Matriz Esparsa e Lista Cruzada

Na matriz (5x6) abaixo, apenas 5 de seus 30 elementos são não nulos.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

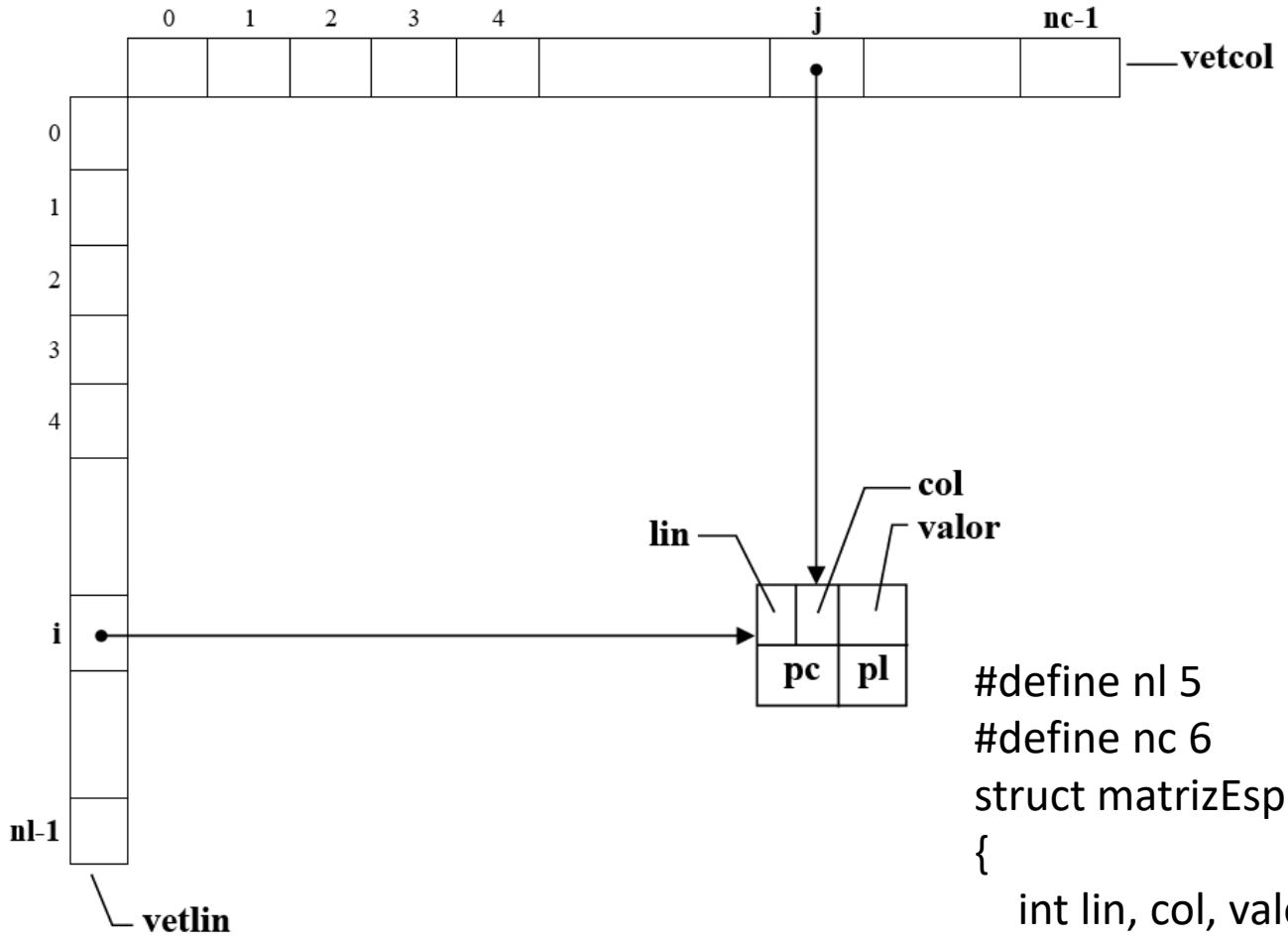
Estrutura de Dados de Listas Cruzadas propõe uma representação que evita o armazenamento de elementos nulos.

Para não representarmos os valores nulos, fazemos **listas de linhas** e **listas de colunas** tais que o elemento m(*i*,*j*) diferente de 0 pertence:

- à lista dos elementos da linha *i*;
- à lista dos elementos da coluna *j*.

Então se a matriz tem *nl* linhas e *nc* colunas, temos *nl* listas de linhas e *nc* listas de colunas.

Graficamente podemos ter algo como:

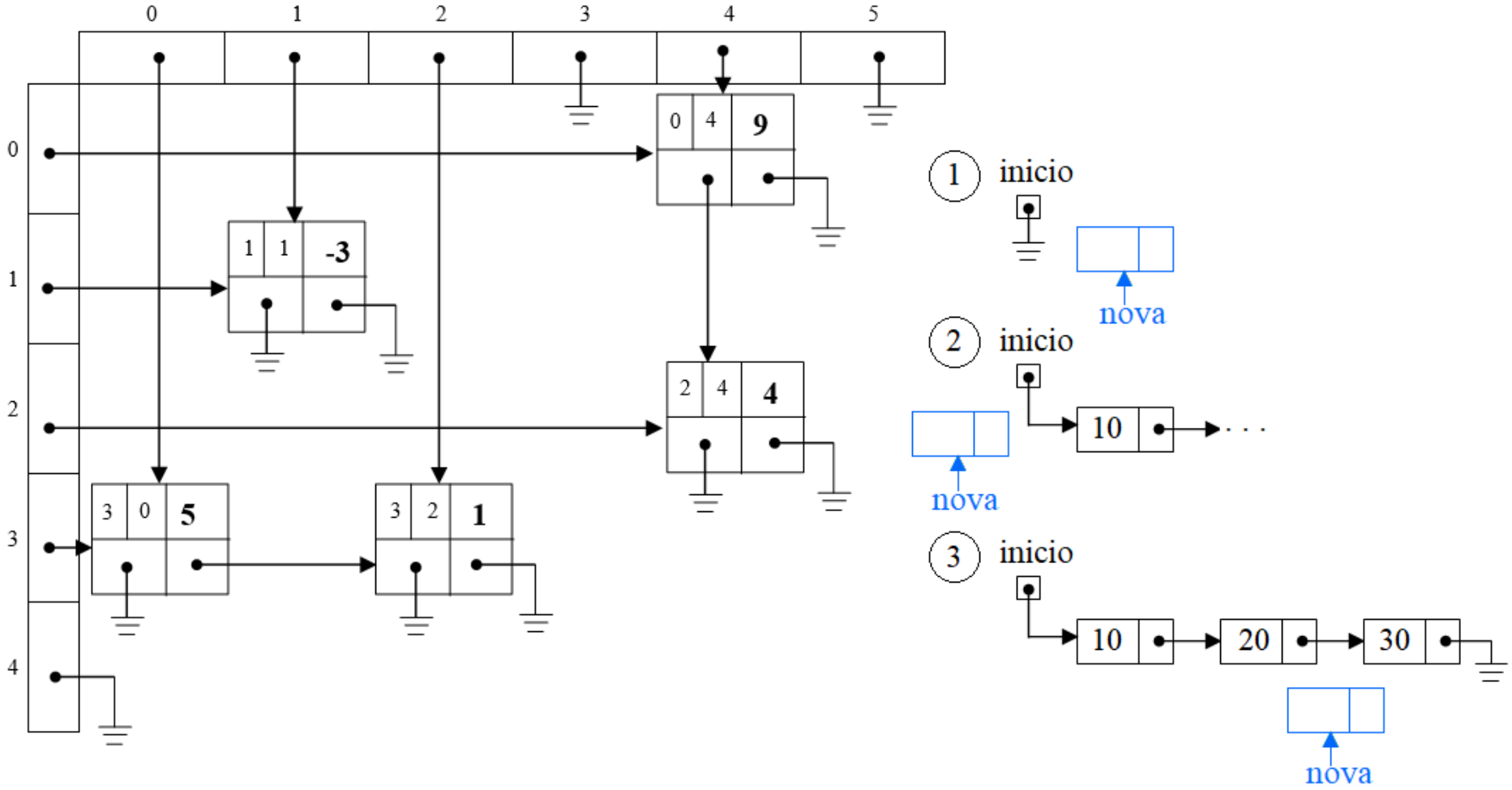


Definição da Estrutura de Dados:

```
#define nl 5
#define nc 6
struct matrizEsp
{
   int lin, col, valor;
   struct matrizEsp *pc, *pl;
};
typedef struct matrizEsp MatEsp;
```

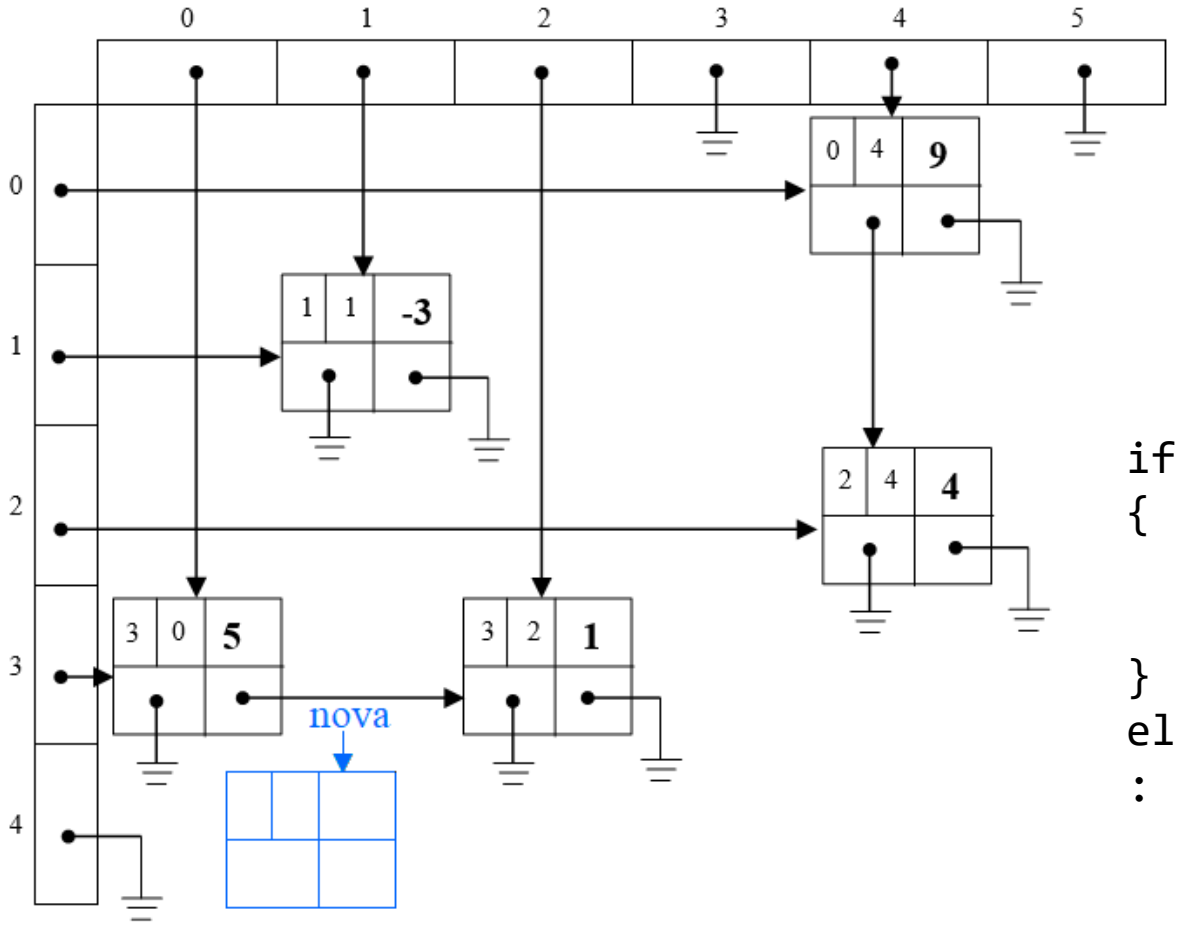
Algoritmo para inserir em um Matriz Esparsa:

3 casos a considerar: 1-aponta para NULL, 2-menor que todos, 3-no meio ou no final.



Algoritmo para inserir em um Matriz Esparsa (horizontal):

1º Caso: aponta para NULL.

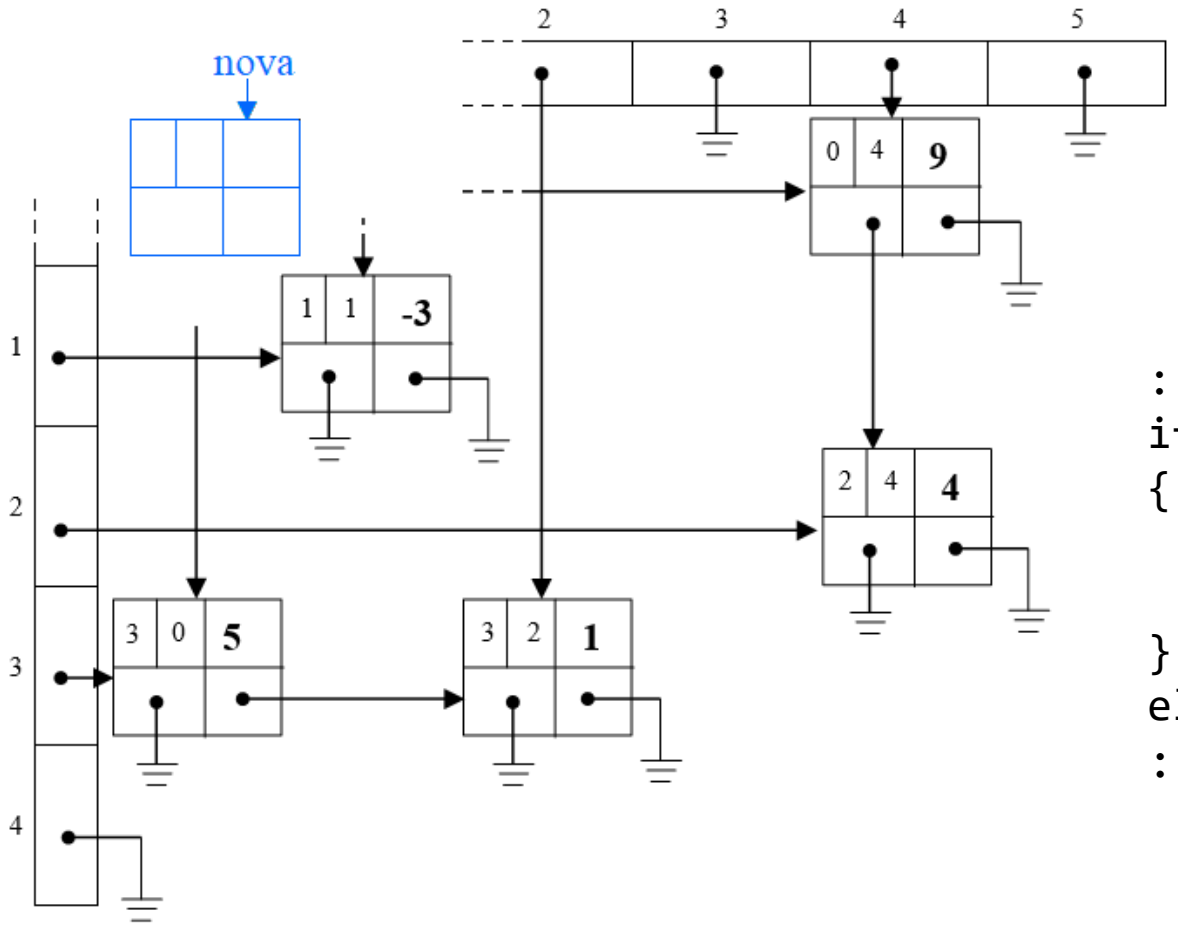



```
lin=4
col=1
valor=30
```

```
if (velin[lin]==NULL)
{
   vetlin[lin]=nova;
   nova->pl=NULL;
}
else
:
```

Algoritmo para inserir em um Matriz Esparsa (horizontal):

2º Caso: menor que todos.

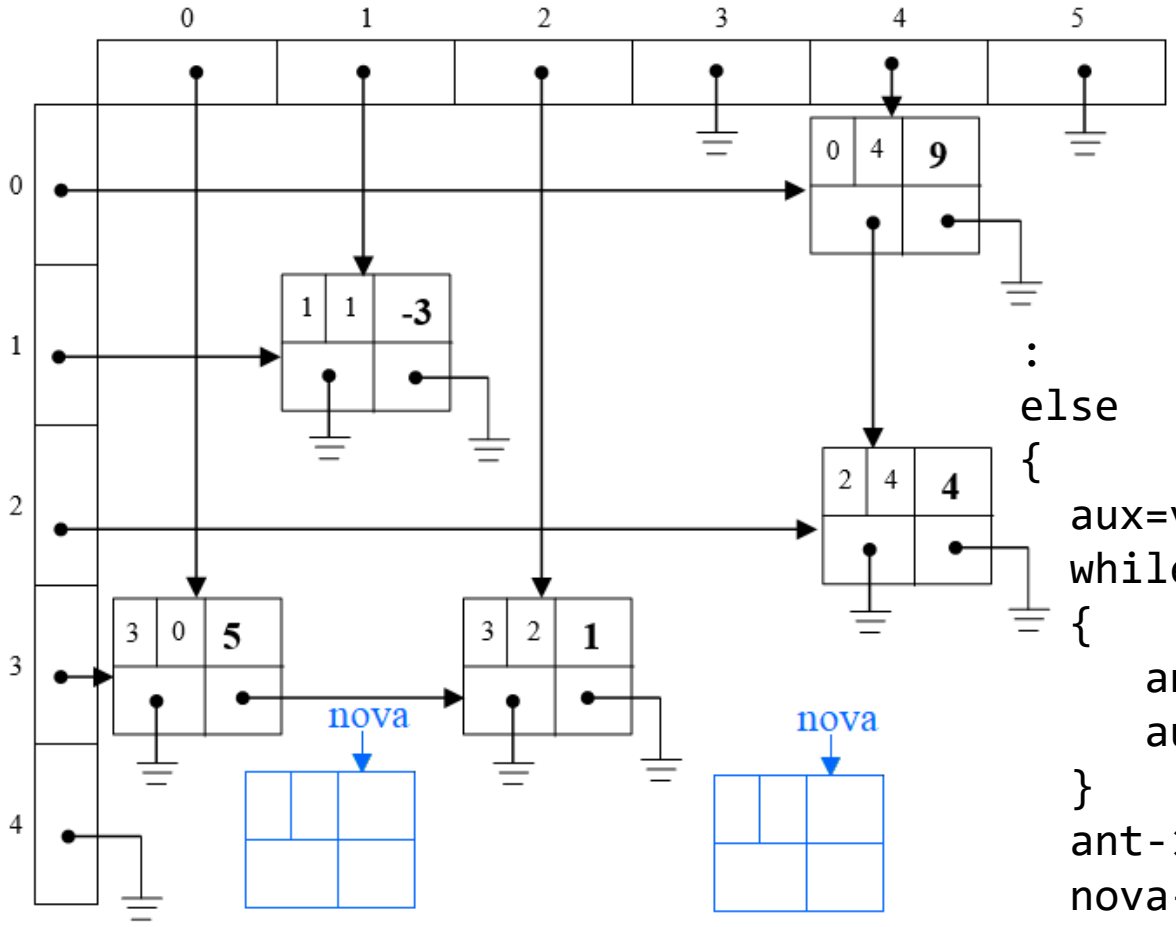



```
lin=1
col=0
valor=30
```

```
:
if (col<vetlin[lin]->col)
{
   nova->pl=vetlin[lin];
   vetlin[lin]=nova;
}
else
:
```

Algoritmo para inserir em um Matriz Esparsa (horizontal):

3º Caso: no meio ou no final.




```
lin=3
col=1
valor=30
```

```
:
else
{
   aux=vetlin[lin];
   while(aux!=NULL && col>aux->col)
   {
      ant=aux;
      aux=aux->pl;
   }
   ant->pl=nova;
   nova->pl=aux;
}
```

Algoritmo para inserir em um Matriz Esparsa:

```
void insere(MatEsp *vetlin[], MatEsp *vetcol[], int lin, int col,
                                                        int valor)
{
  MatEsp *ant,*aux,*nova;
  if (lin>=0 && lin<nl && col>=0 && col<nc)
  {
     verificaOcupado(vetlin,lin,col,&aux);
     if (aux!=NULL)
        aux->valor=valor;
     else
     {
       nova=(MatEsp*)malloc(sizeof(MatEsp));
       nova->lin=lin;
       nova->col=col;
       nova->valor=valor;
       //Ligação Horizontal
       //Ligação Vertical
    }
  }
  else
    printf("As coordenadas estão fora da Matriz!");
}
```

Algoritmos a serem implementados no TAD Matriz Esparsa:

a) inicializar uma matriz esparsa;

b) inserir um determinado elemento na posição *i*, *j*;

c) excluir um elemento da posição *i*, *j*;

d) exibir uma matriz esparsa;

e) somar duas matrizes esparsas e gerar uma terceira;

f) fazer a multiplicação de duas matrizes esparsas;

g) excluir uma matriz esparsa.