SIN110 – Algoritmos e Grafos
Prof. Rafael Frinhani

ATIVIDADE 4 (ATV4)

Assunto: 2. Teoria dos Grafos – Isoformismo e Planaridade.

Data de Entrega: 19/09 até as 21:00h.

1. **Objetivo:** Verificar o aprendizado de conceitos básicos sobre grafos isomorfos e grafos planares, implementar uma lista de adjacências para representação de grafos e funções para operações básicas nesta estrutura.

2. **Descrição:** A atividade consiste na realização de (*i*) Teste para verificação de conhecimentos teóricos sobre grafos; (*ii*) Implementação de uma lista de adjacências e operações básicas. As atividades a serem executadas são:

   i. Estude o conteúdo sobre "2. Teoria dos Grafos – Isomorfismo e Planaridade" pelos *slides* da aula, se possível complementando com as referências bibliográficas da disciplina. Em seguida responda o teste relacionado a este conteúdo, o qual tem duração de 80 minutos. Atenção: Você terá apenas 01 (uma) tentativa. Estude primeiro e faça o teste depois.

   ii. Considerando o protótipo desenvolvido na ATV1 ou o código disponibilizado no SIGAA (app_grafos), implemente as funções descritas a seguir. Utilize a função de visualização para plotar os grafos gerados e validar suas implementações (se necessário adapte-a). Inclua as imagens no relatório como evidência do resultado (ex. antes e depois da execução da função):

   - `criaListaAdjacencias(matriz)`
     Descrição: Cria uma lista de adjacências de um grafo representado por uma matriz de adjacências.
     Entrada: matriz de adjacências (arquivo .txt)
     Saída: lista de adjacências (tipo *Dictionary*)

   - `tipoGrafo(listaAdj)`
     Descrição: Retorna o tipo do grafo representado por uma dada lista de adjacências.
     Entrada: lista de adjacências (tipo *Dictionary*)
     Saída: Integer (0 – simples; 1 – dígrafo; 2 – multigrafo; 3 – pseudografo)

   - `verificaAdjacencia(listaAdj, vi, vj)`
     Descrição: Verifica se os vértices v_i e v_j são adjacentes.
     Entrada: lista de adjacências (tipo *Dictionary*), v_i e v_j (ambos números inteiros que indica o id do vértice)
     Saída: Boolean (*True* se os vértices são adjacentes; *False* caso contrário)

   - `calcDensidade(listaAdj)`
     Descrição: Retorna o valor da densidade do grafo.
     Entrada: lista de adjacências (tipo *Dictionary*)
     Saída: Float (valor da densidade com precisão de três casas decimais)

   - `insereAresta(listaAdj, vi, vj)`
     Descrição: Insere uma aresta no grafo considerando o par de vértices v_i e v_j.
     Entrada: lista de adjacências (tipo *Dictionary*), v_i e v_j (ambos são números inteiros que indicam o id do vértice)
     Saída: lista de adjacências (tipo *Dictionary*) com a aresta inserida.

   - `insereVertice(listaAdj, vi)`
     Descrição: Insere um vértice no grafo.
     Entrada: lista de adjacências (tipo *Dictionary*), v_i (número inteiro que indica o id do vértice)
     Saída: lista de adjacências (tipo *Dictionary*) com o vértice inserido.

   - `removeAresta(listaAdj, vi, vj)`
     Descrição: Remove uma aresta do grafo considerando o par de vértices v_i e v_j.
     Entrada: lista de adjacências (tipo *Dictionary*), v_i e v_j (ambos são números inteiros que indicam os ids dos vértices)
     Saída: lista de adjacências (tipo *Dictionary*) com a aresta removida.

- `removeVertice(listaAdj, vᵢ)`
  Descrição: Remove um vértice do grafo.
  Entrada: lista de adjacências (tipo *Dictionary*), v_i (número inteiro que indica o id do vértice)
  Saída: lista de adjacências (tipo *Dictionary*) com o vértice removido.

3. **Entrega:** A entrega deverá ser feita exclusivamente pelo SIGAA (e-mails não serão aceitos). Enviar 01 (um) arquivo .pdf com o seguinte:

   - O documento deverá incluir um cabeçalho com o nome da disciplina, seu nome e número de matrícula, além de identificar o número da atividade que ele corresponde e a data de entrega.

   - Este documento deverá conter um relatório com no máximo 03 (três) páginas que mostre as evidências da sua implementação e descreva brevemente as questões técnicas do código. Inclua imagens de trechos de código considerados mais importantes, bem como telas ou imagens de grafos e resultados. Finalize o relatório comentando as dificuldades que teve para realizar a atividade.

   - Na implementação siga fielmente a máscara da função (nome, parâmetros de entrada e de saída, tipos de dados) conforme a seção “2. Descrição” no item *ii*.

   - O relatório deverá conter o *link* do GitHub com o código fonte do protótipo implementado (devidamente comentado, esse é um item de avaliação).

   - Se preferir, no lugar do relatório elabore um vídeo (máximo 2 minutos) em que você faz os comentários sobre as questões técnicas do código, fala das dificuldades que teve no desenvolvimento da atividade e mostra o funcionamento do programa. No arquivo .pdf coloque o link do YouTube para visualização do vídeo, além do link do GitHub com o código. **Obs.** Você deverá aparecer no vídeo.

     ATENÇÃO! Não serão aceitos trabalhos incompletos ou entregues fora do prazo.