CI1338 - Primeiro Trabalho

Eduardo Mathias de Souza GRR20190428
ems19@inf.ufpr.br

Introdução

O trabalho consiste em fazer a implementação da classificação de polígonos e pontos interiores.
Dados um conjunto S de linhas poligonais fechadas no plano e um conjunto P de pontos no plano, deve-se encontrar a classificação de cada uma das linhas poligonais fechadas do conjunto S como polígonos “não simples”, “simples e não convexos” ou “simples e convexos”. E, também, encontrar se cada ponto do conjunto P de pontos se encontra dentro ou fora de cada polígono do conjunto S.

Implementação

A resolução do trabalho foi feita por um algoritmo desenvolvido em C que possui duas estruturas de dados Point (definida por um double x e um double y, para representar respectivamente a coordenada x e y do ponto) e Polygon (definida por um array de Point e um inteiro para guardar o número de pontos).
Lê-se da linha de entrada com scanf e criamos um vetor da estrutura Polygon para guardarmos todos os polígonos passados. Após essa primeira etapa, criamos também um vetor de Point para guardar os pontos do conjunto P que serão passados.
Após definida a estrutura de dados, definimos os algoritmos da seguinte maneira:

- Descobrir se um polígono é simples ou complexo:
    1. Iterar através de cada aresta do polígono.
    2. Para cada aresta, iterar para todas as arestas após ela (em ordem anti-horária) e checar se ela se cruza com a aresta atual.
        a. Se sim, o polígono não é simples.
        b. Se não houver interseções, continuar a iteração através das arestas até que todos os pares tenham sido verificados.
    3. Depois de verificar todas, se não houver interseções, o polígono é simples.

Para implementação desse algoritmo, utilizamos uma função chamada 'do edges intersect' que recebe duas arestas/dois segmentos dentro do nosso laço e verifica se elas interceptam usando o método do Determinante, assim como apresentado na imagem abaixo:

$$P_x = \frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

$$P_y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}$$

Se os determinantes tiverem sinais opostos, as linhas se interceptam, a função retorna True e o laço acaba. Caso contrário, a função retorna False para aqueles dois segmentos e continuamos as verificações par a par.

A corretude do algoritmo é de $O(n^2)$, onde n é o número de vértices do polígono. Isto porque, para calcular o Determinante, utilizamos equações lineares que levam O(1) e no pior caso, o algoritmo vai fazer esse cálculo para todos os pares de arestas possíveis para interseções do polígono, $O(n^2)$. $O(1)*O(n^2) = O(n^2)$.

- Descobrir se o polígono simples é convexo
    1. Iterar através de cada um dos vértices do polígono e verificar o ângulo entre ele e dois vértices subsequentes:
        a. Se o ângulo for maior que 180 graus, então o polígono não é convexo e encerramos a iteração
        b. Se o ângulo for menor que 180 graus, o continuamos a iteração
    2. Para um polígono ser convexo, todos os ângulos devem ser menor que 180 graus entre arestas adjacentes

Para implementação desse algoritmo, utilizamos a função de produto vetorial que recebe os três vértices da iteração (p_i, p_j, p_k) , e calcula o produto vetorial entre os vetores (p_i-p_j e p_j-p_k) para determinar se o ângulo formado pelos vértices. Se o produto vetorial for negativo, o polígono não é convexo.

A corretude do algoritmo é O(n), onde n é o número de vértices do polígono. O custo para calcular o produto vetorial é O(1), e no pior caso, o algoritmo vai realizar esse cálculo para todos os vértices do polígono, O(n). $O(1)*O(n) = O(n)$.

- Descobrir se um ponto está dentro de um dado polígono

  A ideia básica do algoritmo é traçar uma linha horizontal do ponto ao infinito, e contar o número de vezes que a linha intercepta o polígono. Se o número de interseções for ímpar, então o ponto está dentro do polígono; se o número de interseções for par, então o ponto está fora do polígono. Ou seja:
    1. Inicializar um contador em 0 e iterar sobre cada aresta do polígono e checar se a linha horizontal do vetor do ponto dado ao infinito intercepta essa aresta
        a. Caso haja essa intersecção, incrementa em 1 o contador
    2. Se o número de vezes que o vetor do ponto ao infinito intersecta as arestas de um polígono for par, então o ponto está fora do polígono
        a. Caso contrário, o ponto está dentro do polígono

  Para implementação desse algoritmo, durante o loop verificamos se os vértices da aresta estão em lados opostos ao vetor do ponto ao infinito (com a condicional: ((polygon[i].y > point.y) != (polygon[j].y > point.y))) e se o ponto de interseção do vetor com a aresta está à esquerda do ponto sendo verificado (com a condicional (point.x < (polygon[j].x - polygon[i].x) * (point.y - polygon[i].y) / (polygon[j].y - polygon[i].y) + polygon[i].x), que pode parecer confusa mas estamos calculando o ponto x da interseção do vetor infinito com a aresta com o auxílio da equação da linha e verificando se o ponto x do ponto dado é menor do que o ponto x da intersecção). Caso seja positivo para as duas condicionais, incrementa o contador de intersecção.

  A corretude do algoritmo é O(n) pois verificamos com custo O(1) as intersecções e realizamos essa verificação para n vértices do polígono

Exemplos utilizados para teste:

1)

3 3
4
1 4
15 4
15 20
1 20
3
8 12
25 8
25 18
5
2 6
2 1
15 2
20 1
20 6
12 10
25 2
5 5

Nesse caso temos como saída:
1 simples e convexo
2 simples e convexo
3 simples e nao convexo
1:1
2:
3:1 3

2)

2 2
6
0 0
5 0
5 5
3 3
5 4
0 4
4
0 0
1 1
2 0
1 -1
1 0.5
0 0

Nesse caso temos como saída:
1 nao simples
2 simples e nao convexo
1:1 2
2:1 2

3)

3 3
4
1 4
15 4
15 20
1 20
3
8 12
25 8
25 18
5
2 6
2 1
15 2
20 1
20 6
12 12
25 2
5 5

Nesse caso temos como saída:
1 simples e convexo
2 simples e convexo
3 simples e nao convexo
1:1 2
2:
3:1 3

Referências:
https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection#Given_two_points_on_each_line
https://stackoverflow.com/questions/471962/how-do-i-efficiently-determine-if-a-polygon-is-convex-non-convex-or-complex
https://web.archive.org/web/20060613060645/http://softsurfer.com/Archive/algorithm_0108/algorithm_0108.htm#Test%20if%20Simple
https://www.geeksforgeeks.org/check-if-given-polygon-is-a-convex-polygon-or-not/
https://rosettacode.org/wiki/Ray-casting_algorithm#top-page