Arreglos

Polinomio de direccionamiento

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

19 de marzo de 2025

Matrices densas

Temas

1. Matrices densas
   - Arreglos bidimencionales
   - Arreglos n-dimencionales

Polinomio de direccionamiento

Renglón mayor (C, Java, Python)

- Conceptualmente un arreglo de dos dimensiones es una matriz

	0	1	2
0	data[0][0]	data[0][1]	data[0][2]
1	data[1][0]	data[1][1]	data[1][2]
2	data[2][0]	data[2][1]	data[2][2]

- Se puede implementar guardando la matriz en un arreglo de una sola dimensión. Los datos de cada renglón van juntos y se concatenan los renglones uno tras otro.

0	1	2	3	4	5	6	7	8
data[0][0]	data[0][1]	data[0][2]	data[1][0]	data[1][1]	data[1][2]	data[2][0]	data[2][1]	data[2][2]

Polinomio

Renglón mayor (C, Java, Python)

0	1	2	3	4	5	6	7	8
data[0][0]	data[0][1]	data[0][2]	data[1][0]	data[1][1]	data[1][2]	data[2][0]	data[2][1]	data[2][2]

- El tamaño D del arreglo está dado por:

$$D = m \times n \tag{1}$$

  donde m es el número de **renglones** y n, el de **columnas**.
- Para que el manejo de los datos en el arreglo sea invisible para el usuario de la matriz, se utiliza un **polinomio de direccionamiento** que calcula el índice del arreglo unidimensional, dados el *renglón* y la *columna* de la matriz.

$$\text{índice}(ren, col) = ren \times n + col \tag{2}$$

Polinomio de direccionamiento

Columna mayor (Fortran, Matlab, Julia)

Alternativamente se pueden guardar juntos los datos de las columnas

- Arreglo 2D = Matriz

	0	1	2
0	data[0][0]	data[0][1]	data[0][2]
1	data[1][0]	data[1][1]	data[1][2]
2	data[2][0]	data[2][1]	data[2][2]

- El arreglo de una dimensión se ve como sigue:

0	1	2	3	4	5	6	7	8
data[0][0]	data[1][0]	data[2][0]	data[0][1]	data[1][1]	data[2][1]	data[0][2]	data[1][2]	data[2][2]

Polinomio

Columna mayor (Fortran, Matlab, Julia)

0	1	2	3	4	5	6	7	8
data[0][0]	data[1][0]	data[2][0]	data[0][1]	data[1][1]	data[2][1]	data[0][2]	data[1][2]	data[2][2]

- En este caso el polinomio cambia de acuerdo al orden en que se acomodaron los datos.

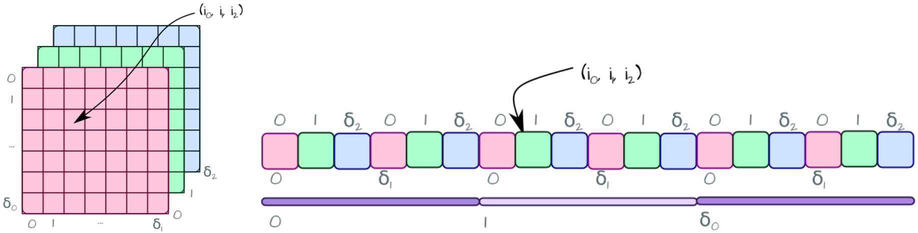
$$\text{índice}(ren, col) = col \times m + ren$$

Temas

1. Matrices densas
   - Arreglos bidimencionales
   - Arreglos n-dimencionales

Arreglos n-dimensionales

- En general, se pueden almacenar arreglos de n dimensiones en arreglos 1D.
- Es más eficiente que la representación por **Vectores de Iliffe**, por lo que se usan en contextos donde el acceso y operación con los elementos será frecuente. Por ejemplo:
  - Matrices con datos para cómputo científico.
  - Imágenes.
  - Capas convolucionales en redes neuronales.



Polinomio en n-dimensiones

δ_0 mayor

Sea δ_i el tamaño de cada dimensión i, entonces se necesita un arreglo 1D de tamaño:

$$D = \delta_0 \times \delta_1 \times \delta_2 \times ... \times \delta_{n-1} \tag{3}$$

La posición del elemento $a[i_0][i_1][...][i_{n-1}]$ esta dada por:

$$\begin{aligned} p(i_0, ..., i_{n-1}) &= (\delta_1 \delta_2 ... \delta_{n-1}) \times i_0 + (\delta_2 ... \delta_{n-1}) \times i_1 + ... + \delta_{n-1} \times i_{n-2} + i_{n-1} \\ &= \sum_{j=0}^{n-1} f_j i_j \end{aligned} \tag{4}$$

donde

$$f_j = \begin{cases} 1 & \text{si } j{=}n-1 \\ \prod_{k=j+1}^{n-1} \delta_k & \text{si } 0 \leqslant j < n-1 \end{cases} \tag{5}$$

Polinomio en n-dimensiones

δ_n mayor

Sea δ_i el tamaño de cada dimensión i, entonces se necesita un arreglo 1D de tamaño:

$$D = \delta_0 \times \delta_1 \times \delta_2 \times ... \times \delta_{n-1} \tag{6}$$

La posición del elemento $a[i_0][i_1][...][i_{n-1}]$ esta dada por:

$$\begin{aligned} p(i_0, ..., i_{n-1}) &= i_0 + \delta_0 \times i_1 + ... + (\delta_0 ... \delta_{n-2}) \times i_{n-1} \\ &= \sum_{j=0}^{n-1} f_j i_j \end{aligned} \tag{7}$$

donde

$$f_j = \begin{cases} 1 & \text{si } j{=}n-1 \\ \prod_{k=0}^{n-2} \delta_k & \text{si } 0 \leqslant j < n-1 \end{cases} \tag{8}$$

Código

Código: Polimonio ineficiente

```
public class ArregloND {
 private int[] arr;
 private int[] deltas;
 private int n;
 // ...
 private int calculaÍndice(int[] índices) {
   int ind = 0;
   for(int i = 0; i < n; i++) {
     int f = 1;
     for(int j = i + 1; j < n; j++) {
       f *= deltas[j];
     }
     ind += f * índices[i];
   }
   return ind;
 }
}
```

NOTA: En este código se muestra el comportamiento escencial, pero falta verificar precondiciones como:

- No se admiten tamaños ni índices negativos.
- Índices fuera del rango.

EFICIENCIA:

- Nótese el cálculo repetitivo del factor f.
- Ambos ciclos `for` provocan una complejidad:

$$T(n) \propto \frac{n(n+1)}{2} = \mathcal{O}(n^2)$$

Versión recursiva

Misma fórmula, pero escrita de otro modo:

$$
\begin{aligned}
\text{índice} = p(i_0, ..., i_{n-1}) = & (\delta_1 \delta_2 ... \delta_{n-1}) \times i_0 + \\
& (\delta_2 ... \delta_{n-1}) \times i_1 + \\
& ... + \\
& \delta_{n-1} \times i_{n-2} + \\
& i_{n-1}
\end{aligned}
$$

$$
\text{índice} = i_{n-1} + \delta_{n-1}(i_{n-2} + \delta_{n-2}(i_{n-3} + (...\delta_1(i_0)))) \tag{9}
$$

- Aunque esté planteada de forma recursiva, se puede programar recursiva o iterativamente.

Código

Código: Polimonio eficiente $\mathcal{O}(n)$

```java
public class ArregloND {
 private int[] arr;
 private int[] deltas;
 private int n;
 // ...
 public int calculaÍndice(int[] índices) { // El arreglo debe tener al menos dimensión 1
   int ind = índices[0];                   // Arreglo de dimensión 1
   for(int i = 1; i < n; i++) {
     ind = índices[i] + deltas[i] * ind;
   }
   return ind;
 }
}
```

Demostración de correctez

Demostración del invariante

- *Teorema:* Al inicio de cada iteración se cumple que la variable `ind` contiene el índice para un arreglo de dimensión $n = \mathtt{i}$.

$$\mathtt{ind} = \text{índice}(i_0, ..., i_{i-1}) = i_{i-1} + \delta_{i-1}(i_{i-2} + ... + (...\delta_1(i_0))) \tag{10}$$

- *Demostración:* Por inducción sobre el número de dimensiones n del arreglo.
  Caso base: Arreglo de tamaño $n = 1$
  - Para este caso, la fórmula para el índice es:

$$\mathtt{ind} \overset{\text{P.D.}}{=} \text{índice}(i_0) = i_0$$

  - Por la línea de código 7 al inicio de la primer iteración $ind = indices[0]$ e $i = 1$, que corresponde con la dimensión del arreglo ✓.

- (Continuación)

  Hip. Ind.: Suponemos que el teorema es válido para un arreglo de tamaño $n = \mathtt{i}$.

  P.D. que también lo es para i', donde i' es el valor de `i` al inicio del ciclo siguiente.

  Ind. Asumimos que $1 \leqslant i < n$ para entrar al ciclo.

  - Por la línea 9 $ind' =$ `índices[i] + deltas[i] * ind;`.
  - Por H.I. $ind = \mathtt{ind} = índice(i_0, ..., i_{i-1})$. Sustituyendo:

    $$ind' = \mathtt{índices[i]\ +\ deltas[i]\ *}\ índice(i_0, ..., i_{i-1})$$

  - Por la última parte del `for` en 8:

    $$i' = \mathtt{i\ +\ 1}$$

  entonces:

  $$\begin{aligned} \mathtt{i} &= i' - 1 \\ ind' &= \mathtt{índices}[i'-1]\ \mathtt{+\ deltas}[i'-1]\ \mathtt{*}\ índice(i_0, ..., i_{i'-1-1}) \\ &= i_{i'-1} + \delta_{i'-1}(índice(i_0, ..., i_{i'-2})) \\ &= i_{i'-1} + \delta_{i'-1}(i_{i'-2} + ... + (...\delta_1(i_0)))\checkmark \end{aligned}$$

Postcondición

- Al cumplirse la condición de terminación: `i=n` sustituimos este valor en el invariante y obtenemos:

$$\mathtt{ind} = \text{índice}(i_0, ..., i_{n-1}) = i_{n-1} + \delta_{n-1}(i_{n-2} + ... + (...\delta_1(i_0))) \checkmark \tag{11}$$

por lo que el índice que buscamos ya está en la variable `ind`.

El ciclo es finito

- Al inicio del ciclo `i=1`.
- Si la condición `i<n` no se cumple quiere decir que el arreglo es de tamaño uno y no entra al ciclo.
- Si no, la varible `i` sólo es modificada al final de cada ciclo con la instrucción `i++`. Esto genera la sucesión creciente $1, 2, 3, ...,$ la cual está acotada por `n`, por la condición de terminación `i<n`. Por lo tanto el número de iteraciones es finito.

Matrices ralas

Temas

2. Matrices ralas
   - Matrices regulares
   - Matrices ralas

Triangular

Sólo hay datos de la diagonal hacia abajo:

$$\begin{bmatrix} x & 0 & 0 & 0 \\ x & x & 0 & 0 \\ x & x & x & 0 \\ x & x & x & x \end{bmatrix}$$

$$D = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \tag{12}$$

$$p(i,j) = \frac{i(i+1)}{2} + j \tag{13}$$

Ejercicio: Diagonal y banda

$$\begin{bmatrix} \color{blue}{x} & x & 0 & 0 \\ x & \color{blue}{x} & x & 0 \\ 0 & x & \color{blue}{x} & x \\ 0 & 0 & x & \color{blue}{x} \end{bmatrix}$$

- **Banda 0:** la diagonal

$$d = 0 \qquad f(i,j) = f(i,i) \tag{14}$$

- **Banda 1:**

$$d = 1 \qquad |i - j| \leqslant d \tag{15}$$

$$p(i,j) = ? \tag{16}$$

Temas

2. Matrices ralas
   - Matrices regulares
   - Matrices ralas

Matriz rala (*sparse matrix*)

- Una *matriz rala* es aquella en la que hay información en muy pocas entradas y por lo tanto **no es eficiente** representarlas todas en memoria.

$$\begin{bmatrix} a & 0 & 0 & 0 & b \\ 0 & 0 & c & 0 & d \\ 0 & 0 & e & 0 & 0 \\ f & 0 & 0 & 0 & g \end{bmatrix}$$

Primer elemento

	Valor	Renglón	Columna	Sig. col.	Reng	Col
	T[]	int[][]			int[]	int[]
0	a	0	0	1	0	0
1	b	0	4	-1	2	-1
2	c	1	2	3	4	2
3	d	1	4	-1	5	3
4	e	2	2	-1		3
5	f	3	0	6		
6	g	3	4	-1		

Bibliografía

Bibliografía I

Notas del curso de la profesora Elisa Viso.

Licencia