GEPROC-ULA

Grupo de Entrenamiento de Programación Competitiva ULA

Universidad de Los Andes

19 de Febrero,2018

Matrices

Definición

1. Las matrices son objetos matemáticos que permiten organizar información numérica (y también de otros tipos) de un modo natural y sencillo.
2. La idea consiste en disponer números en forma de tabla, con una estructura de filas y columnas, de manera que cada elemento (cada número) de la tabla puede ser identificado mediante su posición: la fila y la columna en las que está situado el elemento.
3. Al igual que los arreglos, es una estructuras de datos básica.

Matriz: Características

$$A = \begin{bmatrix} 1 & 3 & 4 \\ 9 & 8 & 7 \\ 3 & 1 & 3 \end{bmatrix}$$

- **Diagonal primaria**: Representan los elementos correspondientes de la matriz A de nxn con posiciones **pos**(ij) tal que i = j.
  - 1 , 8 , 3
- **Diagonal secundaria**: Representan los elementos correspondientes a de la matriz A de nxn con posiciones **pos**(ij) tal que i+j = n-1.
  - 4, 8, 3

Matriz: Características

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 9 & 8 & 0 \\ 3 & 1 & 3 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 3 & 4 \\ 0 & 8 & 7 \\ 0 & 0 & 3 \end{bmatrix}$$

- **Matriz diagonal superior**: Una matriz se le denomina triangular superior cuando todos los elementos por debajo de la diagonal primaria son ceros. Ejemplo: Matriz B.
- **Matriz diagonal inferior**: Una matriz se le denomina triangular inferior cuando todos los elementos por encima de la diagonal primaria son ceros. Ejemplo: Matriz A.

Matriz: Características

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 8 & 7 \\ 2 & 0 & 0 \end{bmatrix}$$

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

- **Matriz esparcida**: Una matriz se denomina esparcida cuando la mayoría de valores es cero. Ejemplo matriz A.
- **Matriz diagonal constante**: Una matriz se denomina diagonal constante si cada diagonal descendiendo de izquierda a derecha es constante.

Matriz: Características

$$A = \begin{bmatrix} 5 & 1 & 3 \\ 1 & 8 & 2 \\ 3 & 2 & 5 \end{bmatrix} \quad A^T = \begin{bmatrix} 5 & 1 & 3 \\ 1 & 8 & 2 \\ 3 & 2 & 5 \end{bmatrix}$$

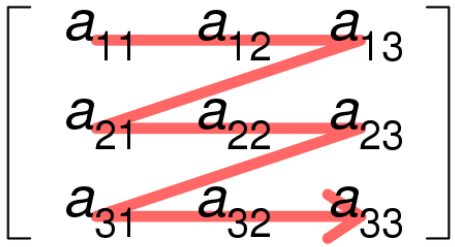
$$B = \begin{bmatrix} 0 & 1 & -8 \\ -1 & 0 & 6 \\ 8 & -6 & 0 \end{bmatrix} \quad B^T = \begin{bmatrix} 0 & -1 & 8 \\ 1 & 0 & -6 \\ -8 & 6 & 0 \end{bmatrix}$$

- **Matriz simétrica**: Una matriz se denomina simétrica cuando es igual a su **traspuesta**. Ejemplo matriz A.
- **Matriz anti-simétrica**: Una matriz se denomina anti-simétrica cuando es igual a su traspuesta pero negativa. Ejemplo matriz B.

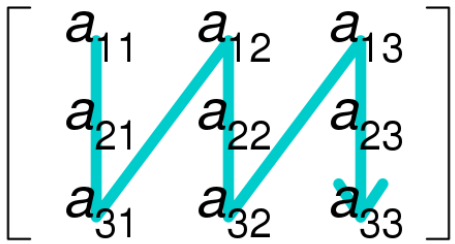
Simular una matriz 2D usando un array 1D

- A veces es necesario simular trabajar una matriz como un arreglo.
- Esto nos permite trabajar con un poco mas de flexibilidad cuando las operaciones que queremos realizar sobre la matriz son secuenciales.
- **Row-major order y col-major order** son los métodos usados para lograr esto.

Row-major order



Column-major order



Row-major order y Col-major order

- Dada una matriz **MxN**, si los elementos están almacenados en **row-major order**. Entonces, el elemento en la posición [i,j] en la matriz estará almacenado en el arreglo en la posición K:
  - **k = j + (i * Numero total de columnas)**
- Dada una matriz **MxN**, si los elementos están almacenados en **column-major order**. Entonces, el elemento en la posición [i,j] en la matriz estará almacenado en el arreglo en la posición K:
  - **k = i + (j * Numero total de filas)**

```
1 int row_mayor_val(int i, int j, int *array,int num_colums)
2 {
3         return array[j + ( i * num_colums)];
4 }
5
6 int col_mayor_val(int i, int j, int *array,int num_rows)
7 {
8         return array[i + ( j * num_rows)];
9 }
```

```
1 int main(int argc, char * argv[]){
2 // Declaring number of rows and columns
3 int n = 3, m = 3;
4 int array[n*m];
5 // Intialising a 2-d array
6 int grid[n][m] = {{1, 2, 3},{4, 5, 6},{7, 8, 9}};
7
8 // storing elements in 1-d array
9 int i, j, k = 0;
10 for (i=0; i<n; i++)
11         for (j=0; j<m; j++){
12                 k = i*m + j;
13                 array[k] = grid[i][j];
14                 k++;
15         }
16 // displaying elements in 1-d array
17 for (i=0; i<n; i++){
18         for (j=0; j<m; j++)
19                 printf("%d ", col_mayor_val(i,j,array,m));
20
21         printf("\n");
22 }
23
24         return 0;
```