Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Algoritmos e Estruturas de Dados

2025/2026 — 1º Semestre

2º Trabalho — O TAD GRAPH e a determinação de conjuntos de vértices dominantes

Data limite de entrega: 22 de dezembro de 2025, às 18 horas

O tipo de dados **GRAPH** foi apresentado nas aulas, e permite representar e operar sobre **grafos e grafos orientados**, com ou sem pesos associados às suas arestas. A estrutura de dados usada é baseada numa **lista de vértices**, e na associação a cada vértice da sua **lista de adjacências**. Estas listas são definidas usando o tipo de dados genérico **SORTED-LIST**.

O tipo **GRAPH** fornece apenas as **operações básicas** sobre grafos. Outros **algoritmos** são implementados em **módulos autónomos**.

A versão do tipo GRAPH usada neste trabalho tem **algumas diferenças** relativamente à versão usada nas aulas. Em particular, para cada grafo:

- Os índices dos vértices pertencem ao conjunto **{0, 1, … , (range – 1)}** e podem **não** ser **estritamente sequenciais**.
- O cabeçalho (*header*) da estrutura de dados tem um campo adicional que referencia o **conjunto dos índices dos vértices**, instanciado usando o tipo **INDICES-SET**.
- Algumas das **funções** do tipo GRAPH devolvem como **resultado**, ou aceitam como um dos seus **argumentos**, um **conjunto de índices de vértices**.

OBJETIVOS

Este trabalho vai permitir desenvolver **algoritmos** sobre **grafos** e efetuar a **análise da eficiência computacional** de algumas das estratégias desenvolvidas.

Assim, é necessário:

1. **TAD INDICES-SET: analisar** a estrutura de dados escolhida (com objetivos didáticos), as funcionalidades disponíveis e a sua utilização no exemplo de aplicação (**IndicesSet.h**, **IndicesSet.c**, **TestIndicesSet.c**).

2. **TAD GRAPH: analisar** a nova versão da estrutura de dados e as principais funcionalidades disponibilizadas (**Graph.h**, **Graph.c**, **TestGraphBasics.c**).

3. **TAD GRAPH:** terminar o desenvolvimento das funções incompletas, por esta ordem:

    1. **GraphGetSetAdjacentsTo** que devolve o conjunto dos vértices adjacentes a um vértice dado.
    2. **GraphComputeVertexWeights** que associa um peso a cada um dos vértices de um grafo não-orientado, tendo em conta as arestas adjacentes a cada vértice do grafo. E devolve um array com os pesos calculados.
    3. **GraphGetSubGraph** que, dado um grafo G e um conjunto de vértices C, devolve o grafo que é o subgrafo de G, definido pelos vértices de C e pelas

arestas (u,v) de G tais que u e v pertencem a C. **--- Tarefa com maior grau de dificuldade**

4. **Módulo DOMINATING-SETS:** terminar o desenvolvimento das funções incompletas, que se aplicam a grafos não-orientados, por esta ordem:

    1. **GraphIsDominatingSet** que verifica se um dado conjunto de vértices é um conjunto de vértices dominantes de um dado grafo não-orientado.
    2. **GraphComputeMinDominatingSet** que determina, usando **procura exaustiva**, um **conjunto de vértices dominantes**, com **menor cardinalidade** possível, para um dado grafo não-orientado.
    3. **GraphComputeMinWeightDominatingSet** que determina, usando **procura exaustiva**, um **conjunto de vértices dominantes**, com **menor peso total**, para um dado grafo não-orientado. O peso de um conjunto dominante é a soma dos pesos dos seus vértices.

    Um **conjunto de vértices dominantes** de um grafo não-orientado G é um conjunto de vértices D, tal que cada vértice não pertencente a D é adjacente a um dos vértices de D. Consultar, por exemplo, a Wikipédia.

5. **Análise da Complexidade:** Caracterizar a complexidade algorítmica das soluções implementadas para a função **GraphComputeMinDominatingSet** e para a função **GraphComputeMinWeightDominatingSet**.

ATENÇÃO

- **Completar o desenvolvimento das funções pedidas nos ficheiros `.c`**
- **Não devem ser alterados os ficheiros `.h`**
- Assegurar que as funções desenvolvidas executam corretamente paras os grafos (muito simples) dos ficheiros de exemplo: **`Test*.c`**
- Testar as funções com grafos mais complexos.
- Assegurar que o código desenvolvido não apresenta fugas de memória.

ANÁLISE DA COMPLEXIDADE – PARA NOTAS SUPERIORES A 16 VALORES

- Escrever um **relatório sucinto** (máx. 6 págs.), com a caracterização da **complexidade** para a função **GraphComputeMinDominatingSet** e para a função **GraphComputeMinWeightDominatingSet**.
- O relatório deverá incluir uma breve descrição da(s) **métrica(s) adotada(s)** para medir a complexidade e **tabelas (gráficos) com os resultados dos testes** efetuados.
- **A entrega de um relatório não significa por si só que a nota do final do trabalho venha a ser superior a 16.**

Atenção – Desenvolvimento do código

- Os índices dos vértices de um grafo pertencem a 0, 1, 2, … , (range – 1). Mas poderão não ser estritamente sequenciais.

- Deve **respeitar os protótipos das funções** definidas nos vários ficheiros cabeçalho.

- Pode criar **funções auxiliares (static)** sempre que achar útil.

- O **código** desenvolvido deverá ser **claro** e **comentado** de modo apropriado: os identificadores escolhidos para as variáveis e a estrutura do código, bem como os eventuais comentários, deverão ser suficientes para a sua compreensão.

- No final do trabalho deverá ser entregue um **ficheiro ZIP** com todo o código e um **ficheiro PDF** com o relatório da análise da complexidade (opcional).

- Não é necessário entregar qualquer relatório relativo ao desenvolvimento do código.

Critérios de Avaliação

- **Colaboração na Avaliação entre Pares (2 valores)**
  - Testar o funcionamento do código de colegas, avaliar a sua qualidade e clareza

- **Desenvolvimento e teste das funções pedidas (14 valores)**
  - Qualidade do código (eficiência, legibilidade, clareza, robustez)
  - Teste do código e verificação de fugas de memória

- **Relatório (4 valores)**
  - Aspetos Gerais/Apresentação/Conclusão
  - Complexidade dos dois algoritmos analisados
    - Dados experimentais

A **nota atribuída ao desenvolvimento e teste das funções pedidas** será obtida pela média da classificação atribuída pelo docente e da classificação atribuída pelos colegas (avaliação entre pares, cada trabalho será avaliado por 3 a 4 estudantes).

A avaliação entre pares está sujeita a validação pelos docentes, podendo ser descartada nos casos em que se verifique que corresponde a uma avaliação manifestamente incorreta do trabalho apresentado.

Atenção:

- O trabalho deve ser realizado em grupos de 2 alunos, mantendo-se os grupos do trabalho anterior, sempre que possível.

- A entrega do trabalho (código + relatório opcional) será feita através da plataforma eLearning.