SMAC03 – Grafos
Prof. Rafael Frinhani

ATIVIDADE 3 (AT3)

Assunto: 2. Teoria dos Grafos – Tipos e Representação – Lista de Adjacências.

Data de Entrega: 02/09 até as 21h.

1. **Objetivo:** Verificar o aprendizado de conceitos básicos sobre grafos, implementar funções para operações em uma lista de adjacências.

2. **Descrição:** A atividade consiste na implementação de funções para operações em uma lista de adjacências. Antes de realizá-la é recomendado o estudo do conteúdo sobre o tópico "2. Teoria dos Grafos – Tipos e Representação" pelos slides da aula, complementando com as referências bibliográficas da disciplina. A atividade a ser executada é:

   Implemente as funções a seguir:

   - **`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; 20 – multigrafo; 21 – multigrafo dirigido; 30 – pseudografo; 31 – pseudografo dirigido)

   - **`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 Moodle (e-mails não serão aceitos).

   Observação: Na implementação siga fielmente a máscara da função (nome, parâmetros de entrada e de saída, tipos de dados).