Construindo um Editor de Grafos

Informações

- Atividade individual;
- Entrega pelo Moodle em <`http://trab.dc.unifil.br/moodle/`>;
- Peso deste trabalho: **4**.

Para a realização das atividades seguintes, parta do projeto `EditorGrafos`, incluso no pacote deste roteiro.

1. Comece sua implementação pelo pacote `Grafos`, necessário ao pleno funcionamento do editor de grafos e todos os algoritmos envolvidos:
   (a) Implemente a classe `Aresta`, lembrando que toda aresta liga dois vértices e possui atributos próprios – definidos pelo usuário programador.
   (b) (20 pontos) Implemente a classe `GrafoAdjacência`, crie *unit tests*, e verifique-a.
   (c) (20 pontos) Faça o mesmo pela classe `GrafoMatriz`.
2. Agora que todo o pacote `Grafos` está implementado e verificado, implemente as funcionalidades de operação do editor de grafos.
   (a) (5 pontos) Implemente a funcionalidade "mover vértice". Já há um botão na interface para abrigá-la[1].
   (b) (5 pontos) Implemente a funcionalidade "ligar vértices", acionada pelo botão homônimo já existente no projeto. Quando ativada, a funcionalidade inicia quando o usuário segurar o mouse em cima de um vértice e arrastar o ponteiro até outro vértice qualquer. Enquanto o usuário arrasta o mouse, o programa deverá mostrar uma linha entre o vértice inicial e o ponteiro do mouse. Se o usuário soltar o botão do mouse sem ligar a aresta a outro vértice, a operação é descartada.
   (c) (5 pontos) Implemente o método `mouseSobreAresta` na classe `DesenhistaGrafo`, que identifica quando o ponteiro do mouse estiver sobre uma aresta qualquer do grafo. Utilize fundamentos de geometria analítica [3]. Esse método será utilizado nas funcionalidades a seguir, em que é necessário escolher uma aresta. Assim que o método estiver implementado, você deverá implementar em `paintComponent` um código para colorir a aresta atual sob o mouse.
   (d) (5 pontos) Implemente a funcionalidade "renomear vértice ou aresta". É necessário cria um novo botão. Quando o modo estiver acionado, o usuário clica em cima de um vértice ou aresta e digita os novos atributos do elemento em uma caixa de texto `JOptionPane`.

[1] Essa questão foi feita em sala de aula com orientação do professor para ambientar os alunos ao código do projeto.

(e) (5 pontos) Implemente a funcionalidade "remover vértice ou aresta". É necessário criar um novo botão.

3. Quando as funcionalidades de edição do programa estiverem funcionando plenamente, implemente os seguintes algoritmos de busca:

   (a) (15 pontos) Busca em extensão (BFS, do inglês *breadth-first search*).

   (b) (15 pontos) Busca em profundidade (DFS, do inglês *depth-first search*).

   Para cada uma dessas buscas, é necessário criar um novo botão. Quando o usuário acionar qualquer uma delas, o programa pergunta o nome do vértice de origem e o nome do vértice buscado. Se for encontrado um caminho, o programa colore as arestas que formam o caminho. Se não for encontrado, o programa avisa através de uma mensagem.

4. (20 pontos) Implemente a funcionalidade "calcular distâncias para um vértice". Para tal, crie um novo botão que, quando acionado, permite ao usuário escolher um vértice com o mouse, e então o programa exibe um diálogo que contém as distâncias de todos os vértices para este. Para resolver o problema, utilize o algoritmo de *Dijkstra*.

Referências

[1] CORMEN, Thomas H. et al. *Algoritmos: teoria e prática*. Rio de Janeiro: Campus, 2012. 926 p. ISBN 978-85-352-3699-6.

[2] GOODRICH, Michael T.; TAMASSIA, Roberto. *Estruturas de dados e algoritmos em Java*. 4. ed. Porto Alegre: Bookman, 2007. 600 p. ISBN 9788560031504.

[3] BIEZUNER, Rodney J.. *Retas e Planos. Seção "Distância entre um Ponto e uma Reta"*. Disponível em: <`http://www.mat.ufmg.br/~rodney/notas_de_aula/retas_e_planos.pdf`>.