Trabalho de Implementação - Relatório

Algoritmos e Grafos — 2021/1 REMOTO

Circuito Euleriano

Marcos Eduardo de Souza – 114097955

1. Implementação da solução

A estratégia abordada foi o algoritmo de Fleury:

1. Verificar se o grafo é conexo e se:
   - **a)** Tem todos os vértices com grau par (grafo de Euler)
   - **b)** Tem 2 vértices com grau ímpar.
2. Escolher um vértice para começar e assinalá-lo, no caso (a) pode ser qualquer um, já no caso (b) terá de ser um dos vértices de grau ímpar.
3. Percorrer qualquer uma das arestas incidentes nesse vértice desde que esta não seja uma aresta de corte (ponte) para a parte não atravessada do grafo, isto é, de modo que o grafo formado pelas arestas que ainda faltam percorrer (e respetivos vértices) se mantenha conexo ao removermos as arestas percorridas, só atravessando a aresta de corte caso não reste outra hipótese.
4. Repetir o passo anterior a partir do vértice onde chegou até que não restem mais arestas a percorrer, no caso (a) terminará no vértice onde começou, já no caso (b) terminará no outro vértice de grau ímpar.

A implementação foi feita em Java, onde no arquivo principal fica o menu para o usuário, com 4 opções, carregar o grafo de um arquivo .txt, imprimir o grafo na tela, executar o algoritmo de Fleury e sair. Quando escolhido o algoritmo de Fleury, o usuário precisará passar o número do vértice que ele deseja que a busca inicie.

Agora falando sobre as funções importantes implementadas, temos a função $getOddVerticesCount( )$ que é responsável por fazer a verificação da quantidade de vértices ímpares no grafo, com a verificação feita podemos analisar em qual caso iremos fazer, caso seja o (b) a função $getEulerPathOrigin( )$ faz uma busca pelo grafo ate encontrar o primeiro vértice de grau ímpar. Caso seja o (a), já adicionaremos na lista o vértice v_0 que o usuário escolheu que inicialize a busca. Dentro da função $FleutyAlgorithm( )$, percorremos os vizinhos do v_0 e a cada interação verificamos se a aresta do vizinho que pretendemos percorrer é uma aresta de corte ou não, através da função $isBridge( )$ que verificar se existe mais de uma possibilidade de aresta, caso existe, verificamos a quantidade de vértices conectados, fazemos a remoção a aresta e refazemos uma nova verificação da quantidade de vértices e colocamos a aresta novamente, caso a segunda verificação seja menor que a primeira, então tiramos uma aresta de corte. Ao retornar para $FleutyAlgorithm( )$, caso não seja uma aresta de corte iremos adicionar o vértice na lista de caminho e agora removendo definitivamente essa aresta. Com isso a função entrar na recursividade fazendo uma buscar em profundidade. Ao terminar iremos imprimir a lista de caminho, começando pelo vértice definido pelo usuário e seguido pelos demais encontrados: $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \cdots \rightarrow v_0$

2. Referências Bibliográficas

- https://pt.wikipedia.org/wiki/Caminho_euleriano
- http://www.inf.ufsc.br/grafos/temas/euleriano/euleriano.htm
- http://www.gpec.ucdb.br/pistori/disciplinas/discreta/aulas/geulham.htm
- https://www.youtube.com/watch?v=kQaDAWGBuGo&ab_channel=PatriciaJaquesMaillard
- https://pt.wikipedia.org/wiki/Algoritmo_de_Fleury#:~:text=O%20algoritmo%20de%20Fleury%20%C3%A9,as%20arestas%20de%20um%20grafo.&text=Um%20grafo%20que%20n%C3%A3o%20cont%C3%A9m,ser%C3%A1%20chamado%20grafo%20semi%2Deuleriano.
- https://www.youtube.com/watch?v=mRvt5yJTw7E&ab_channel=Matem%C3%A1ticaParaGenteGrande
- http://www.decom.ufop.br/marco/site_media/uploads/bcc204/19_aula_19.pdf