Universidade de São Paulo

Instituto de Matemática e Estatística

MAC 0331 - Geometria Computacional

Visibilidade de Arestas

Projeto Final do curso de Geometria Computacional do curso de graduação do bacharelado em Ciência da computação, ministrado e supervisionado pela professora Cristina Gomes Fernandes.

Daniel Paulino Alves, #USP 7156894

Felipe Yamaguti, #USP 7295336

São Paulo, Dezembro de 2014.

Descrição do problema

Dado um conjunto S contendo n segmentos de retas disjuntos no plano e um ponto p não pertencente a nenhuma dessas arestas, desejamos determinar quais arestas são visíveis a partir de p, isto é, determinar todos os segmentos de reta s tais que s contém um ponto q e o segmento de reta pq não intersecta nenhum outro segmento r pertencente a S. Ademais, desejamos também saber quais os trechos destes segmentos dados que são visíveis a partir de p, e não somente os segmentos. O algoritmo deve ter consumo de tempo proporcional a $O(n \lg n)$.

Proposta

Um segmento de reta em *S* é dado por dois pontos. Definimos para todo segmento de reta de *S* seus extremos como sendo um *extremo esquerdo* ou um *extremo direito* de acordo com a posição de *p* em relação a essa reta, de forma que *p* esteja sempre à esquerda de toda reta em *S* considerando a direção do *extremo esquerdo* para o *extremo direito.*

Segmentos de reta colineares em relação a *p* têm como *extremo esquerdo* o ponto mais próximo de *p*, e como *extremo direito* o outro extremo do segmento.

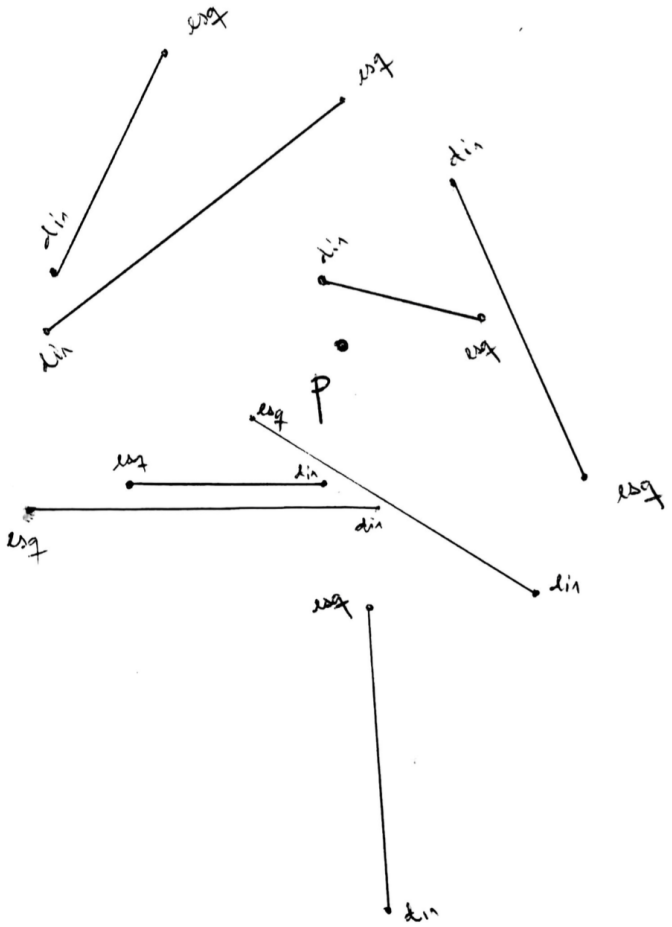

fig. 1: Um exemplo da convenção adotada de extremo esquerdo e direito do segmento de reta.

Propomos como forma de resolver o problema acima a implementação de uma linha de varredura em coordenadas polares dada por uma semirreta cuja origem é o ponto p dado, e sua direção inicial é $(p, (p_x + 1, p_y))$. Essa semirreta percorre todo o plano e, à medida que atravessa os segmentos de S, tem como invariante a seguinte proposição:

Todo ponto contido em uma reta de S já percorrido pela linha de varredura já teve sua visibilidade determinada.

Em um dado instante de tempo, o segmento de reta mais próximo de p intersectado pela linha de varredura nesse instante é visível. Em particular, sabemos que o ponto de interseção da linha de varredura com o segmento de reta é visível, e os pontos das retas mais distantes em relação a p não são visíveis.

Podemos descrever um trecho de visibilidade de um segmento de reta como um segundo segmento de reta, seus extremos sendo os pontos onde a visibilidade do segmento se inicia e onde essa visibilidade termina. O problema se resume a encontrar os pontos de início e término de visibilidade de um segmento de reta. Claramente, um segmento pode ter mais do que um trecho de visibilidade.

Os segmentos de reta intersectados pela linha de varredura podem ser ordenados de acordo com a distância do ponto u de interseção em relação a p. O segmento intersectado que apresenta a menor distância de u a p é visível, e o trecho de visibilidade se estende até o ponto em que o segmento termina, ou até o ponto onde um novo segmento se sobrepõe a sua área de visibilidade. Nesse caso, esse novo segmento recém-intersectado é percebido pelo seu

extremo esquerdo e, então, encerramos o trecho de visibilidade do antigo segmento mais próximo, à medida que abrimos um trecho novo para o segmento recém-inserido na linha de varredura.

Uma forma análoga de iniciar um novo trecho de visibilidade é pelo processamento de um *extremo direito* de um segmento. Nesse caso, o segmento de reta que contém o *extremo direito* é removido da linha de varredura, e o novo extremo mais próximo de *p* tem um trecho de visibilidade aberto.

Percebemos naturalmente que o comportamento da linha de varredura se diferencia no processamento de *extremos esquerdos* e *extremos direitos*. Quando um *extremo esquerdo* é processado, se o seu segmento for o mais próximo de *p*, ele abre um trecho de visibilidade. Quando um *extremo direito* é processado, se ele for o mais próximo de *p*, ele fecha seu trecho de visibilidade e abre um trecho para o segmento que, com sua remoção, se torna o mais próximo da reta (se tal segmento existir).

Logo, nossa linha de varredura tem dois pontos-eventos, o tratamento de *extremos direitos* e *extremos esquerdos*. Em cada um desses eventos, a linha de varredura atualiza qual o segmento de reta mais próximo de *p* e registra pela abertura e fechamento dos trechos de visibilidade quais trechos de aresta são visíveis.

Estrutura de dados

Os pontos-eventos de nossa linha de varredura proposta são os extremos dos segmentos de retas. Dado que cada reta tem dois extremos e temos n retas em S, precisávamos de uma estrutura que fizesse as operações de mínimo, inserção e remoção em $O(\lg n)$, de forma que a solução satisfizesse a restrição de consumo de tempo da resposta em $O(n \lg n)$. Outras estruturas utilizadas não têm impacto na questão da complexidade do algoritmo, apenas no encapsulamento das primitivas e do comportamento dos elementos de geometria. Estas serão explicadas nas seções posteriores desse relatório.

A linha de varredura de nosso algoritmo é representada por uma árvore AVL que armazena segmentos de reta; ela permite a execução de tais operações na complexidade requerida dado que a árvore AVL é uma Árvore de Busca Balanceada. Esta estrutura pode ser compreendida como o conjunto de segmentos de reta que são intersectados pela linha de varredura (semirreta com origem em p) em função do ângulo de rotação atual. Assim, a linha de varredura está intersectando um segmento se e, somente se, ele está inserido na árvore.

Também é importante ressaltar que não há interseção entre dois segmentos de S; disto decorre a seguinte propriedade:

(*) Segmentos de reta de S jamais trocam de ordem na árvore. Sendo assim, se um segmento a for dito menor que outro segmento b, isto permanecerá válido durante toda a coexistência dos segmentos na árvore.

Árvore de segmentos

Sejam *r* e *s* segmentos de reta atravessados pela linha de varredura em uma posição angular α. O critério de comparação entre estes segmentos na árvore é definido como segue:

i. *r* e *s* possuem os respectivos *extremos esquerdos* colineares em relação à origem *p* (caso degenerado)

Evidentemente, o menor segmento será aquele cujo *extremo esquerdo* estiver a menor distância de *p*. E, por (*****), esta é a única comparação necessária.

ii. *r* e *s* não possuem *extremos esquerdos* colineares em relação à origem *p*

Suponhamos, sem perda de generalidade, que o *extremo esquerdo* de *r* possui ângulo inferior ao ângulo do *extremo esquerdo* de *s* em relação à linha de varredura, isto é, *s* não estava presente na árvore quando *r* foi inserido. Logo, todos os pontos de *s* estão do mesmo lado de *r*, visto que ambos não se interseccionam. Por (*****), é suficiente determinar de que lado de *r* está o *extremo esquerdo* de *s*: se estiver do lado direito, temos que $r < s$; senão, $r > s$.

É interessante notar que o critério de comparação adotado é válido para quaisquer dois segmentos presentes na árvore em um determinado momento, independente do ângulo exato em que se encontra a linha de varredura. Portanto, o critério é consistente e bastante simples, podendo ser adotado tanto para a inserção quanto para a remoção de segmentos da árvore.

Geometria e detalhes de implementação

Utilizamos as primitivas Esquerda, Intersecta e Colinear na resolução desse problema. Além disso, implementamos também as classes *Point* e *LineSegment*, para melhor modelagem e modularização do problema. Problemas de ponto flutuante (igualdade, por exemplo) foram resolvidos com a classe *FloatingPoint* implementada em *precision.py*.

Solução

Nossa elegante solução tem como base a linha de varredura explicada acima e sua iteração sobre os pontos-eventos. Ordenamos todos os pontos-eventos de acordo com sua coordenada polar de centro em p (isto é, primeiro o ângulo, com critério de desempate de distância do ponto *p)*. A ordenação dos pontos é feita em $O(n \lg n)$. O algoritmo itera uma vez em cada ponto. Caso o ponto da iteração seja um *extremo esquerdo*, ele executa o caso 1. Senão, esse ponto é um *extremo direito*, e o algoritmo executa o caso 2.

No caso 1, chamado também de evento de inserção, um segmento novo é inserido na linha de varredura, representada pela árvore AVL. Esse segmento cujo *extremo esquerdo* está sendo processado é então inserido na árvore. Se o novo segmento for o elemento mínimo na árvore AVL então ele encerra o trecho de visibilidade do último tratado (o segundo menor). O encerramento do trecho de visibilidade é calculado a partir da interseção da linha de varredura

(definida com origem em *p* e o outro extremo no *extremo esquerdo* em tratamento nessa iteração) com o segmento cujo trecho de visibilidade acabou de ser encerrado.

O caso 2, chamado também de evento de remoção, ocorre quando um *extremo direito* é alvo da iteração do algoritmo. Neste caso, quando o *extremo direito* é processado (ou seja, seu segmento é removido da árvore), se o segmento mínimo na árvore foi alterado por esta remoção, um trecho de visibilidade é aberto para tal segmento. O cálculo do trecho de abertura segue analogamente ao caso 1.

Também vale ressaltar que, considerando como nosso algoritmo começa o processo de rotação a partir de um ângulo fixo (no nosso caso, trata-se da semirreta paralela ao eixo *x* com origem em *p* e direção +*x*), às vezes, é possível que a linha de varredura já contenha algum segmento aberto desde o início.

Na verdade, em alguns casos, é possível que a todo instante haja ao menos um segmento de reta na árvore, de forma que não existe ângulo onde a árvore inicial não necessite ser calculada. Este caso pode ser visualizado na figura abaixo:

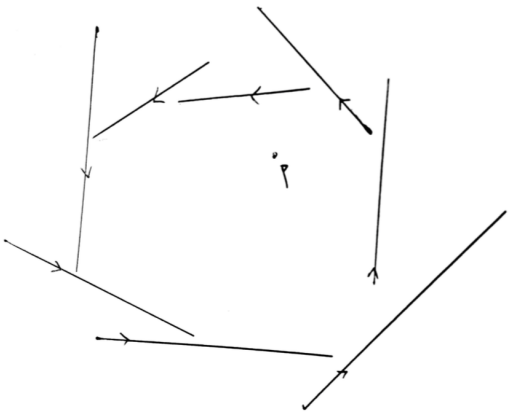

fig. 2: Um exemplo de entrada onde não há direção d cuja árvore inicial seja vazia.

Desta maneira, inicialmente verificamos quais segmentos são intersectados pela linha de varredura linearmente, através de uma rotação prévia; assim, os segmentos de reta cujos *extremos direitos* forem encontrados antes do que os respectivos *extremos esquerdos* são inseridos na árvore. É importante lembrar que convencionamos que segmentos cujos *extremos esquerdos* possuem mesma coordenada y e maior coordenada x que p são segmentos abertos.

Assim que todos os pontos-eventos forem processados, o algoritmo para e exibe a lista dos trechos de visibilidade.

Integração com plataforma Python

A animação do nosso algoritmo foi implementada utilizando as funções gráficas presentes na plataforma disponibilizada. A animação consiste na rotação da linha de varredura, representada por uma reta verde, em torno do ponto *p* recebido na entrada.

Os segmentos de reta fornecidos pelo problema são desenhados na cor vermelha, ao passo que o único ponto dado possui a cor branca. À medida que a linha de varredura rotaciona ao redor do ponto branco, pontos verdes são rapidamente desenhados e apagados, apenas para indicar os pontos que delimitam os segmentos de reta visíveis a partir de *p*. Ao final do algoritmo, todos os segmentos que forem visíveis estarão coloridos de amarelo, sendo que os demais trechos dos segmentos permanecerão na cor original (vermelho).

Problemas de implementação

A implementação original desse problema, como constava no exercício 7 da lista 3, não considerava a impressão dos trechos de visibilidade dos segmentos do conjunto *S,* apenas a verificação se um dado segmento era ou não visível. A adaptação de nossa primeira versão contou com diversos pequenos problemas que foram contornados à medida que nosso entendimento sobre a proposição amadurecia.

Pontos positivos

Familiarização com a linguagem Python, aprendizado aprofundado sobre o funcionamento, uso e implementação das primitivas de geometria e do problema proposto estão entre os pontos positivos da realização desse projeto.

Referências

Notas de aula, cadernos, slides e listas de exercício expostos no site da professora Cristina.