Trabalho Prático 1
Problema da Galeria de Arte

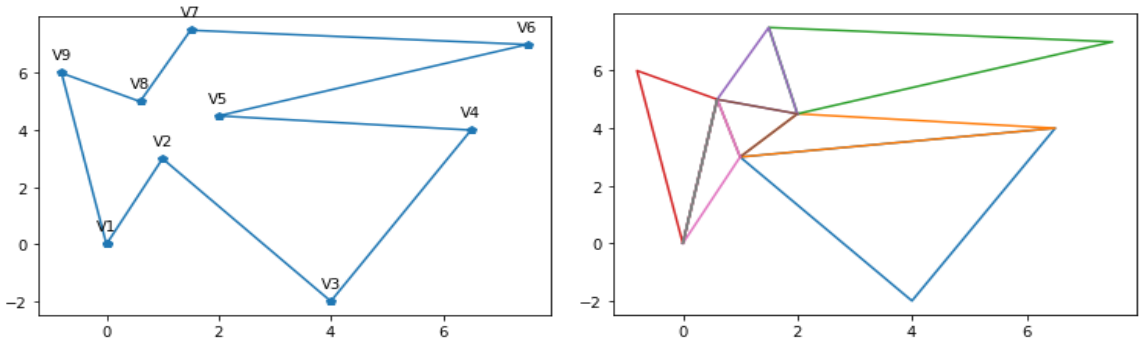
Arthur Veloso Kuahara

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

arthurvk@ufmg.br

1. Introdução

O primeiro trabalho prático da disciplina de Algoritmos II consiste em resolver o problema da galeria de arte, isto é, triangular um espaço (no caso do TP, um polígono convexo) para determinar a quantidade e o posicionamento de guardas que observem todo o espaço. Para isso, devemos implementar o algoritmo Ear-Clipping, que triangula um polígono, e depois 3-colorizar o grafo resultante para determinar o posicionamento dos guardinhas. As imagens abaixo mostram o polígono original e depois ele já triangulado.



2. Implementação

Para começar, vamos usar as bibliotecas do matplotlib e ggplot para exibir os triângulos. Esses e outros imports (utilizados no arquivo dos polígonos de teste) são feitos logo no começo do notebook.

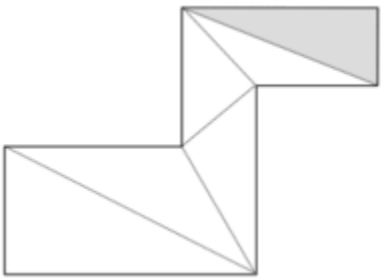
```
#imports

import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from functools import reduce
from itertools import chain
from plotnine import ggplot, aes, geom_polygon, geom_segment, geom_point

import math
import sys
from collections import namedtuple
Point = namedtuple('Point', ['x', 'y'])
EPSILON = math.sqrt(sys.float_info.epsilon)
```

Depois, iremos implementar algumas funções que auxiliarão no processo de triangulação - funções que identificam se o sentido de rotação da aresta a seguir é o sentido horário (conforme visto na aula), que retornam área do triângulo, que identificam se um certo ponto está dentro de um triângulo, entre outros.

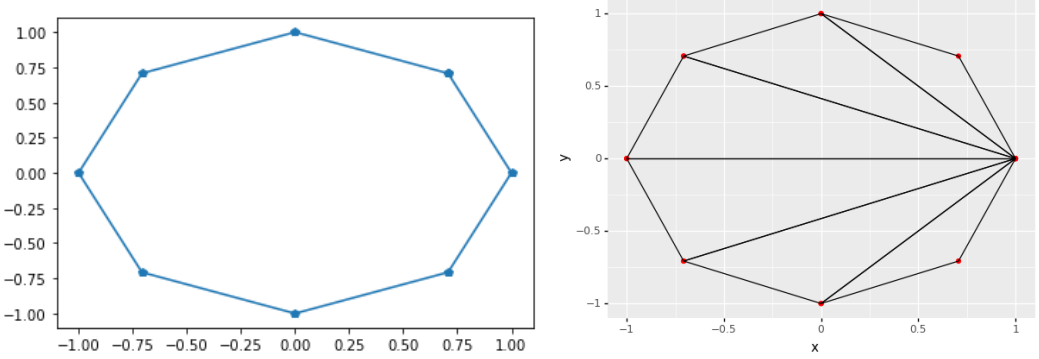
Por fim, com a utilização dessas funções auxiliares, finalmente podemos começar a implementação do algoritmo Ear-Clipping.



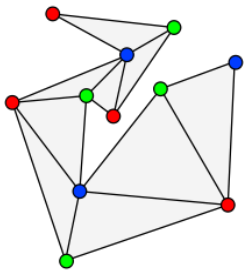
O algoritmo funciona identificando as 'orelhas' do polígono e as removendo, até sobrar apenas um triângulo. A complexidade de tal algoritmo é $O(n^2)$.

Uma 'orelha' de um polígono é um triângulo que possui dois de seus lados sendo as bordas do próprio polígono, e um lado sendo interior a ele. Encontrar tais 'orelhas' possibilita a triangulação.

A função 'earclip' implementada recebe os pontos do polígono e retorna uma lista contendo os triângulos formados a partir da triangulação. Plotar a lista resultante utilizando a biblioteca ggplot resulta em um plot como esse a seguir. A imagem à esquerda é o matplotlib do polígono original, e a da direita é o ggplot do polígono já triangulado.



Uma vez triangulados os polígonos, devemos então começar a realizar a 3-coloração dos mesmos. A 3-coloração de um polígono revela corretamente os conjuntos de guardas possíveis para um mesmo espaço.

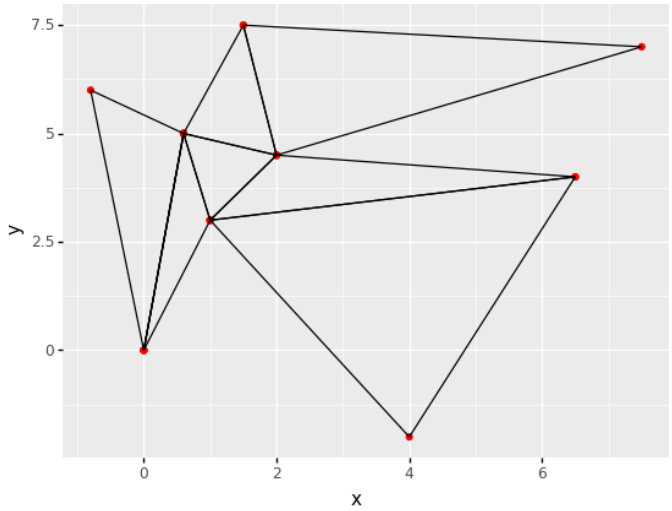


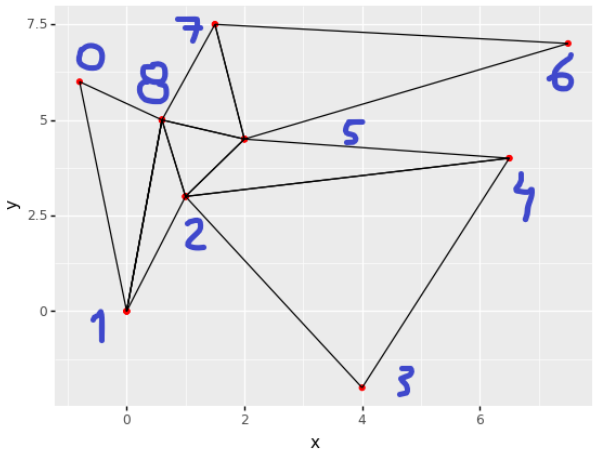
Na imagem à esquerda, temos uma 3-coloração de um polígono já triangulado. Qualquer uma das 3 cores pode ser utilizada, entretanto, como só temos 3 pontos em azul, concluímos que é possível termos apenas 3 guardas para esse espaço, sem a necessidade de usar 4 , como sugerido pelas cores vermelha e verde.

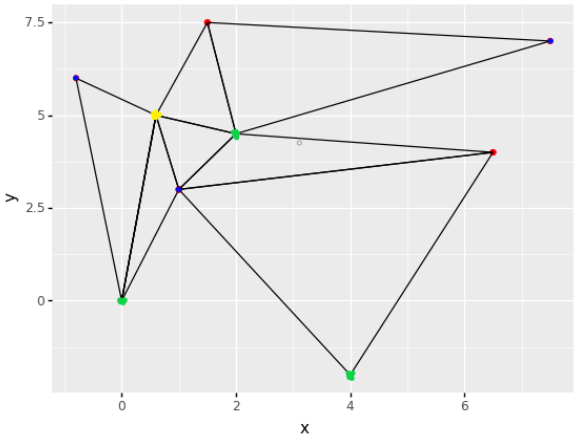
Para fazer isso, implementamos uma classe ‘Graph’ que armazena o grafo que será utilizado para a coloração, e uma função colorGraph que recebe o grafo criado, e irá fazer a coloração em si.

Implementadas, podemos de fato fazer a coloração e finalmente encontrar a solução para o problema.

O grafo utilizado como exemplo é o gerado através de *points2,* isso é, [(0,0), (1,3), (4,-2),(6.5,4),(2,4.5),(7.5,7),(1.5,7.5),(0.6,5),(-0.8,6),(0,0)].







Com isso, concluímos que apenas dois guardas são suficientes para cobrir o espaço inteiro da galeria - os que estão localizados onde a cor do ponto é vermelho.