Trabalho 2 – Matriz Esparsa

SCC-502 Algoritmos e Estruturas de Dados 1 M. Cristina/Jorge

1 Introdução

O trabalho deverá ser feito **individualmente** e submetido para o sistema SSP (Sistema de Submissão de Programas), do ICMC.

Data de entrega: 05 de novembro de 2013.

1.1 Instruções de submissão

Para entrar no sistema e submeter os trabalhos, seguir os passos:

1. Entre no site <https://ssp.icmc.usp.br/>;
2. Faça login/cadastro no site;
3. Matricule-se na disciplina SCC0502 - Algoritmos e Estruturas de dados (M. Cristina) (2013/2-Turma A)
4. Clique em Submeter exercicio -> "Trabalho 2. Matriz Esparsa";
5. Submeta seu exercício;
6. O sistema fará a correção automática do seu programa, seguindo alguns casos de teste (padrões de entrada e saída) e apresentará o número de acertos e erros. Você pode submeter o programa quantas vezes desejar (até o prazo de entrega).

1.2 Critério de Avaliação

Dados:

$N_{acertos}$ = Número de casos de teste corretos;
N_{casos} = Número total de casos de teste;
$T2$ = Nota do trabalho 2.
$T2 = \frac{N_{acertos} * 10}{N_{casos}}$

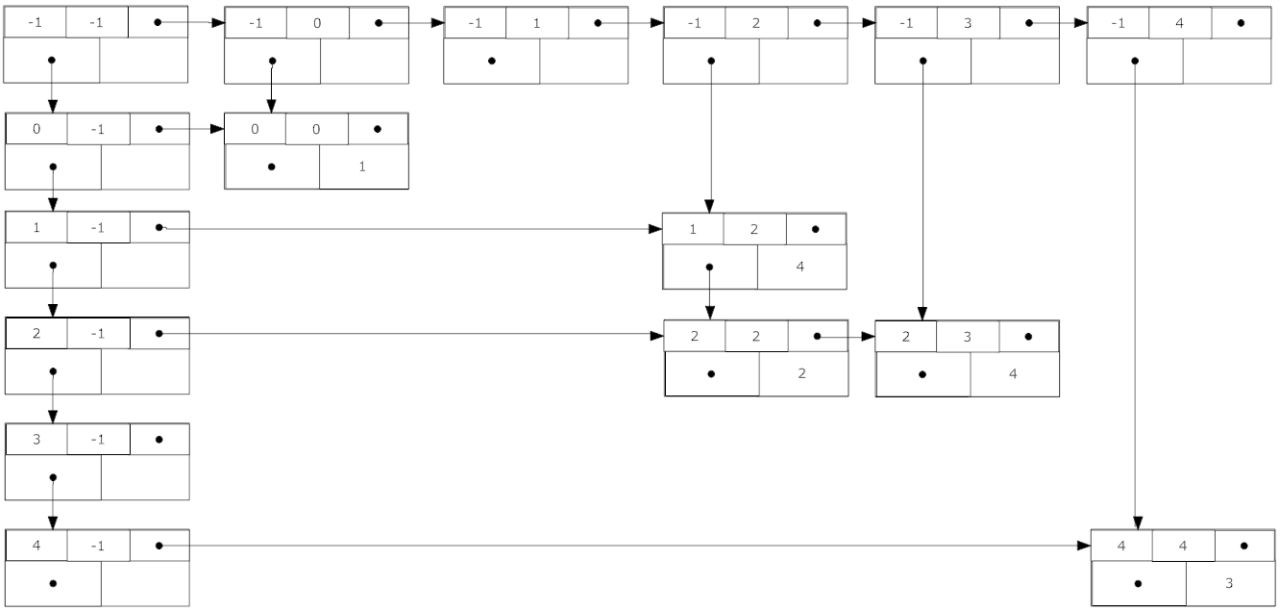
2 Descrição do Trabalho

Matrizes esparsas são matrizes em que a maioria das posições são preenchidas por zeros. Para estas matrizes, podemos economizar um espaço significativo de memória se apenas os termos diferentes de zero forem armazenados. Operações como a adição e a multiplicação de matrizes também podem ser feitas em tempo muito menor se não armazenarmos as posições que contém zeros.

Uma maneira eficiente de representar estruturas com tamanho variável e/ou desconhecido é através de alocação encadeada, utilizando listas. Cada coluna da matriz será representada por uma lista linear. Da mesma maneira, cada linha da matriz também será representada por uma lista linear.

Por exemplo, considere a matriz $A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 2 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$

Sua representação como lista é apresentada abaixo:



Considere que a estrutura para armazenar a matriz esparsa será:

```
typedef struct t_celula{
    struct t_celula *direita, *abaixo;
    int linha, coluna;
    float valor;
} Celula;

typedef struct {
    Celula *cabeca;
    int nlin,ncol;
} Matriz;
```

Seu programa deve, para cada caso de teste:

1. Ler duas matrizes esparsas;
2. Realizar a multiplicação das duas matrizes;
3. Imprimir a matriz resultante na tela;
4. Apagar as matrizes alocadas da memória;
5. Cumprir as especificações de Entrada e Saída, descritas nas seções **??** e **??**.

3 Entrada

A entrada deve ser lida do teclado (stdin).

A entrada contém vários casos de teste.

A primeira linha de cada caso de teste corresponde à entrada da primeira matriz (M_1) e contem três números inteiros, L_1, C_1 e N_1, representando o número de linhas (L_1), o número de colunas (C_1) e o número de elementos a serem inseridos na matriz (N_1).

As próximas N_1 linhas contêm dois números inteiros e um real, i_1, j_1 e v_1, que indicam a linha (i_1) e a coluna (j_1) em que o elemento (v_1) será inserido.

A próxima linha corresponde à segunda matriz (M_2) e contem três números inteiros, L_2, C_2 e N_2, representando o número de linhas (L_2), o número de colunas (C_2) e o número de elementos a serem inseridos na matriz (N_2).

As próximas N_2 linhas contêm dois números inteiros e um real, i_2 e j_2 e v_2, que indicam a linha (i_2) e a coluna (j_2) em que o elemento (v_2) será inserido.

ATENÇÃO: Todos os casos de teste devem ser processados em uma única execução do programa (utilize uma estrutura de repetição para percorrer até o fim do arquivo). As matrizes podem ter até 30000 linhas e 30000 colunas.

MUITA ATENÇÃO: Os elementos da matriz esparsa devem ser indexados a partir do zero (assim como na linguagem C).

4 Saída

A saída deve ser apresentada na saída padrão, através do comando printf

Para cada caso de teste, seu programa deve imprimir os elementos não nulos (não zeros) da matriz resultante, no mesmo formato da entrada, ou seja, <linha> <coluna> <valor>. O valor da matriz deve ter uma casa decimal, ou seja, deve-se utilizar o comando:

```
printf("%.1f",.....);
```

ATENÇÃO: Não devem ser adicionados espaços ou quebras de linha entre uma matriz e outra.

MUITA ATENÇÃO: A saída deve ser idêntica a dos exemplos de execução (Seção **??**). Varia-se, primeiramente, as colunas e, em seguida, as linhas. Um trecho de código que imprime uma matriz tradicional (não esparsa) desta maneira é dado abaixo.

```
int main(void){
    float matriz[10][10];
    int i,j;
    //......
    //...... Preenchimento da matriz
    //......
    //......
    //......
    for (j = 0; j < 10; j++){
        for (i = 0; i < 10; i++){
            if (matriz[i][j] != 0){
                printf("%d %d %.1f\n",i,j,matriz[i][j]);
            }
        }
    }
}
```

5 Exemplo de execução

Entrada	Saída
2 2 2 0 0 2.0 1 1 2.0 2 2 2 0 0 3.0 1 1 6.0	0 0 6.0 1 1 12.0
30000 30000 4 32 100 1.7 32 101 1.7 32 102 1.7 32 103 1.7 30000 30000 2 101 32 1.3 102 32 1.3	32 32 4.4
3 10000 4 0 2 3.0 1 2 3.0 0 3 3.0 1 3 3.0 10000 3 4 2 0 3.0 3 0 3.0 2 1 3.0 3 1 3.0	0 0 18.0 1 0 18.0 0 1 18.0 1 1 18.0
1 30000 3 0 9 5.0 0 10 5.0 0 11 5.0 30000 1 3 9 0 1.2 10 0 1.2 11 0 1.2	0 0 18.0

6 Considerações Finais

6.1 Tempo de execução

Para cada caso de teste, seu programa deverá ser executado em, no máximo, 30 segundos. Se a execução demorar mais do que isso, verifique se não há problemas em sua implementação. Lembre-se: aproveite o fato de que a matriz é esparsa nas funções de multiplicação de matrizes e de impressão dos elementos não nulos.

6.2 Dicas de Implementação

Num primeiro momento, você pode ter pensado em criar uma função

```
float acessa(Matriz *m, int lin, int col);
```

E trabalhar com a multiplicação (e impressão) de matrizes de maneira tradicional, por exemplo:

```
Matriz* multiplica(Matriz *m1, Matriz *m2){
    int i,j,x;
    float valor;
    Matriz *mtp; //matriz resultante
    mtp = (Matriz *)malloc(sizeof(Matriz)); //aloca a matriz
    if (mtp == NULL) return NULL;
    criaMatriz(mtp,m1->nlin,m2->ncol); //cria a matriz com o numero
    //de linhas da matriz 1 e de colunas da matriz 2
    for (i=0;i<mtp->nlin;i++){
        for (j=0;j<mtp->ncol;j++){
            valor = 0;
            for (x=0;x<m1->ncol;x++){
                valor+=acessa(m1,i,x)*acessa(m2,x,j);
            }
            if (valor != 0){
                insere(mtp,i,j,valor);
            }
        }
    }
    return mtp;
}
```

```
void imprimeNaoNulos(Matriz *m){
    int i,j;
    for (j = 0; j < m->ncol; j++){
        for (i = 0; i < m->nlin; i++){
            if (acessa(m,i,j) != 0){
                printf("%d %d %.1f\n",i,j,acessa(m,i,j));
            }
        }
    }
}
```

Lembre-se: Sua matriz é esparsa. Você está aproveitando esse fato para fazer seus cálculos? **Faça seu programa sem a função "acessa"e aproveite o grande numero de elementos nulos na sua matriz.**

6.3 Plágio

ATENÇÃO: O sistema SSP utiliza a ferramenta MOSS, de Stanford, para detecção de plágio. Para mais informações, visite: <http://theory.stanford.edu/ aiken/moss/>.