Algoritmos II
Trabalho Prático 1:
Problema da Galeria de Arte

Antônio Isaac Silva Lima - Matrícula 2019006361

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

11 de julho de 2021

isaaclima1934@ufmg.br

1 Introdução

O problema da Galeria de Arte se trata em descobrir qual é, aproximadamente, o menor número de camêras necessárias para vigiar o interior de uma galeria de arte por inteiro.

Para resolver tal questão, começamos a partir de um polígono simples o qual representa o interior da galeria de arte e realizamos dois passos: primeiramente fazemos a triangulação do polígono, quebrando-o em triângulos e fazemos a 3-coloração dos vértices.

Com a 3-coloração, colorimos todos os vértices do polígono utilizando 3 cores distintas de forma a que nenhum par de vértices conectados por uma aresta (seja do polígono original ou uma aresta criada para formar os triângulos) tenham a mesma cor.

Tendo completado esses passos, uma boa resposta para o problema da Galeria de Arte é que são necessárias n camêras, sendo n o número de vezes que a cor menos utilizada foi usada para colorir os vértices.

2 Implementação

O programa foi feito na linguagem Python, utilizando o *Jupyter Notebook*.

2.1 Estrutura de Dados

A única estrutura de dados utilizada para a realização desse trabalho prático foi a lista (a *built-in* de Python). Todas as outas estruturas de dados utilizadas foram as classes *Point*, utilizada para representar pontos em um sistema cartesiano de 2 dimensões, e *Polygon*, utilizada para representar um polígono no mesmo sistema cartesiano de 2 dimensões.

Ambas as estruruas de dados criadas serão detalhadas na próxima seção.

2.2 Classes

À fins de organização, as estruturas descritas acima foram encapsuladas classes que modelam um polígono em um plano bidimensional.

A primeira classe criada, *Point*, foi usada, especificamente, para definir os vértices do polígono com dois inteiros x e y e possui apenas métodos básicos de representação, inicialização e subtração (esse último foi criado a fins de conveniência e organização).

Já a segunda classe criada, *Polygon*, é a representação do polígono em si. Ela possui uma lista de objetos do tipo *Point* que atuam como os vértices do polígono. Além disso, essa classe

possui algumas listas auxiliares que facilitam a lógica necessária para a resolução do programa e um objeto do tipo *HoloView.HoloMap*, o qual é utilizado para criar as representações gráficas dos passos tomados pelos algoritmos de triangulação e 3-coloração.

A classe *Polygon* também possui os seguintes métodos:

- Um construtor que inicializa todas as variáveis pertinentes à classe;
- Métodos que fazem a representação do polígono de forma apropriada para a função *print*;
- Um método *add_vertice* que realiza a adição de vértices (objetos do tipo *Point*) ao polígono;
- Um método que realiza a triangulação e guarda o resultado (os triângulos formados) em uma lista pertencente à classe;
- Um método que realiza a 3-coloração à partir do resultado da triangulação;
- E, por fim, um método que chama ambos os métodos de triangulação e 3-coloração e, com a resposta de ambos, calcula a solução para o problema da Galeria de Arte e retorna a resposta no formato de um inteiro;

Os detalhes minuciosos sobre a implementação das estruturas de dados e classes comentadas (além de função ajudantes) podem ser encontradas no código em si, através de comentários.

3 Conclusão

O trabalho teve grande efetividade em reforçar os conceitos vistos em aula acerca de programação geométrica, além de introduzir os alunos à bibliotecas não comuns, como a *HoloView*.