Aula prática 8

Esta aula tem como objetivo estudar a estrutura de dados "grafo", explorando a implementação baseada na matriz de adjacências e alguns dos algoritmos base.

1 Pretende-se implementar um **dígrafo** recorrendo ao uso de uma *matriz de adjacências*. Um grafo com *n* vértices pode ser representado por uma matriz de *n* por *n*, cujas entradas são valores booleanos (representado por int, em que 0 indica falso e 1 indica verdadeiro), que indicam se dois vértices estão ligados.

Considere a biblioteca da estrutura de dados *grafo.* São fornecidos os ficheiros ".c" e ".h" com as estruturas definidas e código parcialmente implementado. O ficheiro **teste_grafo.c** contém a função main(), que pode ser utilizada para implementar testes às funções pedidas.

Estude cuidadosamente a estrutura de dados fornecida, observando que o *grafo* guarda o tamanho e um double pointer para uma matriz de adjacências.

1.1 Implemente as seguintes funções de adição, remoção e teste de existência de arestas:

int *grafo_adiciona*(**grafo*** *g,* **int** *origem,* **int** *dest*)

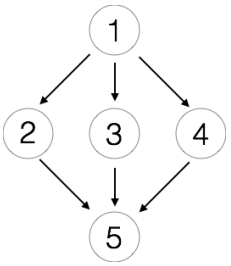
int *grafo_remove*(**grafo*** *g,* **int** *origem,* **int** *dest*)

int *grafo_aresta*(**grafo*** *g,* **int** *origem,* **int** *dest*)

As funções devem efetuar as verificações necessárias, incluindo se as variáveis *origem* e *dest* estão dentro dos intervalos esperados. Em caso de erro devem retornar -1.

1.2 Implemente uma função que dada uma *sequência de pares de adjacências* e o número de arestas, crie o respectivo grafo. Note que o tamanho do grafo é determinado com base no valor máximo dos vértices constantes da lista de adjacências.

grafo* *grafo_deLista*(**int*** *adjacencias,* **int** *n_arestas*)



No exemplo ao lado, os parâmetros seriam:

adjacencias = {1, 2, 1, 3, 1, 4, 2, 5, 3, 5, 4, 5}

n_arestas = 6

1.3 Implemente as seguintes funções, que devolvem os vértices de todas as arestas que saem de/chegam a um determinado vértice. Utilize a implementação de vetor de números inteiros fornecida.

vetor* *grafo_arestasSaida*(**grafo*** *g,* **int** *i*)

vetor* *grafo_arestasEntrada*(**grafo*** *g,* **int** *i*)

As funções devem efetuar as verificações necessárias. Teste a sua implementação, verificando se no exemplo da linha anterior arestasSaida(1) = arestasEntrada(5) = {2, 3, 4}. Indique a complexidade algorítmica de cada uma destas funções.

1.4 Implemente a seguinte função que testa se um grafo é completo. Relembre-se que um grafo completo é aquele que tem uma aresta para qualquer par de vértices:

int *grafo_completo*(**grafo*** *g*)