

Aplicações de listas e outras estruturas

SCC-502 – Algoritmos e Estruturas de
Dados I

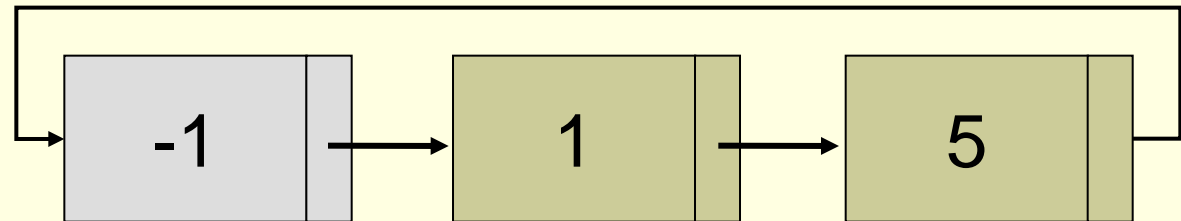
Grandes números

- **Problema:** lidar com números muito grandes
 - Em C, inteiros (mesmo long int) são limitados
 - Como somar números inteiros maiores do que o tamanho do tipo permite?
 - Listas!

Grandes números

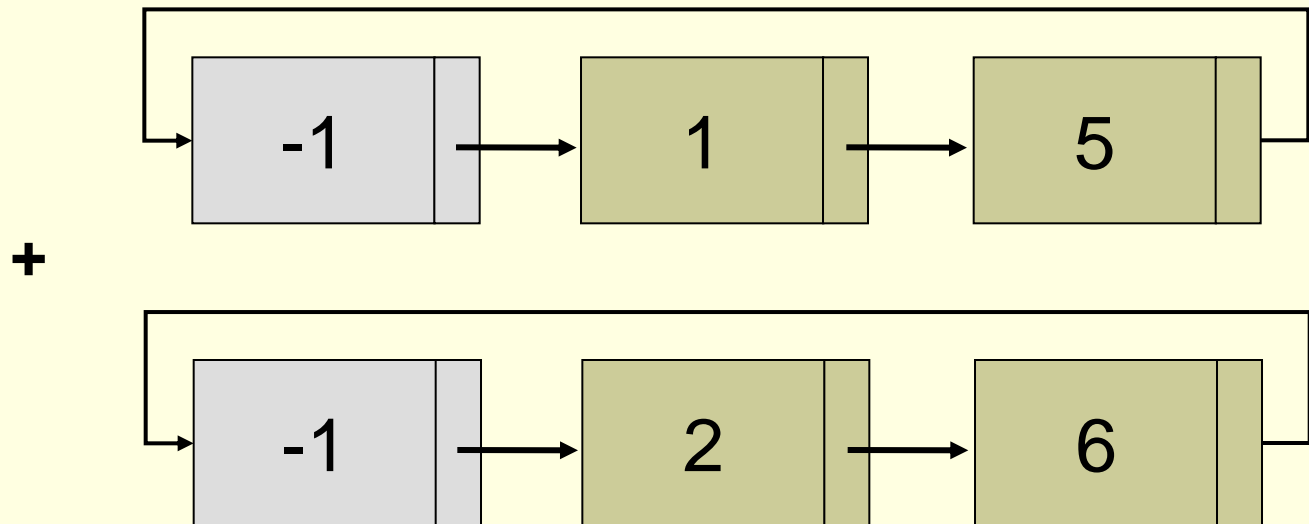
- Representando números como listas

- 15



Grandes números

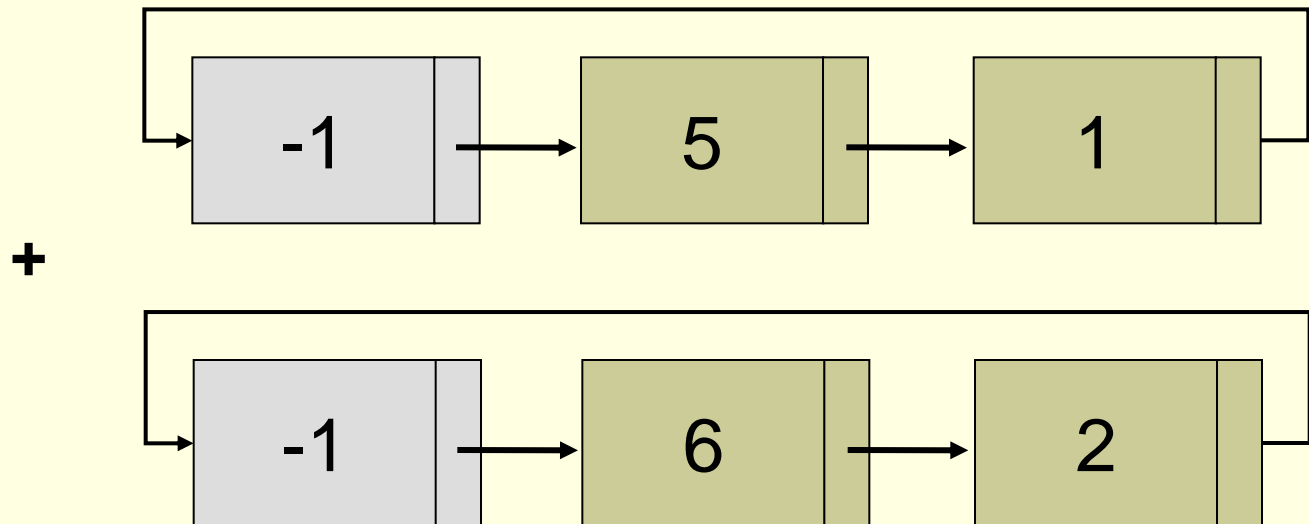
- Soma de dois números
 - Bloco somados dois a dois, da direita para a esquerda
 - Exemplo: $15+26$



Grandes números

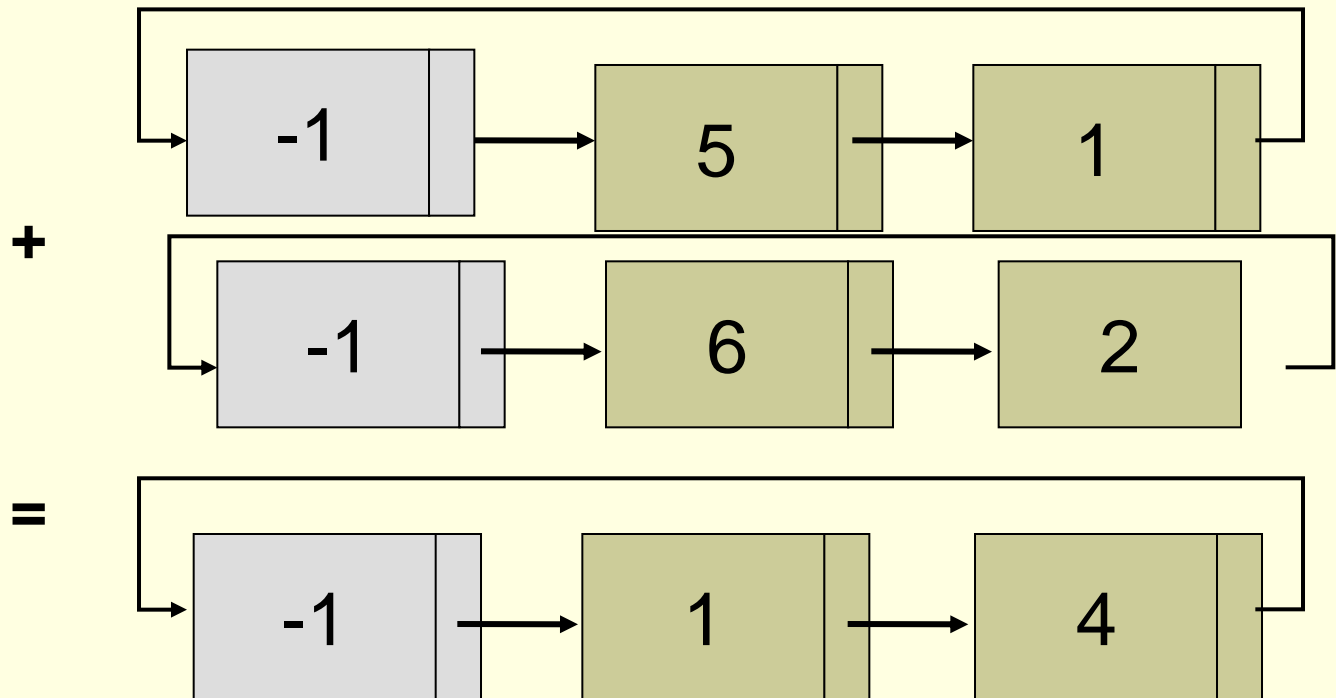
- Para facilitar nossa vida, números já são representados ao contrário

- Exemplo: $15+26$



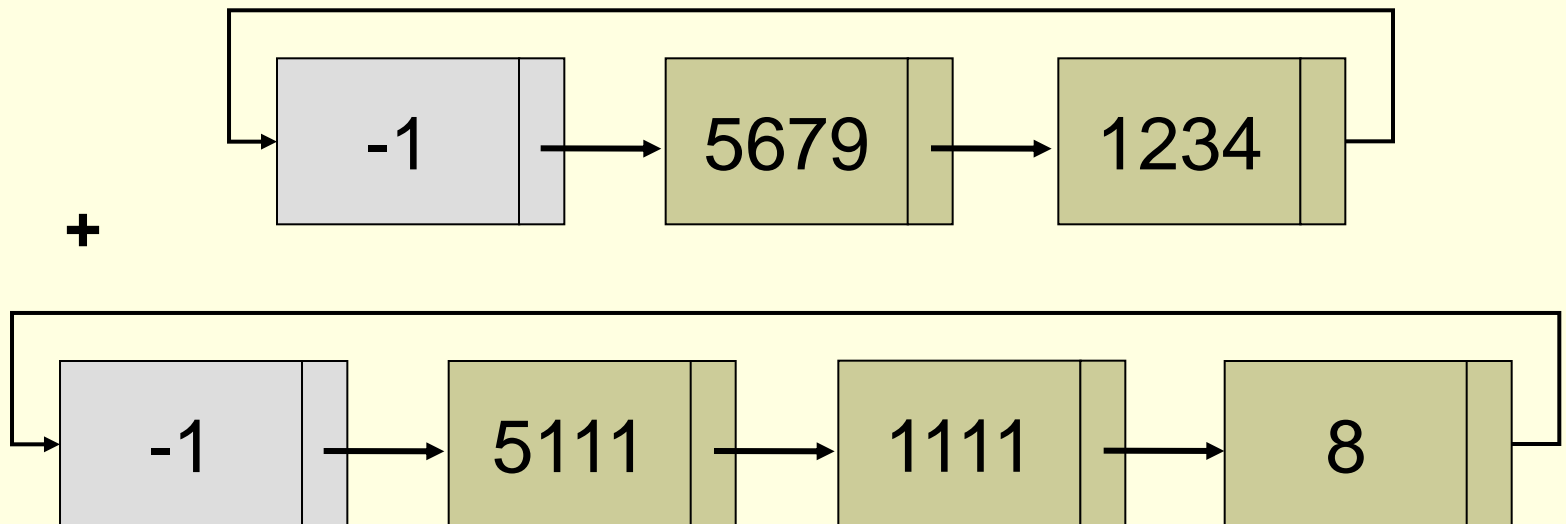
Grandes números

- Para facilitar nossa vida, números já são representados ao contrário
 - Exemplo: $15+26 = 41$



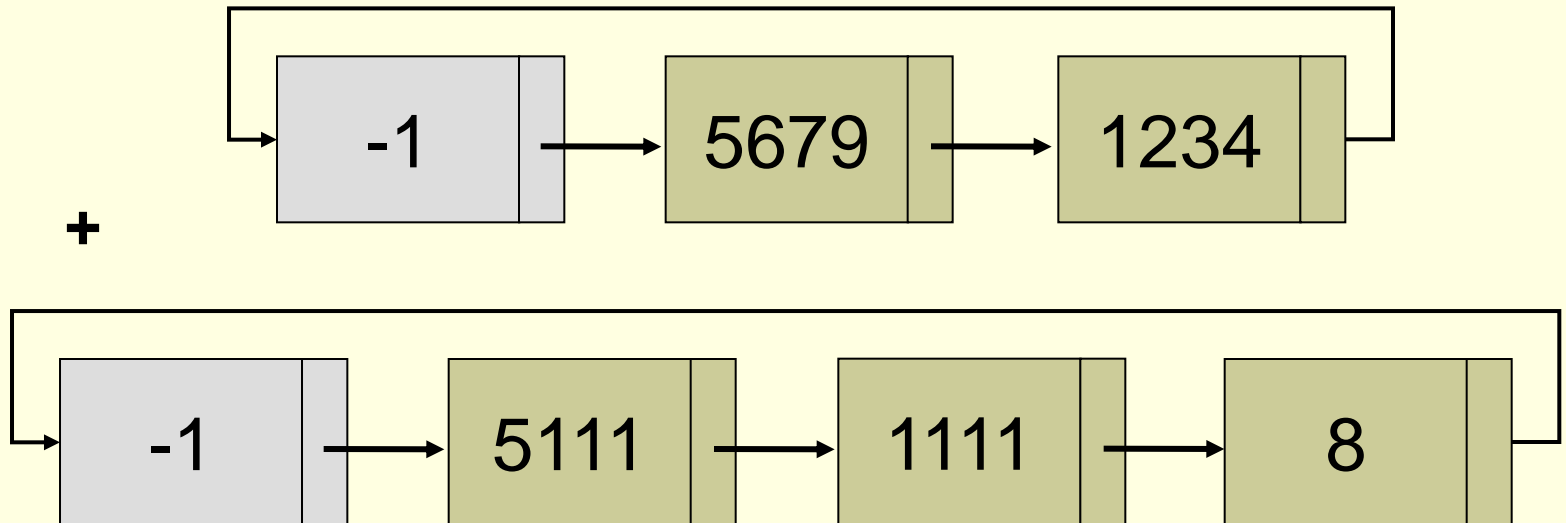
Grandes números

- Como os números tratados por esse mecanismo são muito grandes, pode-se aproveitar melhor o tipo inteiro: uso otimizado de memória
 - Exemplo: $12.345.679 + 811.115.111 = 823.460.790$
 - Produz-se uma outra lista como resultado



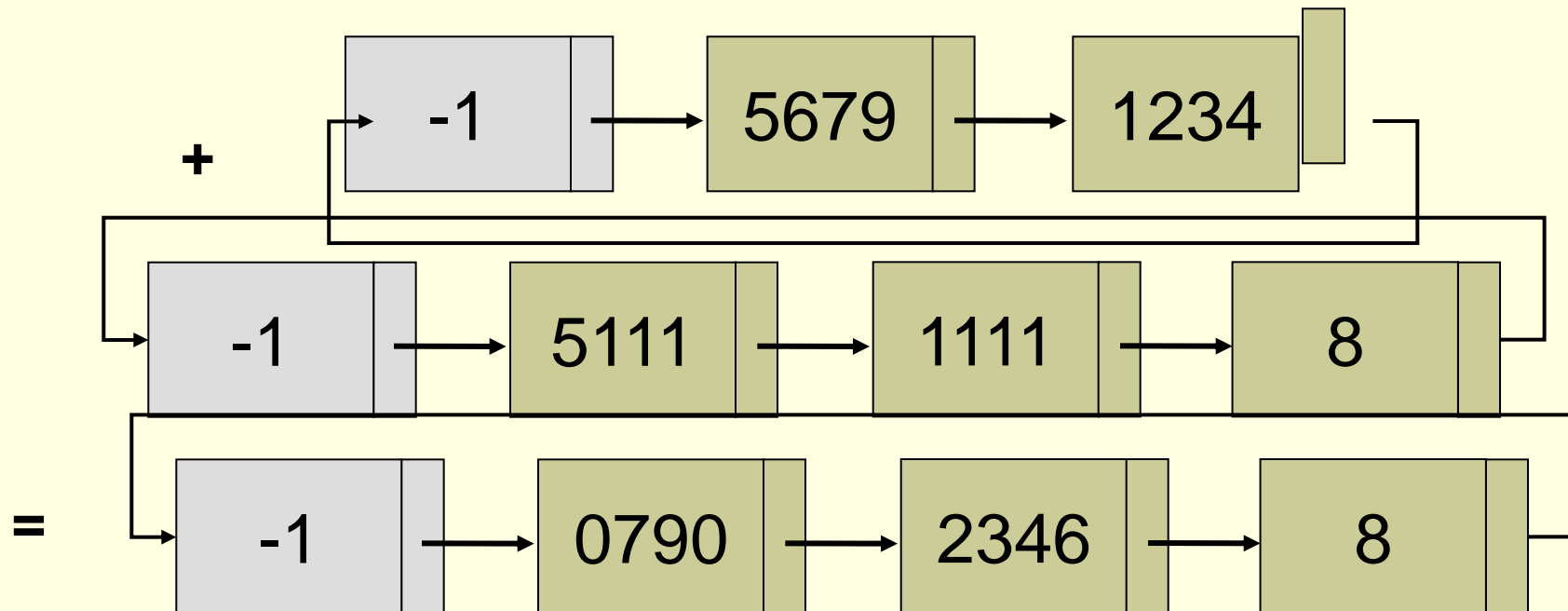
Grandes números

- Como recuperar o número somado para colocar na nova lista?
- Como recuperar o “sobe 1”?



Grandes números

- Como recuperar o número somado para colocar na nova lista?
 - Soma % 10.000 (por que 4 zeros?)
- Como recuperar o “sobe 1”?
 - Soma / 10.000



Grandes números

- Exercício para casa
 - Implemente em C uma sub-rotina para somar dois grandes números utilizando uma lista circular com nó de cabeçalho
 - As duas listas a serem somadas devem ser passadas por parâmetros, sendo que o ponteiro para a nova lista contendo a soma deve ser retornado em um outro ponteiro (por parâmetro também).

Matrizes

- Matriz é um arranjo (tabela) retangular de números dispostos em linhas e colunas

$$A \begin{matrix} 3 \times 4 \\ \left[\begin{array}{cccc} 1 & 0 & 4 & -3 \\ 2 & 5 & 3 & 4 \\ 9 & 8 & -2 & 1 \end{array} \right] \end{matrix}$$

$$B \begin{matrix} 3 \times 3 \\ \left[\begin{array}{ccc} 3 & 7 & 4 \\ 1 & 0 & 6 \\ 9 & 2 & 8 \end{array} \right] \end{matrix}$$

n° de elementos = n° de linhas * n° de colunas

Matriz = Arranjo bidimensional

Matrizes especiais

$$A \begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 4 & 5 & 6 \end{bmatrix}$$

3x3

Triangular inferior

$$B \begin{bmatrix} 1 & 2 & 0 & 0 \\ 2 & 4 & 5 & 0 \\ 0 & 6 & 7 & 8 \\ 0 & 0 & 9 & 10 \end{bmatrix}$$

4x4

Tri-diagonal

$$C \begin{bmatrix} 1 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

7x9

Matriz esparsa:
excessivo n° de
elementos nulos (0)

Matrizes esparsas

$$C_{700 \times 900} \begin{bmatrix} 1 & 0 & 0 & 3 & 0 & 0 & 0 & \dots & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & -1 & 4 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 \end{bmatrix}$$

$700 \times 900 = 630.000$ elementos

Matriz esparsa com **9** elementos **não nulos**

Matrizes esparsas

- Uso da **matriz tradicional**

- Vantagem

- Ao se representar dessa forma, preserva-se o acesso direto a cada elemento da matriz
 - Algoritmos simples

- Desvantagem

- Muito espaço para armazenar zeros

Matrizes esparsas

- Necessidade
 - **Método alternativo** para representação de matrizes esparsas
- Solução
 - Estrutura de lista encadeada contendo somente os elementos não nulos

Matrizes esparsas - solução 1

■ Listas simples encadeadas

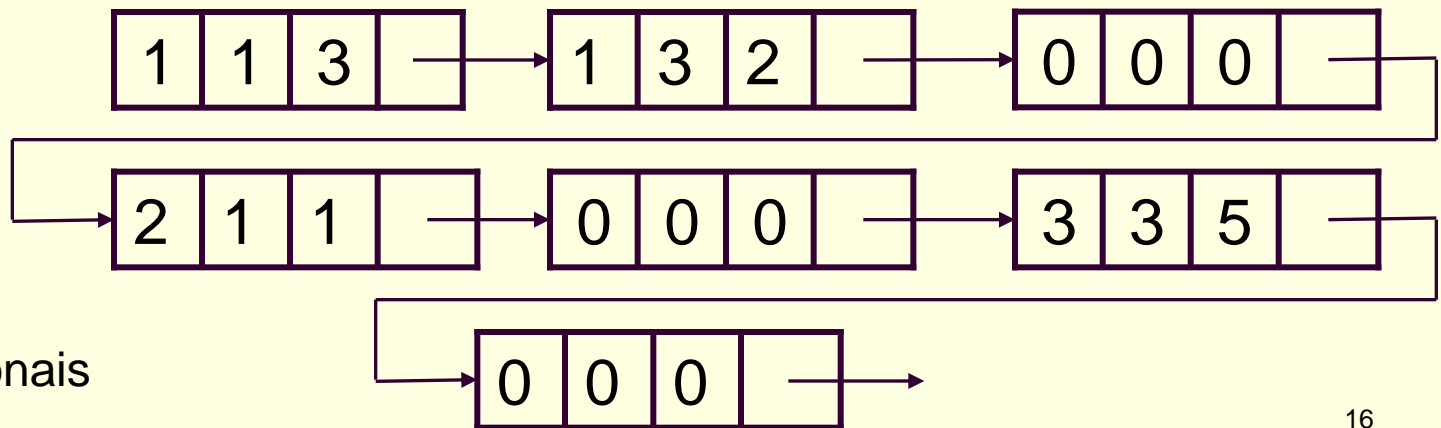
$$A \begin{bmatrix} 3 & 0 & 2 \\ 1 & 0 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

3x3

| linha | coluna | valor | prox |
|-------|--------|-------|------|
|-------|--------|-------|------|

Estrutura de um nó:

- linha, coluna: posição
- valor: \neq zero
- prox: próximo nó



Nós zerados opcionais para auxiliar na divisão de linhas

Matrizes esparsas - solução 1

■ Desvantagens

- Perda da natureza bidimensional de matriz
- Acesso ineficiente à linha
 - Para acessar o elemento na i -ésima linha, deve-se atravessar as $i-1$ linhas anteriores
- Acesso ineficiente à coluna
 - Para acessar os elementos na j -ésima coluna, tem que se passar por várias outras antes

■ Questão

- Como organizar essa lista, preservando a natureza bidimensional de matriz?

Matrizes esparsas - solução 2

■ Listas cruzadas

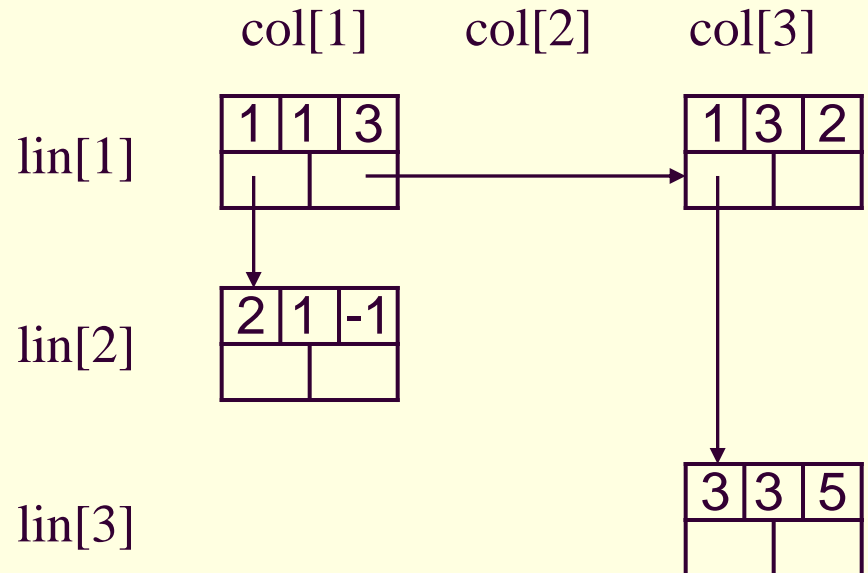
- Para cada matriz, usam-se dois vetores com N ponteiros para as linhas e M ponteiros para as colunas

$$A \begin{bmatrix} 3 & 0 & 2 \\ -1 & 0 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

3x3

Estrutura de um nó:

| | | |
|---------|--------|---------|
| linha | coluna | valor |
| proxlin | | proxcol |



Matrizes esparsas - solução 2

- Listas cruzadas
 - Cada elemento não nulo é mantido **simultaneamente em duas listas**
 - Uma para sua linha
 - Uma para sua coluna

Matrizes esparsas

- Listas cruzadas vs. matriz tradicional
 - Em termos de espaço
 - Supor que inteiro e ponteiro para inteiro ocupam um bloco de memória
 - Listas cruzadas: tamanho do vetor de linhas (nl) + tamanho do vetor de colunas (nc) + n elementos não nulos * tamanho do nó
 - $nl+nc+5n$
 - Matriz tradicional bidimensional
 - $nl*nc$

Matrizes esparsas

- Listas cruzadas vs. matriz tradicional
 - Em termos de tempo
 - Operações mais lentas em listas cruzadas: acesso não é direto

Matrizes esparsas

- Listas cruzadas vs. matriz tradicional
 - Necessidade de avaliação tempo-espaço para cada aplicação
 - Em geral, usam-se listas cruzadas quando no máximo 1/5 dos elementos forem não nulos
 - De onde vem isso?

Matrizes esparsas

- Listas cruzadas vs. matriz tradicional
 - Necessidade de avaliação tempo-espaço para cada aplicação
 - Em geral, usam-se listas cruzadas quando no máximo 1/5 dos elementos forem não nulos
 - De onde vem isso?

Dica: $nl+nc+5n < nl*nc$

Matrizes esparsas - operações

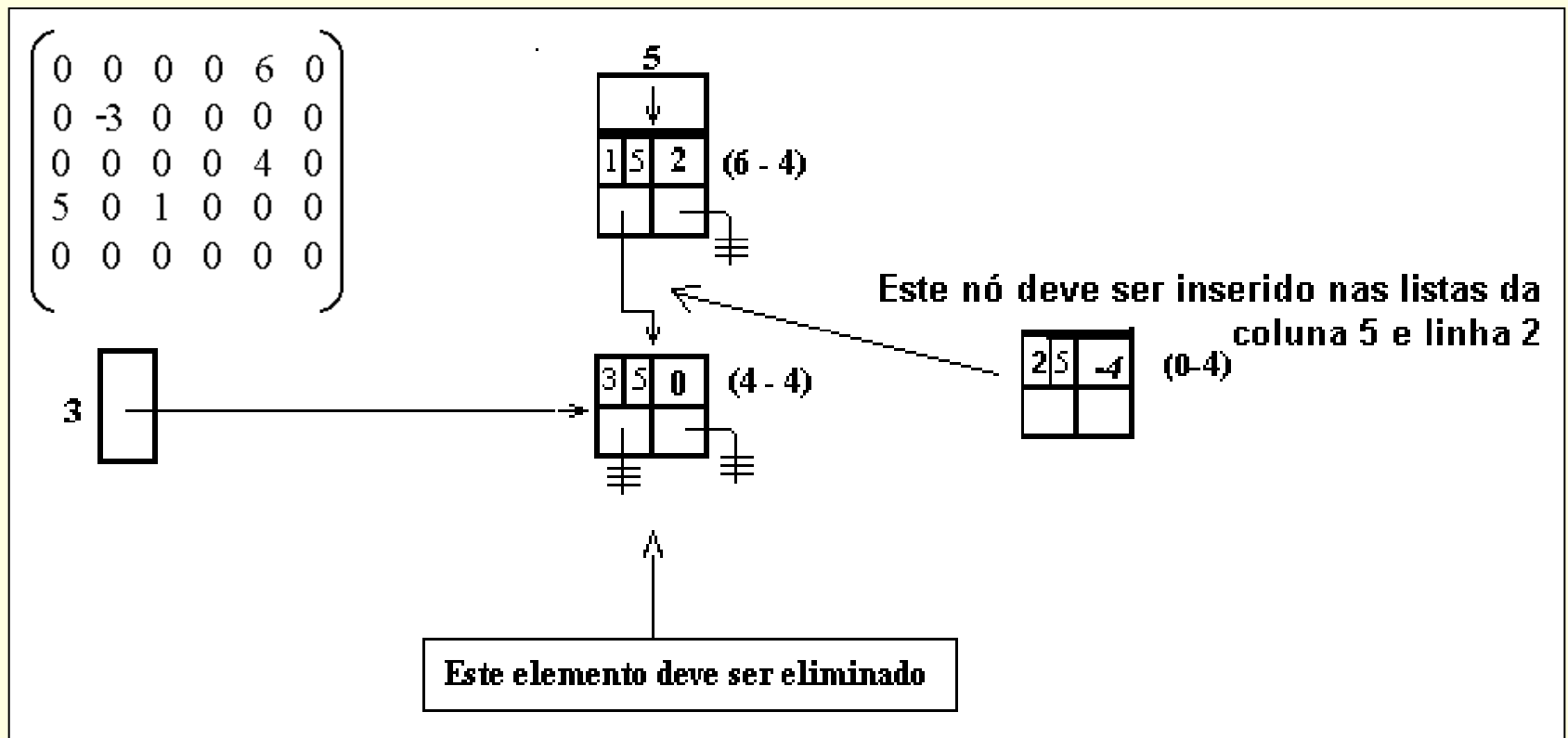
- Em geral
 - Multiplicar uma dada linha ou coluna por uma constante
 - Somar uma constante a todos os elementos de uma linha ou coluna
 - Somar duas matrizes esparsas de igual dimensão
 - Multiplicar matrizes esparsas
 - Transpor matrizes esparsas
 - Inserir, remover ou alterar elementos
 - Etc.

Matrizes esparsas - operações

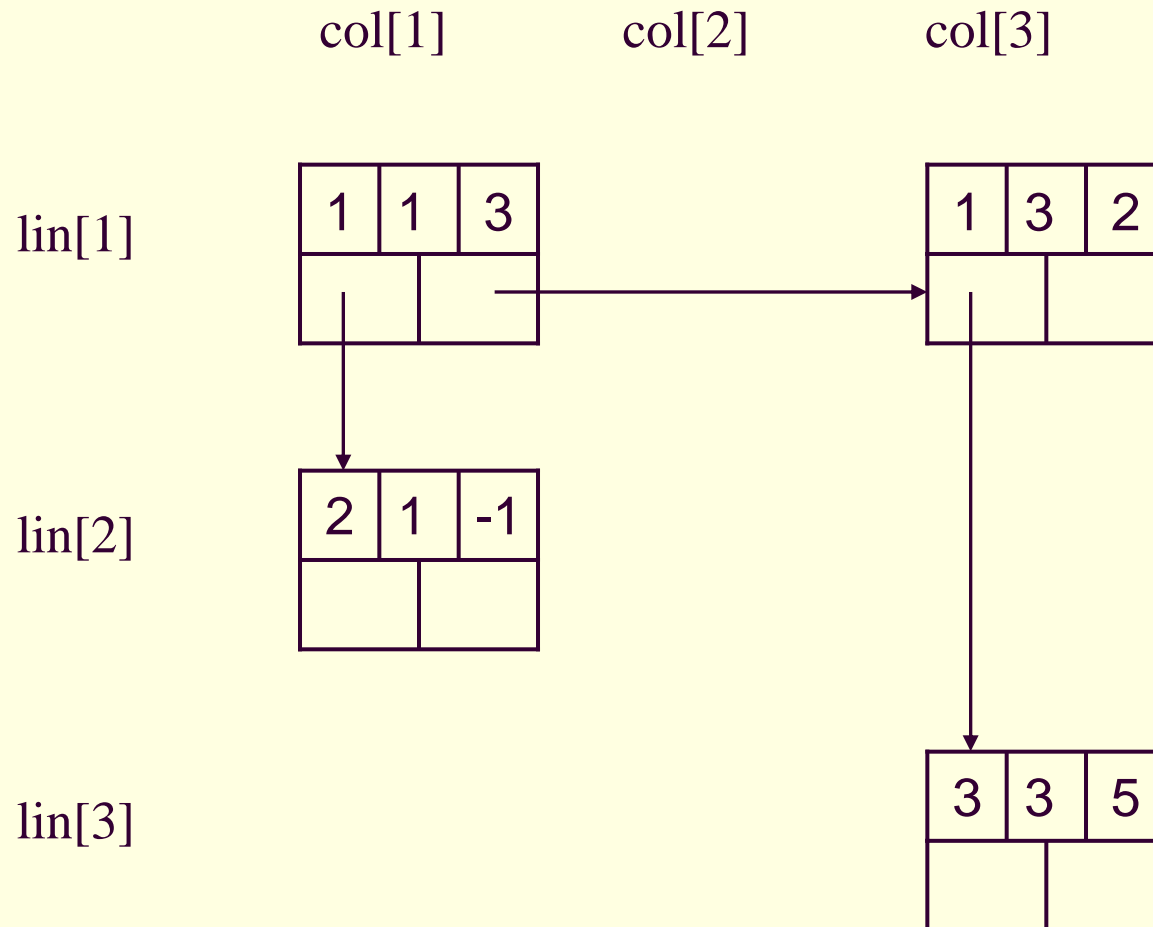
- Após a realização de alguma operação sobre a matriz
 - Quando um elemento da matriz se torna nulo
 - Remoção do elemento
 - Quando algum elemento se torna não nulo
 - Inserção do elemento

Matrizes esparsas - operações

- Por exemplo, ao se somar -4 a coluna 5 do exemplo



Exercício: somar -5 a coluna 3



Exercício

- Declare em C a estrutura da matriz esparsa representada via lista cruzada

Exercício

- Declare em C a estrutura da matriz esparsa representada via lista cruzada

```
#define n ... //número de linhas  
#define m ... //número de colunas
```

```
typedef struct no {  
    int lin, col, val;  
    struct no *proxlin, *proxcol;  
} no;
```

```
typedef struct {  
    no *L[n], *C[m];  
} MatrizEsparsa;
```

Matrizes esparsas

- Exercício para casa
 - Implementar uma sub-rotina para somar todos os elementos (inteiros) não nulos de uma matriz esparsa
 - Usando listas cruzadas

```
int soma(MatrizEsparsa *M) {
    int res=0;
    no *linha, *coluna;
    linha=M->inicio->proxlin;
    while (linha!=M->inicio) {
        coluna=linha->proxcol;
        while (coluna!=linha) {
            res+=coluna->val;
            coluna=coluna->proxcol;
        }
        linha=linha->proxlin;
    }
    return(res);
}
```

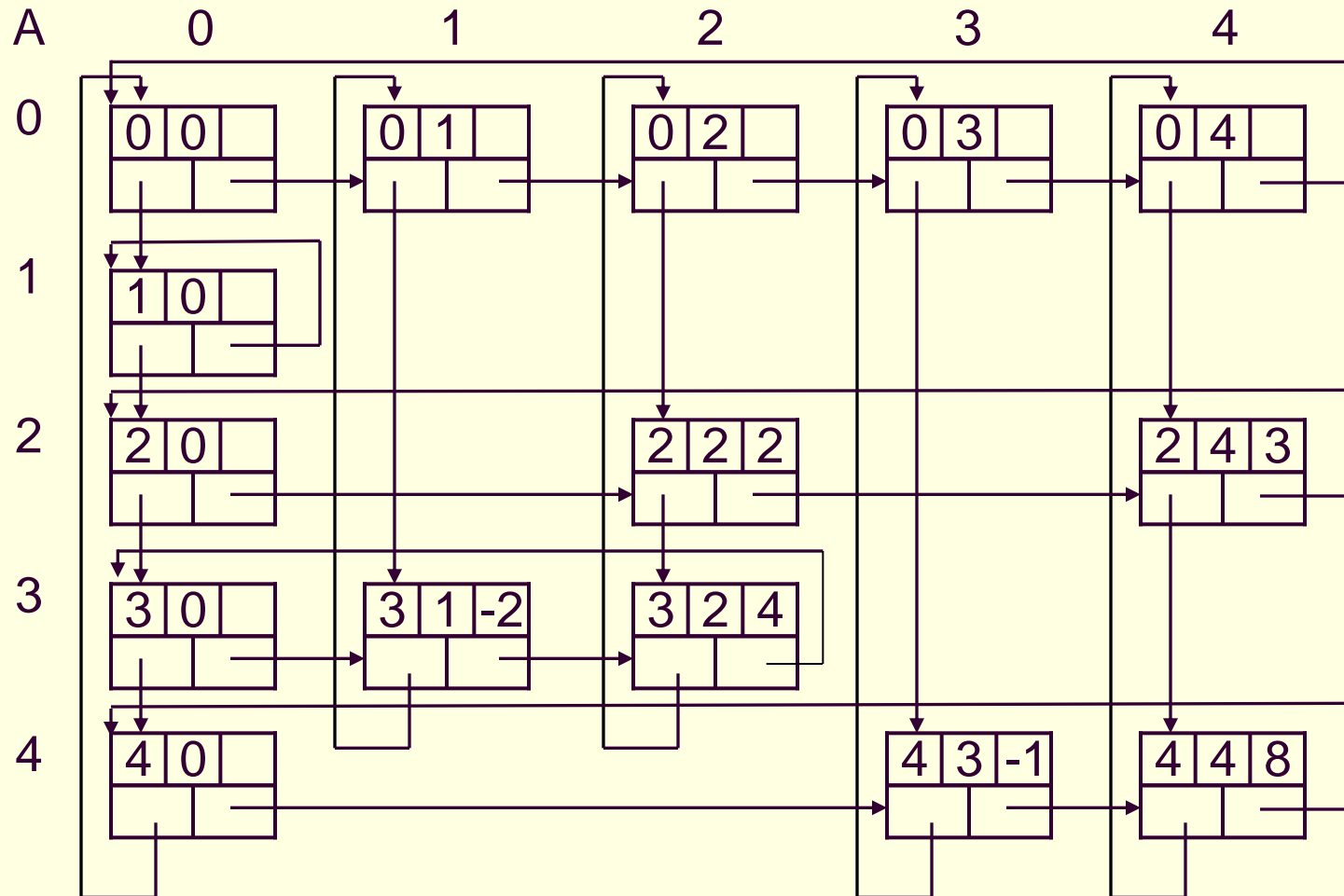
Matrizes esparsas - outra solução

- Listas circulares com nós de cabeçalho
 - Ao invés de vetores de ponteiros, linhas e colunas são listas circulares com nós de cabeçalho
 - Nós de cabeçalho: reconhecidos por um 0 no campo linha ou coluna
 - 1 único ponteiro para a matriz: navegação em qualquer sentido

■ Exemplo

$$A \begin{matrix} 4 \times 4 \\ \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 3 \\ -2 & 4 & 0 & 0 \\ 0 & 0 & -1 & 8 \end{bmatrix} \end{matrix}$$

Matrizes esparsas - outra solução



Matrizes esparsas - outra solução

■ Exercício

1. Representar a matriz abaixo com listas circulares com nós de cabeçalho

$$A \begin{matrix} 3 \times 3 \\ \left[\begin{array}{ccc} 3 & 0 & 2 \\ -1 & 0 & 0 \\ 0 & 0 & 5 \end{array} \right] \end{matrix}$$

2. Implementar em C uma sub-rotina que some todos os elementos de uma matriz qualquer representada dessa forma

Matrizes esparsas - outra solução

- Representação da matriz?

$$A \begin{matrix} 3 \times 3 \\ \left[\begin{array}{ccc} 3 & 0 & 2 \\ -1 & 0 & 0 \\ 0 & 0 & 5 \end{array} \right] \end{matrix}$$

Matrizes esparsas - outra solução

- Implementação da sub-rotina com análise de complexidade

Matrizes esparsas - outra solução

- Quais as desvantagens dessa representação?
- Quais as vantagens dessa representação?

Matrizes esparsas - outra solução

- Quais as desvantagens dessa representação?
 - Mais complexa de se manipular
- Quais as vantagens dessa representação?
 - A matriz pode crescer dinamicamente

Matrizes esparsas

- **Desafio para casa!**
 - Implemente o TAD matriz esparsa com listas circulares e nós de cabeçalho

Deque

- **deque** = *double-ended queue*
 - Fila de extremidade dupla
- Estrutura de dados relativamente comum, mas pouco conhecida por esse nome
- Enquanto pilha e fila exigem inserções e remoções em extremidades específicas da estrutura, deque permite inserções e remoções em quaisquer extremidades
 - Se insere em uma extremidade e remove dela: pilha
 - Se insere em uma extremidade e remove da outra: fila
 - Deque permite inserções e remoções dos dois tipos, “misturadas”

Deque

- Implementação do TAD deque
 - Estrutura de dados tradicional
 - Funções diferenciadas para inserção e remoção em ambas as extremidades