



ALGORITMOS E ESTRUTURAS DE DADOS  
AULA DE LABORATÓRIO #06 - GRAFOS

## Objectivos

Neste laboratório abordam-se os tópicos da representação e manipulação de grafos. Na 1ª parte recorre-se à representação em matriz de adjacências, sendo solicitado o cálculo de algumas propriedades de um grafo e a produção de um ficheiro de saída com todas as arestas do grafo. Na 2ª parte pretende-se ler os resultados da primeira e usar a representação em lista de adjacências.

## Plano da Aula

A tabela abaixo mostra um exemplo de representações de um grafo. Este grafo é ponderado e não direccionado, com  $V = 6$  vértices e  $E = 5$  arestas; tem densidade  $2E/V = 1.666...$  e grau máximo 3. Os vários formatos são especificados em mais detalhe no texto das questões.

Matriz de adjacência	Vector de arestas	Lista de adjacências
6 0 2 1 3 0 0 2 0 0 0 0 0 1 0 0 4 5 0 3 0 4 0 0 0 0 0 5 0 0 0 0 0 0 0 0 0	6 5 0 1 2 0 2 1 0 3 3 2 3 4 2 4 5	6 1:2 2:1 3:3 -1 0:2 -1 0:1 3:4 4:5 -1 0:3 2:4 -1 2:5 -1 -1
grafo6.adj	grafo6.edge	grafo6.ladj

Para ambas as partes seguintes deverá completar uma **Makefile** que permita a compilação dos programas e facilite a execução de testes. Sugere-se a consulta de alguns tutoriais facilmente encontrados na net como [este](#) ou [este](#).

## 1 Matriz de Adjacências

Para a representação em matriz de adjacências de um grafo não direccionado, ponderado, será usado um ficheiro de dados em formato texto, com a extensão **adj**,

`<nome_grafo>.adj`

A primeira linha tem um inteiro,  $V$ , que indica o número de vértices (ou nós) do grafo. A matriz de adjacências (simétrica) é dada pelas  $V$  linhas seguintes, cada uma com  $V$  colunas de inteiros não negativos, em que um valor positivo na posição  $(i, j)$  indica o peso da aresta entre o nó  $i$  e o nó  $j$ , sendo que o valor 0 na posição  $(i, j)$  indica a ausência de aresta entre o nó  $i$  e o nó  $j$ . Os nós são numerados de 0 a  $V-1$ .

Escreva um programa, designado **pg1**, invocado pelo comando

`pg1 <nome_grafo>.adj`

que lê o ficheiro `<nome_grafo>.adj` e realiza as seguintes tarefas:

- 1.1. Constrói, a partir dos dados lidos, uma estrutura de dados que representa o grafo, usando a representação em matriz de adjacências.
- 1.2. Imprime no terminal a densidade do grafo.
- 1.3. Imprime no terminal o grau máximo do grafo.
- 1.4. Escreve no ficheiro de saída em formato texto, com a extensão `edge`,

`<nome_grafo>.edge`

todas as arestas do grafo. A primeira linha do ficheiro deve indicar o número de nós,  $V$ , seguido do número de arestas (ou ramos),  $E$ , e as  $E$  linhas seguintes devem conter os dois nós de cada ramo seguidos do seu peso, escrevendo um ramo por linha.

## 2 Listas de Adjacências

Na 2<sup>a</sup> parte pretende-se fazer uso da representação de grafos em lista de adjacências. Escreva um programa, designado `pg2`, invocado com o comando

`pg2 <nome_grafo>.edge`

que lê de um ficheiro do tipo `.edge`, como especificado acima, as arestas de um grafo não direccionado, ponderado, e que implementa as seguintes tarefas:

- 2.1. Constrói, a partir dos dados lidos, uma estrutura de dados que representa o grafo, usando a representação em lista de adjacências.
- 2.2. Escreve um ficheiro de saída, com a extensão `ladj`,

`<nome_grafo>.ladj`

com uma representação em texto da lista de adjacências. A primeira linha especifica o número de vértices do grafo; cada linha seguinte diz respeito a um vértice, por ordem, e indica a sua lista de vértices adjacentes. Para cada adjacência são dados o vértice e o peso da aresta, unidos por `:'`. A linha termina com `-1`. Quando um vértice é isolado, a sua linha existe para manter a ordenação, mas contém apenas `-1`.

Nos ficheiros `LinkedList.h` e `LinkedList.c` encontram-se definidos os protótipos para manipulação de uma lista simplesmente ligada de `Items` e as suas implementações, respectivamente, que deverá usar no seu programa. O tipo `Item` é definido no ficheiro `defs.h`.