Trabalho Prático 01 – AEDS 1

Professora: Thais R. M. Braga Silva

Valor: 10 pontos

Data de Entrega: 24/09/2019

Forma de Entrega: PVANet (formato .zip ou .tar.gz)

O objetivo deste trabalho consiste em concretizar os conceitos de TAD (Tipo Abstrato de Dados) e Listas implementadas por encadeamento através de uma aplicação: Matrizes esparsas.

Matrizes esparsas são matrizes nas quais a maioria das posições é preenchida por zeros (ou nulos). Para essas matrizes, podemos economizar um espaço significativo de memória se apenas os termos diferentes de zero (ou nulo) forem armazenados. As operações usais sobre essas matrizes (somar, multiplicar, inverter, pivotar) também podem ser feitas em tempo muito menor se não armazenarmos as posições que contêm zeros.

Uma maneira eficiente de apresentar estruturas com tamanho variável e/ou desconhecido é com o emprego de alocação encadeada, utilizando listas. Vamos usar essa representação para armazenar as matrizes esparsas. Cada coluna da matriz será representada por uma lista linear circular com uma célula cabeça. Da mesma maneira, cada linha da matriz também será representada por uma lista linear circular com uma célula cabeça. Cada célula da estrutura, além das células cabeça, representará os termos diferentes de zero da matriz e deverá ser como no código abaixo:

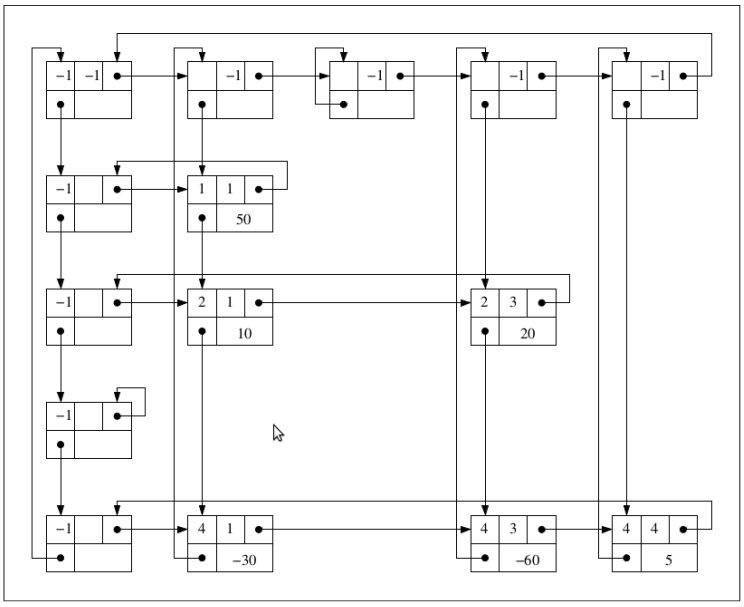
```
typedef struct Celula {
      struct Celula* direita, abaixo;
      int linha, coluna;
      double valor;
} TCelula;
```

O campo ***abaixo*** deve ser usado para referenciar o elemento diferente de zero (ou nulo) na mesma coluna. O campo ***direita*** deve ser usado para referenciar o próximo elemento diferente de zero (ou nulo) na mesma linha. Dada uma matriz A, para um valor A(i,j) diferente de zero (ou nulo), deverá haver uma célula com o campo ***valor*** contendo A(i,j), o campo ***linha*** contendo i e o campo ***coluna*** contendo j. Essa célula deverá pertencer a lista circular da linha i e também deverá pertencer à lista circular da coluna j. Ou seja, cada célula pertencerá a duas listas ao mesmo tempo. Para diferenciar as células cabeça, coloque -1 nos campos linha e coluna dessas células. Considere a seguinte matriz esparsa:

$$A = \begin{pmatrix} 50 & 0 & 0 & 0 \\ 10 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 \\ -30 & 0 & -60 & 5 \end{pmatrix}$$

A representação da matriz A pode ser vista na figura abaixo. Com essa representação, uma matriz esparsa m x n com r elementos diferentes de zero (ou nulo) gastará (m + n + r) células. É bem verdade que cada célula ocupa vários bytes na memória; no entanto, o total de memória usado será menor do que as m x n posições necessárias para representar a matriz toda, desde que r seja suficientemente pequeno.

Retirado do livro “Projeto de Algoritmos” do autor Nivio Ziviani



Dada a representação de listas duplamente encadeadas, o trabalho consiste em desenvolver em C um tipo abstrato de dados "**TAD Matriz_Esparsa"** com as seguintes operações, conforme esta especificação:

a) void imprimeMatriz()

Esse método imprime (uma linha da matriz por linha na saída) a matriz A, inclusive os elementos iguais a zero.

b) void leMatriz()

Esse método lê, de algum arquivo de entrada, os elementos diferentes de zero de uma matriz e monta a estrutura especificada anteriormente. Considere que a entrada consiste dos valores de m e n (número de linhas e de colunas da matriz) seguidos de triplas (i, j, valor) para os elementos diferentes de zero da matriz. Por exemplo, para a matriz anterior, a entrada seria:

4, 4

1, 1, 50.0

2, 1, 10.0

2, 3, 20.0

4, 1, -30.0

4, 3, -60.0

4, 4, -5.0

Para inserir células das listas que formam a matriz, crie métodos especiais para esse fim. Por exemplo,

void insere(int i, int j, double v)

para inserir o valor v na linha i, coluna j da matriz A. Note que você deve criar todas as operações necessárias para viabilizar a implementação do TAD.

As matrizes a serem lidas para testar as funções são:

a) a mesma matriz mostrada no enunciado deste problema;

Retirado do livro "Projeto de Algoritmos" do autor Nivio Ziviani

b) $\begin{pmatrix} 50 & 30 & 0 & 0 \\ 10 & 0 & -20 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5 \end{pmatrix}$

c) $\begin{pmatrix} 3 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix}$

d) Vamos considerar agora que o **TAD Matriz_Esparsa** deverá ser refeito para ser utilizado por uma empresa de vendas online que deseja registrar em uma matriz Clientes X Produtos as compras de um determinado produto realizadas por um mesmo cliente. Para tanto, dentro de cada célula (i,j) da matriz, ao invés de haver apenas um número em ponto flutuante, deverá constar uma lista de compras feitas pelo usuário i do produto j. Cada compra é registrada pela data em que foi realizada e pela quantidade de unidades do produto compradas. Uma lista de compras contém 1 ou mais compras realizadas. Para os casos em que um cliente nunca realizou uma compra de determinado produto, a célula relacionada não deve constar da matriz. Neste cenário, o seu programa vai receber de um arquivo de entrada como aqueles usados para as implementações anteriores, exceto pelo fato de que, em cada linha ao invés de apenas um valor real após os números da linha e coluna, haverá uma lista de 1 ou mais compras, representadas pelo par data e quantidade. Exemplo:

```
4 8 25/05/2019 5 06/08/2019 10
```

Nessa modificação do TAD Matriz_Esparsa, as operações devem ser imprimeMatriz, leMatriz, quantidadeComprasPorProduto, quantidadeComprasPorCliente. Você deverá ser reponsável por projetar os TADs necessários, bem como suas operações.

É obrigatório o uso de alocação dinâmica de memória para implementar as listas de adjacência que representam as matrizes.

As funções deverão ser testadas utilizando-se um programa main.

O que deve ser entregue:

a) Listagem do programa em C

b) Listagem dos testes executados

c) Descrição sucinta (por exemplo, desenho) das estruturas de dados e as decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.

Todos os itens acima devem ser compactados em um único arquivo com extensão .zip ou .tar.gz e submetido, dentro do prazo máximo determinado, através do PVANet.

Uma entrevista será feita em laboratório, para a apresentação de cada trabalho. Os TPs serão testados com as entradas descritas nesta especificação, bem como com uma entrada surpresa. Perguntas serão feitas a todos os membros e, é esperado domínio sobre as ideias do projeto e da codificação por todos.