Universidad Tecnológica Equinoccial

Tecnología Superior en Desarrollo de Software

Programación

Nombre: Jossue Guerrero

Paralelo: "B"

ARREGLOS Y MATRICES

Es una tarea funcional y fundamental para la manipulación de datos. En la programación, estos se almacenan en **arreglos y matrices** y en mucho de esos casos es necesario organizarlos para ayudar a optimizar sus búsquedas y cálculos.

Hay varios algoritmos de ordenamiento, algunos más adecuados para arreglos unidimensionales y otros que pueden aplicarse a matrices.

1. Ordenamiento en Arreglos (Vectores)

Un arreglo es una lista de elementos que se pueden organizar utilizando los algoritmos clásicos de ordenamiento.

a) Ordenamiento Burbuja en Arreglo

```
PS C:\Users\alumnos\Downloads> & C:/Users/alumnos/AppData/Local/Programs/Python/Python313/python.exe c:/Users/alumnos/Downloads/EJEMPLOS
Arreglo ordenado: [5, 7, 23, 32, 34, 62]
```

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:  # Intercambia si están en orden incorrecto
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

# Ejemplo de uso
arreglo = [34, 7, 23, 32, 5, 62]
print("Arreglo ordenado:", bubble_sort(arreglo))
```

b) Ordenamiento por Inserción en Arreglo

```python
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i - 1
        while j >= 0 and arr[j] > key:
            arr[j + 1] = arr[j]  # Desplaza elementos
            j -= 1
        arr[j + 1] = key  # Inserta en su lugar
    return arr

# Ejemplo de uso
arreglo = [34, 7, 23, 32, 5, 62]
print("Arreglo ordenado:", insertion_sort(arreglo))

```

```
PS C:\Users\alumnos\Downloads> & C:/Users/alumnos/AppData/Local/Programs/Python/Python313/python.exe c:/Users/alumnos/Downloads/EJEMPLOS
Arreglo ordenado: [5, 7, 23, 32, 34, 62]
```

c) Ordenamiento por Selección en Arreglo

```python
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:  # Encuentra el menor
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]  # Intercambio
    return arr

# Ejemplo de uso
arreglo = [34, 7, 23, 32, 5, 62]
print("Arreglo ordenado:", selection_sort(arreglo))

```

```
PS C:\Users\alumnos\Downloads> & C:/Users/alumnos/AppData/Local/Programs/Python/Python313/python.exe c:/Users/alumnos/Downloads/EJEMPLOS
Arreglo ordenado: [5, 7, 23, 32, 34, 62]
```

2. Ordenamiento en Matrices (Arreglos Bidimensionales)

Una matriz es un conjunto de datos organizados en filas y columnas. Ayudan a ordenar una matriz es mas complejo que ordenar a un arreglo.

.-Ordenar **cada fila** individualmente.

.-Ordenar **cada columna** individualmente.

.- Convertir la matriz en un arreglo, ordenarlo y reconstruir la matriz.

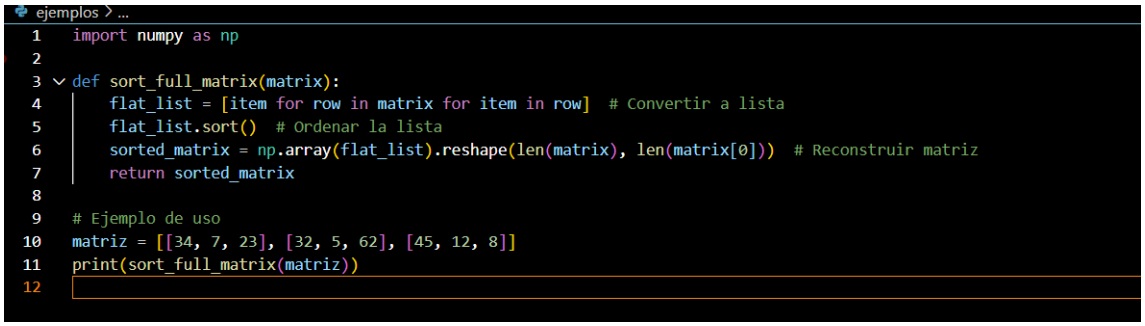
a) Ordenar una Matriz por Filas usando Bubble Sort

```python
def sort_matrix(matrix):
    # Convertir la matriz en un arreglo unidimensional
    arr = [elemento for fila in matrix for elemento in fila]
    # Ordenar el arreglo (puedes usar cualquier algoritmo de ordenamiento)
    arr.sort()
    # Reconstruir la matriz
    n = len(matrix[0])  # Número de columnas
    matriz_ordenada = [arr[i*n:(i+1)*n] for i in range(len(matrix))]
    return matriz_ordenada

# Ejemplo de uso
matriz = [[34, 7, 23], [32, 5, 62], [45, 12, 8]]
matriz_ordenada = sort_matrix(matriz)

for fila in matriz_ordenada:
    print(fila)
```

```
PS C:\Users\alumnos\Downloads> & C:/Users/alumnos/AppData/Local/Programs/Python/Python313/python.exe c:/Users/alumnos/Downloads/EJEMPLOS
[5, 7, 8]
[12, 23, 32]
[34, 45, 62]
PS C:\Users\alumnos\Downloads>
```

b) Ordenar Matriz Completa Convertida en Arreglo

Este método **convierte la matriz en un arreglo, lo ordena y reconstruye la matriz**.



```python
import numpy as np

def sort_full_matrix(matrix):
    flat_list = [item for row in matrix for item in row]  # Convertir a lista
    flat_list.sort()  # Ordenar la lista
    sorted_matrix = np.array(flat_list).reshape(len(matrix), len(matrix[0]))  # Reconstruir matriz
    return sorted_matrix

# Ejemplo de uso
matriz = [[34, 7, 23], [32, 5, 62], [45, 12, 8]]
print(sort_full_matrix(matriz))
```

```
PS C:\Users\alumnos\Downloads> & C:/Users/alumnos/AppData/Local/Programs/Python/Python313/python.exe c:/Users/alumnos/Downloads/EJEMPLOS
[5, 7, 8]
[12, 23, 32]
[34, 45, 62]
PS C:\Users\alumnos\Downloads>
```

Conclusión:

El ordenamiento en arreglos y matrices es una técnica funcional para organizar los datos y mejorar su manipulación:

-Para **arreglos unidimensionales**, los métodos como **Bubble Sort, Inserción y Selección** son simples pero poco eficientes para grandes volúmenes de datos.

-Para **matrices bidimensionales**, se pueden ordenar **fila por fila** o convertir la matriz en un **arreglo lineal**, ordenarlo y luego reconstruirla.