Estruturas de Dados I - ARA 7125

Primeiro Exercício-Programa - Entregar até dia 30/10/2015

Tipo de Dado Abstrato - Matriz

Este Exercício-Programa (EP) consiste na implementação do tipo de dado abstrato: *Matriz*. Uma matriz pode ser esparsa, ou seja, ela pode conter muitas entradas iguais a zero (0). Não guardar as entradas iguais a zero de uma matriz esparsa é uma estratégia usada para economizar memória (muitas vezes, com essa estratégia torna-se possível o armazenamento de matrizes esparsas absurdamente grandes). Neste EP, o tipo de dado abstrato *Matriz* **não poderá** guardar as entradas iguais a zero. Os atributos de uma matriz que devem ser considerados são (pelo menos):

- o número de linhas da matriz: m;
- o número de colunas da matriz: n;
- um vetor com m listas: V (uma lista para cada linha da matriz).

A lista em $V[i]$ de uma matriz A deve guardar as entradas (não-nulas) da i-ésima linha de A. As células das listas devem guardar: a entrada (não-nula), e a coluna dessa entrada. Em seguida, temos um exemplo de uma matriz que chamamos de matriz A.

	0	1	2	3	4
0	50	0	8	0	0
1	9	0	0	0	0
2	0	0	0	0	1
3	0	0	1	1	0
4	3	9	4	1	0
5	5	35	1	0	3
6	0	37	0	3	1

Na Figura 1, temos o vetor V de A[1].

As operações `LêM`, `DesalocaM`, `EscreveM`, `AtribuiM`, `SomaMs`, `SubtraiMs`, `MultiplicaMs`, `PotênciaM` devem ser implementadas para este novo tipo de dado com os seguintes protótipos:

- `Matriz LêM (FILE * arq)`: Recebe um ponteiro para um arquivo que armazena uma matriz, lê a matriz do arquivo, e devolve a matriz lida;
- `void DesalocaM (Matriz A)`: Recebe uma matriz `A`, e desaloca o espaço reservado para `A`;
- `void EscreveM (Matriz A)`: Recebe uma matriz `A`, e escreve a matriz na saída padrão do computador;
- `void AtribuiM (Matriz D, Matriz O)`: Recebe duas matrizes `D` (com m linhas e n colunas) e `O` (com m linhas e n colunas), e realiza uma atribuição da matriz origem `O` para a matriz destino `D`;

[1]Neste exemplo foram usadas para cada linha da matriz, listas simplesmente encadeadas sem cabeça. Você pode usar qualquer implementação de lista.

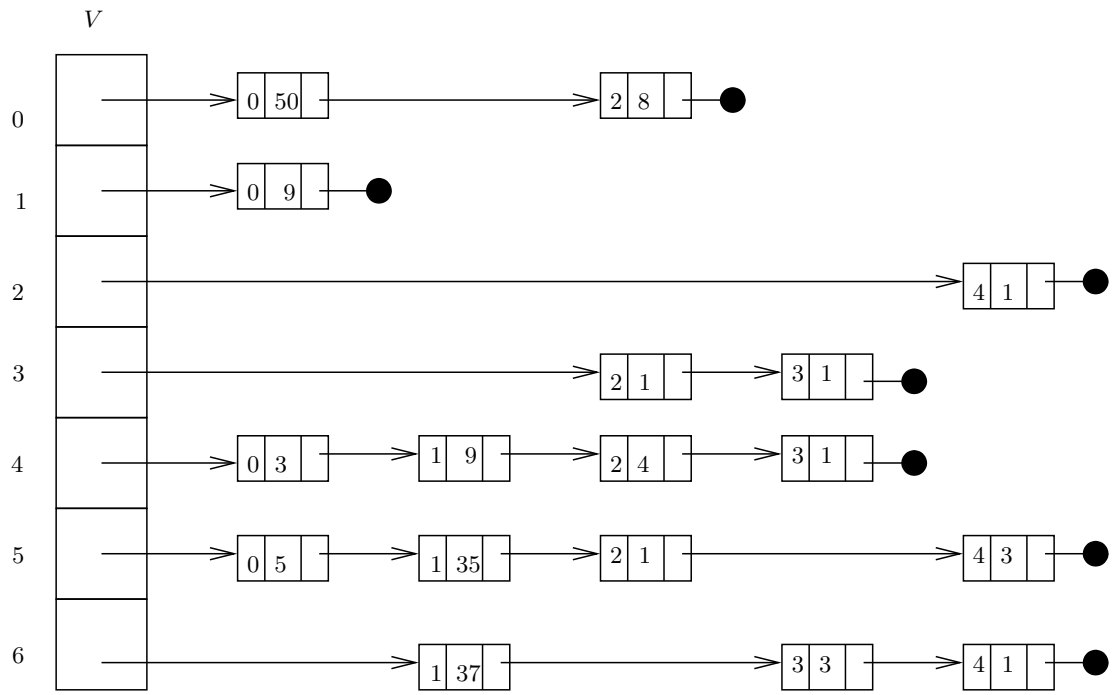

Figura 1: Representação de V da matriz A. Se uma célula de $V[i]$ possui dois inteiros j e d, então temos $A[i][j] = d$. Por exemplo, a terceira célula de $V[4]$ possui os inteiros 2 e 4, portanto, temos $A[4][2] = 4$. Outro exemplo, a primeira célula de $V[6]$ possui os inteiros 1 e 37, então temos $A[6][1] = 37$. Você pode confirir as entradas na matriz A.

- `Matriz SomaMs (Matriz A, Matriz B)`: Recebe duas matrizes `A` (com m linhas e n colunas) e `B` (com m linhas e n colunas), e devolve uma nova matriz (com m linhas e n colunas), resultado da soma de `A` e `B`;
- `Matriz SubtraiMs (Matriz A, Matriz B)`: Recebe duas matrizes `A` (com m linhas e n colunas) e `B` (com m linhas e n colunas), e devolve uma nova matriz (com m linhas e n colunas), resultado da subtração de `A` e `B`;
- `Matriz MultiplicaMs (Matriz A, Matriz B)`: Recebe duas matrizes `A` (com m linhas e n colunas) e `B` (com n linhas e k colunas), e devolve uma nova matriz com m linhas e k colunas, resultado da multiplicação de `A` por `B`;
- `Matriz PotênciaM (Matriz A, int p)`: Recebe uma matriz `A` quadrada, ou seja, com o mesmo número de linhas e colunas, e um inteiro `p`, e devolve uma nova matriz quadrada, resultado da potência $\texttt{A}^{\texttt{p}}$.

Observações importantes:
1. Você **deve** considerar que as matrizes lidas de arquivos são dadas com todas as entradas, inclusive, as entradas iguais a 0.
2. O seu programa deve ser feito em linguagem C. O compilador deve ser o gcc.
3. O formato do arquivo que armazena uma matriz é definido da seguinte forma:
   - a primeira linha do arquivo possui dois inteiros m e n que representam, respectivamente, o número de linhas e colunas da matriz;

- as m linhas seguintes do arquivo possuem n inteiros, cada uma.

Exemplo de um arquivo.:

```
 7   5
50   0  8  0  0
 9   0  0  0  0
 0   0  0  0  1
 0   0  1  1  0
 3   9  4  1  0
 5  35  1  0  3
 0  37  0  3  1
```

4. Você **deve** usar listas para armazenar uma matriz. Se você não usar, então a sua nota é ZERO. Faça a sua própria implementação. Você pode usar qualquer função vista em sala de aula.
5. Exercícios-Programas atrasados **não** serão aceitos.
6. Programas com erros de sintaxe (ou seja, existem erros durante a compilação do programa), receberão nota ZERO. Programas com *warning* na compilação terão diminuição da nota. Selecione as opções -Wall -ansi -pedantic -O3 do compilador gcc.
7. Utilize apenas os recursos que foram vistos em aula.
8. É importante que seu programa esteja escrito (digitado) de maneira a destacar a estrutura do programa (boa formatação).
9. O Exercício-Programa pode ser feito por grupos com até 3 pessoas. Não copie o programa de outra pessoa (ou grupo), não empreste o seu programa para outra pessoa (ou grupo), e tome cuidado para que não copiem seu programa sem a sua permissão. Todos os programas envolvidos em cópia terão nota ZERO.
10. Coloque comentários em pontos convenientes do programa.
11. A entrega do Exercício-Programa deverá ser feita no MOODLE.
12. O seu programa deve começar com um cabeçalho (linhas em comentários) contendo pelo menos o nome dos integrantes do grupo, e as matrículas.
13. As operações de soma, subtração, multiplicação e potência em matrizes podem ser obtidas em livros do ensino médio, ou livros de álgebra linear (primeiros capítulos ou apêndice).