4 | Polinomio de direccionamiento

Meta

Que el alumno domine el manejo de información almacenada en arreglos multidimensionales.

Objetivos

Al finalizar la práctica el alumno será capaz de:

- Almacenar un arreglo de n dimensiones en uno de una sola dimensión.
- Encapsular el comportamiento de la estructura de datos.

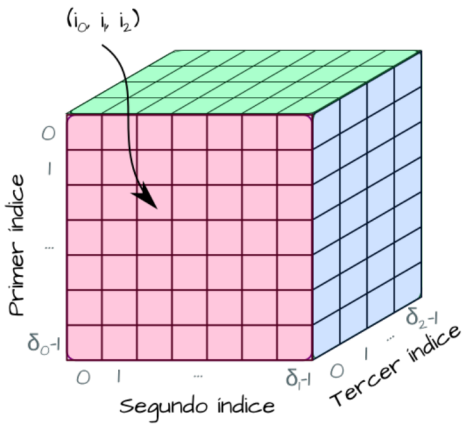

Figura 4.1 Arreglo tridimensional con $\delta_0 \times \delta_1 \times \delta_2$ elementos.

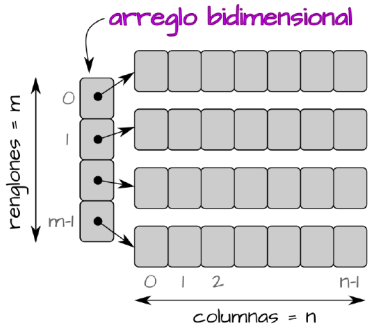

Figura 4.2 Arreglo de arreglos.

Antecedentes

Los arreglos multidimensionales son aquellos que tienen más de una dimensión, los más comunes son de dos dimensiones, conocidos como matrices, o de tres, como en la Figura 4.1.

Vectores de Iliffe

En Java los arreglos multidimensionales se almacenan como arreglos de arreglos, a los cuales se llama *Vectores de Iliffe* Figura 4.2.

Existen dos momentos fundamentales en la creación de arreglos:

1. Declaración: en esta parte no se reserva memoria, sólo se crea una referencia.

```
int [][] arreglo;
float[][][] b;
```

2. Reservación de la memoria y la especificación del número de filas y columnas.

```
//matriz con 10 filas y 5 columnas
arreglo = new int[10][5];

//cubo con 13 filas, 25 columnas y 4 planos
b = new float [13][25][4];
```

Nota: Se puede declarar el número de elementos en algunas dimensiones primero y dejar otras sin especificar, pero siempre debe de ser en orden de izquierda a derecha. Por ejemplo, `arreglo = new int[10][];` apartará la memoria para un arreglo de 10 renglones y un número aún indeterminado de columnas. Por otro lado, `arreglo = new`

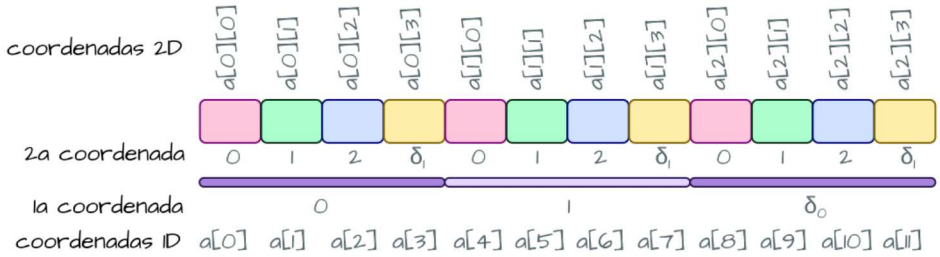

Figura 4.3 Elementos de un arreglo 2D almacenados en celdas contiguas en la memoria.

`int[][10];` es erróneo pues las filas quedan indeterminadas y `Java` no sabría de dónde sostener los arreglos de 10 elementos que corresponden a la segunda coordenada. Hecho de forma correcta, esta libertad permite crear arreglos de forma irregular, como:

```
int[][][] arreglo = new int[3][][];
arreglo[0] = new int[2][];
arreglo[0][1] = {1,2,3};
arreglo[2] = new int[1][2];
// Se ve como:
// {{null, {1,2,3}}, null, {0,0}}
```

Polinomio de direccionamiento

Dado que la memoria de la computadora es esencialmente lineal es natural pensar en almacenar los elementos de un arreglo multidimensional en un arreglo de una dimensión. Por ejemplo, considera un arreglo de dos dimensiones, como en la Figura 4.3.

Generalicemos el ejemplo anterior a uno de n-dimensiones, donde el tamaño de cada dimensión i esta dada por δ_i, entonces tenemos un arreglo nD de tamaño:

$$T = \delta_0 \times \delta_1 \times \delta_2 \times ... \times \delta_{n-1} \tag{4.1}$$

donde la posición del elemento $a[i_0][i_1][...][i_{n-1}]$ está dada por:

$$p(i_0, i_1, ..., i_{n-1}) = \sum_{j=0}^{n-1} f_j i_j \tag{4.2}$$

con

$$f_j = \begin{cases} 1 & \text{si } j{=}n-1 \\ \prod_{k=j+1}^{n-1} \delta_k & \text{si } 0 \leqslant j < n-1 \end{cases} \tag{4.3}$$

Es posible calcular el índice con menos operaciones si se factorizan las δ de modo que sólo se multiplique una vez por cada una de ellas.

Actividad 4.1

Reescribe las fórmulas para $p(i_0, i_1, ..., i_{n-1})$ de modo que sólo se multiplique una vez por cada δ. Usarás esta fórmula en tu código.

Desarrollo

La práctica consiste en implementar los métodos definidos en la interfaz `Arreglo`, la cual define una estructura que almacena los datos de un arreglo de enteros, conceptualmente de n-dimensiones, en uno de una dimensión.

1. Crea una clase llamada `ArregloPolinomio` que implemente la interfaz `Arreglo` dentro del paquete `ed.estructuras.lineales`.
2. El constructor de la clase deberá recibir como parámetro un arreglo de ints que representen las dimensiones del arreglo. Por ejemplo para crear un arreglo tridimensional $(3 \times 10 \times 5)$ funcionaría del modo siguiente:

```
1  Arreglo a = new ArregloPolinomio(new int [] {3,10,5});
```

   Observa que entonces:

   - Este arreglo debe contener al menos un elemento, si no se lanzará `IllegalSizeException`
   - El número de dimensiones en la matriz estará dada por la longitud de este primer arreglo.
   - Todas las dimensiones deben ser mayores que cero, si alguna no lo es se lanzará `IllegalSizeException`.

   Agrega a tu clase los atributos que vayas considerando necesarios para el funcionamiento del arreglo.
3. Implementa el método auxiliar `obtenerÍndice`, que será utilizado por los otros dos para leer y escribir en el arreglo. Cuida cumplir con las condiciones descritas en la documentación, pues verificar la validez de los índices es importante para la seguridad de tus datos y el correcto funcionamiento de tu clase.
4. Implementa los métodos `obtenerElemento` y `almacenarElemento` que reciben arreglos con los índices y manipulan el dato en el arreglo unidimensional subyacente.

Al final asegúrate de que tu programa pasa las pruebas unitarias. Observa que no son exhaustivas, de todos modos revisa que tu código sea correcto.

Preguntas

1. Explica la estructura de tu código, explica en más detalle tu implementación del método `obtenerÍndice`.
2. ¿Cuál es el orden de complejidad de cada método?
3. ¿Qué consecuencias negativas pudiera tener si no verificas que los índices en cada dimensión estén contenidos en el rango válido? En particular, describe un escenario en el que el código compile y ejecute sin objeciones, pero el resultado sea incorrecto.