Trabalho de Algoritmos 2

Daniel Augusto Costa de Sá

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil
danielaugusto191@ufmg.br
https://github.com/DanielAugusto191

1. Modelagem

Para atender as especificações do problema, separei o problema inteiro em duas partes: Encontrar a triangulação do poligono usando o algoritmo Ear Clipping, e encontrar o grafo 3-Colorado para essa triangulação.

Para o algoritmo de Ear Clipping, foram usados listas de tuplas para armazenar os pontos do poligono, outra lista onde cada posição da lista é um triangulo na triagulação do poligono que consiste em uma tupla com os indices de cada ponto do triangulo. Além disso, foram usados outras listas e variaveis auxiliares e um set para armazenar os vertices “orelhas”.

Para a coloração do grafo, usei o algoritmo de Breadth-first search (BFS) para o resultado, que por sua vez, usei uma biblioteca externa Queue que implementa filas, alem da lista de visitados, cores, e apartir daqui para obter o passo-a-passo da coloração os pontos foram redefinidos como uma lista de dicionarios que possui as chaves “ponto” e “color” que tem como item o ponto atual e sua coloração respectivamente.

2. Implementação

Para o algoritmo de Ear Clipping, começo usando uma função Clockwise para saber se os pontos foram dados em ordem correta, senão invertemos a ordem. Depois crio uma lista de adjacencia onde a i-esima possição armazena os vertices adjacentes a I.

Com a lista de pontos e de adjacencia prontas podemos identificar quais vertices são vertices “orelhas” (usei *ears* no código), para isso usei a função ear que usa o triangulo formado por I e seus vertices adjacentes e verifica se há um ponto P dentro deste triangulo, se houver não é vertices orelha. Entao removemos cada vertices orelha atualizando seus

adjacentes podendo se tornar ou deixar de ser um vertice orelha, e adicionando a resposta à lista final.

Enfim, obtemos a triangulação do poligono armazenado na lista *polig*.

Para a coloração, seguimos uma simples BFS, adiciono uma cor qualquer a todos os pontos e inicializamos a queue. Adicionamos um novo ponto se houver 2 verticies que pertecem ao triangulo atual e também pertecem a outro triangulo, e alteramos a cor do verticies que pertece ao primeiro triangulo mas não ao segundou ou que pertence ao segundo triangulo mas não ao primeiro. Com isso conseguismos a coloração do grafo.

3. Complexidade

Seja n o número de pontos (vertices), a complexidade de tempo geral, para ambos os metodos é $O(n^2)$.

Em Ear Clipping temos que verificar se cada ponto é um vertices “orelha” para isso temos que passar por todos os outros pontos.

Na BFS para cada triangulo temos que verificar todos os outros se são adjacentes.

A complexidade de espaço permanece O(N) pois apenas adicionamos e removemos os pontos do poligno original.

4. Referencias

Queue - https://docs.python.org/3/library/queue.html

BFS - https://cp-algorithms.com/graph/breadth-first-search.html

Ear Clipping - http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/97/Ian/introduction.html + Material dado em Aula