Universidad de Cundinamarca

Estructura de Datos

R1-A5-S7 Complejidad Computacional

Javier Mateo Barrero Vanegas

Jim Alejandro Quiñones Martínez

Luis Ángel Martínez Cuenca

Harick Yesid Villarraga Rincon

Cristian Esteban Ruiz Parra

Docente: Dilia Ines Molina Cubillos

2 octubre de 2025

Índice

---

1. Introducción

Este programa implementa un **arreglo bidimensional asimétrico** en Python, también conocido como arreglo irregular o "jagged array". A diferencia de un arreglo rectangular tradicional donde todas las filas tienen la misma cantidad de columnas, este arreglo tiene una estructura triangular donde cada fila tiene un número diferente de elementos.

Fundamentos de la Complejidad Temporal

La complejidad temporal es un criterio fundamental para evaluar la eficiencia de un algoritmo, midiendo cómo su tiempo de ejecución varía en función del tamaño de la entrada (n). Esta métrica, expresada mediante la notación Big O, permite predecir el comportamiento de un algoritmo en escenarios de crecimiento asintótico, lo que es crucial para aplicaciones que manejan grandes volúmenes de datos.

Objetivos:

1. Definir los principios teóricos de la complejidad temporal.
2. Clasificar las principales categorías de complejidad y su impacto en la escalabilidad.
3. Destacar la relevancia de este análisis en el diseño de sistemas eficientes.

Desarrollo:

1. **Fundamentos Teóricos:**

- La complejidad temporal se calcula contando las operaciones elementales del algoritmo, asumiendo que cada una consume un tiempo constante.
- El **peor caso** es el escenario más utilizado para garantizar cotas superiores de rendimiento.

2. **Clasificación de Complejidades:**
   - $O(1)$**:** Operaciones directas (ej: acceso a un índice en un array).
   - $O(\log\ n)$**:** Algoritmos que dividen el problema (ej: búsqueda binaria).
   - $O(n)$**:** Procesamiento secuencial de datos (ej: búsqueda lineal).
   - $O(n\log\ n)$**:** Algoritmos óptimos de ordenamiento (ej: Merge Sort, Quick Sort).
   - $O(n^2)$**:** Algoritmos ineficientes en grandes datasets (ej: Bubble Sort).
   - $O(2^n)$**:** Problemas NP-completos (ej: Torre de Hanói).
3. **Importancia en la Computación Moderna:**
   - Permite seleccionar algoritmos escalables, evitando cuellos de botella en aplicaciones críticas.
   - Facilita la optimización de recursos en sistemas con restricciones de tiempo (ej: tiempo real).

Objetivo del Programa

Demostrar el uso de estructuras de datos no uniformes y aplicar conceptos de complejidad computacional en operaciones básicas como creación, carga e impresión de datos.

Requisitos Funcionales

- Crear un arreglo con estructura triangular específica
- Cargar datos mediante interacción con el usuario
- Mostrar los datos en formato legible
- Documentar la complejidad computacional de cada operación

---

2. Estructura del Arreglo

2.1 Representación Visual

Fila 0: [ ]

Fila 1: [ ] [ ]

Fila 2: [ ] [ ] [ ]

Fila 3: [ ] [ ] [ ] [ ]

Fila 4: [ ] [ ] [ ] [ ] [ ]

Fila 5: [ ] [ ] [ ] [ ] [ ] [ ]

2.2 Características Matemáticas

- **Número de filas**: 6
- **Elementos por fila**: La fila i contiene (i + 1) elementos
- **Total de elementos**: 1 + 2 + 3 + 4 + 5 + 6 = 21 elementos
- **Fórmula general**: n(n+1)/2, donde n = número de filas

2.3 Implementación en Python

```
arreglo = [
  [elemento_0_0],                  # Fila 0: 1 elemento
  [elemento_1_0, elemento_1_1],          # Fila 1: 2 elementos
  [elemento_2_0, elemento_2_1, elemento_2_2], # Fila 2: 3 elementos
  ...
]
```

El arreglo se representa como una **lista de listas** donde cada sublista puede tener diferente longitud.

2.4 Screenshots del código

```python
"""
ARREGLO BIDIMENSIONAL ASIMÉTRICO

Este programa implementa un arreglo bidimensional asimétrico con estructura triangular.
Incluye análisis de complejidad computacional en sus operaciones principales.

Estructura del arreglo:
Fila 0: 1 elemento
Fila 1: 2 elementos
Fila 2: 3 elementos
Fila 3: 4 elementos
Fila 4: 5 elementos
Fila 5: 6 elementos
"""

def crear_arreglo():
    """
    Crea la estructura del arreglo bidimensional asimétrico.

    Complejidad Temporal: O(n²) donde n es el número de filas
    - El bucle externo se ejecuta n veces (6 filas)
    - Cada iteración crea una lista de tamaño i+1
    - Total de operaciones: 1+2+3+4+5+6 = 21 = n(n+1)/2

    Complejidad Espacial: O(n²)
    - Se almacenan n(n+1)/2 elementos en total

    Returns:
        list: Arreglo bidimensional asimétrico vacío (con None)
    """
    # Número de filas del arreglo triangular
    num_filas = 6

    # Crear el arreglo asimétrico
    # Cada fila i tiene (i+1) elementos
    arreglo = []
```

```python
    for i in range(num_filas):
        # Crear una fila con (i+1) elementos inicializados en None
        fila = [None] * (i + 1)
        arreglo.append(fila)

    return arreglo


def cargar(arreglo):
    """
    Carga datos en el arreglo bidimensional asimétrico.
    Los datos se solicitan al usuario elemento por elemento.

    Complejidad Temporal: O(n²) donde n es el número de filas
    - Se recorren todas las filas (bucle externo: n iteraciones)
    - Para cada fila i, se recorren i+1 elementos (bucle interno)
    - Total: 1+2+3+4+5+6 = 21 operaciones = n(n+1)/2

    Complejidad Espacial: O(1)
    - No se crea espacio adicional, solo se modifica el arreglo existente

    Args:
        arreglo (list): Arreglo bidimensional a cargar
    """
    print("\n" + "="*50)
    print("CARGA DE DATOS EN EL ARREGLO ASIMÉTRICO")
    print("="*50)

    # Recorrer cada fila del arreglo
    for i in range(len(arreglo)):
        print(f"\nFila {i} (tiene {len(arreglo[i])} elementos):")

        # Recorrer cada columna de la fila actual
        for j in range(len(arreglo[i])):
            # Solicitar dato al usuario con validación
            while True:
                try:
```

```
            while True:
                try:
                    dato = input(f"  Ingrese el valor para posición [{i}][{j}]: ")
                    # Intentar convertir a número (int o float)
                    try:
                        valor = int(dato)
                    except ValueError:
                        valor = float(dato)

                    # Asignar el valor al arreglo
                    arreglo[i][j] = valor
                    break
                except ValueError:
                    # Si no es número, guardar como string
                    arreglo[i][j] = dato
                    break

    print("\n✓ Datos cargados exitosamente")


def imprimir(arreglo):
    """
    Imprime el arreglo bidimensional asimétrico en formato visual triangular.

    Complejidad Temporal: O(n²) donde n es el número de filas
    - Se recorren todas las filas (n iteraciones)
    - Para cada fila i, se imprimen i+1 elementos
    - Total de operaciones de impresión: n(n+1)/2

    Complejidad Espacial: O(1)
    - Solo se usan variables temporales para la impresión

    Args:
        arreglo (list): Arreglo bidimensional a imprimir
    """
    print("\n" + "="*50)
    print("CONTENIDO DEL ARREGLO ASIMÉTRICO")
    print("="*50)

    # Calcular el ancho máximo para formateo
    max_ancho = 10

    # Recorrer cada fila
    for i in range(len(arreglo)):
        # Imprimir número de fila
        print(f"Fila {i}: ", end="")

        # Recorrer cada elemento de la fila
        for j in range(len(arreglo[i])):
            # Formatear y imprimir el elemento
```

```
            elemento = str(arreglo[i][j])
            print(f"[{elemento:^{max_ancho}}]", end=" ")

        # Nueva línea al terminar la fila
        print()

    print("="*50)


def imprimir_estadisticas(arreglo):
    """
    Calcula e imprime estadísticas del arreglo (elementos totales, valores numéricos).

    Complejidad Temporal: O(n²)
    - Se recorren todos los elementos del arreglo

    Complejidad Espacial: O(1)
    - Solo se usan variables para contar y sumar

    Args:
        arreglo (list): Arreglo bidimensional
    """
    total_elementos = 0
    elementos_numericos = 0
    suma_numericos = 0

    # Recorrer todo el arreglo para calcular estadísticas
    # Complejidad: O(n²)
    for i in range(len(arreglo)):
        for j in range(len(arreglo[i])):
            total_elementos += 1

            # Verificar si el elemento es numérico
            if isinstance(arreglo[i][j], (int, float)):
                elementos_numericos += 1
                suma_numericos += arreglo[i][j]

    print("\n" + "="*50)
    print("ESTADÍSTICAS DEL ARREGLO")
    print("="*50)
    print(f"Total de elementos: {total_elementos}")
    print(f"Elementos numéricos: {elementos_numericos}")

    if elementos_numericos > 0:
        promedio = suma_numericos / elementos_numericos
        print(f"Suma de valores numéricos: {suma_numericos}")
        print(f"Promedio de valores numéricos: {promedio:.2f}")

    print("="*50)
```

```python
def menu():
    """
    Muestra el menú principal y maneja las opciones del usuario.

    Complejidad Temporal: O(1) para el menú, pero las operaciones llamadas
    tienen sus propias complejidades (O(n²) para cargar e imprimir)

    Complejidad Espacial: O(n²) por el arreglo almacenado
    """
    # Crear el arreglo asimétrico vacío
    # Complejidad: O(n²)
    arreglo = crear_arreglo()

    # Variable para controlar si el arreglo ha sido cargado
    arreglo_cargado = False

    while True:
        print("\n" + "="*50)
        print("MENÚ PRINCIPAL - ARREGLO BIDIMENSIONAL ASIMÉTRICO")
        print("="*50)
        print("1. Cargar datos en el arreglo")
        print("2. Imprimir arreglo")
        print("3. Mostrar estadísticas")
        print("4. Visualizar estructura del arreglo")
        print("5. Salir")
        print("="*50)

        opcion = input("Seleccione una opción: ")

        if opcion == "1":
            # Cargar datos - Complejidad: O(n²)
            cargar(arreglo)
            arreglo_cargado = True

        elif opcion == "2":
            # Imprimir arreglo - Complejidad: O(n²)
            if arreglo_cargado:
                imprimir(arreglo)
            else:
                print("\n⚠ Primero debe cargar datos en el arreglo (opción 1)")

        elif opcion == "3":
            # Mostrar estadísticas - Complejidad: O(n²)
            if arreglo_cargado:
                imprimir_estadisticas(arreglo)
            else:
                print("\n⚠ Primero debe cargar datos en el arreglo (opción 1)")
```

```
        elif opcion == "4":
            # Visualizar estructura - Complejidad: O(n²)
            print("\n" + "="*50)
            print("ESTRUCTURA DEL ARREGLO ASIMÉTRICO")
            print("="*50)
            for i in range(len(arreglo)):
                print(f"Fila {i}: {len(arreglo[i])} elementos → {arreglo[i]}")
            print("="*50)

        elif opcion == "5":
            # Salir del programa
            print("\n¡Gracias por usar el programa!")
            print("="*50)
            break

        else:
            print("\n✗ Opción inválida. Por favor seleccione una opción del 1 al 5.")


def main():
    """
    Función principal del programa.

    Complejidad Total del Programa:
    - Temporal: O(n²) dominada por las operaciones de carga e impresión
    - Espacial: O(n²) por el almacenamiento del arreglo triangular

    Donde n = número de filas (6 en este caso)
    Elementos totales: n(n+1)/2 = 6(7)/2 = 21 elementos
    """
    print("="*50)
    print("PROGRAMA: ARREGLO BIDIMENSIONAL ASIMÉTRICO")
    print("="*50)
    print("\nEste programa maneja un arreglo con estructura triangular:")
    print("  Fila 0: 1 elemento")
    print("  Fila 1: 2 elementos")
    print("  Fila 2: 3 elementos")
    print("  Fila 3: 4 elementos")
    print("  Fila 4: 5 elementos")
    print("  Fila 5: 6 elementos")
    print("\nTotal: 21 elementos")

    # Iniciar el menú interactivo
    menu()


# Punto de entrada del programa
# Complejidad global: O(n²) en tiempo y espacio
if __name__ == "__main__":
    main()
```

2.5 Codigo compilando

```
================================================
PROGRAMA: ARREGLO BIDIMENSIONAL ASIMÉTRICO
================================================

Este programa maneja un arreglo con estructura triangular:
  Fila 0: 1 elemento
  Fila 1: 2 elementos
  Fila 2: 3 elementos
  Fila 3: 4 elementos
  Fila 4: 5 elementos
  Fila 5: 6 elementos

Total: 21 elementos

================================================
MENÚ PRINCIPAL - ARREGLO BIDIMENSIONAL ASIMÉTRICO
================================================
1. Cargar datos en el arreglo
2. Imprimir arreglo
3. Mostrar estadísticas
4. Visualizar estructura del arreglo
5. Salir
================================================
Seleccione una opción:
```

```
Seleccione una opción: 1

================================================
CARGA DE DATOS EN EL ARREGLO ASIMÉTRICO
================================================

Fila 0 (tiene 1 elementos):
  Ingrese el valor para posición [0][0]:
```

```
    Fila 0 (tiene 1 elementos):
      Ingrese el valor para posición [0][0]: 1

    Fila 1 (tiene 2 elementos):
      Ingrese el valor para posición [1][0]:
```

```
    Fila 0 (tiene 1 elementos):
      Ingrese el valor para posición [0][0]: 1

    Fila 1 (tiene 2 elementos):
      Ingrese el valor para posición [1][0]: 2
      Ingrese el valor para posición [1][1]:
```

```
   ------------------------------------------------

  Fila 0 (tiene 1 elementos):
    Ingrese el valor para posición [0][0]: 1

  Fila 1 (tiene 2 elementos):
    Ingrese el valor para posición [1][0]: 2
    Ingrese el valor para posición [1][1]: 3

  Fila 2 (tiene 3 elementos):
    Ingrese el valor para posición [2][0]: 4
    Ingrese el valor para posición [2][1]:
```

```
    ================================================
    MENÚ PRINCIPAL - ARREGLO BIDIMENSIONAL ASIMÉTRICO
    ================================================
    1. Cargar datos en el arreglo
    2. Imprimir arreglo
    3. Mostrar estadísticas
    4. Visualizar estructura del arreglo
    5. Salir
    ================================================
    Seleccione una opción: 1

    ================================================
    CARGA DE DATOS EN EL ARREGLO ASIMÉTRICO
    ================================================

    Fila 0 (tiene 1 elementos):
      Ingrese el valor para posición [0][0]: 1

    Fila 1 (tiene 2 elementos):
      Ingrese el valor para posición [1][0]: 2
      Ingrese el valor para posición [1][1]: 3

    Fila 2 (tiene 3 elementos):
      Ingrese el valor para posición [2][0]: 4
      Ingrese el valor para posición [2][1]: 5
      Ingrese el valor para posición [2][2]:
```

```
  Fila 2 (tiene 3 elementos):
    Ingrese el valor para posición [2][0]: 4
    Ingrese el valor para posición [2][1]: 5
    Ingrese el valor para posición [2][2]: 6

  Fila 3 (tiene 4 elementos):
    Ingrese el valor para posición [3][0]:
```

```
 Fila 3 (tiene 4 elementos):
   Ingrese el valor para posición [3][0]: 7
   Ingrese el valor para posición [3][1]: 8
   Ingrese el valor para posición [3][2]: 9
   Ingrese el valor para posición [3][3]: 0
```

```
Fila 4 (tiene 5 elementos):
  Ingrese el valor para posición [4][0]: 1
  Ingrese el valor para posición [4][1]: 2
  Ingrese el valor para posición [4][2]: 3
  Ingrese el valor para posición [4][3]: 4
  Ingrese el valor para posición [4][4]: 5
```

```
Fila 5 (tiene 6 elementos):
  Ingrese el valor para posición [5][0]: 6
  Ingrese el valor para posición [5][1]: 7
  Ingrese el valor para posición [5][2]: 8
  Ingrese el valor para posición [5][3]: 9
  Ingrese el valor para posición [5][4]: 0
  Ingrese el valor para posición [5][5]: 1

✓ Datos cargados exitosamente
```

```
================================================
MENÚ PRINCIPAL - ARREGLO BIDIMENSIONAL ASIMÉTRICO
================================================
1. Cargar datos en el arreglo
2. Imprimir arreglo
3. Mostrar estadísticas
4. Visualizar estructura del arreglo
5. Salir
================================================
Seleccione una opción: 2

================================================
CONTENIDO DEL ARREGLO ASIMÉTRICO
================================================
Fila 0: [    1    ]
Fila 1: [    2    ] [    3    ]
Fila 2: [    4    ] [    5    ] [    6    ]
Fila 3: [    7    ] [    8    ] [    9    ] [    0    ]
Fila 4: [    1    ] [    2    ] [    3    ] [    4    ] [    5    ]
Fila 5: [    6    ] [    7    ] [    8    ] [    9    ] [    0    ] [    1    ]
================================================
```

```
================================================
MENÚ PRINCIPAL - ARREGLO BIDIMENSIONAL ASIMÉTRICO
================================================
1. Cargar datos en el arreglo
2. Imprimir arreglo
3. Mostrar estadísticas
4. Visualizar estructura del arreglo
5. Salir
================================================
Seleccione una opción: 3

================================================
ESTADÍSTICAS DEL ARREGLO
================================================
Total de elementos: 21
Elementos numéricos: 21
Suma de valores numéricos: 91
Promedio de valores numéricos: 4.33
================================================
```

```
================================================
ESTRUCTURA DEL ARREGLO ASIMÉTRICO
================================================
Fila 0: 1 elementos → [1]
Fila 1: 2 elementos → [2, 3]
Fila 2: 3 elementos → [4, 5, 6]
Fila 3: 4 elementos → [7, 8, 9, 0]
Fila 4: 5 elementos → [1, 2, 3, 4, 5]
Fila 5: 6 elementos → [6, 7, 8, 9, 0, 1]
================================================
```

```
  ================================================
  MENÚ PRINCIPAL - ARREGLO BIDIMENSIONAL ASIMÉTRICO
  ================================================
  1. Cargar datos en el arreglo
  2. Imprimir arreglo
  3. Mostrar estadísticas
  4. Visualizar estructura del arreglo
  5. Salir
  ================================================
  Seleccione una opción: 5

  ¡Gracias por usar el programa!
  ================================================
```

2.6 Código literal

"""

ARREGLO BIDIMENSIONAL ASIMÉTRICO

Este programa implementa un arreglo bidimensional asimétrico con estructura triangular.

Incluye análisis de complejidad computacional en sus operaciones principales.

Estructura del arreglo:

Fila 0: 1 elemento

Fila 1: 2 elementos

Fila 2: 3 elementos

Fila 3: 4 elementos

Fila 4: 5 elementos

```
Fila 5: 6 elementos
"""

def crear_arreglo():
    """
    Crea la estructura del arreglo bidimensional asimétrico.

    Complejidad Temporal: O(n^2) donde n es el número de filas
    - El bucle externo se ejecuta n veces (6 filas)
    - Cada iteración crea una lista de tamaño i+1
    - Total de operaciones: 1+2+3+4+5+6 = 21 = n(n+1)/2

    Complejidad Espacial: O(n^2)
    - Se almacenan n(n+1)/2 elementos en total

    Returns:
        list: Arreglo bidimensional asimétrico vacío (con None)
    """
    # Número de filas del arreglo triangular
    num_filas = 6

    # Crear el arreglo asimétrico
    # Cada fila i tiene (i+1) elementos
    arreglo = []

    for i in range(num_filas):
```

```
        # Crear una fila con (i+1) elementos inicializados en None
        fila = [None] * (i + 1)
        arreglo.append(fila)

    return arreglo


def cargar(arreglo):
    """
    Carga datos en el arreglo bidimensional asimétrico.
    Los datos se solicitan al usuario elemento por elemento.

    Complejidad Temporal: O(n^2) donde n es el número de filas
    - Se recorren todas las filas (bucle externo: n iteraciones)
    - Para cada fila i, se recorren i+1 elementos (bucle interno)
    - Total: 1+2+3+4+5+6 = 21 operaciones = n(n+1)/2

    Complejidad Espacial: O(1)
    - No se crea espacio adicional, solo se modifica el arreglo existente

    Args:
        arreglo (list): Arreglo bidimensional a cargar
    """
    print("\n" + "="*50)
    print("CARGA DE DATOS EN EL ARREGLO ASIMÉTRICO")
    print("="*50)
```

```
# Recorrer cada fila del arreglo
for i in range(len(arreglo)):
    print(f"\nFila {i} (tiene {len(arreglo[i])} elementos):")

    # Recorrer cada columna de la fila actual
    for j in range(len(arreglo[i])):
        # Solicitar dato al usuario con validación
        while True:
            try:
                dato = input(f"  Ingrese el valor para posición [{i}][{j}]: ")
                # Intentar convertir a número (int o float)
                try:
                    valor = int(dato)
                except ValueError:
                    valor = float(dato)

                # Asignar el valor al arreglo
                arreglo[i][j] = valor
                break
            except ValueError:
                # Si no es número, guardar como string
                arreglo[i][j] = dato
                break

print("\n✓ Datos cargados exitosamente")
```

```
def imprimir(arreglo):
    """
    Imprime el arreglo bidimensional asimétrico en formato visual triangular.

    Complejidad Temporal: O(n^2) donde n es el número de filas
    - Se recorren todas las filas (n iteraciones)
    - Para cada fila i, se imprimen i+1 elementos
    - Total de operaciones de impresión: n(n+1)/2

    Complejidad Espacial: O(1)
    - Solo se usan variables temporales para la impresión

    Args:
        arreglo (list): Arreglo bidimensional a imprimir
    """
    print("\n" + "="*50)
    print("CONTENIDO DEL ARREGLO ASIMÉTRICO")
    print("="*50)

    # Calcular el ancho máximo para formateo
    max_ancho = 10

    # Recorrer cada fila
    for i in range(len(arreglo)):
```

```
        # Imprimir número de fila
        print(f"Fila {i}: ", end="")

        # Recorrer cada elemento de la fila
        for j in range(len(arreglo[i])):
            # Formatear y imprimir el elemento
            elemento = str(arreglo[i][j])
            print(f"[{elemento:^{max_ancho}}]", end=" ")

        # Nueva línea al terminar la fila
        print()

    print("="*50)


def imprimir_estadisticas(arreglo):
    """
    Calcula e imprime estadísticas del arreglo (elementos totales, valores numéricos).

    Complejidad Temporal: O(n^2)
    - Se recorren todos los elementos del arreglo

    Complejidad Espacial: O(1)
    - Solo se usan variables para contar y sumar
```

```
Args:
    arreglo (list): Arreglo bidimensional
"""
total_elementos = 0
elementos_numericos = 0
suma_numericos = 0

# Recorrer todo el arreglo para calcular estadísticas
# Complejidad: O(n^2)
for i in range(len(arreglo)):
    for j in range(len(arreglo[i])):
        total_elementos += 1

        # Verificar si el elemento es numérico
        if isinstance(arreglo[i][j], (int, float)):
            elementos_numericos += 1
            suma_numericos += arreglo[i][j]

print("\n" + "="*50)
print("ESTADÍSTICAS DEL ARREGLO")
print("="*50)
print(f"Total de elementos: {total_elementos}")
print(f"Elementos numéricos: {elementos_numericos}")

if elementos_numericos > 0:
    promedio = suma_numericos / elementos_numericos
```

```
        print(f"Suma de valores numéricos: {suma_numericos}")
        print(f"Promedio de valores numéricos: {promedio:.2f}")

    print("="*50)


def menu():
    """
    Muestra el menú principal y maneja las opciones del usuario.

    Complejidad Temporal: O(1) para el menú, pero las operaciones llamadas
    tienen sus propias complejidades (O(n^2) para cargar e imprimir)

    Complejidad Espacial: O(n^2) por el arreglo almacenado
    """
    # Crear el arreglo asimétrico vacío
    # Complejidad: O(n^2)
    arreglo = crear_arreglo()

    # Variable para controlar si el arreglo ha sido cargado
    arreglo_cargado = False

    while True:
        print("\n" + "="*50)
        print("MENÚ PRINCIPAL - ARREGLO BIDIMENSIONAL ASIMÉTRICO")
        print("="*50)
```

```
print("1. Cargar datos en el arreglo")
print("2. Imprimir arreglo")
print("3. Mostrar estadísticas")
print("4. Visualizar estructura del arreglo")
print("5. Salir")
print("="*50)

opcion = input("Seleccione una opción: ")

if opcion == "1":
    # Cargar datos - Complejidad: O(n^2)
    cargar(arreglo)
    arreglo_cargado = True

elif opcion == "2":
    # Imprimir arreglo - Complejidad: O(n^2)
    if arreglo_cargado:
        imprimir(arreglo)
    else:
        print("\n⚠ Primero debe cargar datos en el arreglo (opción 1)")

elif opcion == "3":
    # Mostrar estadísticas - Complejidad: O(n^2)
    if arreglo_cargado:
        imprimir_estadisticas(arreglo)
    else:
```

```
            print("\n⚠ Primero debe cargar datos en el arreglo (opción 1)")

        elif opcion == "4":
            # Visualizar estructura - Complejidad: O(n²)
            print("\n" + "="*50)
            print("ESTRUCTURA DEL ARREGLO ASIMÉTRICO")
            print("="*50)
            for i in range(len(arreglo)):
                print(f"Fila {i}: {len(arreglo[i])} elementos → {arreglo[i]}")
            print("="*50)

        elif opcion == "5":
            # Salir del programa
            print("\n¡Gracias por usar el programa!")
            print("="*50)
            break

        else:
            print("\n✗ Opción inválida. Por favor seleccione una opción del 1 al 5.")


def main():
    """
    Función principal del programa.

    Complejidad Total del Programa:
```

- Temporal: O(n^2) dominada por las operaciones de carga e impresión

- Espacial: O(n^2) por el almacenamiento del arreglo triangular

Donde n = número de filas (6 en este caso)

Elementos totales: n(n+1)/2 = 6(7)/2 = 21 elementos

"""

```
    print("="*50)
    print("PROGRAMA: ARREGLO BIDIMENSIONAL ASIMÉTRICO")
    print("="*50)
    print("\nEste programa maneja un arreglo con estructura triangular:")
    print("  Fila 0: 1 elemento")
    print("  Fila 1: 2 elementos")
    print("  Fila 2: 3 elementos")
    print("  Fila 3: 4 elementos")
    print("  Fila 4: 5 elementos")
    print("  Fila 5: 6 elementos")
    print("\nTotal: 21 elementos")

    # Iniciar el menú interactivo
    menu()


# Punto de entrada del programa
# Complejidad global: O(n^2) en tiempo y espacio
if __name__ == "__main__":
    main()
```

---

3. Análisis de Complejidad Computacional

3.1 Notación Big O

La **notación Big O** describe el comportamiento de un algoritmo en función del tamaño de entrada, específicamente en el peor caso.

3.2 Complejidad de las Operaciones Principales

Creación del Arreglo - crear_arreglo()

Complejidad Temporal: $O(n^2)$

- El bucle externo se ejecuta **n veces** (6 filas)
- Cada iteración i crea una lista de tamaño (i+1)
- Total de operaciones: 1 + 2 + 3 + 4 + 5 + 6 = 21 = n(n+1)/2
- Simplificando: $O(n^2)$

Complejidad Espacial: $O(n^2)$

- Se almacenan n(n+1)/2 elementos
- En términos asintóticos: $O(n^2)$

Carga de Datos - cargar()

Complejidad Temporal: $O(n^2)$

Operaciones:

- Fila 0: 1 lectura

- Fila 1: 2 lecturas

- Fila 2: 3 lecturas

- ...

- Fila n-1: n lecturas

Total: $n(n+1)/2 = O(n^2)$

Complejidad Espacial: O(1)

- No se crea espacio adicional

- Solo se modifican elementos existentes

Impresión - imprimir()

Complejidad Temporal: $O(n^2)$

- Se recorren todos los elementos para imprimir
- Número de operaciones: $n(n+1)/2 = O(n^2)$

Complejidad Espacial: O(1)

- Solo se usan variables temporales para formateo

Estadísticas - imprimir_estadisticas()

Complejidad Temporal: $O(n^2)$

- Se recorre cada elemento para calcular suma y promedio
- Operaciones: $n(n+1)/2 = O(n^2)$

Complejidad Espacial: O(1)

- Variables de conteo y suma ocupan espacio constante

3.3 Tabla Resumen de Complejidades

Función	Complejidad Temporal	Complejidad Espacial
crear_arreglo()	$O(n^2)$	$O(n^2)$
cargar()	$O(n^2)$	O(1)
imprimir()	$O(n^2)$	O(1)
imprimir_estadisticas()	$O(n^2)$	O(1)
menu()	O(1) por iteración	$O(n^2)$ total

---

4. Descripción de Funciones

4.1 crear_arreglo()

Propósito: Inicializa la estructura del arreglo asimétrico.

Parámetros: Ninguno

Retorna: Lista bidimensional asimétrica con valores None

Proceso:

1. Define el número de filas (6)
2. Itera de 0 a 5
3. Para cada iteración i, crea una fila con (i+1) elementos
4. Agrega cada fila al arreglo principal

Código clave:

```
for i in range(num_filas):
  fila = [None] * (i + 1)  # Crea fila de tamaño i+1
  arreglo.append(fila)
```

4.2 cargar(arreglo)

Propósito: Permite al usuario ingresar datos en el arreglo.

Parámetros:

- arreglo: Lista bidimensional a llenar

Retorna: Nada (modifica el arreglo directamente)

Características:

- Solicita datos elemento por elemento
- Valida entrada numérica (int o float)
- Acepta texto si no es numérico
- Muestra progreso por fila

Proceso:

1. Recorre cada fila del arreglo
2. Para cada elemento, solicita valor al usuario
3. Intenta convertir a número (int → float → string)

4. Asigna el valor validado a la posición correspondiente

4.3 imprimir(arreglo)

Propósito: Muestra el contenido del arreglo en formato tabular.

Parámetros:

- arreglo: Lista bidimensional a imprimir

Retorna: Nada (imprime en consola)

Características:

- Formato visual con corchetes [ ]
- Alineación centrada de elementos
- Identificación de número de fila
- Separadores visuales

Formato de salida:

==========

CONTENIDO DEL ARREGLO ASIMÉTRICO

==========

Fila 0: [10]

Fila 1: [20] [30]

Fila 2: [40] [50] [60]

...

4.4 imprimir_estadisticas(arreglo)

Propósito: Calcula y muestra estadísticas del arreglo.

Parámetros:

- arreglo: Lista bidimensional a analizar

Retorna: Nada (imprime estadísticas)

Estadísticas calculadas:

- Total de elementos

- Cantidad de elementos numéricos
- Suma de valores numéricos
- Promedio de valores numéricos

Proceso:

1. Inicializa contadores en 0
2. Recorre todos los elementos
3. Identifica tipos de datos con isinstance()
4. Calcula suma y promedio de numéricos
5. Muestra resultados formateados

4.5 menu()

Propósito: Gestiona la interfaz de usuario y opciones del programa.

Parámetros: Ninguno

Retorna: Nada

Opciones del menú:

1. **Cargar datos**: Ejecuta cargar()
2. **Imprimir arreglo**: Ejecuta imprimir()
3. **Mostrar estadísticas**: Ejecuta imprimir_estadisticas()
4. **Visualizar estructura**: Muestra la estructura del arreglo
5. **Salir**: Termina el programa

Características:

- Bucle infinito hasta opción "Salir"
- Validación de arreglo cargado
- Mensajes de error informativos
- Control de flujo con variable arreglo_cargado

4.6 main()

Propósito: Función principal que inicia el programa.

Parámetros: Ninguno

Retorna: Nada

Proceso:

1. Muestra información introductoria
2. Explica la estructura del arreglo
3. Llama a la función menu()

---

5. Flujo del Programa

5.1 Diagrama de Flujo General

INICIO

↓

[main()] - Muestra introducción

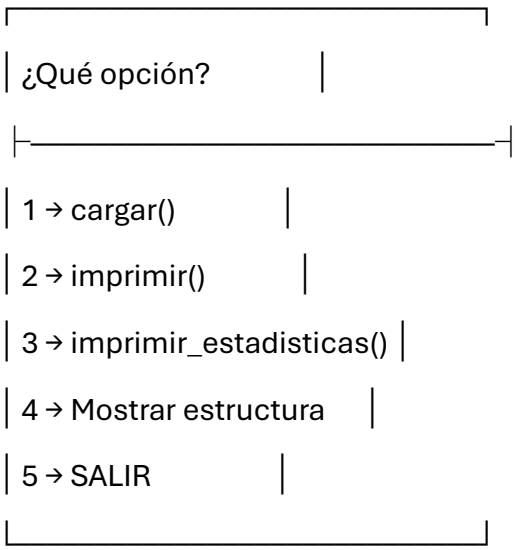
↓

[menu()] - Muestra opciones

↓

Usuario selecciona opción

↓



↓

¿Salir? → NO → Volver al menú

↓

SÍ

↓

FIN

5.2 Secuencia Típica de Uso

1. Usuario ejecuta el programa
2. Ve la introducción y menú
3. Selecciona opción 1 (Cargar datos)
4. Ingresa 21 valores (uno por cada posición)
5. Selecciona opción 2 (Imprimir arreglo)
6. Ve el arreglo completo formateado
7. Selecciona opción 3 (Estadísticas)
8. Ve análisis de los datos ingresados
9. Selecciona opción 5 (Salir)

---

6. Ejemplos de Uso

6.1 Ejemplo: Carga de Datos Numéricos

Entrada del usuario:

Fila 0, posición [0][0]: 10

Fila 1, posición [1][0]: 20

Fila 1, posición [1][1]: 25

Fila 2, posición [2][0]: 30

Fila 2, posición [2][1]: 35

Fila 2, posición [2][2]: 40

...

Resultado en memoria:

arreglo = [

  [10],

  [20, 25],

  [30, 35, 40],

  [45, 50, 55, 60],

  [65, 70, 75, 80, 85],

  [90, 95, 100, 105, 110, 115]

]

6.2 Ejemplo: Salida de Impresión

==========

CONTENIDO DEL ARREGLO ASIMÉTRICO

==========

Fila 0: [10]

Fila 1: [20] [25]

Fila 2: [30] [35] [40]

Fila 3: [45] [50] [55] [60]

Fila 4: [65] [70] [75] [80] [85]

Fila 5: [90] [95] [100] [105] [110] [115]

==========

6.3 Ejemplo: Estadísticas

==========

ESTADÍSTICAS DEL ARREGLO

==========

Total de elementos: 21

Elementos numéricos: 21

Suma de valores numéricos: 1260

Promedio de valores numéricos: 60.00

==========

6.4 Ejemplo: Datos Mixtos (Números y Texto)

Entrada:

Fila 0: "Inicio"

Fila 1: 10, 20

Fila 2: "A", "B", "C"

...

Resultado:

```
arreglo = [
  ["Inicio"],
  [10, 20],
  ["A", "B", "C"],
  ...
]
```

---

7. Conclusiones

7.1 Ventajas del Diseño

Flexibilidad: El arreglo asimétrico permite estructuras de datos no uniformes

Eficiencia de memoria: Solo se asigna el espacio necesario para cada fila

Documentación completa: Cada función está ampliamente documentada

Análisis de complejidad: Se incluye análisis detallado de rendimiento

Interfaz amigable: Menú interactivo fácil de usar

7.2 Conceptos Aprendidos

1. **Estructuras de datos asimétricas**: Listas de listas con longitudes variables
2. **Complejidad computacional**:
    - Análisis de tiempo y espacio
    - Notación Big O
    - Cálculo de operaciones
3. **Buenas prácticas de programación**:
    - Funciones modulares
    - Documentación interna
    - Validación de entrada
    - Manejo de errores
4. **Operaciones sobre arreglos**:
    - Creación
    - Carga de datos
    - Recorrido
    - Impresión formateada

7.3 Aplicaciones Prácticas

Este tipo de estructura es útil en:

- **Matrices triangulares** (como la matriz de Pascal)
- **Tablas de multiplicar** asimétricas
- **Almacenamiento de datos jerárquicos**
- **Representación de grafos** (listas de adyacencia)
- **Sistemas de almacenamiento eficiente** donde no todas las filas requieren la misma capacidad

7.4 Complejidad Final del Programa

Complejidad Temporal Total: $O(n^2)$

La operación dominante es el recorrido completo del arreglo (carga, impresión, estadísticas), que requiere visitar los n(n+1)/2 elementos.

Complejidad Espacial Total: $O(n^2)$

El espacio principal lo consume el arreglo bidimensional con sus 21 elementos.

Para n = 6 filas:

- Elementos = 6(7)/2 = 21
- Complejidad = $O(6^2) = O(36) \approx O(21)$

Referencias

- **Complejidad computacional**: Análisis de algoritmos y estructuras de datos
- **Python Lists**: Documentación oficial de Python sobre listas
- **Big O Notation**: Teoría de complejidad algorítmica
- **Arreglos irregulares (Jagged Arrays)**: Estructuras de datos no uniformes

---

Información del Programa

Lenguaje: Python 3.x

Archivo fuente: arreglo_asimetrico.py

Requisitos: Python 3.6 o superior

Ejecución: python arreglo_asimetrico.py