Módulo 5 - Laboratório prático – Exercícios

Em cada módulo, os exercícios assinalados com ♣ são mais importantes. Os exercícios extra apresentam treino extra ou aplicações importantes dos TAD do módulo. São exercícios cuja prática complementa os conceitos e ajuda a perceber como os aplicar.

Exercícios:

1. Implemente o interface dado na aula para o tipo abstrato de dados (TAD) *Lista* usando uma representação de lista simplesmente ligada com inserção à cabeça. Use o identificador *ListaSL* para a classe a construir.
2. ♣ O comportamento de inserção da lista ligada implementada no exercício 1 é já conhecido. Aproveite o código possível e construa uma nova implementação para o TAD Pilha, desta vez usando uma implementação de lista simplesmente ligada.
3. ♣ Implemente o interface dado na aula para o tipo abstrato de dados (TAD) *Lista* usando uma representação de lista simplesmente ligada com um ponteiro para a cabeça e outro para a cauda da lista. Use o identificador *ListaL* para a classe a construir. Os métodos para consultar e para remover elementos em posições dadas na lista devem emitir um erro caso a posição não exista (instrução *raise IndexError()*).
4. Implemente o TAD Fila usando uma lista simplesmente ligada.
5. ♣ Analise as ordens de complexidade dos algoritmos que implementam os métodos de *Lista* no exercício 3).
6. ♣ Implemente o TAD *Lista* usando uma estrutura de lista duplamente ligada. (Use o identificador *ListaDL* para a classe a construir).
7. Estenda o conjunto de operações possíveis da sua *ListaDL* para incluir as seguintes operações: *ins_idx(idx, item)* para inserir elemento com valor item na posição *idx*, *after(p)*, para devolver o valor guardado na posição seguinte a p e *previous(p)*, para devolver o valor guardado na posição anterior a p. Note que p será um valor inteiro para indicar uma posição na lista.
8. ♣ Implemente um algoritmo para converter uma expressão *infix* aritmética inserida pelo utilizador numa expressão *postfix* que deve ser devolvida numa *ListaDL*. Use a classe do exercício 2, onde implementou o TAD *Pilha*.

Exercícios Extra

1. ♣ Estenda as classes de listas ligadas desenvolvidas anteriormente para permitirem gerar e devolver um iterador, através dos métodos especiais de classe __iter__ e __next__.
2. ♣ Uma matriz esparsa é uma matriz $n \times m$, onde apenas k valores são não nulos e $k \ll n \times m$. Por exemplo, a matriz 8×8, com apenas 12 valores não nulos entre os 64 possíveis é uma matriz esparsa:

$$\begin{pmatrix} 1.0 & 0 & 5.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.0 & 0 & 0 & 0 & 0 & 11.0 & 0 \\ 0 & 0 & 0 & 0 & 9.0 & 0 & 0 & 0 \\ 0 & 0 & 6.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.0 & 0 & 0 & 0 & 0 \\ 2.0 & 0 & 0 & 0 & 0 & 10.0 & 0 & 0 \\ 0 & 0 & 0 & 8.0 & 0 & 0 & 0 & 0 \\ 0 & 4.0 & 0 & 0 & 0 & 0 & 0 & 12.0 \end{pmatrix}$$

Matrizes com um elevado número de entradas nulas ocorrem em muitos domínios em que são usadas para representação de dados, por exemplo, em redes sociais, redes de computadores, métodos de elementos finitos, etc. Neste caso, não faz sentido utilizar um array multi-dimensional convencional, uma vez que a grande maioria das células do array vão guardar o valor nulo, para além de não ser eficiente se necessitarmos de operar com a matriz.

a) Analise a complexidade espacial de uma matriz matemática clássica (array bidimensional).

b) Implemente um TAD *MatrizEsparsa* usando uma representação com nós duplamente ligados e vetores de indexação (um vetor para as linhas e um vetor para as colunas). A sua classe deve conter as seguintes operações:

- **Criar** uma matriz "vazia".
- ***Devolver*** o número de colunas (m); **Devolver** o número de linhas (n); **Devolver** a quantidade (k) de valores não nulos; **Devolver** o valor na linha *i*, coluna *j*;
- ***Inserir*** o valor x dado na posição (i, j) indicada.
- ***Remover*** na posição (i, j) indicada.
- ***Multiplicar*** um dado valor a cada uma das posições não nulas, na matriz.
- ***Adicionar*** duas matrizes esparsas.
- ***Multiplicar*** duas matrizes esparsas.