AP2 Projeto Matriz Esparsa

Airton Douglas, Vladimir Saraiva

12 de novembro de 2022

1 Introdução

Esse projeto teve como objetivo a criação de uma Matriz Esparsa.
A matriz esparsa é um tipo especial de matriz que possui elementos nulos que podem estar entre valores existentes esses elementos são mostrados como iguais a 0.Segue a imagem, exemplo de uma matriz esparsa:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \end{bmatrix}$$

Figura 1: Os valores nulos são definidos como 0.

Para essa implementação em c++ será usado a estrutura de dados de lista encadeada circular com nó sentinela implementado através de um TAD no qual os arquivos são:
Node.h, SparseMatrix.h, SparseMatrix.cpp e main.cpp que serão especificados na subseção abaixo.

1.1 Node.h

Classe Node com as atributos de linha e Coluna que vão identificar a posição que aquele nó pertence. Os atributos next e below são uma referencia para um nó a direita e abaixo respectivamente e o valor do tipo double.

```cpp
#ifndef NODE_H
#define NODE_H
#include <iostream>

class Node{
public:
    int linha;
    int coluna;
    Node *next;
    Node *below;
    double valor;

    Node(int linha,int coluna,Node*prox,Node*abaixo,double valor){
        this->linha= linha;
        this->coluna = coluna;
        next = prox;
        below = abaixo;
        this->valor = valor;
    }
};




#endif
```

Figura 2: Node.h com os atributos e construtor definidos.

1.2 SparseMatrix.h

```
#ifndef SPARSEMATRIX_H
#define SPARSEMATRIX_H
#include "Node.h"
class SparseMatrix{
public:
    //ATRIBUTOS DA MATRIZ ESPARAS
    //Node cabeça dos cabeças
    Node *m_head;
    //Node cabeça das linha
    Node *i_head;
    //Node cabeça das colunas
    Node *j_head;
    //Nº de linhas
    int i_size;
    //Nº de colunas
    int j_size;
```

Figura 3: Atributos da classe SparseMatrix.

A classe SparseMatrix tem um nó cabeça mhead, que aponta para o primeiro nó de linha ihead, que está abaixo dele e para o primeiro nó das colunas jhead, que é o proximo nó dele. O arquivo de SparseMatrix.h serve de cabeçalho para as funções que serão implementadas no SparseMatrix.cpp.

1.3 SparseMatrix.cpp

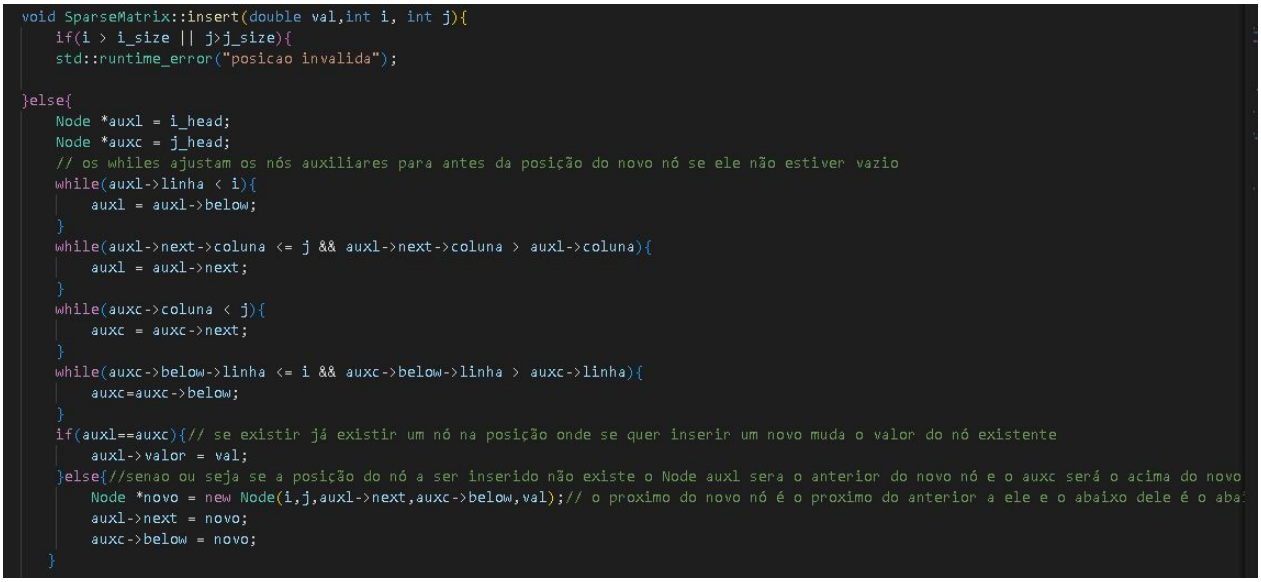
Onde são implementada as funções:

SparseMatrix::SparseMatrix(int m , int n)

```
SparseMatrix::SparseMatrix(int m, int n){
    if(m > 0 && n > 0){
        m_head = new Node(0,0,m_head,m_head,0.0);// m_head inicializa com valores zerados e apontando pra ele mesmo.
        i_size = m;// i_size representa o numero de linhas
        j_size = n;// j_size representa o numero de colunas
        j_head = new Node(0,1,m_head,j_head,0.0); //Node cabeça da primeira coluna inicializa com 1(indicando a primeira coluna matriz)
        i_head = new Node(1,0,i_head,m_head,0.0); //Node cabeça das linhas incializa com 1 (indicando a primeira linha)
        m_head->next = j_head;
        m_head->below = i_head;
        Node *auxc = j_head;//auxiliar para percorrer as colunas
        Node *auxl = i_head;//auxiliar para percorrer as linhas
        for(int i = 2; i <= m-1; i++){//cria os nós cabeças pra cada linha
            auxl->below = new Node(i,0,auxl->below,auxl->below,0.0);
            auxl=auxl->below;
        }
        auxl->below = new Node(m,0,auxl->below,m_head,0.0);// coloca o ultimo nó apontando para o nó cabeça
        for(int j = 2; j <= n-1; j++){//cria os nós cabeças de cada coluna
           auxc->next = new Node(0,j,auxc->next,auxc->next,0.0);
            auxc=auxc->next;
        }
        auxc->next = new Node(0,n,m_head,auxc->next,0.0); // o mesmo da linha 21 só que para o nó das colunas
    }else{
        std::runtime_error("A matriz nao pode ser criada valores de tamanho invalidos");
    }
}
```

Figura 4: Construtor que cria uma Matriz vazia com tamanho (m x n).Complexidade do pior caso O(n)

void SparseMatrix::insert(int i,int j, double val)



```
void SparseMatrix::insert(double val,int i, int j){
    if(i > i_size || j>j_size){
    std::runtime_error("posicao invalida");

}else{
    Node *auxl = i_head;
    Node *auxc = j_head;
    // os whiles ajustam os nós auxiliares para antes da posição do novo nó se ele não estiver vazio
    while(auxl->linha < i){
        auxl = auxl->below;
    }
    while(auxl->next->coluna <= j && auxl->next->coluna > auxl->coluna){
        auxl = auxl->next;
    }
    while(auxc->coluna < j){
        auxc = auxc->next;
    }
    while(auxc->below->linha <= i && auxc->below->linha > auxc->linha){
        auxc=auxc->below;
    }
    if(auxl==auxc){// se existir já existir um nó na posição onde se quer inserir um novo muda o valor do nó existente
        auxl->valor = val;
    }else{//senao ou seja se a posição do nó a ser inserido não existe o Node auxl sera o anterior do novo nó e o auxc será o acima do novo
        Node *novo = new Node(i,j,auxl->next,auxc->below,val);// o proximo do novo nó é o proximo do anterior a ele e o abaixo dele é o aba
        auxl->next = novo;
        auxc->below = novo;
    }
```

Figura 5: Insere nó com val na posição (i,j) caso houver um nó na posição muda o valor dele para val.(Complexidade O(n))

void SparseMatrix::print()

```
void SparseMatrix::print(){ // O(n^2)
    Node * aux;
    Node * aux1 = i_head;
    for(int j = 1; j <= j_size ; j++){
        aux=aux1->next;
        for(int i = 1; i <= i_size;i++){
            if(aux->coluna == i){
                std::cout  << aux->valor << " ";
                aux = aux->next;
            }else{
                std::cout << "0" << " ";
            }

        }
        std::cout << "\n";
        aux1=aux1->below;
    }
```

Figura 6: Imprime os valores da Matriz caso não haja valor nó na posição imprime 0.Complexidade do pior caso $O(n^2)$

1.4 main.cpp

Arquivo main.cpp contém funções adicionas: sum multiply e Read também possui um menu interativo que será especificado na subseção abaixo.

1.5 comandos do menu interativo

create i j : Cria uma matriz de tamanho (i,j) vazia onde i = número de linhas e j = número de colunas .
show: Mostra as matrizes criadas, ordenadas pelo indice que varia de 0 a n-1 (onde n é o numero de matrizes criadas).
get m i j : Exibe o valor da matriz m nas cordenadas (i,j) onde i é a linha e j é a coluna.
insert m i j v: Insere na matriz m na linha i e coluna j o valor v.
sum A B : soma a matriz A com a matriz B e cria uma nova matriz resultante desta soma (os valores de A e B indicam o indicie das matrizes que são especificados no comando show).

multiply A B : multiplica a matriz A com a matriz B e cria uma nova matriz resultante desta multiplicação (os valores de A e B indicam o indicie das matrizes que são especificados no comando show).
read "nome do arquivo.txt": lê um arquivo .txt especificado na pasta do programa.

2 Funções adicionais da main.cpp

2.1 Função de soma

```
// Soma duas matrizes e retorna uma matriz resultante complexidade : O(n^2)
SparseMatrix* sum (SparseMatrix* A,SparseMatrix* B){
    if(A->i_size != B->i_size || A->j_size != B->j_size){
       cout << "Matrizes de tamanhos diferentes nao podem ser somadas" << endl;
    }
    SparseMatrix * C = new SparseMatrix(A->i_size,A->j_size);
    for(int i = 1; i <= A->i_size;i++){
        for(int j=1; j <= A->j_size; j++){
            C->insert( ( A->get(i,j) + B->get(i,j)), i, j);
        }
    }
    return C;

}
```

Figura 7: Soma duas matrizes de mesmo tamanho e retorna a matriz resultante.(Complexidade $O(n^2)$
)

2.2 Função de multiplicação

```
SparseMatrix* multiply(SparseMatrix* A, SparseMatrix * B){ // Complexidade O(n^3)
    if(A->i_size != B->j_size){
       cout << "Nao e possivel multiplicar as matrizes"  << endl;
    }
    double aux1;
    double aux2;
    SparseMatrix *C = new SparseMatrix (A->i_size,B->j_size);

    for(int i = 1; i <= C->i_size; i++){
        for(int j = 1; j <= C->j_size; j++) {
            aux2 = 0;
            for(int k = 1; k <= A->j_size; k++){
                aux1 = A->get(i,k) * B->get(k,j);
                aux2+=aux1;
            }
            C->insert(aux2,i,j);
        }
    }
    return C;
}
```

Figura 8: multiplica duas matrizes com número de linhas de A igual número de colunas de B e retorna a matriz resultante.(Complexidade $O(n^3)$
)

2.3 Função de leitura de arquivos

```
SparseMatrix *ReadSparseMatrix(string file){// le um arquivo na
    ifstream arquivo;
    string linha;
    stringstream s;
    int numl,numc;
    double v;
    arquivo.open(file);
    if(arquivo.is_open()){
        getline(arquivo,linha);
        s << linha;
        s >> numl >> numc;
        SparseMatrix *m = new SparseMatrix(numl,numc);
        while(getline(arquivo,linha)){
            s << linha;
            s >> numl >> numc;
            s >> v;
            m->insert(v,numl,numc);
        }
        return m;

    }else{
        cout << "Arquivo nao foi encontrado " << endl;
    }


}
```

Figura 9: lê uma string com o nome exato do arquivo a ser lido (Complexidade $O(n)$)

3 Erros e Dificuldades Encontradas

3.1 Função print

O problema encontrado na função print na hora de imprimir os valores vazios existentes na matriz esparsas preenchendo-os com 0.

3.2 Função insert

Na hora de inserir um valor foi usado o parametro de perguntar se a coluna do proximo era maior que a do anterior e se a linha do acima era menor que a linha do abaixo caso fosse um valor ja existia ali então era só mudá-lo. porem um problema acontecia ao tentar inserir um valor no começo da linha. Ele não era inserido.
Os testes foram segundo main.cpp(Testes feitos antes da criação do menu interativo): SparseMatrix m(3,3);//Matriz 3x3 criada //entradas: inserir 3.3 na posicao (3,2); 3.9 na (3,3) , 5.0 na (3,2) e 3.3 na (3,1)
m.insert(3.3,3,2);
m.insert(3.9,3,3)

m.insert(5.0,3,2);
m.insert(3.3,3,1);
m.print();
std::cout « std::endl;
Saída do programa
0 0 0
0 0 0
0 5 3.9
Saída esperada
0 0 0
0 0 0
3.3 5 3.9
Outro problema encontrado na função insert atribuir valor na primeira coluna de qualquer linha resultava em um erro de segmentação que fechava o programa

valores de entrada:
insert 0 3 1 3.3 (inserido na matriz 0 na posição (3,1) o valor 3.3)
Saída esperada
0 0 0
0 0 0
3.3 0 0

Porém, o programa parou a criação do nó na primeira coluna não funcionava. Foi feito outros testes nas primeiras colunas e resultava no mesmo.

3.3 Função get

Na função get foi encontrado um erro ao acessar valores dentro do tamanho da matriz mas, que não tinha posição definida(ou seja eram nulos) isso era percibido ao chamar essa função na função de sum da main.cpp, pois na hora de somar valores da em posições que não existessem a variavel auxiliar que estava percorrendo a matriz poderia pegar um valor já usado para somar e botá-lo em outra posição. O teste que esse erro foi percebido foi o seguinte:
Entradas:
create 3 3 (criada matriz 3x3)
create 3 3 (criada matriz 3x3)
insert 0 1 1 2.5 (inserir na matriz 0 posicao 1x1 valor 2.5)
insert 1 1 1 2.5
sum 0 1
show
saídas:
Matriz 0:
2.5 0 0
0 0 0
0 0 0

Matriz 1:
2.5 0 0
0 0 0
0 0 0
Matriz 2:
5 0 5
0 0 0
0 0 0
saída esperada:
Matriz 2:
5 0 0
0 0 0
0 0 0

no caso esse problema com a soma ocorria porque a função get retornava um valor mesmo se as coordenadas não correspondessem a nenhum elemento da matriz.
outro erro encontrado no get e percebido ao chamar a função de somar duas matrizes. Nesse teste foram criadas duas matrizes 3x3 com valores setados em posições iguais a saida do programa foi esta:
O certo seria na posição (2,2) da matriz 2 que é a resultante da soma da matriz 0 com a

```
Matriz 0:
3.4 5.4 7.4
0 4.4 0
5.7 5 0
Matriz 1:
6.4 6.4 1.4
0 4.5 0
7.7 8.1 0
Matriz 2:
9.8 11.8 8.8
0 0 0
13.4 13.1 0
```

matriz 1 o valor seria 8.9 que é soma de 4.4 com 4.5 mas essa posição está zerada.

4 Divisão de Atividades

A separação de tarefas ficou a Seguinte:
Airton:Ficou responsável pela criação do TAD e da implementação das funções da Sparse-Matrix.h(get,insert,construtor e destrutor) criação do menu interativo e da função read da main.cpp.
Vladimir: Ficou responsável pela criação da lógica das funções de soma e multiplicação de matrizes e por fazer a análise do grau de complexidade das funções.

5 Referências

https://www.youtube.com/watch?v=C$_e$PgrEbLs0
Engenharia de Computação Univesp - Estrutura de Dados Curso de Engenharia de Computação Disciplina EID-001 - Estrutura de Dados Univesp - Universidade Virtual do Estado de São Paulo. Professor responsável: Norton T. Roman Professor ministrante: Luciano A. Digiampietri
https://www.educba.com/c-plus-plus-fstream/
ISO 10004:2018 ISO 9001:2015 Certified © 2022 - EDUCBA. ALL RIGHTS RESERVED. THE CERTIFICATION NAMES ARE THE TRADEMARKS OF THEIR RESPECTIVE OWNERS.
https://www.youtube.com/watch?v=1fi2P08vNWI
Produção: Bruno P. Campos / CFB Cursos
Edição: Bruno P. Campos / CFB Cursos
Licença padrão do YouTube