Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

Mateus Barros Rodrigues

Implementação de algoritmos para consultas de segmentos em janelas

São Paulo
Setembro de 2016

Implementação de algoritmos para consultas de segmentos em janelas

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Carlos Eduardo Ferreira

São Paulo
Setembro de 2016

Resumo

Este trabalho de conclusão de curso fundamentou-se na compreensão e implementação em linguagem *python* de um algoritmo para consultas de intersecções de segmentos de retas com janelas retangulares no espaço, um subproblema de geometria computacional conhecido por: buscas em regiões ortogonais. Este algoritmo foi o foco da tese de mestrado de Álvaro Junio Pereira Franco. Além da implementação, foi feita também a adaptação do visualizador de algoritmos geométricos feito por Alexis Sakurai Landgraf para exposição dos resultados obtidos.

Palavras-chave: Geometria, janelas, segmentos, buscas.

Sumário

Capítulo 1

Introdução

Neste trabalho de conclusão de curso foi abordado o problema de *consultas de segmentos em janelas*, um problema de *buscas em intervalos ortogonais*, que é um dos tópicos fundamentais da área de geometria computacional.

Dado um conjunto S de segmentos no espaço (Seja no $\mathbb{R}$, $\mathbb{R}^2$, etc.) e uma janela W de lados paralelos, queremos responder rapidamente a seguinte pergunta: *quais segmentos de S estão contidos na ou intersectam a janela W?*

Este trabalho foi baseado em *Consultas de segmentos em janelas: algoritmos e estruturas de dados* de Álvaro Junio (2009), portanto seguiremos a mesma divisão do problema que foi proposta nessa dissertação: Encontrar pontos contidos em janelas e achar todos os segmentos que intersectam com um dado segmento (Horizontal ou vertical). Seguiremos também a mesma divisão de capítulos: Primeiramente apresentaremos definições e primitivas geométricas, dedicaremos um capítulo para falar de consultas de pontos em janelas, um para falar de encontrar intersecção de segmentos e finalmente um onde agregaremos esses algoritmos para resolver o problema proposto. Todo o código desenvolvido foi escrito em linguagem *python* e está disponível no gitHub.

Capítulo 2

Definições e Primitivas

Explicaremos a seguir algumas das noções fundamentais que serão utilizadas ao longo do trabalho:

2.1 Pontos e Segmentos

Neste trabalho trataremos basicamente com pontos e segmentos de reta no espaço ($\mathbb{R}$ e $\mathbb{R}^2$). Sejam $x, y \in \mathbb{R}$ definimos um ponto no $\mathbb{R}^2$ como um par $p = (x, y)$. Um segmento s é da forma $s = \overline{(x_1, y_1)(x_2, y_2)}$ onde $u = (x_1, y_1)$ e $v = (x_2, y_2)$ são pontos chamados de pontos extremos de s.

2.2 Comparações

Uma outra definição que será usada copiosamente ao longo desta monografia é a relação de desigualdade associada à uma dada coordenada. Sejam u, v pontos, dizemos que $u \leq_x v$ caso $x(u) < x(v)$ ou $x(u) = x(v)$ e $y(u) \leq y(v)$ (Simetricamente definido para desigualdades em relação à coordenada y), ou seja, sempre comparamos primeiro a coordenada de maior interesse e desempatamos pela segunda coordenada nas comparações.

2.3 Posição Relativa

Usaremos também bastante a noção de posição relativa entre pontos e segmentos, isto é, dado um ponto p e um segmento s, queremos saber se p se encontra *à esquerda*, *à direita* ou *sobre* o segmento s.

Sejam $p = (x_1, y_1)$, $s = \overline{(x_2, y_2), (x_3, y_3)}$ e $d = \det \begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{pmatrix}$

Dizemos que p está **à esquerda** de s caso $d > 0$, que está **sobre** s caso $d = 0$ e que está **à direita** de s caso contrário. Seguem a seguir os trechos de código que foram usados no trabalho para realizarmos essas verificações:

Algoritmo 1 Retorna **TRUE** caso p esteja à esquerda de s

```
1 def left(p,s):
2     b = s.beg
3     c = s.end
4     if b.x == c.x and p.x == b.x: return p.y > c.y
5     if b.y == c.y and p.y == b.y: return p.x < c.x
6     return (b.x-p.x)*(c.y-p.y) - (b.y-p.y)*(c.x-p.x) > 0
```

Algoritmo 2 Retorna **TRUE** caso p esteja à direita de s

```
1 def right(p,s):
2     b = s.beg
3     c = s.end
4     if b.x == c.x and p.x == b.x: return p.y < b.y
5     if b.y == c.y and p.y == b.y: return p.x > c.x
6     return not(left_on(p,s))
```

Algumas ressalvas sobre essas funções:

- A única diferença da função *left_ on* em relação à função *left* é que ela também retorna *true* caso o ponto esteja sobre o segmento dado.
- As modificações presentes nas linhas 4 e 5 foram adicionadas apenas para resolverem os casos degenerados apresentados no capítulo x.

Capítulo 3

Desenvolvimentos

Embora neste exemplo tenhamos apenas um capítulo, entre a introdução e a conclusão de uma monografia podemos ter uma sequência de capítulos descrevendo o trabalho e os resultados. Estes podem descrever fundamentos, trabalhos relacionados, método/modelo/algoritmo proposto, experimentos realizados, resulatdos obtidos.

Cada capítulo pode ser organizado em seções, que por sua vez pode conter subseções.

Um exemplo de figura está na figura 3.1.

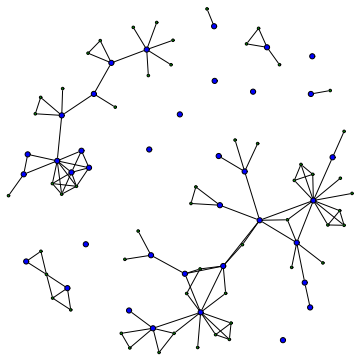

Figura 3.1: *Exemplo de uma figura.*

Capítulo 4

Conclusões

Texto texto[1].

[1]Exemplo de referência para página Web: www.vision.ime.usp.br/~jmena/stuff/tese-exemplo

Apêndice A

Título do apêndice

Texto texto.

Referências Bibliográficas

Álvaro Junio(2009) Álvaro Junio. Consultas de segmentos em janelas: algoritmos e estruturas de dados. Dissertação de Mestrado, Instituto de Matemática e Estatística, Universidade de São Paulo, Brasil. Citado na pág. 1