Matrizes Esparsas

SCC0202 - Algoritmos e Estruturas de Dados I

Prof. Fernando V. Paulovich
**Baseado no material do Prof. Gustavo Batista*
http://www.icmc.usp.br/~paulovic
paulovic@icmc.usp.br

Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)

23 de outubro de 2017

Sumário

1 Matrizes Esparsas – Listas Cruzadas

2 Representação Alternativa – Listas Cruzadas Circulares

Sumário

1 Matrizes Esparsas – Listas Cruzadas

2 Representação Alternativa – Listas Cruzadas Circulares

O Problema

- Representação de matrizes com muitos elementos nulos

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

O Problema

- Representação de matrizes com muitos elementos nulos
  - P.ex., matriz abaixo, de 5 linhas por 6 colunas: apenas 5 dos 30 elementos são não nulos

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

O Problema

- Representação de matrizes com muitos elementos nulos
  - P.ex., matriz abaixo, de 5 linhas por 6 colunas: apenas 5 dos 30 elementos são não nulos
  - Precisamos de uma representação que evite o armazenamento de tantos zeros

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

O Problema

- Representação de matrizes com muitos elementos nulos
  - P.ex., matriz abaixo, de 5 linhas por 6 colunas: apenas 5 dos 30 elementos são não nulos
  - Precisamos de uma representação que evite o armazenamento de tantos zeros
  - Solução: utilizar listas cruzadas como estruturas de dados

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Representação por Listas Cruzadas

- Cada elemento identificado pela sua linha, coluna, e valor

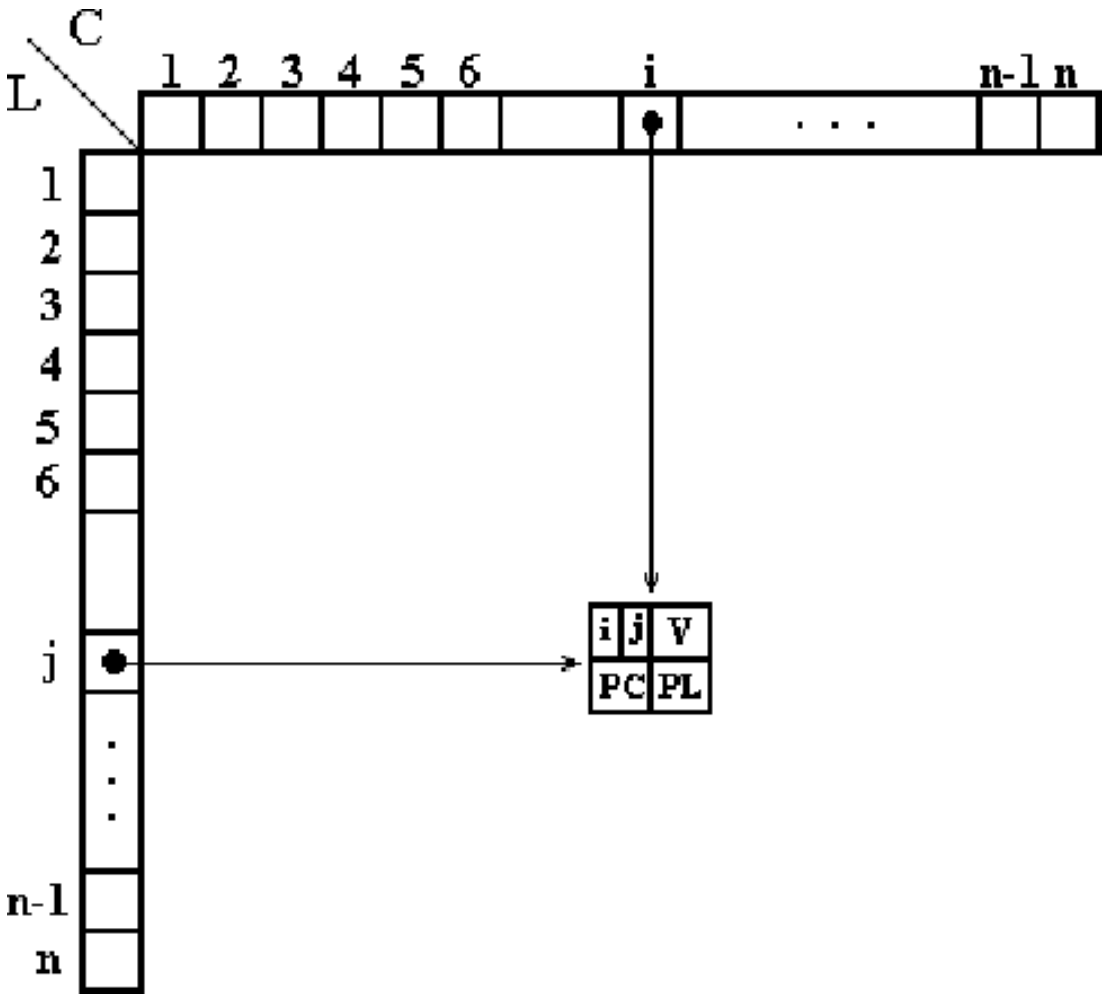
Representação por Listas Cruzadas

- Cada elemento identificado pela sua linha, coluna, e valor
- Cada elemento a_{ij} não-nulo pertence a uma lista de valores não nulos da linha i e também a uma lista de valores não nulos da coluna j

Representação por Listas Cruzadas

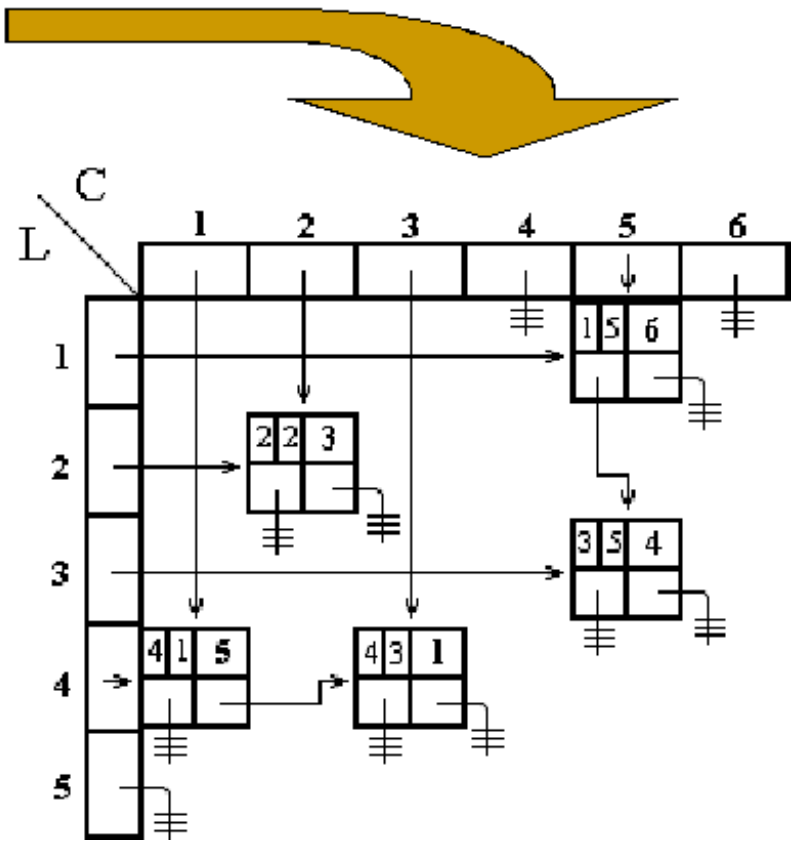
- Cada elemento identificado pela sua linha, coluna, e valor
- Cada elemento a_{ij} não-nulo pertence a uma lista de valores não nulos da linha i e também a uma lista de valores não nulos da coluna j
- Assim, para matriz de nl linhas e nc colunas, teremos nl listas de linhas e nc listas de colunas

Listas Cruzadas



Listas Cruzadas

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



TAD Matriz Esparsa

- Pode-se criar um TAD bastante simples para matrizes esparsas

TAD Matriz Esparsa

- Pode-se criar um TAD bastante simples para matrizes esparsas
- Operações principais

TAD Matriz Esparsa

- Pode-se criar um TAD bastante simples para matrizes esparsas
- Operações principais
  - *criar_matriz(n, m)* : cria uma nova matriz esparsa vazia com n linhas e m colunas

TAD Matriz Esparsa

- Pode-se criar um TAD bastante simples para matrizes esparsas
- Operações principais
  - *criar_matriz(n, m)* : cria uma nova matriz esparsa vazia com n linhas e m colunas
  - *set(M, lin, col, valor)* : define um valor na posição *(lin, col)* da matriz esparsa *M*

TAD Matriz Esparsa

- Pode-se criar um TAD bastante simples para matrizes esparsas
- Operações principais
  - *criar_matriz(n, m)* : cria uma nova matriz esparsa vazia com n linhas e m colunas
  - *set(M, lin, col, valor)* : define um valor na posição *(lin, col)* da matriz esparsa *M*
  - *get(M, lin, col)* : retorna o valor na posição *(lin, col)* da matriz esparsa *M*

TAD Matriz Esparsa

- Operações auxiliares (podem ser criadas a partir das operações principais)

TAD Matriz Esparsa

- Operações auxiliares (podem ser criadas a partir das operações principais)
  - *somar_matriz(M1, M2, R)* : Soma as matrizes *M1* e *M2* e armazena o resultado em *R*

TAD Matriz Esparsa

- Operações auxiliares (podem ser criadas a partir das operações principais)
  - *somar_matriz(M1, M2, R)* : Soma as matrizes *M1* e *M2* e armazena o resultado em *R*
  - *multiplicar_matriz(M1, M2, R)* : Multiplica as matrizes *M1* e *M2* e armazena o resultado em *R*

TAD Matriz Esparsa

- Operações auxiliares (podem ser criadas a partir das operações principais)
  - *somar_matriz(M1, M2, R)* : Soma as matrizes *M1* e *M2* e armazena o resultado em *R*
  - *multiplicar_matriz(M1, M2, R)* : Multiplica as matrizes *M1* e *M2* e armazena o resultado em *R*
  - *somar_coluna(M, V, col)* : Soma uma constante *V* a todos os elementos da coluna *col* da Matriz *M*

TAD Matriz Esparsa

- Operações auxiliares (podem ser criadas a partir das operações principais)
  - *somar_matriz(M1, M2, R)* : Soma as matrizes *M1* e *M2* e armazena o resultado em *R*
  - *multiplicar_matriz(M1, M2, R)* : Multiplica as matrizes *M1* e *M2* e armazena o resultado em *R*
  - *somar_coluna(M, V, col)* : Soma uma constante *V* a todos os elementos da coluna *col* da Matriz *M*
  - *somar_linha(M, V, lin)* : Soma uma constante *V* a todos os elementos da linha *lin* da Matriz *M*

TAD Matriz Esparsa

- Operações auxiliares (podem ser criadas a partir das operações principais)
  - *somar_matriz(M1, M2, R)* : Soma as matrizes *M1* e *M2* e armazena o resultado em *R*
  - *multiplicar_matriz(M1, M2, R)* : Multiplica as matrizes *M1* e *M2* e armazena o resultado em *R*
  - *somar_coluna(M, V, col)* : Soma uma constante *V* a todos os elementos da coluna *col* da Matriz *M*
  - *somar_linha(M, V, lin)* : Soma uma constante *V* a todos os elementos da linha *lin* da Matriz *M*
  - E mais: inverter, transpor, calcular determinante, etc...

Estrutura de Dados

- A implementação é facilitada se as listas contêm o nó cabeça

```
1  typedef struct matriz_esparsa MATRIZ_ESPARSA;
2
3  typedef struct CELULA {
4    int linha;
5    int coluna;
6    float valor;
7    struct CELULA *direita;
8    struct CELULA *abaixo;
9  } CELULA;
10
11 struct matriz_esparsa {
12   CELULA **linhas;
13   CELULA **colunas;
14   int nr_linhas;
15   int nr_colunas;
16 };
```

Operações

- Vamos implementar as operações **criar_matriz(...)**, **apagar_matriz(...)**, **set(...)** e **get(...)** do conjunto de operações principais

Operações

- Vamos implementar as operações **criar_matriz(...)**, **apagar_matriz(...)**, **set(...)** e **get(...)** do conjunto de operações principais
- As demais operações principais e auxiliares ficam como exercício

Operações

- Vamos implementar as operações **criar_matriz(...)**, **apagar_matriz(...)**, **set(...)** e **get(...)** do conjunto de operações principais
- As demais operações principais e auxiliares ficam como exercício
- Entretanto, vamos discutir alguns aspectos importantes dessas operações

TAD Matriz Esparsa

```
1  #ifndef MATRIZ_ESPARSA_H
2  #define MATRIZ_ESPARSA_H
3
4  typedef struct matriz_esparsa MATRIZ_ESPARSA;
5
6  MATRIZ_ESPARSA *criar_matriz(int nr_linhas, int nr_colunas);
7  void apagar_matriz(MATRIZ_ESPARSA **matriz);
8
9  int set(MATRIZ_ESPARSA *matriz, int lin, int col, float val);
10 float get(MATRIZ_ESPARSA *matriz, int lin, int col);
11
12 void imprimir_matriz(MATRIZ_ESPARSA *matriz);
13
14 #endif
```

Criar Matriz

```
1  MATRIZ_ESPARSA *criar_matriz(int nr_linhas, int nr_colunas) {
2
3
4  }
```

Apagar Matriz

```
1  void apagar_matriz(MATRIZ_ESPARSA **matriz) {
2
3
4  }
```

Definir Valor

```
1  void set(MATRIZ_ESPARSA *matriz, int lin, int col, float val) {
2
3
4  }
```

Retornar Valor

```
1  float get(MATRIZ_ESPARSA *matriz, int lin, int col) {
2
3
4  }
```

- Estratégia muito simplista, pode ser melhorada. Como?

Operações

- E quando um elemento da matriz original se torna não nulo, em consequência de alguma operação? É necessário inserir na estrutura?

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Operações

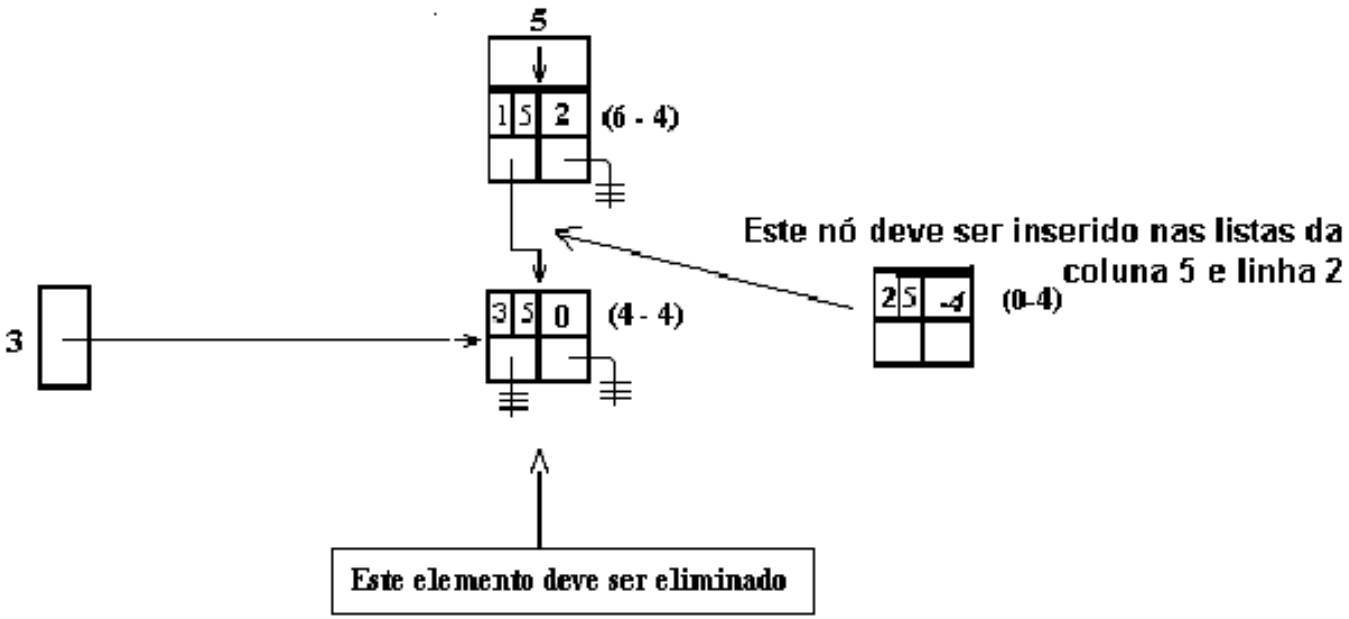
- E quando um elemento da matriz original se torna não nulo, em consequência de alguma operação? É necessário inserir na estrutura?
- E quando um elemento da matriz original se tornar nulo? É necessário eliminar da estrutura?

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Operações

- P.ex., somar -4 à coluna 5

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & -3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 \\ 5 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?

Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?
- Fator Espaço. Suponhamos

Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?
- Fator Espaço. Suponhamos
  - matriz esparsa que armazena inteiros

Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?
- Fator Espaço. Suponhamos
  - matriz esparsa que armazena inteiros
  - ponteiro ocupa o mesmo espaço de memória que um inteiro

Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?
- Fator Espaço. Suponhamos
  - matriz esparsa que armazena inteiros
  - ponteiro ocupa o mesmo espaço de memória que um inteiro
- Matriz Esparsa (Listas Cruzadas)

Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?
- Fator Espaço. Suponhamos
  - matriz esparsa que armazena inteiros
  - ponteiro ocupa o mesmo espaço de memória que um inteiro
- Matriz Esparsa (Listas Cruzadas)
  - Espaço ocupado por matriz de nl linhas, nc colunas e n valores não-nulos

Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?
- Fator Espaço. Suponhamos
  - matriz esparsa que armazena inteiros
  - ponteiro ocupa o mesmo espaço de memória que um inteiro
- Matriz Esparsa (Listas Cruzadas)
  - Espaço ocupado por matriz de nl linhas, nc colunas e n valores não-nulos
    - $5n$ espaços para ponteiros (um para cada campo do registro: $linha$, $coluna$, $valor$, $direita$, $abaixo$)

Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?
- Fator Espaço. Suponhamos
  - matriz esparsa que armazena inteiros
  - ponteiro ocupa o mesmo espaço de memória que um inteiro
- Matriz Esparsa (Listas Cruzadas)
  - Espaço ocupado por matriz de nl linhas, nc colunas e n valores não-nulos
    - $5n$ espaços para ponteiros (um para cada campo do registro: $linha$, $coluna$, $valor$, $direita$, $abaixo$)
    - $5nl$ espaços para ponteiros para o vetor $linha$

Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?
- Fator Espaço. Suponhamos
  - matriz esparsa que armazena inteiros
  - ponteiro ocupa o mesmo espaço de memória que um inteiro
- Matriz Esparsa (Listas Cruzadas)
  - Espaço ocupado por matriz de nl linhas, nc colunas e n valores não-nulos
    - $5n$ espaços para ponteiros (um para cada campo do registro: $linha$, $coluna$, $valor$, $direita$, $abaixo$)
    - $5nl$ espaços para ponteiros para o vetor $linha$
    - $5nc$ espaços para ponteiros para o vetor $coluna$

Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?
- Fator Espaço. Suponhamos
  - matriz esparsa que armazena inteiros
  - ponteiro ocupa o mesmo espaço de memória que um inteiro
- Matriz Esparsa (Listas Cruzadas)
  - Espaço ocupado por matriz de nl linhas, nc colunas e n valores não-nulos
    - $5n$ espaços para ponteiros (um para cada campo do registro: $linha$, $coluna$, $valor$, $direita$, $abaixo$)
    - $5nl$ espaços para ponteiros para o vetor $linha$
    - $5nc$ espaços para ponteiros para o vetor $coluna$
    - espaço total: $5n + 5nl + 5nc$

Desempenho

- Quando a representação por listas cruzadas é vantajosa em relação à representação convencional?
- Fator Espaço. Suponhamos
  - matriz esparsa que armazena inteiros
  - ponteiro ocupa o mesmo espaço de memória que um inteiro
- Matriz Esparsa (Listas Cruzadas)
  - Espaço ocupado por matriz de nl linhas, nc colunas e n valores não-nulos
    - $5n$ espaços para ponteiros (um para cada campo do registro: $linha$, $coluna$, $valor$, $direita$, $abaixo$)
    - $5nl$ espaços para ponteiros para o vetor $linha$
    - $5nc$ espaços para ponteiros para o vetor $coluna$
    - espaço total: $5n + 5nl + 5nc$
- Na representação bidimensional: espaço total: $nl \times nc$

Desempenho (Fator Espaço)

Conclusão

- Em termos de espaço ocupado, é vantajoso utilizar a representação de listas cruzadas quando

Desempenho (Fator Espaço)

Conclusão

- Em termos de espaço ocupado, é vantajoso utilizar a representação de listas cruzadas quando
  - $5n + 5nl + 5nc < nl \times nc$

Desempenho (Fator Espaço)

Conclusão

- Em termos de espaço ocupado, é vantajoso utilizar a representação de listas cruzadas quando
  - $5n + 5nl + 5nc < nl \times nc$
  - ou seja, quando $n < [(nl - 5) \times (nc - 5) - 25]/5$

Desempenho (Fator Espaço)

Conclusão

- Em termos de espaço ocupado, é vantajoso utilizar a representação de listas cruzadas quando
  - $5n + 5nl + 5nc < nl \times nc$
  - ou seja, quando $n < [(nl - 5) \times (nc - 5) - 25]/5$
  - Como $(nl - 5) \times (nc - 5)$ é aproximadamente o tamanho da matriz, pode-se dizer, de uma maneira geral, que há ganho de espaço, quando um número inferior a $1/5$ dos elementos da matriz forem não nulos

Desempenho (Fator Tempo)

- As operações sobre listas cruzadas podem ser mais lentas e complexas do que para o caso bidimensional

Desempenho (Fator Tempo)

- As operações sobre listas cruzadas podem ser mais lentas e complexas do que para o caso bidimensional
- Portanto, para algumas aplicações, deve ser feita uma avaliação do compromisso entre tempo de execução e espaço alocado

Sumário

1 Matrizes Esparsas – Listas Cruzadas

2 Representação Alternativa – Listas Cruzadas Circulares

Representação alternativa

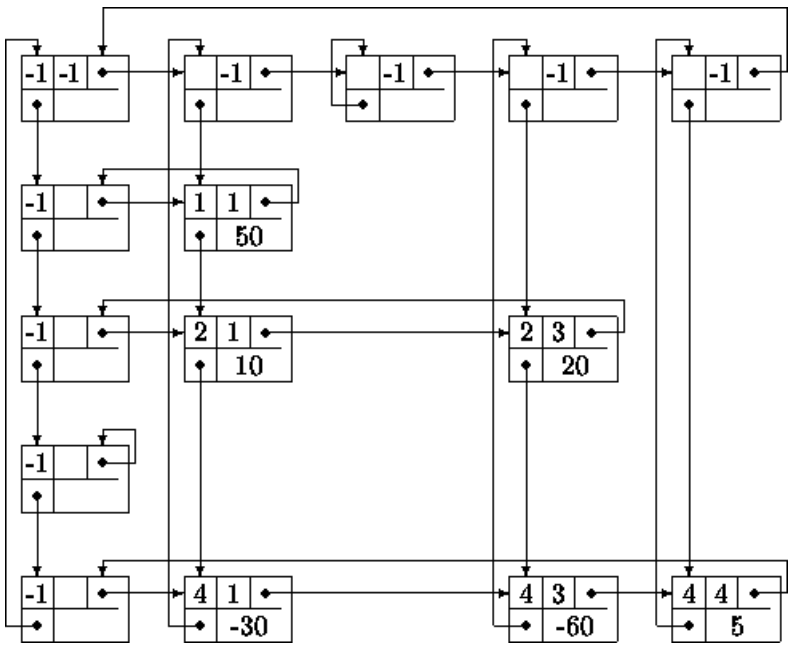
- Existem ocasiões nas quais não se sabe a princípio qual será o número máximo de linhas ou colunas da matriz esparsa

Representação alternativa

- Existem ocasiões nas quais não se sabe a princípio qual será o número máximo de linhas ou colunas da matriz esparsa
- Nessas situações, os vetores *Coluna* e *Linha* podem ser substituídos por listas ligadas circulares

Representação alternativa

$$\begin{pmatrix} 50 & 0 & 0 & 0 \\ 10 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 \\ -30 & 0 & -60 & 5 \end{pmatrix}$$



Exercícios

- Desenvolva procedimentos para

Exercícios

- Desenvolva procedimentos para
  - Definir o elemento a_{ij}

Exercícios

- Desenvolva procedimentos para
  - Definir o elemento a_{ij}
  - Retornar a_{ij} da matriz

Exercícios

- Desenvolva procedimentos para
  - Definir o elemento a_{ij}
  - Retornar a_{ij} da matriz
  - Somar a constante c todos os elementos da coluna j

Exercícios

- Desenvolva procedimentos para
  - Definir o elemento a_{ij}
  - Retornar a_{ij} da matriz
  - Somar a constante c todos os elementos da coluna j
    - Pode resultar em inserção ou eliminação nas listas.