(\log n)$ para responder a consulta de cada ponto, onde N é o número total de vértices dos polígonos. Foi usada apenas uma árvore de busca binária balanceada para a construção do algoritmo.

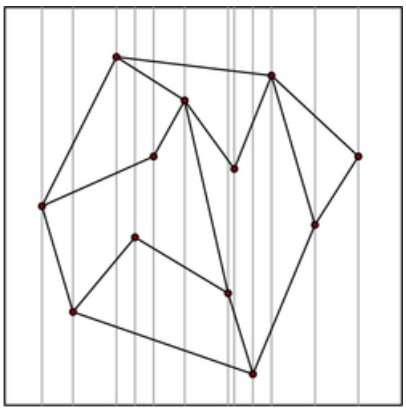


Figura 1: Decomposição em faixas de um mapa retilinear

2 Implementação

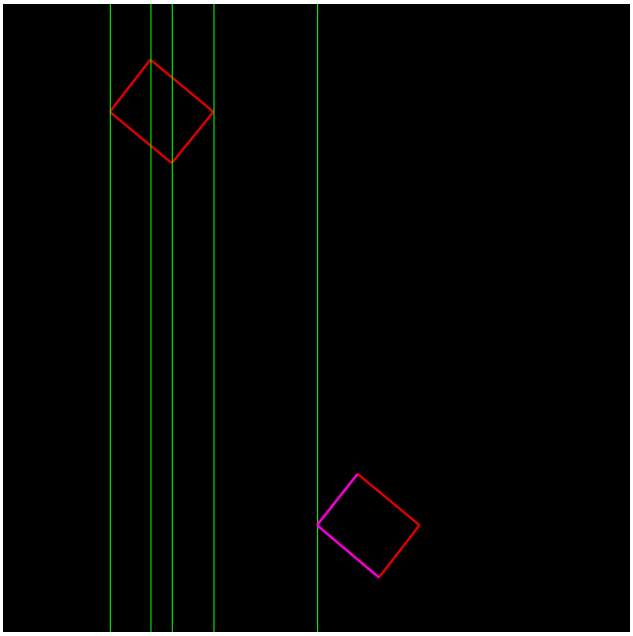
A implementação do algoritmo foi feita em python e utilizamos uma árvore *AVL* baseada na implementação encontrada no site *geeksforgeeks* [2]. O primeiro passo do pré processamento é criar uma árvore de eventos. Para isso, modificamos a estrutura da *AVL* para receber dois valores: chave e valor. A chave de cada evento é a x-coordenada que o define e o valor é uma lista com todas as arestas que possuem um dos pontos extremos em tal x e a informação de qual polígono é delimitado pela aresta.

Após criada a árvore de eventos, criamos as faixas. Cada evento delimita uma faixa: a primeira começa em "menos infinito"e termina no primeiro evento, a segunda começa no primeiro evento e termina no segundo, etc. Para criar as faixas, utilizamos outra árvore de busca binária com as arestas que cruzam a faixa e cada evento atualizamos a árvore com as arestas que entraram e que saíram. Essa etapa é o gargalo do programa, já que a árvore pode ter a cada momento $O(n)$ arestas e devemos copiá-la $O(n)$ vezes (uma vez para cada faixa). A árvore armazena objetos de uma classe criada por nós, HOR_LINE, que na verdade armazena arestas não verticais e criamos também um comparador próprio para a ordenação, que faz até quatro chamadas de ESQUERDA por comparação.

Após concluída a construção das faixas, o pré processamento está terminado e para cada ponto devemos apenas fazer duas buscas binárias para descobrir sua localização. A primeira busca binária determina em qual faixa o ponto está e a segunda determina entre quais arestas o ponto está na faixa. Dependendo da paridade do índice da aresta encontrada definimos se o ponto está dentro ou fora do polígono a que pertence a aresta.

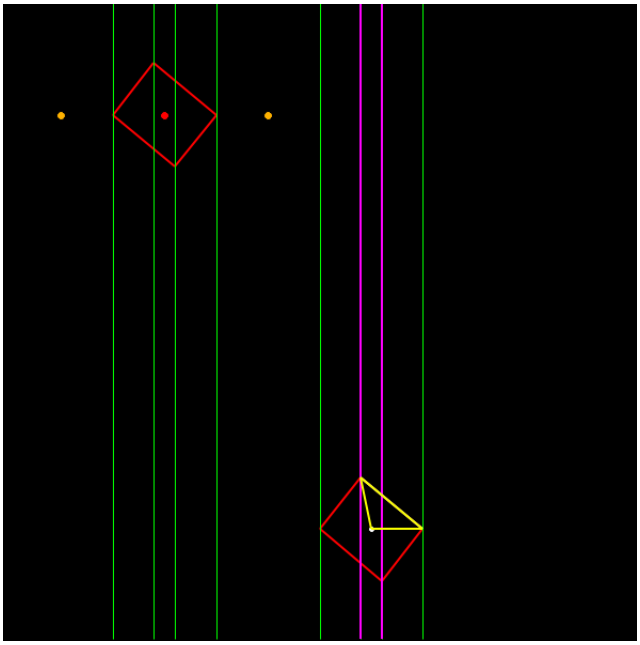
3 Animação

A animação tem duas fases. No início, os polígonos (ou seções) são desenhados em vermelho. A primeira fase ilustra a linha de varredura e testes de esquerda. A ordem das arestas na árvore é de baixo para cima. Note que as arestas não se intersectam internamente na faixa, logo essa ordem está bem definida.



Ao processar cada ponto evento, as arestas na faixa imediatamente anterior ao ponto são armazenadas em um vetor na ordem em que se encontram na árvore. O resultado da varredura é uma coleção de vetores com as arestas que atravessam cada faixa ordenadas como acima. Esse espaço é $O(n^2)$ e o tempo para guardar esses vetores é $O(n^2)$.

Então as consultas dos pontos são feitas, identificando onde estão os pontos em $O(\log N)$. São feitas diversas comparações entre as arestas (mais precisamente, entre a aresta que está sendo inserida ou removida da faixa e uma aresta já presente na faixa, que se encontra na árvore). As arestas comprovadas são mostradas em roxo. Primeiro as faixas são desenhadas em verde e então são destacadas (em azul claro) quando estão sendo analisadas. Na primeira busca binária determinamos a faixa em que o ponto está, pintamos a faixa em questão de roxo e analisamos as arestas que atravessam as faixas os pintando também. Quando um ponto está fora de um polígono ele é pintado de laranja e quando está dentro é pintado de vermelho. Quando o ponto está dentro de um polígono, este também é pintado de roxo.



O projeto dispoível no github no endereço `https://en.wikipedia.org/wiki/Point_location`.

Referências

[1] `https://en.wikipedia.org/wiki/Point_location`

[2] `https://www.geeksforgeeks.org/avl-tree-set-2-deletion/`