



Representação de Arranjos

Introdução

- ❑ Lembre-se que a memória é unidimensional e pode ser considerada como palavras numeradas de 1 até m (ou, alternativamente, de 0 a $m-1$)
- ❑ Assim, o objetivo é a representação de arranjos n dimensionais em uma memória unidimensional
 - $n=1$: vetor (arranjo unidimensional)
 - $n=2$: matriz (arranjo bidimensional)
 - $n>2$: arranjo (ou matriz) multidimensional

Introdução

- ❑ Dentre as várias representações vamos utilizar uma na qual a localização na memória de um elemento do arranjo arbitrário possa ser definida de modo eficiente, por exemplo $A[i_1, i_2, \dots, i_n]$
- ❑ Isso é necessário, pois, em geral, os programas que utilizam arranjos podem fazer uso dos elementos dele em qualquer ordem, inclusive em ordem aleatória
- ❑ Além disso, para poder recuperar facilmente os elementos de um arranjo será também necessário ter condições para determinar a quantidade de espaço da memória a ser reservada para um arranjo em particular
- ❑ Assumindo que cada elemento de um arranjo necessita de apenas uma palavra da memória, a quantidade de palavras necessária corresponderá à quantidade de elementos dentro do arranjo

Número de Elementos

- Seja um arranjo n -dimensional declarado como $\mathbf{A}[l_1:u_1, l_2:u_2, \dots, l_n:u_n]$ (ou, equivalentemente $\mathbf{A}[l_1..u_1, l_2..u_2, \dots, l_n..u_n]$), onde $l_i:u_i$ ($l_i..u_i$) representam os limites inferior e superior de variação dos índices, respectivamente ($1 \leq i \leq n$)

- A quantidade dos elementos do arranjo \mathbf{A} é:

$$\prod_{i=1}^n (u_i - l_i + 1)$$

- Por exemplo, dado o arranjo declarado como

- $\mathbf{A}[4:5, 2:4, 1:2, 3:4]$
- temos um total de $(5-4+1)*(4-2+1)*(2-1+1)*(4-3+1) = 2*3*2*2 = 24$ elementos

Ordem de Armazenamento

- ❑ Os arranjos podem ser armazenados por **ordem de linhas** ou por **ordem de colunas**
 - No armazenamento por linhas, todos os elementos da primeira linha são armazenados; a seguir todos os elementos da segunda linha e assim sucessivamente
 - No armazenamento por colunas, todos os elementos da primeira coluna são armazenados; a seguir todos os elementos da segunda coluna e assim por diante

Exemplo

- ❑ No caso do arranjo bidimensional $B[1:2, 1:4]$

	1	2	3	4
1	10	20	30	40
2	50	60	70	80

- ❑ O armazenamento por linhas resultaria na seguinte disposição na memória:

$B[1,1]$	$B[1,2]$	$B[1,3]$	$B[1,4]$	$B[2,1]$	$B[2,2]$	$B[2,3]$	$B[2,4]$
10	20	30	40	50	60	70	80

- ❑ Esse mesmo arranjo se armazenado por colunas ficaria:

$B[1,1]$	$B[2,1]$	$B[1,2]$	$B[2,2]$	$B[1,3]$	$B[2,3]$	$B[1,4]$	$B[2,4]$
10	50	20	60	30	70	40	80

Armazenamento por Linhas

- ❑ O arranjo $A[4:5, 2:4, 1:2, 3:4]$ armazenado por linhas tem seus elementos na seguinte ordem
 - $A[4,2,1,3]$, $A[4,2,1,4]$, $A[4,2,2,3]$, $A[4,2,2,4]$ seguidos de
 - $A[4,3,1,3]$, $A[4,3,1,4]$, $A[4,3,2,3]$, $A[4,3,2,4]$ seguidos de
 - $A[4,4,1,3]$, $A[4,4,1,4]$, $A[4,4,2,3]$, $A[4,4,2,4]$ seguidos de
 - $A[5,2,1,3]$, $A[5,2,1,4]$, $A[5,2,2,3]$, $A[5,2,2,4]$ seguidos de
 - $A[5,3,1,3]$, $A[5,3,1,4]$, $A[5,3,2,3]$, $A[5,3,2,4]$ seguidos de
 - $A[5,4,1,3]$, $A[5,4,1,4]$, $A[5,4,2,3]$, $A[5,4,2,4]$
- ❑ Nota-se que o índice da direita move-se mais rápido
- ❑ De fato, se considerarmos os subscritos como números, observamos que eles aumentam: 4213, 4214, ..., 5423, 5424
- ❑ Assim, a ordem de armazenamento por linhas é também denominada **ordem lexicográfica**

Armazenamento por Linhas

- ❑ Do ponto de vista do compilador, o problema é como traduzir do nome $A[i_1, i_2, \dots, i_n]$ para a localização correta na memória
- ❑ Supondo que $A[4,2,1,3]$ está armazenado na posição 100, então $A[4,2,1,4]$ está na posição 101 e $A[5,4,2,4]$ na posição 123
- ❑ De maneira geral podemos deduzir uma fórmula para o endereço de qualquer elemento usando apenas o endereço inicial do arranjo, mais as dimensões declaradas
- ❑ Sem perda de generalidade, vamos assumir que os limites inferiores são 1 em cada dimensão i_j
- ❑ Antes de encontrar a fórmula para o caso de um arranjo n -dimensional, veremos a representação de arranjos armazenados por linhas para 1, 2 e 3 dimensões

Arranjo Unidimensional

- Se A está declarado como $A[1:u_1]$ (totalizando u_1 elementos) e assumindo uma palavra de memória por elemento, o arranjo pode ser representado na memória seqüencial da forma:

$A[1]$	$A[2]$	$A[3]$...	$A[i]$...	$A[u_1]$
α	$\alpha+1$	$\alpha+2$...	$\alpha+i-1$...	$\alpha+u_1-1$

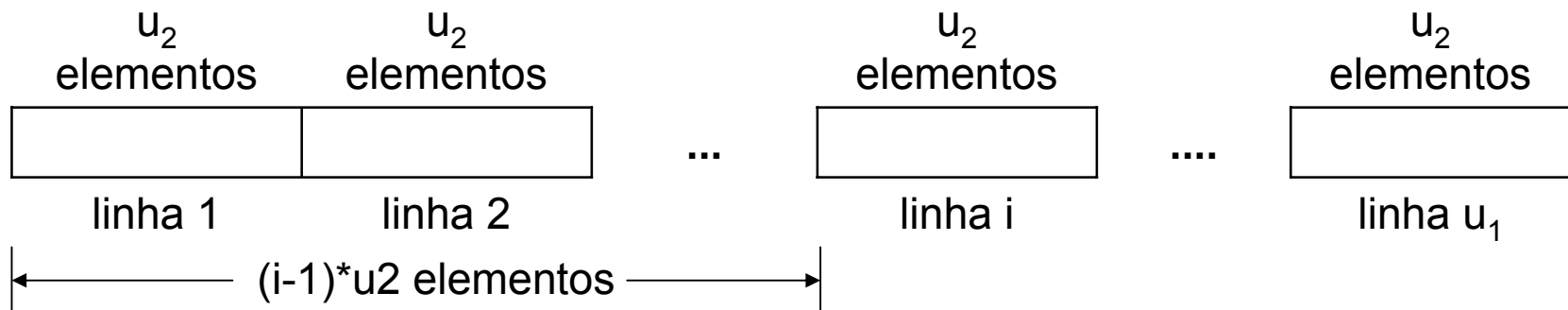
- Se α é o endereço de $A[1]$ então o endereço de um elemento arbitrário $A[i]$ é $\alpha+(i-1)$

Arranjo Bidimensional

- ❑ O arranjo $A[1:u_1, 1:u_2]$ pode ser interpretado como u_1 linhas: linha 1, linha 2, ..., linha u_1 sendo cada linha composta de u_2 elementos
- ❑ Essas linhas são representadas na memória como:

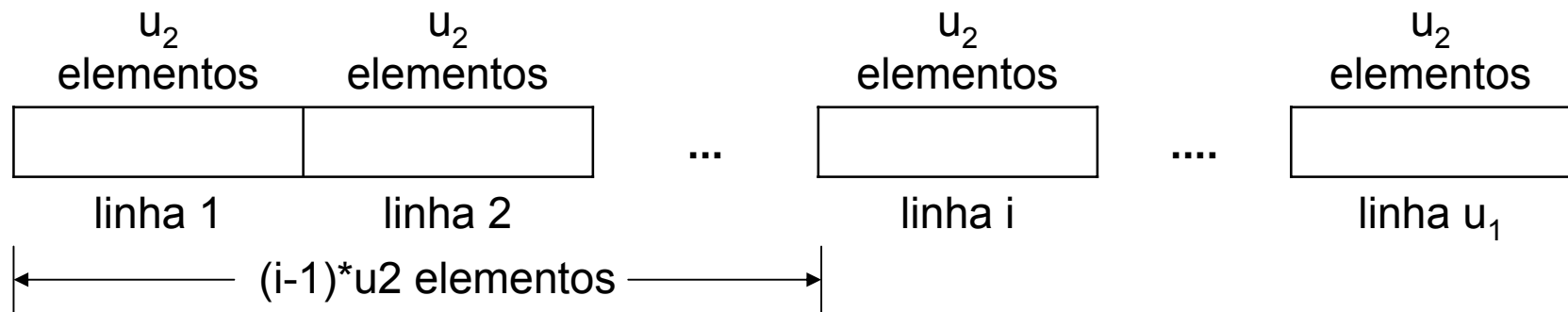
	coluna 1	coluna 2	...	coluna u_2
linha 1				
linha 2				
...				
linha u_1				

- ❑ ou seja:



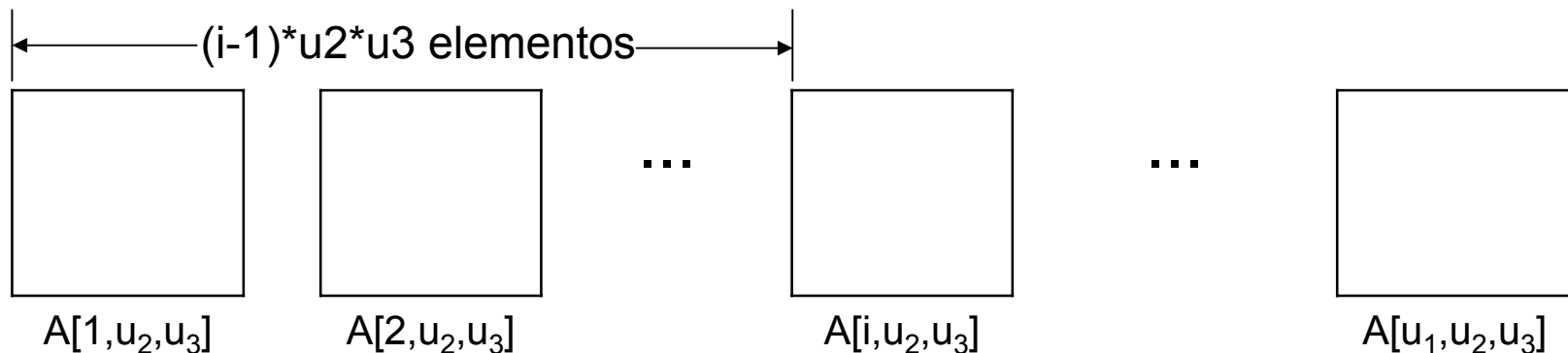
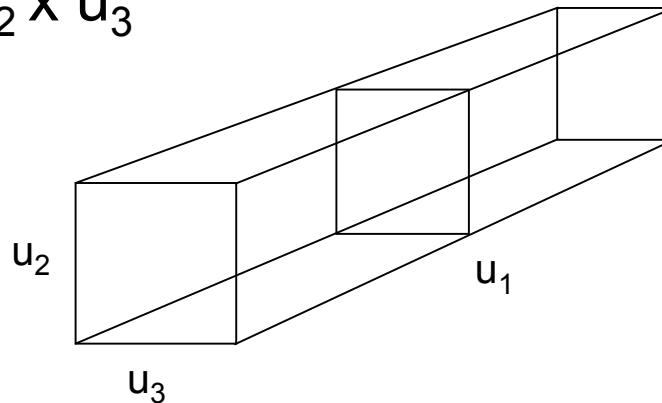
Arranjo Bidimensional

- ❑ Se α é o endereço de $A[1,1]$ então o endereço de $A[1,j]$ é $\alpha+(j-1)$
- ❑ Como existem $(i-1)$ linhas, todas de tamanho u_2 , precedendo o primeiro elemento da i -ésima linha o endereço de $A[i,j]$ é $\alpha+(i-1)*u_2+(j-1)$



Arranjo Tridimensional

- Arranjos tridimensionais $A[1:u_1, 1:u_2, 1:u_3]$ podem ser interpretados como u_1 arranjos bidimensionais com dimensões $u_2 \times u_3$



Arranjo Tridimensional

- ❑ Para localizar $A[i,j,k]$ obtemos primeiro $\alpha + (i-1) \cdot u_2 \cdot u_3$ como o endereço para $A[i,1,1]$, desde que haja $(i-1)$ arranjos bidimensionais de tamanho $u_2 \times u_3$ precedendo este elemento
- ❑ Partindo disso e da fórmula para endereçamento de um arranjo bidimensional, temos que como endereço de $A[i,j,k]$ é $\alpha + (i-1) \cdot u_2 \cdot u_3 + (j-1) \cdot u_3 + (k-1)$

Arranjos Multidimensionais

- ❑ A fórmula de endereçamento para qualquer elemento $A[i_1, i_2, \dots, i_n]$ em um arranjo n -dimensional, declarado como $A[1:u_1, 1:u_2, \dots, 1:u_n]$ pode ser conseguida facilmente
- ❑ Se α é o endereço para $A[1, 1, \dots, 1]$ então $\alpha + (i_1 - 1) * u_2 * u_3 * \dots * u_n$ é o endereço para $A[i_1, 1, \dots, 1]$
- ❑ O endereço para $A[i_1, i_2, 1, \dots, 1]$ será então $\alpha + (i_1 - 1) * u_2 * u_3 * \dots * u_n + (i_2 - 1) * u_3 * u_4 * \dots * u_n$

Arranjos Multidimensionais

□ Repetindo dessa maneira, o endereço para $A[i_1, i_2, \dots, i_n]$ é

- $\alpha + (i_1 - 1) * u_2 * u_3 * \dots * u_n$
- $+ (i_2 - 1) * u_3 * u_4 * \dots * u_n$
- $+ (i_3 - 1) * u_4 * u_5 * \dots * u_n$
- ...
- $+ (i_{n-1} - 1) * u_n$
- $+ (i_n - 1)$

□ Ou seja:

$$\alpha + \sum_{j=1}^n (i_j - 1) \times p_j$$

□ onde:

$$\begin{cases} p_j = \prod_{k=j+1}^n u_k & 1 \leq j < n \\ p_n = 1 \end{cases}$$

Matrizes Esparsas

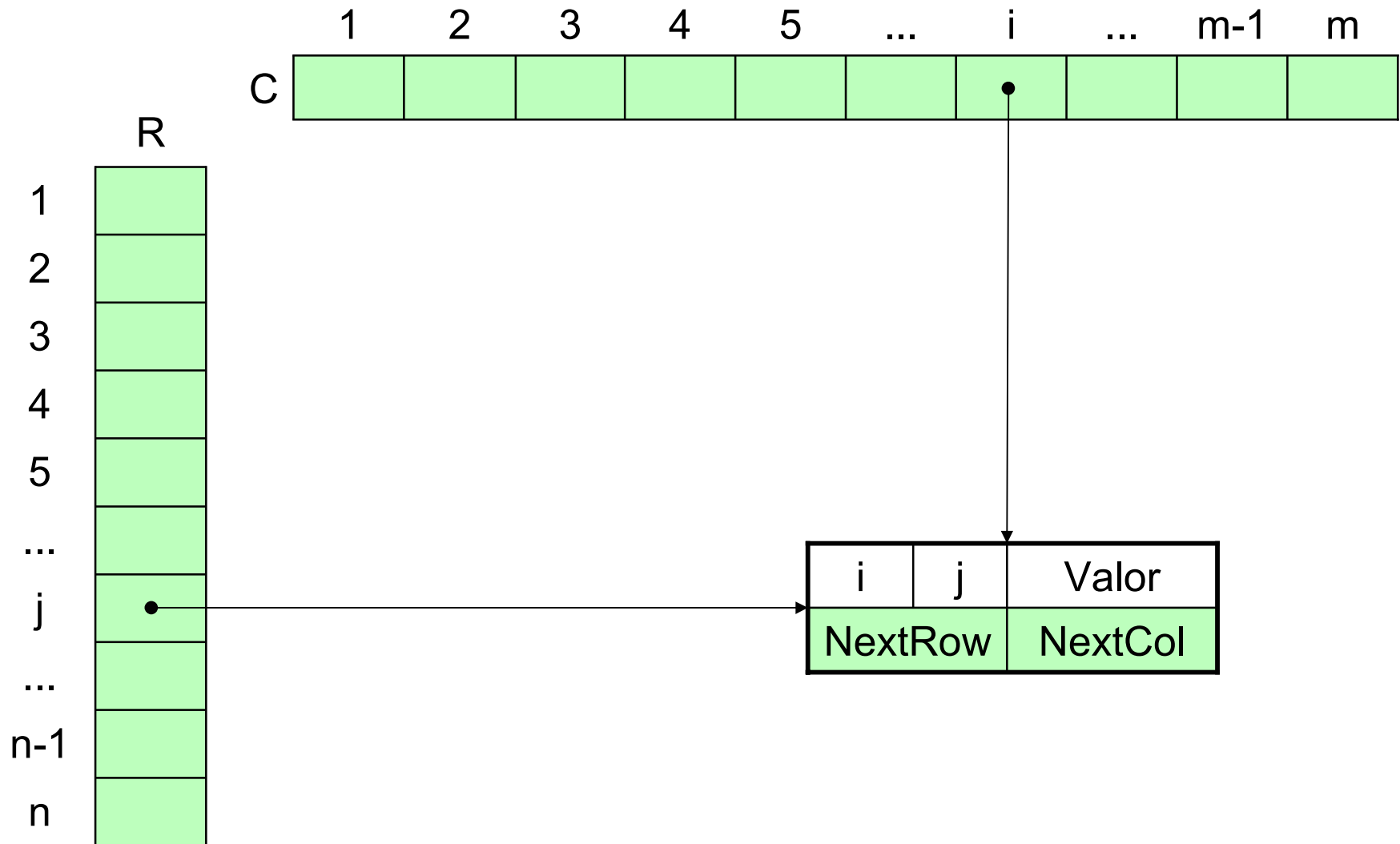
- ❑ Embora não exista uma definição precisa, uma matriz é **esparsa** quando ela tem muitas entradas iguais a zero (elementos nulos)
- ❑ Por exemplo, a matriz de ordem 5 ao lado possui um total de 25 elementos sendo que somente 7 (28%) são diferentes de zero

$$\begin{bmatrix} 15 & 0 & 0 & 22 & 0 \\ 0 & 11 & 3 & 0 & 0 \\ 0 & 0 & 0 & -6 & 0 \\ 91 & 0 & 0 & 0 & 0 \\ 0 & 0 & 28 & 0 & 0 \end{bmatrix}$$

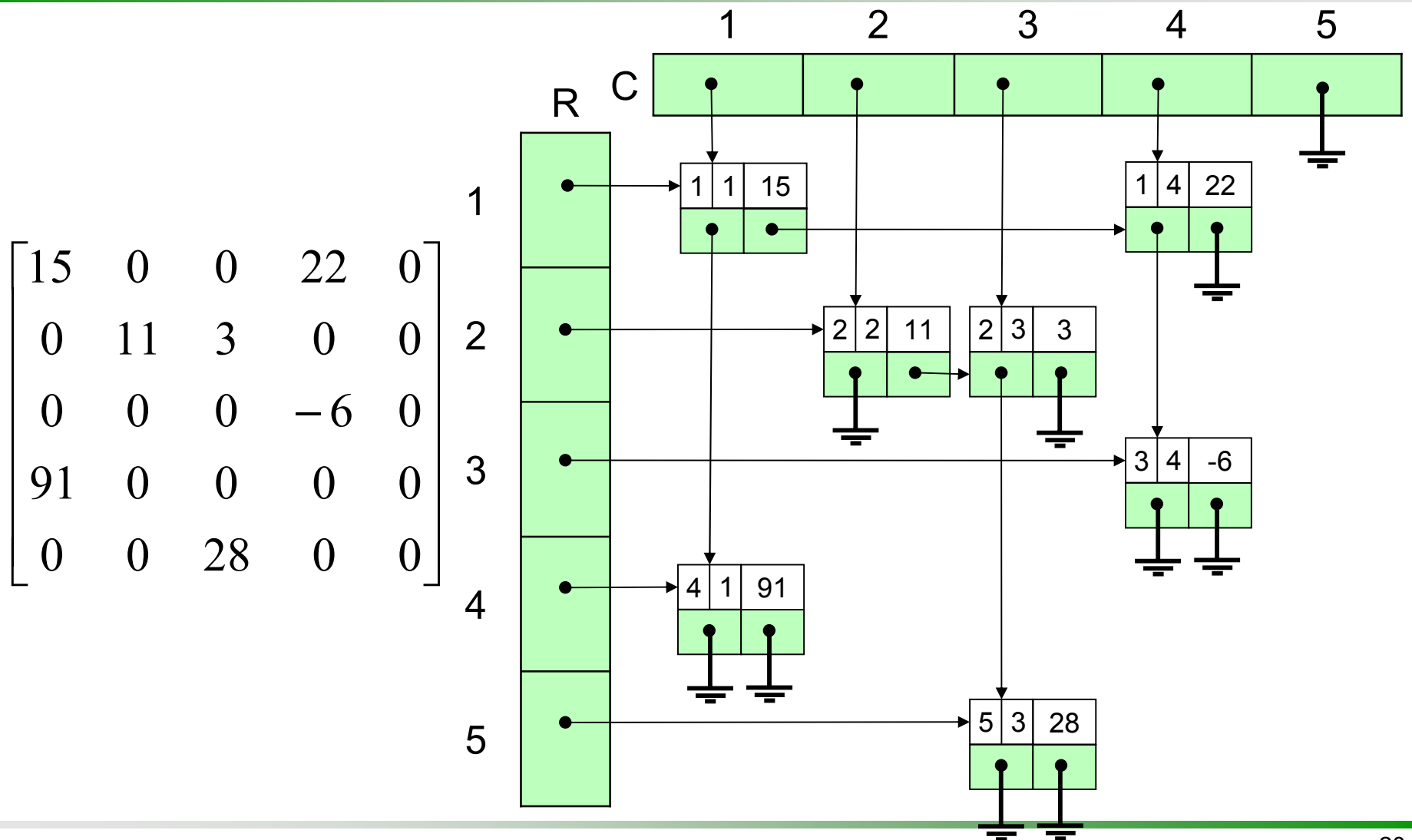
Matrizes Esparsas

- ❑ Assim, o problema consiste em representar matrizes esparsas de forma a economizar memória, definindo o ADT Matriz Esparsa
- ❑ Seja A uma matriz com n linhas e m colunas, ou seja, $A[1:n, 1:m]$ contendo k valores distintos de zero
- ❑ Uma forma de representação conhecida como **Listas Cruzadas** utiliza:
 - Um vetor de colunas $C[1:m]$
 - Um vetor de linhas $R[1:n]$
 - O elemento $A[i,j]$ é representado por uma estrutura contendo:
 - ❖ i, j , Valor de $A[i,j]$
 - ❖ NextCol: ponteiro para a coluna do próximo elemento não nulo na linha i
 - ❖ NextRow: ponteiro para a linha do próximo elemento não nulo na coluna j
 - Ambos vetores C e R são do tipo ponteiro para a estrutura descrita acima

Listas Cruzadas



Listas Cruzadas



Listas Cruzadas

- ❑ Uma matriz esparsa armazenada como uma matriz bidimensional ocupa $n*m$ palavras de memória
- ❑ Usando listas cruzadas, assumindo que a matriz esparsa armazena somente inteiros e que o espaço ocupado por ponteiros seja igual ao espaço ocupado por um inteiro (uma palavra), temos:
 - $5*k$ (linha, coluna, valor, NextRow, NextCol)
 - n palavras para vetor R
 - m palavras para vetor C
 - Espaço total: $5*k + n + m$

Listas Cruzadas

- ❑ Assim, em termos de espaço, há vantagem em armazenar uma matriz esparsa usando listas cruzadas se:
 - $5*k + n + m < n*m$
- ❑ Ou seja, quando
 - $k < ((n - 1) * (m - 1) - 1) / 5$
- ❑ Como $(n-1)*(m-1)$ é aproximadamente o tamanho da matriz original, em geral, há ganho em termos de espaço usando listas cruzadas quando um número inferior a 1/5 dos elementos da matriz forem não nulos
- ❑ Todavia, as operações sobre listas cruzadas podem ser mais lentas e complexas do que no caso bidimensional

Resumo

- ❑ Nesta apresentação foi fornecida a idéia básica de como os arranjos multidimensionais são representados na memória do computador que é unidimensional (linear)
- ❑ Adicionalmente, foi visto o conceito de matriz esparsa bem com uma forma de representação utilizando listas encadeadas