Documentação das Funções: Análise de Grafos

Mensagens gerais

retornoInsiraValor() e retornoValorInvalido()

- **Propósito:** São funções simples que retornam strings para interações com o usuário. Servem como mensagens padrão para solicitar um valor ou indicar que o valor inserido é inválido.
- **Conexão com Grafos:** Não diretamente relacionado à teoria dos grafos, mas é útil para interação com o usuário na inserção de dados sobre o grafo.

Grafo conexo

bfsConexo()

- **Propósito:** Realiza uma busca em largura (BFS) para verificar a conectividade de um grafo a partir de um vértice inicial.
- **Entrada:** Recebe o mapa de adjacência do grafo, o vértice inicial e um conjunto de vértices visitados.
- **Saída:** Marca os vértices conectados ao vértice inicial como visitados.
- **Conexão com Grafos:** Utiliza o algoritmo de BFS, que é fundamental na análise de conectividade de grafos.

ehConexo()

- **Propósito:** Verifica se um grafo é conexo, ou seja, se existe um caminho entre qualquer par de vértices.
- **Entrada:** Recebe uma lista de arestas, vértices e um booleano que indica se o grafo é direcionado.
- **Saída:** Retorna `true` se o grafo for conexo e `false` caso contrário.
- **Conexão com Grafos:** Fundamental para determinar a conectividade do grafo. Usa o resultado de `bfsConexo()` para verificar se todos os vértices podem ser alcançados.

Grafo bipartido

ehBipartido()

- **Propósito:** Verifica se um grafo é bipartido, ou seja, se é possível dividir os vértices do grafo em dois conjuntos disjuntos, de forma que nenhuma aresta conecte vértices do mesmo conjunto.
- **Entrada:** Recebe uma lista de arestas, vértices e um booleano que indica se o grafo é direcionado.
- **Saída:** Retorna `true` se o grafo for bipartido e `false` caso contrário.
- **Conexão com Grafos:** Utiliza uma coloração dos vértices (bipartição) para verificar se o grafo pode ser dividido sem que haja conexões internas em qualquer um dos dois conjuntos.

Grafo eureliano

ehEureliano ()

- **Propósito:** Determina se um grafo é Euleriano, ou seja, se existe um circuito que passa por todas as arestas do grafo exatamente uma vez.
- **Entrada:** Recebe uma lista de arestas, vértices e um booleano que indica se o grafo é direcionado.
- **Saída:** Retorna `true` se o grafo for Euleriano e `false` caso contrário.
- **Conexão com Grafos:** Um grafo não direcionado é Euleriano se for conexo e todos os vértices têm grau par. Para grafos direcionados, todos os vértices devem ter grau de entrada igual ao grau de saída.

Arestas pontes

contarArestasPonte()

- **Propósito:** Verifica se uma aresta em um grafo é uma "ponte", ou seja, uma aresta cuja remoção desconecta o grafo.
- **Entrada:** Recebe uma lista de arestas, vértices e a aresta que será verificada.
- **Saída:** Retorna `true` se a aresta for uma ponte e `false` caso contrário.
- **Conexão com Grafos:** Identificar pontes em um grafo é crucial para entender sua robustez e vulnerabilidade, especialmente em redes.

Grafo de articulação

imprimirArticulacoes()

- **Propósito:** Verifica se um vértice é uma "articulação", ou seja, um vértice cuja remoção aumenta o número de componentes conexos do grafo.
- **Entrada:** Recebe uma lista de arestas, vértices e o vértice que será verificado.
- **Saída:** Retorna `true` se o vértice for uma articulação e `false` caso contrário.
- **Conexão com Grafos:** A identificação de articulações é importante para a análise da conectividade e resiliência de redes complexas.

Leitura de grafo

entradaDados()

- **Propósito:** Inicializa e cria a estrutura básica do grafo, incluindo os vértices e arestas a partir dos dados fornecidos.
- **Entrada:** Recebe os vértices e arestas como parâmetros e constrói a representação do grafo.
- **Saída:** Retorna a estrutura de dados que representa o grafo (geralmente uma matriz de adjacência ou lista de adjacência).
- **Conexão com Grafos:** Esta função é fundamental, pois configura a estrutura de base sobre a qual todas as outras operações e análises de grafos serão realizadas.

Análise de Grafo

- **Propósito:** Exibe a representação do grafo de uma maneira compreensível.
- **Entrada:** Recebe a estrutura do grafo criada pela leitura realizada no início da função
- **Saída:** Imprime a estrutura do grafo no console.
- **Conexão com Grafos:** Facilita a visualização do grafo, o que é essencial para a compreensão e análise das suas propriedades.

Análise de grafo

entradaDados()

- **Propósito:** Exibe a representação do grafo de uma maneira compreensível.
- **Entrada:** Recebe a estrutura do grafo criada pela leitura realizada no início da função
- **Saída:** Imprime a estrutura do grafo no console.
- **Conexão com Grafos:** Facilita a visualização do grafo, o que é essencial para a compreensão e análise das suas propriedades.

Execução

menu()

- **Propósito:** Controla o fluxo do programa, oferecendo opções para o usuário executar diferentes análises e operações no grafo.
- **Entrada:** Não recebe parâmetros, mas interage com o usuário para capturar as opções escolhidas.
- **Saída:** Chama as funções correspondentes com base nas escolhas do usuário.
- **Conexão com Grafos:** Serve como interface de interação, conectando o usuário às diferentes funcionalidades de análise de grafos implementadas no código.

Execução

main()

- **Propósito:** Função principal que inicializa o programa, chama o menu e gerencia a execução geral.
- **Entrada:** Não recebe parâmetros diretamente, mas pode processar entradas do usuário através do menu.
- **Saída:** Controla a execução do programa até que o usuário decida encerrar.
- **Conexão com Grafos:** É o ponto de entrada do programa que coordena todas as operações relacionadas à criação e análise de grafos.