Trabalho 03: Matrizes Esparsas

SSC0300 – Linguagens de Programação e Aplicações

Alunos:

1. Felipe Augusto Senger Lopes de Souza - Nº USP: 5513317
2. Bruno Henrique de Souza - Nº USP: 9311685
3. André Luthold - Nº USP: 9022071
4. Bruno Ávila - Nº USP: 9012667

Introdução: Este projeto contém o exercício do 3º Trabalho da disciplina de Linguagens de Programação e Aplicações. Ele é um programa em C que implementa uma matriz esparsa.

Definição: A estrutura da matriz esparsa é utilizada para armazenar matrizes de grandes ordens no computador, que obedecem a uma condição: possuem uma grande quantidade de zeros (0). Quando uma matriz tem ordem de cerca de 10^6 x 10^6, por exemplo, é necessário muita memória para armazená-la. Entretanto, se ela possui muitos zeros (cerca de 90% da matriz), é possível implementar uma estrutura que otimize o uso da memória, armazenando apenas algarismos diferentes do 0. Essa implementação usa uma série de listas encadeadas para armazenar as linhas da matriz, e se uma determinada posição não consta na lista, assume-se que ela é zero. Mais detalhes sobre a utilização de listas foram incluídos a seguir.

Descrição do Projeto: Todo o programa foi desenvolvido no Windows 7 64 bits, no programa Dev C++. O compilador utilizado foi o default do Dev C++, v5.10, nas configurações TDM-GCC 4.8.1 64-bit Release. Todas as bibliotecas usadas foram as consideradas padrão do Dev C++. A compilação não requer nenhum parâmetro especial e os programas foram compilados usando o botão de compilação do Dev C++.

Segue a descrição de como o programa funciona e é executado:

Algoritmo: O programa recebe a entrada de dois números, que representam o número de linhas e o número de colunas que a matriz irá possuir. Ao receber esses números, ele utiliza o primeiro (número de linhas) para criar uma lista encadeada “base”, fazendo uso de *structs*. Ela será a base para o início de cada uma das linhas da matriz (uma ilustração se encontra mais adiante). Após isso, o usuário pode inserir números em sua matriz.

Cada item da lista base está aterrado, apontando para *NULL,* que posteriormente irão apontar para o início de cada lista que representa as linhas da matriz, que são listas utilizando um segundo tipo de *struct* definido. Quando o usuário insere um elemento na linha, o programa chama a função *malloc* para alocar espaço para o novo item da matriz. Então ele é inserido em sua respectiva linha, e possui um marcador “coluna” para saber de qual coluna ele pertence.

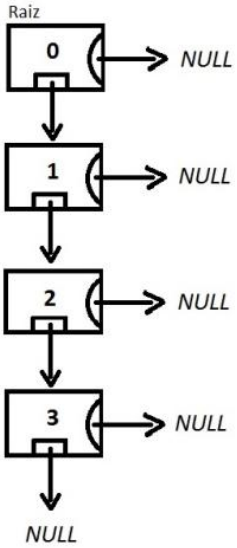

Figura 1 - Ilustração da lista-base

Mais detalhes sobre o algoritmo do programa se encontram no próprio código, nos comentários feitos antes de cada função.

Tutorial de uso: O programa deve ser compilado nas configurações citadas acima, que são as consideradas "default" do Dev C++ (versão 5.10). Primeiramente deve-se entrar com as dimensões da matriz, na forma linha x coluna. Então um menu aparecerá, pedindo que o usuário entre com uma das opções. Na primeira vez, é necessário inserir números na matriz, então o programa irá pedir que o usuário entre com quantos números ele deseja colocar na matriz. Depois deve-se digitar os números seguidos de suas posições (linha x coluna).

Por exemplo, ao inicializar uma matriz **4 x 4**, inserindo **3** números:

3 na posição **0 x 2**

42 na posição **3 x 1**

27 na posição **2 x 2**

O programa será algo semelhante a isso:

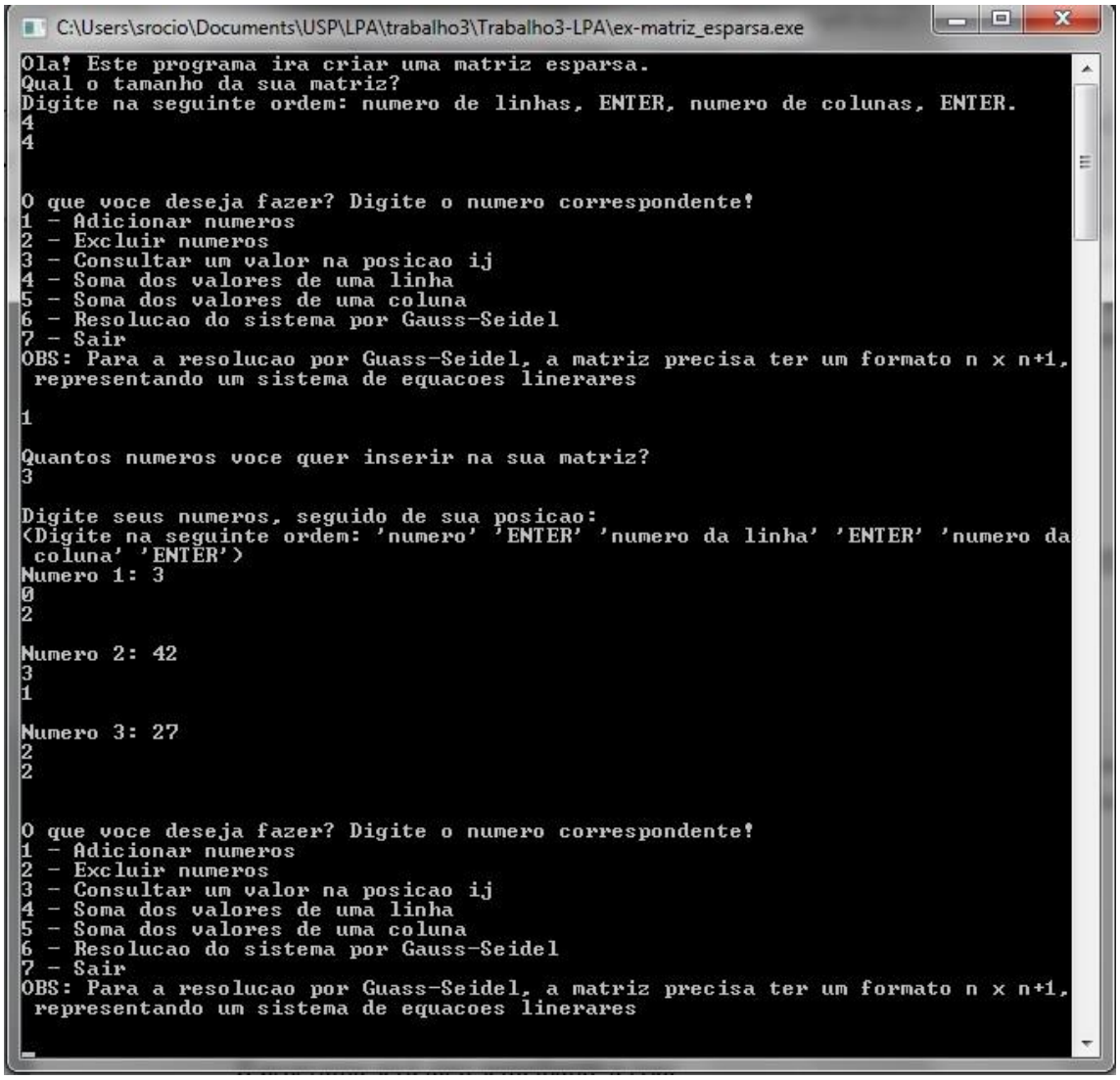

Figura 2 screenshot do programa em funcionamento

Após isso, o programa terá implementado uma matriz que pode ser representada da seguinte maneira:

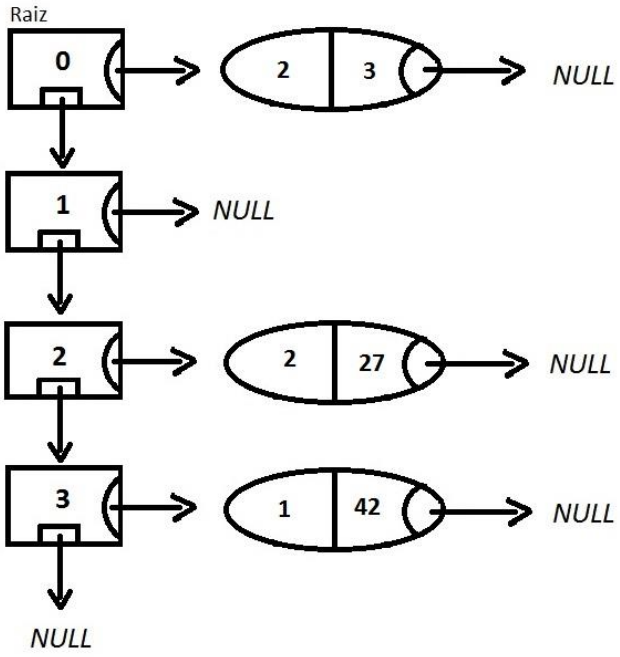

Figura 3 - Exemplo da matriz entrada

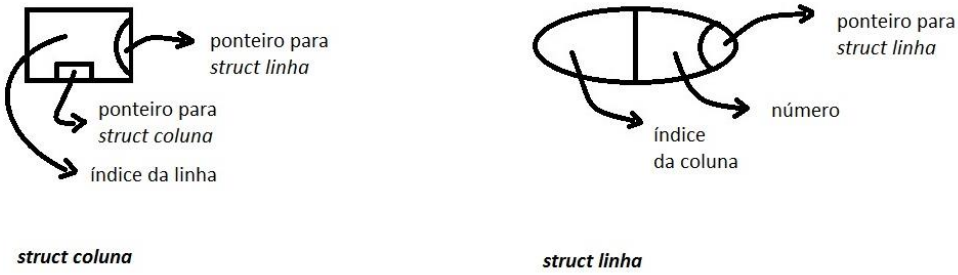

Figura 4 - Legenda

Essa matriz seria representada normalmente da seguinte forma:

$$\begin{bmatrix} 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 27 & 0 \\ 0 & 42 & 0 & 0 \end{bmatrix}$$

Nota-se a quantidade de zeros que são armazenados desnecessariamente. Assim, a estrutura da matriz esparsa otimiza o uso da memória armazenando apenas os algarismos não nulos pertencentes a matriz.

Figura 5 - Matriz normalmente representada

No menu do programa, basta digitar a opção desejada e apertar “Enter”. Por exemplo, se deseja obter a soma dos valores de uma linha da matriz, basta digitar **4** e depois **Enter**.

Questão Bônus -- Resolução de sistema por Gauss-Seidel: Foram incluídas 3 novas funções no programa: *check, gauss* e *calc.* A função *check* vai conferir se a matriz é diagonal dominante (o módulo do elemento da diagonal é maior do que a soma dos módulos dos números da mesma linha), a função *gauss* vai implementar o método de Gauss-Seidel e a função *calc* vai calcular o valor das novas variáveis, simplificando o código da função *gauss.*

Método: O método consiste em resolver um sistema de equações lineares, atribuindo primeiramente valores aleatórios às variáveis do sistema (nesse programa, o número 0 foi adotado), e a partir destes números, calcular o novo valor de cada variável diversas vezes, renovando seus valores. Após certo ponto, os valores das variáveis terão uma variação ínfima após cada renovação, então o resultado irá convergir para a solução do sistema. Para que isso funcione, basta que a matriz seja diagonal dominante. Isso é condição suficiente para assegurar que este método de solução irá funcionar.

Algoritmo: O método foi implementado com as 3 funções citadas acima. Uma descrição mais detalhada de como cada função funciona está contida nos comentários no código fonte do programa. Para este programa, o método de Gauss-Seidel é repetido 100 vezes antes de fornecer o resultado final, truncado e arredondado para 3 casas decimais. Caso seja necessário mudar, basta mudar o valor da variável global R definida logo no início do programa.

Tutorial de uso: Ao inicializar o programa, basta entrar com uma matriz no formato m x m+1. A matriz representará o sistema de equações a ser resolvido pelo método. Deve-se assumir que os elementos da coluna 0 são os coeficientes de X_0, os da coluna 1 coeficientes de X_1, os da coluna *m* coeficientes de X_m, e os da coluna *m+1* os resultados de cada equação.

Dessa forma, a matriz

$$\begin{bmatrix} 10 & 4 & 3 & 30 \\ 2 & -5 & 2 & 3 \\ 4 & 6 & 15 & 44 \end{bmatrix}$$

Representa o seguinte sistema de equações:

$10X_0 + 4X_1 + 3X_2 = 30$
$2X_0 - 5X_1 + 2X_2 = 3$
$4X_0 + 6X_1 + 15X_2 = 44$

Obs: Vale lembrar que a matriz armazenada será armazenada na estrutura de uma matriz esparsa, não como a ilustrada acima.

Após a entrada da matriz, no menu do programa, escolher a opção **6** para a resolução do sistema, e programa fornecerá os valores de X_0 X_2 X_3 ... X_{m-1}.

Observações gerais:

- Deve-se observar que a numeração das linhas e colunas da matriz inicia-se em 0. Portanto uma matriz de 4 linhas e 5 colunas tem suas linhas numeradas de 0 a 3 e colunas numeradas de 0 a 4.
- É preciso saber a dimensão da matriz previamente. A dimensão da matriz não pode ser alterada dinamicamente, por isso é definido no início do programa um tamanho para a matriz.
- Na resolução por Gauss-Seidel, os resultados são fornecidos com uma precisão de 3 casas decimais.
- A quantidade de repetições do método foi deixada como 100 vezes por default. Caso Sejam necessárias mais repetições, mudar o valor da variável global R. Ela define quantas vezes o algoritmo será repetido.