Tarefa – Grafo Dirigido como lista encadeada

Um Grafo (*Graph*) é um conjunto de vértices (ou nós) ligados por Arestas (*Edges*). Grafos Dirigidos (*Directed Graphs*) são grafos cujas arestas são pares ordenados de vértices (i.e., há uma direção imposta). O primeiro vértice do par é o vértice inicial da aresta e o segundo é o vértice final. Entre as várias aplicações de grafos está a representação de "quem segue quem" numa rede social. A Fig. 1 ilustra um grafo dirigido onde o indivíduo *0* segue o indivíduo *1*.

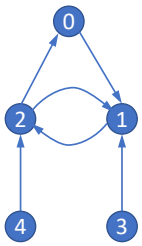

Fig. 1

Uma maneira de especificar um grafo é através do seu conjunto de arestas. Por exemplo, o grafo da Fig. 1 pode ser especificado por:

$$(0 \rightarrow 1)\quad (1 \rightarrow 2)\quad (2 \rightarrow 0)\quad (2 \rightarrow 1)\quad (3 \rightarrow 1)\quad (4 \rightarrow 2)$$

Uma forma de representar um grafo na memória de um computador é através de um vetor de ponteiros para uma lista simplesmente encadeada de vértices, onde para o vértice *i* se coloca a lista de todos os vértices adjacentes que definem uma aresta com *i*. A Fig. 2 ilustra esta representação. Note que ***Graph*** é uma estrutura que consiste apenas de um ponteiro para o vetor (i.e., uma estrutura com apenas um componente, que é um vetor).

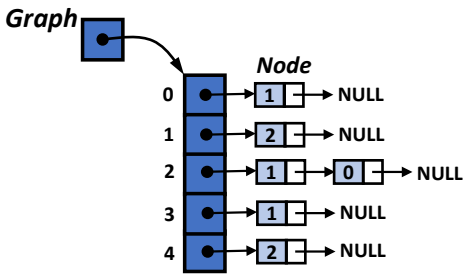

Fig. 2

Primeiro defina as seguintes estruturas: ***Node*** (que contém o vértice ponta final da aresta e o ponteiro para o próximo ***Node***), ***Graph*** (que contém um vetor **estático** de ponteiros para ***Node***)[1] e ***Edge*** (que especifica uma aresta com o vértice inicial e o vértice final).

Na *main*, especifique o grafo através de um vetor de arestas. Por exemplo, para a Fig.1, temos:

```
Edge edges1[] =
{
   {0,1}, {1,2}, {2,0}, {2,1}, {3,1}, {4,2}
};
```

Depois defina e teste as seguintes funções:

- Função ***createGraph*** para criar e inicializar o grafo com um vetor de ponteiros para NULL. Essa função recebe o número de vértices e retorna um grafo (i.e., retorna um ponteiro para ***Graph***).
- Função ***setGraph*** para montar o grafo (como na Fig. 2). Essa função recebe um ponteiro para ***Graph***, o vetor de arestas (*edges1*) e o tamanho desse vetor.
- Função ***printGraph*** que imprime o grafo, dados um ponteiro para ***Graph*** e o número total de vértices.
- Função que testa se dois vértices dados são uma aresta do grafo dirigido. Essa função recebe um ponteiro para ***Graph*** e os dois vértices e retorna verdadeiro ou falso. Por exemplo, na Fig. 1, $2 \rightarrow 0$ formam uma aresta, mas $2 \rightarrow 4$ não. Faça uma versão iterativa ***isEdgeIt*** e outra recursiva ***isEdgeRec***.[2]

Não faça leituras de arquivos ou do teclado. Mensagens de insuficiência de memória e interrupções devem ocorrer apenas na *main*. Libere as memórias ao final (*Nodes* e o *Graph*) com a função ***freeGraph***. Exemplo de *output* na Fig.3.

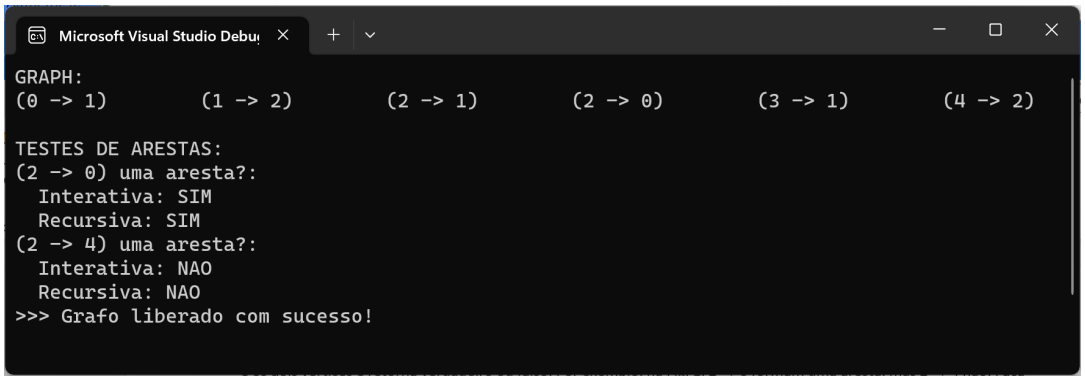

Fig.3

[1] Considere uma constante simbólica (#define) para estabelecer o tamanho desse vetor.

[2] *Dica*: para a versão recursiva, use uma função auxilar que é recursiva