

**AULA 12 – O TIPO ABSTRATO DE DADOS GRAPH – PARTE I**

**\*\*\* As funcionalidades implementadas serão apenas entregues junto com o próximo guião \*\*\***

O tipo de dados **GRAPH** foi apresentado nas aulas teóricas, 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 **SORTEDLIST** 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 os vários tipos de travessias estão já definidos os módulos:

- - travessia em profundidade: **GRAPHDFSREC** e **GRAPHDFSWITHSTACK**.
- -travessia por níveis: **GRAPHBFSWITHQUEUE**.

Os tipos de dados auxiliares **INTEGERSSTACK** e **INTEGERSQUEUE** são também fornecidos e permitem usar essas estruturas de dados auxiliares nalguns dos algoritmos implementados.

É também fornecido o tipo de dados **PRIORITYQUEUE** que irá suportar, na próxima semana, o desenvolvimento de algoritmos “*greedy*” como o Algoritmo de Dijkstra.

**TAREFAS**

- Analisar o tipo de dados genérico **PRIORITYQUEUE** – fila com prioridade baseada numa MIN-HEAP binária – e executar o seu programa de simulação para diferentes dados de entrada. Notar as diferenças relativamente ao tipo **MINHEAP** do guião anterior.
- Analisar o módulo **GRAPHDFSREC** que implementa o algoritmo recursivo de travessia em profundidade, 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 **GRAPHDFSWITHSTACK** que implementa o algoritmo iterativo de travessia em profundidade usando uma **pilha**.
- 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 **GRAPHBFSWITHQUEUE** que implementa o algoritmo iterativo de travessia por níveis usando uma **fila**.
- Para um grafo não orientado e sem pesos associados às 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** para o vértice inicial e (2) o **predecessor** de cada vértice na árvore dos caminhos mais curtos.
- Concluir o desenvolvimento do tipo abstrato **GRAPH**: as funções incompletas estão assinaladas.

- Desenvolver **exemplos de aplicação**: (1) quais os vértices alcançáveis a partir de um vértice inicial? (2) verificar se um grafo (orientado) é (fortemente) conexo.

**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.