GUIÃO 12 – O TIPO ABSTRATO DE DADOS GRAPH – TRAVESSIAS

O tipo de dados **GRAPH** foi apresentado nas aulas teórico-práticas, e permite representar e operar sobre **grafos** e **grafos orientados**, **com ou sem pesos associados às suas arestas**.

A estrutura de dados usada é constituída uma **lista de vértices** e, para cada vértice, pela sua **lista de adjacências**. Estas listas são definidas usando o tipo de dados genérico **SORTED-LIST** que já conhece de um guião anterior.

Por decisão de projeto, o tipo **GRAPH** fornece apenas as **operações básicas** sobre grafos. **Outros algoritmos** são implementados em **módulos autónomos**.

Por exemplo, para as travessias apresentadas nas aulas teórico-práticas estão já definidos os módulos:

- travessia em profundidade: **GRAPH-DFS-REC** e **GRAPH-DFS-WITH-STACK**.
- travessia por níveis: **GRAPH-BFS-WITH-QUEUE**.

Os tipos de dados auxiliares **INTEGERS-STACK** e **INTEGERS-QUEUE** permitem usar essas estruturas de dados auxiliares nalguns dos algoritmos implementados.

TAREFAS

- Analisar o módulo **GRAPH-DFS-REC** que implementa o **algoritmo recursivo** de **travessia em profundidade** (**depth-first**), a partir de um vértice dado. Ter em atenção o modo como é atravessado o grafo e como se regista a informação associada à travessia.
- Por analogia, **completar e testar** o módulo **GRAPH-DFS-WITH-STACK** que implementa o **algoritmo iterativo** de **travessia em profundidade** usando uma **pilha (stack)**.
- Para um mesmo grafo, **comparar** a ordem pela qual os vértices são visitados pelas duas travessias anteriores. **Em que circunstâncias é que a essa ordem é exatamente a mesma ou é diferente?**
- Por analogia, **completar e testar** o módulo **GRAPH-BFS-WITH-QUEUE** que implementa o **algoritmo iterativo** de **travessia por níveis** usando uma **fila (queue)**.
- Para um grafo sem pesos associados às suas arestas, a travessia por níveis a partir de um dado vértice inicial permite construir **a árvore dos caminhos mais curtos** com raiz nesse vértice. Notar a existência de dois *arrays* permitindo armazenar (1) a **distância (i.e., o número de arestas)** no caminho mais curto definido a partir do vértice inicial, e (2) o **predecessor** de cada vértice na árvore dos caminhos mais curtos.
- **Desenvolver exemplos de aplicação**:
    1. Identificar os **vértices alcançáveis** a partir de um dado vértice inicial, usando **a travessia em profundidade**.
    2. Listar os **caminhos mais curtos** definidos a partir de um dado vértice inicial, usando a **travessia por níveis**.

Atenção:

Os vértices de um grafo estão sequencialmente numerados: 0, 1, 2, …

Deve respeitar os protótipos das funções definidos nos vários ficheiros cabeçalho.

Pode criar funções auxiliares (**static**) locais a cada módulo.