Matrizes Esparsas

Prof. Silvio Luiz Bragatto Boss

Algoritmos em Grafos (AG44CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

DAINF
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CÂMPUS PATO BRANCO

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Matrizes Esparsas

- **Matrizes Esparsas** são matrizes nas quais a maioria das posições é preenchida por zeros;

Matrizes Esparsas

- **Matrizes Esparsas** são matrizes nas quais a maioria das posições é preenchida por zeros;
- Para sua representação computacional, podemos reduzir o espaço ocupado por ela significativamente armazenando apenas os elementos diferentes de zero.

Matrizes Esparsas

- Quanta memória ocupa a matriz abaixo quando representada em C?

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 34 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 5 & 0 & 12 & 0 & 0 & 0 \end{bmatrix}$$

Matrizes Esparsas

- Como armazenar uma matriz esparsa em C gastando pouca memória?

Matrizes Esparsas

- Como armazenar uma matriz esparsa em C gastando pouca memória?

Existem algumas alternativas

Matrizes Esparsas

- Como armazenar uma matriz esparsa em C gastando pouca memória?

Existem algumas alternativas

1. Vetor de ponteiros para linhas apenas;

Matrizes Esparsas

- Como armazenar uma matriz esparsa em C gastando pouca memória?

Existem algumas alternativas

1. Vetor de ponteiros para linhas apenas;
2. Vetor de ponteiros para colunas apenas;

Matrizes Esparsas

- Como armazenar uma matriz esparsa em C gastando pouca memória?

Existem algumas alternativas

1. Vetor de ponteiros para linhas apenas;
2. Vetor de ponteiros para colunas apenas;
3. Utilizar simultaneamente ambas;

Matrizes Esparsas

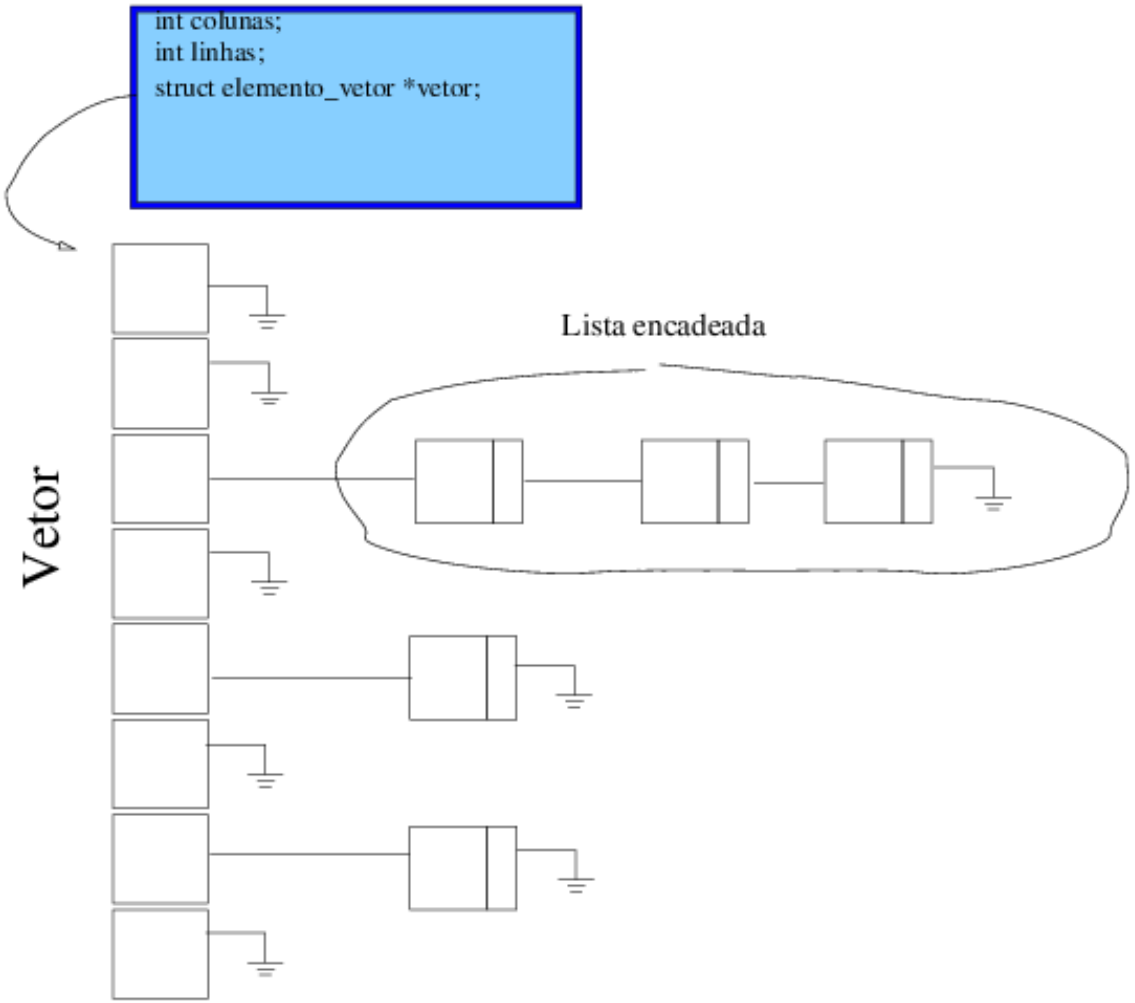
- Como armazenar uma matriz esparsa em C gastando pouca memória?

Existem algumas alternativas

1. Vetor de ponteiros para linhas apenas;
2. Vetor de ponteiros para colunas apenas;
3. Utilizar simultaneamente ambas;
4. Utilizar listas ao invés de vetores. Cada elemento da lista aponta para a lista de linha/coluna

Matrizes Esparsas

- Para o primeiro caso, temos a seguinte representação:



Matrizes Esparsas

- Nesta aula, focaremos a apresentação para resolver o problema para o quarto caso;

Matrizes Esparsas

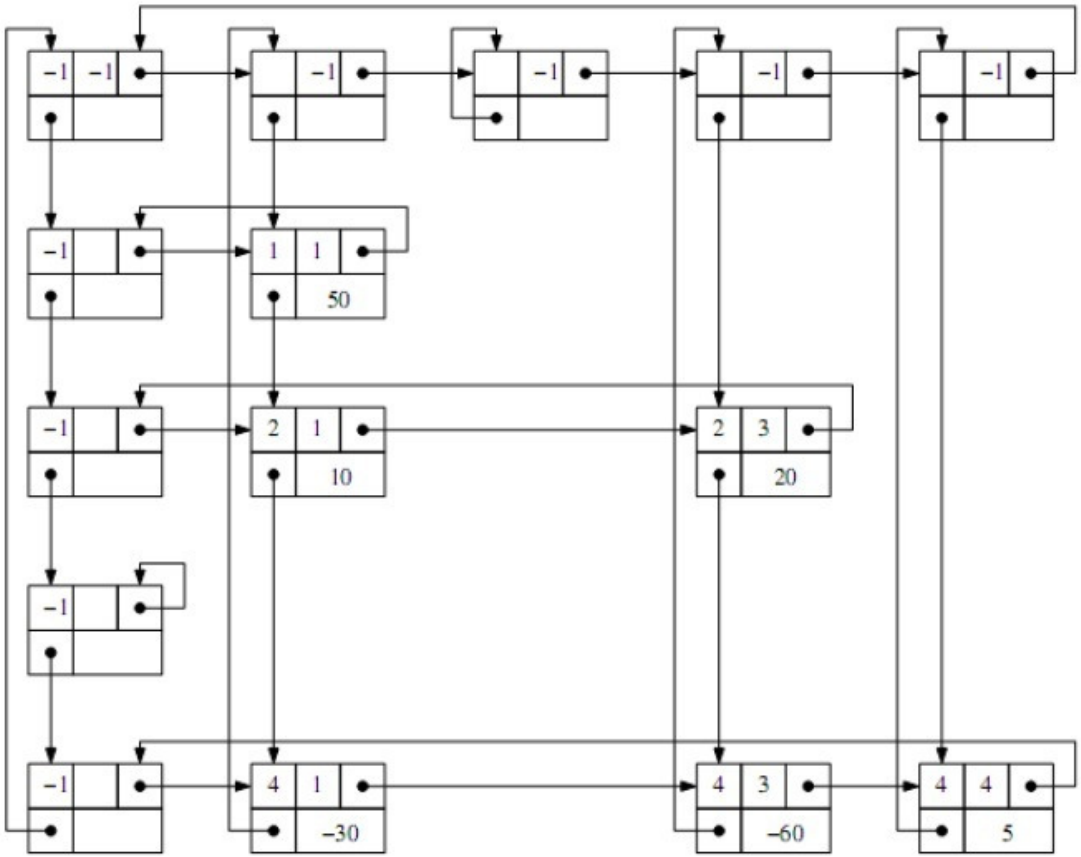
- Nesta aula, focaremos a apresentação para resolver o problema para o quarto caso;
- Assim, a solução utiliza listas circulares encadeadas com uma célula cabeça para a composição da matriz;

Matrizes Esparsas

- Nesta aula, focaremos a apresentação para resolver o problema para o quarto caso;
- Assim, a solução utiliza listas circulares encadeadas com uma célula cabeça para a composição da matriz;
- Cada linha e coluna é representada por uma lista circular.

Matrizes Esparsas

- Representação de uma matriz esparsa utilizando listas encadeadas circulares.



Matrizes Esparsas

Tipo Abstrato de Dados

- Para a representação das matrizes esparsas, declaramos duas estruturas de dados;

Matrizes Esparsas

Tipo Abstrato de Dados

- Para a representação das matrizes esparsas, declaramos duas estruturas de dados;
- A primeira é **tCelula**, que representa cada posição da matriz;

Matrizes Esparsas

Tipo Abstrato de Dados

- Para a representação das matrizes esparsas, declaramos duas estruturas de dados;
- A primeira é **tCelula**, que representa cada posição da matriz;
- Temos um ponteiro que aponta para a célula a direita, e um outro ponteiro que aponta para a célula que vem abaixo;

Matrizes Esparsas

Tipo Abstrato de Dados

- Para a representação das matrizes esparsas, declaramos duas estruturas de dados;
- A primeira é **tCelula**, que representa cada posição da matriz;
- Temos um ponteiro que aponta para a célula a direita, e um outro ponteiro que aponta para a célula que vem abaixo;
- **linha** e **coluna** contém a posição da célula na matriz;

Matrizes Esparsas

Tipo Abstrato de Dados

- Para a representação das matrizes esparsas, declaramos duas estruturas de dados;
- A primeira é **tCelula**, que representa cada posição da matriz;
- Temos um ponteiro que aponta para a célula a direita, e um outro ponteiro que aponta para a célula que vem abaixo;
- **linha** e **coluna** contém a posição da célula na matriz;
- **valor** armazena o dado inserido pelo usuário.

Matrizes Esparsas

- A segunda estrutura, **tMatriz**, é composta por três ponteiros para outras **tCelula**: *inicio*, *fimLinha*, *fimColuna*;

Matrizes Esparsas

- A segunda estrutura, **tMatriz**, é composta por três ponteiros para outras **tCelula**: *inicio*, *fimLinha*, *fimColuna*;
  - *inicio* aponta para a célula cabeça principal;

Matrizes Esparsas

- A segunda estrutura, **tMatriz**, é composta por três ponteiros para outras **tCelula**: *inicio*, *fimLinha*, *fimColuna*;
  - *inicio* aponta para a célula cabeça principal;
  - *fimLinha* aponta para a última célula cabeça de lista-linha;

Matrizes Esparsas

- A segunda estrutura, **tMatriz**, é composta por três ponteiros para outras **tCelula**: *inicio*, *fimLinha*, *fimColuna*;
  - *inicio* aponta para a célula cabeça principal;
  - *fimLinha* aponta para a última célula cabeça de lista-linha;
  - *fimColuna* aponta para a última célula cabeça de lista-coluna.

Matrizes Esparsas

- A segunda estrutura, **tMatriz**, é composta por três ponteiros para outras **tCelula**: *inicio*, *fimLinha*, *fimColuna*;
  - *inicio* aponta para a célula cabeça principal;
  - *fimLinha* aponta para a última célula cabeça de lista-linha;
  - *fimColuna* aponta para a última célula cabeça de lista-coluna.
- Os inteiros **m** e **n** representam a dimensão da matriz.

Matrizes Esparsas

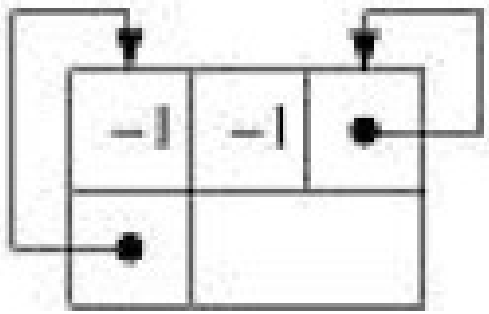
```
typedef struct tCelula
{
  int linha, coluna;
  int valor;
  tCelula *direita, *abaixo;
}tCelula;
```

```
typedef struct
{
  int m,n //dimensao da matriz
  tCelula *inicio, *fimLinha, *fimColuna;
}tMatriz;
```

Matrizes Esparsas

Inicialização da Matriz

- Para inicializar a Matriz é criado um nó denominado **célula cabeça**;
- Para computar se uma Matriz é vazia, checa se a matriz é uma célula cabeça:

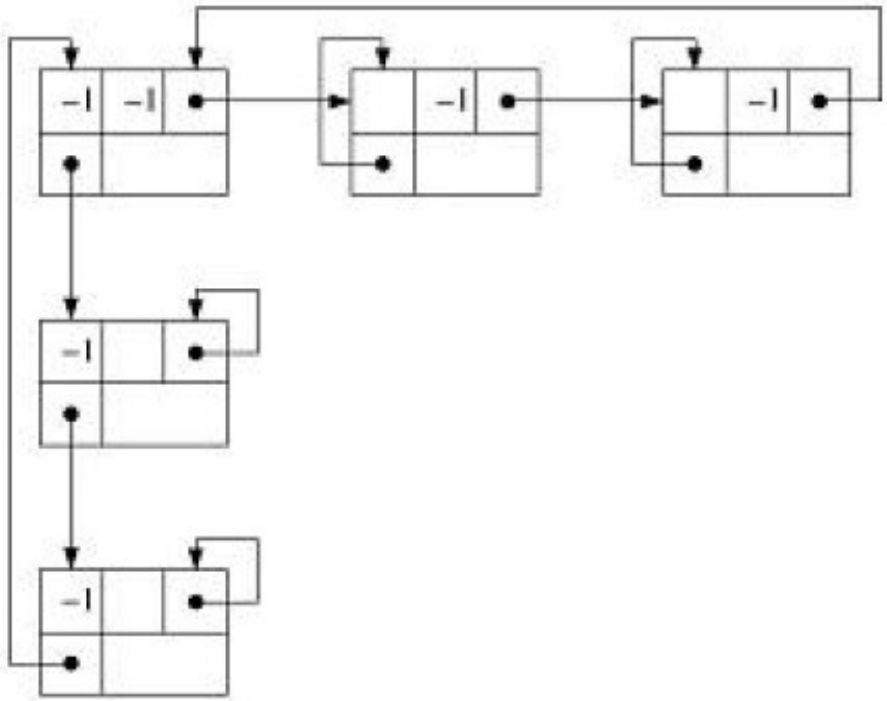


Matrizes Esparsas

- Após criada a célula cabeça, instancia-se novos elementos através da quantidades de linhas e colunas da matriz.
- Com isso, deve-se desenvolver:
  - Uma função para instanciar a quantidade de colunas a matriz terá e,
  - Uma função para instanciar a quantidade de linhas a matriz terá.

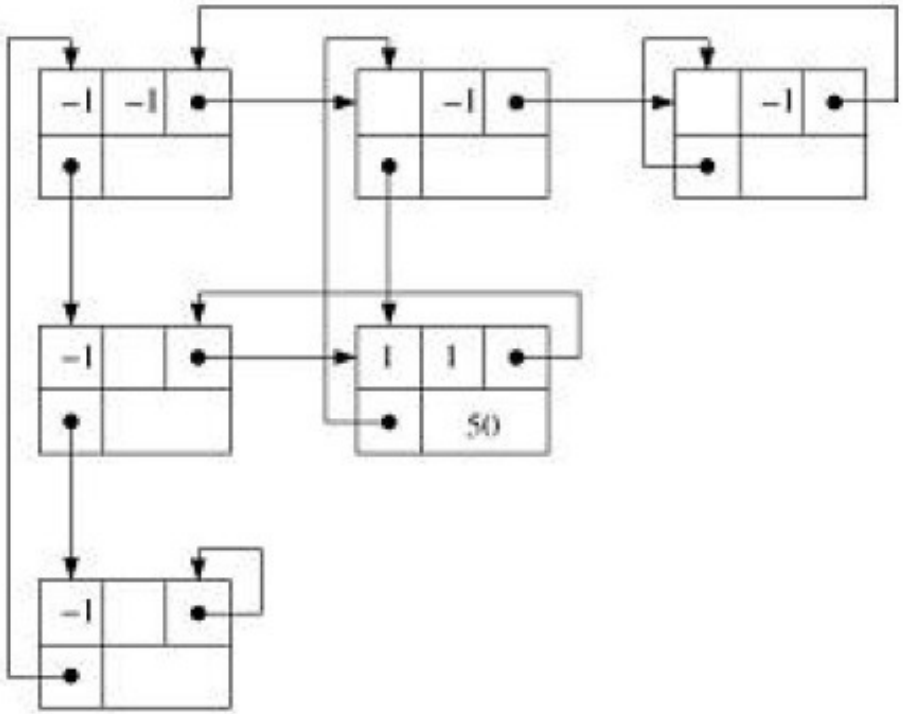
Matrizes Esparsas

- A Figura abaixo apresenta uma matriz esparsa instanciada de dimensão **2x2**



Inserção de um Elemento na Matriz

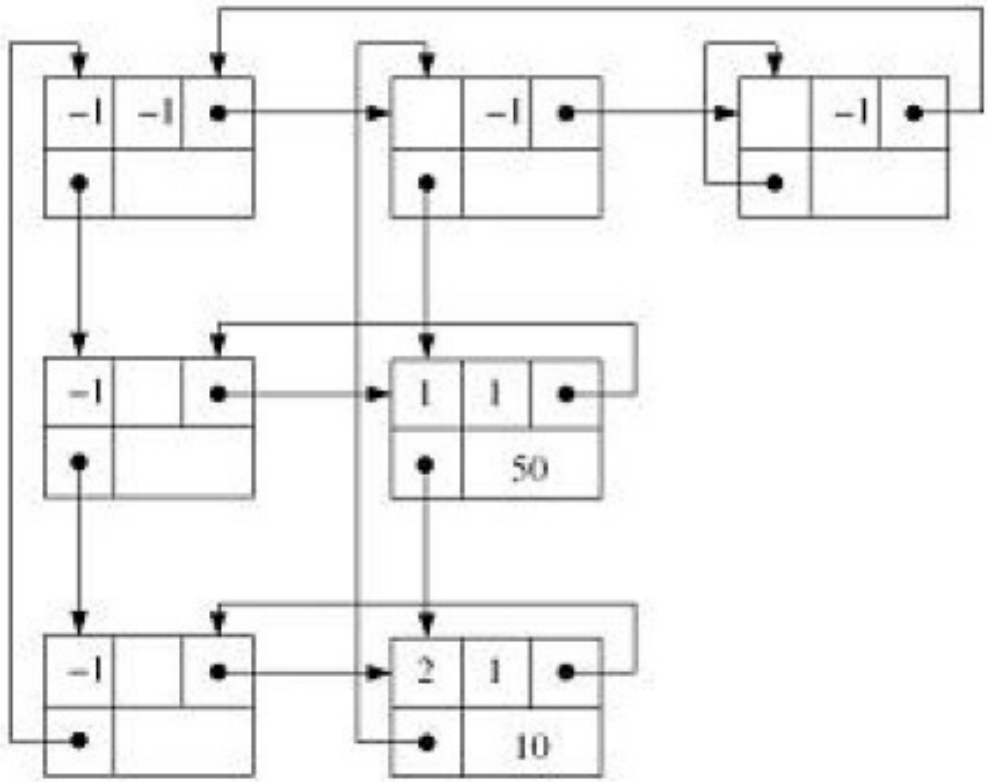
- Para inserir um elemento na matriz, informamos apenas as coordenadas onde ele será inserido e o valor da célula.
- Supondo a chamada da função **insere(1,1,50)**, temos:



Matrizes Esparsas

Inserção de um Elemento na Matriz

- Com a chamada da função **insere(2,1,10)**, temos:



Dúvidas?