

Especificando uma Matriz como um Tipo Abstrato de Dados

Obs.: Para avaliar cada função da interface, construa uma pequena aplicação de testes.

Construir um TAD matriz (numérica) com as seguintes características:

a) Requisito: a matriz não deve armazenar elementos de valor zero.

b) Modelo/Estrutura de dados: Os elementos que implementam o tad devem ser modelados via dicionário e seguir as orientações vistas nas discussões realizadas nas aulas de laboratório. Consulte suas anotações de aula para saber do que se trata.

c) Interface:

c.1) cria(quant. linhas, quant. colunas): Retorna uma estrutura de dados do tipo tad matriz representando uma matriz de dimensões [quant. linhas X quant. colunas].

c.2) criaLst(matriz lista de listas): Retorna uma estrutura de dados do tipo tad matriz equivalente à matriz lista de listas passada como argumento de entrada.

c.3) destroi(): Retorna *None*.

c.4) getElem(tadMat, lin, col): Retorna o elemento armazenado na posição *lin*, *col* da matriz de entrada. Retorna *None* se o elemento não existir.

c.5) setElem(tadMat, lin, col, valor): Armazena o elemento na posição *lin*, *col* da matriz de entrada. Retorna *None* se a posição não existir.

c.6) soma(tadMatA, tadMatB): Soma duas matrizes e retorna uma terceira matriz com o resultado da soma. Retorna *None* se as matrizes não puderem ser somadas.

c.7) vezesK(tadMat, k): Altera a matriz de entrada multiplicando os seus elementos por *k*. Retorna uma referência para *tadMat*.

c.8) multi(tadMatA, tadMatB): Multiplica duas matrizes e retorna uma terceira matriz com o produto resultante. Retorna *None* se as matrizes não puderem ser multiplicadas.

c.9) clona(tadMat): Retorna uma matriz cópia da matriz de entrada.

c.10) diagP(tadMat): Retorna uma lista com a diagonal principal. Retorna *None* se a diagonal principal não existir (matriz não quadrada).

c.11) diagS(tadMat): Retorna uma lista com a diagonal secundária. Retorna *None* se a diagonal secundária não existir (matriz não quadrada).

c.12) quantLinhas(tadMat): Retorna a quantidade de linhas da matriz de entrada.

c.13) quantColunas(tadMat): Retorna a quantidade de colunas da matriz de entrada.

c.14) vizinhos(tadMat, lin, col): Retorna uma lista com os 8 vizinhos do elemento localizado em *lin, col* da matriz de entrada. O primeiro vizinho está localizado na posição quina esquerda superior. O restante segue em sentido horário ao redor do elemento *lin, col*. Usar o valor *None* para representar os vizinhos que por ventura não existam. (ver figura 2.g)

c.15) extrai(tadMat, lin, col, tamLin, tamCol): Extrai e retorna um submatriz (do tipo *tadmatriz*) da matriz de entrada. A matriz retornada começa na posição *lin, col* da matriz de entrada. Essa posição será o elemento esquerdo superior da nova matriz. A matriz extraída possui dimensões [*tamLin* linhas X *tamCol* colunas]. Retorna *None* se a matriz solicitada não puder ser extraída. (Ver figura 1.a)

c.16) insere(tadMatA, lin, col, tadMatB): Insere a matriz *tadMatB* a partir da posição *lin, col* da matriz *tadMatA*. Retorna a *tadMatA* alterada. Retorna *None* se a matriz *tadMatB* não puder ser toda inserida em *tadMatA*. (Ver figura 1.b)

c.17) deslocaEsq(tadMat): Desloca todas as colunas da matriz de entrada 1 posição à esquerda. Última coluna é preenchida com zeros. Retorna uma referência para a *tad* matriz de entrada. (Ver figura 1.c)

c.18) deslocaDir(tadMat): Desloca todas as colunas da matriz de entrada 1 posição à direita. Primeira coluna é preenchida com zeros. Retorna uma referência para a *tad* matriz de entrada. (Ver figura 1.d)

c.19) rotEsq(tadMat): Desloca todas as colunas da matriz de entrada 1 posição à esquerda. Primeira coluna é movida para a última coluna. Retorna uma referência para a *tad* matriz de entrada. (Ver figura 2.f)

c.20) rotDir(tadMat): Desloca todas as colunas da matriz de entrada 1 posição à direita. Última coluna é movida para a primeira coluna. Retorna uma referência para a *tad* matriz de entrada. (Ver figura 2.e)

c.21) carrega(<arquivo>): Carrega uma matriz a partir de um arquivo texto de nome <arquivo>. Retorna uma matriz do tipo *tad* matriz preenchida com o conteúdo arquivo. No arquivo, a matriz está representada da seguinte forma: elementos separados por espaço, cada linha de texto é uma linha da matriz.

c.22) salva(tadMat, <arquivo>): Salva uma matriz em um arquivo texto de nome <arquivo>. Retorna uma referência para o *tad* matriz de entrada. No arquivo, a matriz está representada da seguinte forma: elementos separados por espaço, cada linha de texto é uma linha da matriz.

Utilize as figuras 1 e 2 a seguir para auxiliar na interpretação do que é pedido para algumas das funções constantes na interface do *tad*.

Algumas funcionalidades da Interface do TAD Matriz (parte I)

a) `extraí(Matriz,1,2,3,3)`

Matriz						Extraída		
0	1	2	3	4	5	0	1	2
0	5	4	1	8	3	6	5	6
1	3	0	5	6	7	4	2	8
2	9	1	2	8	4	3	9	0
3	4	5	1	9	0	2		
4	9	0	7	4	1	3		

b) `insere(Matriz,1,2,Inserida)`

Matriz						Inserida		
0	1	2	3	4	5	0	1	2
0	5	4	1	8	3	6	1	1
1	3	0	1	1	1	4	1	1
2	9	1	1	1	1	3	1	1
3	4	5	1	1	1	2		
4	9	0	7	4	1	3		

c) `deslocaEsq(Matriz)`

Matriz						Matriz deslocada à esquerda					
0	1	2	3	4	5	0	1	2	3	4	5
0	5	4	1	8	3	6	4	1	8	3	6
1	3	0	5	6	7	4	0	5	6	7	4
2	9	1	2	8	4	3	1	2	8	4	3
3	4	5	1	9	0	2	5	1	9	0	2
4	9	0	7	4	1	3	0	7	4	1	3

d) `deslocaDir(Matriz)`

Matriz						Matriz deslocada à direita					
0	1	2	3	4	5	0	1	2	3	4	5
0	5	4	1	8	3	6	0	5	4	1	8
1	3	0	5	6	7	4	0	3	0	5	6
2	9	1	2	8	4	3	0	9	1	2	8
3	4	5	1	9	0	2	0	4	5	1	9
4	9	0	7	4	1	3	0	9	0	7	4

Figura 1

Algumas funcionalidades da Interface do TAD Matriz (parte II)

e) `rotDir(Matriz)`

Matriz						Matriz rotacionada à direita					
0	1	2	3	4	5	0	1	2	3	4	5
0	5	4	1	8	3	6	6	5	4	1	8
1	3	0	5	6	7	4	4	3	0	5	6
2	9	1	2	8	4	3	3	9	1	2	8
3	4	5	1	9	0	2	2	4	5	1	9
4	9	0	7	4	1	3	3	9	0	7	4

f) `rotEsq(Matriz)`

Matriz						Matriz rotacionada à esquerda					
0	1	2	3	4	5	0	1	2	3	4	5
0	5	4	1	8	3	6	4	1	8	3	6
1	3	0	5	6	7	4	0	5	6	7	4
2	9	1	2	8	4	3	1	2	8	4	3
3	4	5	1	9	0	2	5	1	9	0	2
4	9	0	7	4	1	3	0	7	4	1	3

g) `vizinhos(Matriz,1,2)`

Matriz						Vizinhos de (1,2)							
0	1	2	3	4	5	0	1	2	3	4	5	6	7
0	5	4	1	8	3	6	4	1	8	6	8	2	1
1	3	0	5	6	7	4							
2	9	1	2	8	4	3							
3	4	5	1	9	0	2							
4	9	0	7	4	1	3							

Figura 2

d) Observações:

d.1) Utilize a aplicação de testes para avaliar a sua implementação do tad matriz. A aplicação está listada a seguir. Os fontes estão incluídos no arquivo *apptadmat.zip*. Leia com atenção os comentários presentes nos fontes python das aplicações exemplo.

d.2) Em caso de dúvidas, contate o utilize o fórum da atividade na sala moodle.

```
-----
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
#  apptadmat.py
#
#  Autor: ifes <ifes@ifes-vbox>
#  Criado em: 14/11/16 12:29:13
#  Usuário: IFES Serra Prog II
#
#  Função do programa: <resumo da tarefa que o programa realiza.>
#  Implementa um tad matriz usando dicionário para os dados.
#  A matriz tem como restrição não armazenar zeros.
#
#  Versão inicial: 1.0
#  -----
import tadmatrizv1

def main():
    # Cria dois tads matriz a partir de uma matriz lista de listas convencional.
    # Você pode substituir a chamada desta função por uma chamada a carrega
    # (<arquivo>). Basta salvar o conteúdo da matriz em um arquivo texto, uma linha
    # da matriz por linha do arquivo, com os elementos separados por espaço.
    X = tadmatrizv1.criaLst([[12,7,3],[4,5,6],[7,8,9]])
    Y = tadmatrizv1.criaLst([[5,8,1,2],[6,7,3,0],[4,5,9,1]])

    S = tadmatrizv1.soma(X,X)
    M = tadmatrizv1.multi(X,Y)

    # Remova os comentários e faça os seus testes.
    #~ print('X')
    #~ print(tadmatrizv1.mat2str(X))
    #~ print()
    #~ print('S = X + X')
    #~ print(tadmatrizv1.mat2str(S))
    #~ print()
    #~ print('M = X * Y')
    #~ print(tadmatrizv1.mat2str(M))
    #~ print()
    #~ Y2 = tadmatrizv1.clona(Y)
    #~ print('Y')
    #~ print(tadmatrizv1.mat2str(Y))
    #~ print()
    #~ print('Y desloca esq')
    #~ print(tadmatrizv1.mat2str(tadmatrizv1.deslocaEsq(Y)))
    #~ print()
    #~ Y = tadmatrizv1.clona(Y2)
    #~ print('Y desloca dir')
    #~ print(tadmatrizv1.mat2str(tadmatrizv1.deslocaDir(Y)))
    #~ print()
    #~ Y = tadmatrizv1.clona(Y2)
    #~ print('Y rota dir')
    #~ print(tadmatrizv1.mat2str(tadmatrizv1.rotaDir(Y)))
```

```

#~ print()
#~ Y = tadmatriizv1.clona(Y2)
#~ print('Y rota esq')
#~ print(tadmatriizv1.mat2str(tadmatriizv1.rotaEsq(Y)))
#~ print()

tadmatriizv1.salva(S,'S.txt')
tadmatriizv1.salva(M,'M.txt')
SA = tadmatriizv1.carrega('S.txt')
MA = tadmatriizv1.carrega('M.txt')

print()
print('S = X + X')
print(tadmatriizv1.mat2str(S))
print()
print('Comparando com o conteúdo carregado do arquivo.')
print(tadmatriizv1.mat2str(SA))
print()
print('M = X * Y')
print(tadmatriizv1.mat2str(M))
print()
print('Comparando com o conteúdo carregado do arquivo.')
print(tadmatriizv1.mat2str(MA))
print()

return 0

if __name__ == '__main__':
    main()

```

Fim (por enquanto) !