Relatório Projeto 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 "Slabs". Um dos exemplos de aplicação do algoritmo é a localização do mouse em um navegador web [1]. O algoritmo implementado tem complexidade $O(N^2)$ no tempo e no espaço para a construção da estrutura de dados e $O(logN)$ para cada query. Foi usada apenas uma árvore de busca binária balanceada para a construção do algoritmo.

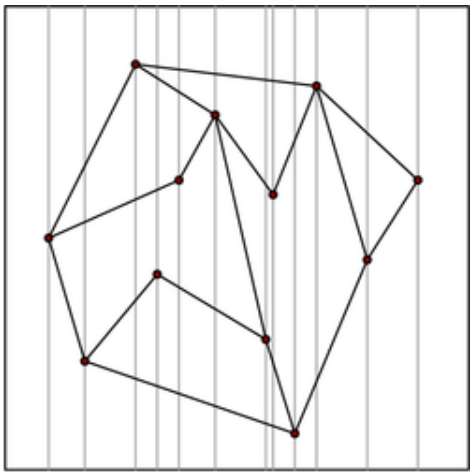


Figure 1: Slab Decomposition de um polígono

2 Implementação

A implementação do algoritmo foi feita em python e utilizamos uma *AVL tree* baseada na implementação encontrada no site *geeksforgeeks* [3]. 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 *slabs*. Cada evento delimita uma *slab*: 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 *slabs*, utilizamos outra árvore de busca binária com as arestas que cruzam a *slab* 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 *slab*). 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.

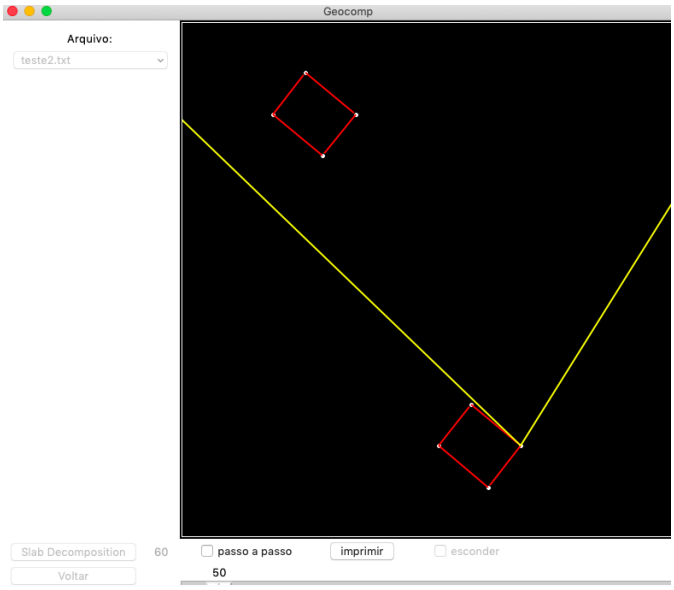
Após concluída a construção das *slabs*, o pré processamento está terminado e para cada ponto devemos apenas fazer duas buscas binárias para descobrir sua localização.

O Input é dado pelo Número de Polígonos (N), então cada polígono é representado pela quantidade de pontos do polígono (P_i) seguidos pelos pontos (x_{ij} y_{ij}). Então temos o número de pontos "queries" (Q) seguidos pelos pontos (x_k y_k). Como o input da plataforma só recebe pontos, fizemos que as linhas que só tem um valor são acompanhadas de 0. Um input exemplo está abaixo (Teste01):
1 0
3 0
-5.0 -1.0
0.0 5.0
5.0 -1.0
3 0
0.0 20.0
0.0 1.0
0.0 -20.0

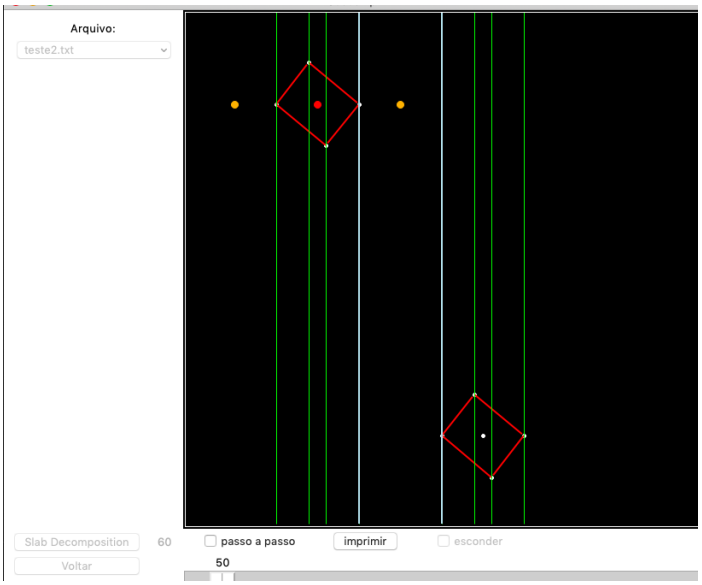
3 Animação

A animação feita tem duas fases. No início os polígonos (ou seções) são desenhados em vermelho. A primeira fase então é a construção da estrutura de dados, que consiste em diversos testes de esquerda:

Então as queries são feitas, identificando onde estão os pontos estão. Primeiro os slabs são desenhados em verde e então são destacados (Em azul claro) quando estão sendo analisados. Quando sabemos que o ponto está dentro do slab pintamos o slab de roxo e analisamos as arestas dos polígonos (ou seções) que cruzam os slabs os pintando da mesma cor que os slabs. 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 também pintamos o poligono que o contém de roxo.



O projeto pode ser encontrado no github [2].

References

[1] Wikipedia, point location, https://en.wikipedia.org/wiki/Point_location

[2] https://github.com/breno-helf/geocomp/

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